

Parallel Grammars

A Phenomenology

Henning Fernau

(fernau@{cs.newcastle.edu.au,informatik.uni-tuebingen.de})

University of Newcastle

School of Electrical Engineering and Computer Science

University Drive, NSW 2308 Callaghan, Australia

Universität Tübingen

Wilhelm-Schickard-Institut für Informatik

Sand 13, D-72076 Tübingen, Germany

Abstract. The aim of this paper is at least twofold:

- to give prospective PhD students in the area hints at where to start a promising research and
- to supplement earlier reference lists on parallel grammars, trying to cover recent papers as well as “older” papers which were somehow neglected in other reviews.

Together with the nowadays classical book on L systems by G. Rozenberg and A. Salomaa and with the articles on L systems in the “Handbook of Formal Languages,” researchers will be equipped with a hopefully comprehensive list of references and ideas around parallel grammars.

Keywords: Formal languages, grammars, automata, rewriting, parallelism

1. Introduction and disclaimer

It is simply impossible to cover all aspects of parallel grammars in the short space and time given to the subject here. Moreover, it is not very meaningful to narrow down the subject to a review on Lindenmayer systems, which is of course tempting. Then, this chapter would be more or less a repetition of material presented elsewhere in an excellent way, see, e. g., [197, Vol. 1, Chapter 5] or [197, Vol. 3, Chapter 9] in the Handbook of Formal Languages.

Instead, I will try to give an overview covering as many aspects of parallel grammars as possible. As probably unavoidable, this overview will be partly biased by the author’s own interests and his personal history and background. I ask anyone who feels that his or her work is underrepresented in this chapter to excuse this bias. In a way, this paper can be also read as an annotated bibliography which largely supplements the quoted chapters from the Handbook of Formal Languages. Seen in this way, we hope that this paper is of help to other researchers in this area, as well.

© 2003 *Kluwer Academic Publishers. Printed in the Netherlands.*

Moreover, we give many possible research directions, hence hopefully inspiring further studies. These directions are classified as Project Proposal if we think the corresponding research leads to a bigger project, including the possibility of writing several papers and/or a PhD thesis on it, while a classification as Paper Proposal means that we think that the idea is worth pursuing but that it will probably only lead to one or two papers but not to a whole thesis. Due to the very nature of research, the Project Proposals tend to be formulated in a rather general, often informal way, while the Paper Proposals are more concrete (mathematical) problems. For people starting their career in research, it might be best to firstly work on some concrete problems and then move towards the more general ones.

We assume throughout the paper some familiarity of the reader with basic concepts of formal language theory. Especially, the Chomsky hierarchy

$$\mathcal{L}(\text{REG}) \subsetneq \mathcal{L}(\text{CF}) \subsetneq \mathcal{L}(\text{CS}) \subsetneq \mathcal{L}(\text{RE})$$

should be known.

In order to do our exposition in due space and time, we will refrain from giving precise induction proofs of the mentioned results but focus on explaining the underlying ideas. This way, we hope that this paper may serve as a good introduction to formal language theory research as such, not restricted just to parallel grammars. Sometimes, we even refrain from giving exact definitions. In any case, we give enough references in order to trace down the exact meanings.

This chapter is organized as follows: In Section 2, we will exhibit the different phenomena which can be encountered within parallel grammars. Section 3 explains several phenomena and their formal language modelling somehow in more depth. In Section 4, we present several definitions important for parallel grammars and for understanding their relations with other language description mechanisms. In Section 5, we present some language theoretic results on parallel grammars, mainly focussing on hierarchy questions. There, we also present some new results concerning *k*ETOL systems. We discuss in Section 6 several interesting open questions concerning parallel grammars which could be a starting point of research, not necessarily giving way to a whole PhD thesis. Section 7 is devoted to the question which phenomena were (originally) modelled by (parallel) grammars and whether it is possible to interpret and communicate experiences and results meaningfully “back” to the origin of the ideas. In Section 8, we discuss ways in which parallel grammars has been applied to other areas and propose a number of possible new application fields in terms of research projects. Mathematical challenges are proposed as research projects in Section 9.

We also provide a list of interesting web-sites and give a list of more than 200 references which—in large part—can be seen as a supplement and update to the huge literature lists of [198] and of [197, Vol. 1, Chapter 5] or of [197, Vol. 3, Chapter 9], as well as of other chapters of the handbook [197].

Finally, we hope to stimulate also people already working for some time in formal language theory by providing several examples of applications of formal language theory which seem to be rather unknown (if not surprising) even to specialists in the field.

2. A phenomenology

From an abstract level, parallel grammars (and automata) try to model phenomena encountered in parallel processing.

Since in grammatical mechanisms for generating formal languages the basic and simplest processing step is that of a replacement of one symbol (or a string of contiguous symbols in the non-context-free case, but this can be seen as a special case of the context-free case if we consider parallel derivations restricted to adjacent symbols) by a string of symbols. The most straightforward way to incorporate or model parallelism here is certainly to consider the possibility of doing a number of derivation sub-steps in parallel—hence defining the notion of a *parallel derivation step*.

One obvious possibility here is to allow (and enforce) the parallel execution of a context-free derivation step in all possible places. This basically leads to the theory of Lindenmayer systems, see [137, 138, 198].

For physical reasons (and often also biologically motivated), it is unreasonable to assume that basic processing units (cells in the biological context) located in the midst of a larger computing facility (body) may grow arbitrary (by “splitting” the cell due to some rule of the form $A \rightarrow BC$). This idea led, on the one hand, to the theory of cellular automata (when we, for the moment, forget about the differentiating aspects of grammars and automata), and on the other hand to considering Lindenmayer systems with apical growth, as done in [5, 7, 6, 12, 156, 179]. In this context, we should also mention array grammars, which adhere to similar growth restriction, although they are not necessarily parallel in nature, but, as often the case in formal language theory, different concepts of rewriting—here the ideas of array grammars and of parallelism in rewriting—can be fruitfully combined [28, 68, 70, 172, 79, 229]. Even more interesting than the usually considered one-dimensional case are the higher-dimensional cases,

where the restrictions of growth in the innermost cells of the body become even more obvious, see [25, 34, 40, 79, 214].

When comparing the growth patterns, i. e., consider the development of the length of the cellular strings as a function of time, which are typically found within (certain types of simple) Lindenmayer systems [227], they mostly show polynomial or exponential growth. On the other hand, in nature, we never encounter unlimited exponential growth; this sort of growth pattern can only be observed over a limited time (even with cancer cells or other cells living as “parasites”), since at some point of time the host will die and this will immediately stop any growth process. The most “natural” way to model this phenomenon is to incorporate environmental factors, which are mostly determining the limitation of growth, within the formal language model itself. In this way, we arrive at several forms of limited parallel rewriting as discussed in [56, 55, 86, 204, 243] or at forms of ecogrammar systems, see [30].

When discussing partial parallel rewriting,¹ we could consider the classical forms of sequential rewriting as a very special extreme case. Coming from this end of the possible spectrum of parallel rewriting, it is quite natural to consider also the parallel replacement of only “a few” symbols as one rewriting step. As usual, both “grammatical” and “automata-based” models can be considered here. Due to the multitude of possible replacement strategies, we mention only a few of them in the following:

- Equipping finite automata with more than one head can enhance the power of finite automata beyond accepting regular languages, see, e. g., [188, 213]. This idea is closely related to certain forms of parallel grammars, as pointed out in [70, 172], basically leading to a form of absolutely parallel (array) grammars further discussed in the next item.
- In a grammatical mechanism where the replacement of terminal symbols is not permitted, allowing a grammar to produce only sentential forms containing a bounded number of nonterminal occurrences is a severe restriction basically known as “finite index restriction.” In fact, regulated grammars of finite index (which in case of most regulated rewriting mechanisms turn out to be of equal generative power, see [72] for a recent account incorporating lots of references) can be equivalently “interpreted” as doing one restricted parallel derivation step involving at most k nonterminals, where k is the finite index of the grammar and all

¹ We prefer to call this form of partial parallel rewriting “partial” instead of “limited,” since the latter term has some nowadays fixed technical meaning.

(ℓ) nonterminals of a sentential form w must be replaced by a rule of the form

$$(A_1, \dots, A_\ell) \rightarrow (w_1, \dots, w_\ell)$$

and the nonterminals A_1 through A_ℓ appear in that order in w . These *absolutely parallel grammars* introduced by V. Rajlich [182, 183] generalize towards *scattered context grammars* if we do not insist on replacing all symbols of a sentential form in parallel, but can select some of them (obeying the order imposed by the rule) [93, 94]. A related notion is that of simple matrix grammars, see [99, 144, 162, 163, 205].

- Taking into account complexity restrictions, it is also interesting to study Turing machines having multiple heads, see [246] or Site 21.
- Instead of allowing or even enforcing that *all* occurrences of symbols of a sentential form are replaced in one step, only a certain limited number of replacements per step is allowed. This leads to the theory of (uniformly) limited Lindenmayer systems, which we discuss further in some detail below.
- Allowing finite automata or regular grammars to work in parallel (in a possibly synchronized fashion) and combining the results in the end is one of the historic roots leading towards what is now known as parallel communicating grammar systems.² Besides allowing combination of results in the end, certain forms of communicating the results have been discussed, see [30, 32, 66, 221, 222]
- It is possible to consider combination of different forms of language models of parallel computation. For example, D. Wätjen recently investigated parallel communicating limited and uniformly limited 0L systems [242], as well as the team mechanism in limited rewriting [240], while Gy. Vaszil studied parallel communicating Lindenmayer systems in a series of papers [222–224]. Another more classical example are the so-called Russian parallel grammars which combine features from sequential and (Indian) parallel rewriting, see [136].

Still other forms of parallelism can be observed in mathematics: when a variable is replaced in an expression, this replacement has to be consistent, meaning that each variable occurrence is replaced in the same way. This phenomenon is also encountered in deterministic Lindenmayer systems, as well as in the very similar Indian parallel

² We will discuss such systems further in RP *VIII* and give references there.

grammars (or Bharat systems), see [127, 206]. Although the way of obtaining languages is somehow different, we also have to mention the notion of pattern languages in this context, see [8] and the Introduction/Synopsis on formal languages found in [197, Vol.1]; in fact, these ideas have also been used to construct sorts of parallel grammars [38, 148, 170]. Applications in the direction of parallel term rewriting [29] point in a similar direction.

We conclude this section with a collection of places on the landscape of formal languages where parallelism popped up in one form or another; due to space limitations, we cannot go into details here, but, generally speaking, showing interrelations between all these mechanisms would be indeed a fruitful stimulation of research.

- Collage grammars [49] are a grammatical mechanism based on hyperedge replacement which has been proven to be very versatile for picture description purposes; often, parallel features are incorporated—or can be easily simulated—here.
- Besides studying grammars or automata, formal language theory also deals with exploring operators on languages. Evidently, there are also many “parallel” operators around. For example, the operation of homomorphism can be seen as modelling a complete parallel replacement operation; accordingly, D. Wätjen introduced “(uniformly) k -limited homomorphism”³ modelling partial parallel replacements [231, 232]. Insertion and deletion can also be seen as parallel operations, as well as different forms of shuffle, see [101, 130, 129, 131]. Another example of this interpretation can be found in [91].
- Given a classical sequential context-free grammar G , it has been studied the maximal height $h(n)$ of possible derivation trees of generatable words of length n . $h(n)$ can be seen as a measure of parallelism within a derivation of a word, see [20, 184].
- In a maybe broader sense, trace languages can be seen as a (very successful) model of distributed computation. The interested reader is referred to [197, Vol.3, Chapter 8] or the monograph [45].

³ which aren’t any morphisms any more from the strict algebraic perspective

3. Selected phenomena

Looking closer at modeling biological phenomena, it is quite obvious how certain cases often distinguished in formal language theory actually reflect different biological phenomena:

erasing production A rule of the form $A \rightarrow \varepsilon$ models the death of a cell (which has been in state A before its disappearance).

chain rule A rule of the form $A \rightarrow B$ reflects a change of state of the corresponding cell.

growing rule A rule of the form $A \rightarrow BC$ models the split of a cell previously being in state A , having two “children” being in state B and C , respectively. As already discussed above, this kind of rule makes sense biologically at the outermost part of the body or in filamentous organisms, as originally targeted by A. Lindenmayer.

context A rule of the form $ABA' \rightarrow ACA'$ shows that the context-free rule $B \rightarrow C$ is only applicable within an appropriate left and right context. This way, the flow of signals can be modelled.

pure grammars They do not differentiate between terminal and non-terminal symbols, so that all sentential forms generatable from the grammar’s axiom are put into the generated language. This notion is well-motivated biologically since the alphabet symbols in L systems should denote certain “states” of cells, and it is not reasonable to consider only organisms consisting of cells being in peculiar states.

But there are more things which find rather straightforward modelisations in the theory of L systems, even though this does not reflect “usual” concepts in formal language theory. Stated positively, it shows (again) how ideas from other areas (like biology) led to interesting theoretical concepts in language theory. We mention a few examples in the following:

offsprings Even some multi-cellular organisms do not (only) rely on sexual reproduction, but they can “split” and produce offsprings in this way. (Sometimes, this sort of reproduction is only by accident, as exemplified by some worms which (may) develop into two separate organisms when split by force.) This phenomenon—called *fragmentation*—was addressed in [191].

grown-up Higher organisms go through a couple of stages until they reach full maturity. In an idealized interpretation, being grown-up

or adult means that the overall “body” does not change anymore, although “local changes” due to the death of some cells which is compensated by the birth of others are possible. Adult languages were considered in [192]. Related to this are the “death word languages” discussed by M. Kudlek [128].

interaction with neighbouring cells For biology, organisms consisting of cells which are not interacting with each other seems to be quite “unnatural.” Possibly, one would not consider such a hypothetical collection of cells as one organism on its own. Hence, it is doubtful whether all the studies undertaken on basically context-free partial parallel grammars can be meaningfully re-interpreted in biological terms. In fact, in the case of Lindenmayer systems, the incorporation of interaction was done in the very first paper [137, 138], and interactionless systems were considered only later [139]. In the case of limited Lindenmayer systems for example, interactionless systems were considered first [231] and only recently limited systems with interactions were investigated [53, 54].

If interaction with neighbouring cells is incorporated in mathematical models and formalisms, then the question arises what neighborhood actually means, especially in the multidimensional case. In the theory of cellular automata, several notions of neighbourhoods were studied, see [228].

signals Closely related to the notion of neighbourhood is the idea of using this neighbourhood in order to transport information from one part of the body to another one. Information flow techniques (or *signals*) are the basis of various algorithms for cellular automata, see [228]. Especially with parameterized L systems, this issue arose also in L system models.

changing environmental conditions The change of growth patterns in organisms can have various reasons, one of them being the change of the environmental conditions which are responsible, e. g., for the annual rings observable in cut trees. This has led to the introduction of so-called *tables* in Lindenmayer systems, where each table is representing a certain special environment [190].

non-observable internal changes Often, it is not possible to directly observe what is going on in living organisms. The observable changes can be due to various reasons.

In the more abstract model of Lindenmayer systems, this means that we are not dealing with the language of words generated by

an L system, but rather by codings of these words, where a coding is a “renaming” homomorphism, see [193].

4. Some definitions and examples

Let us start with defining L systems (without interaction) and then discuss several variants.

An *interactionless Lindenmayer system with tables*, for short a *TOL system*, is given by a triple $G = (\Sigma, H, \omega)$, where the components fulfill the following requirements:

- Σ is the alphabet.
- H is a finite set of finite substitutions $H = \{h_1, \dots, h_t\}$, i. e., $h_i : a \mapsto \{w_1, \dots, w_{n_{i,a}}\}$, which means that each h_i can be represented by a list of context-free rules $a \rightarrow w$ such that $a \in \Sigma$, $w \in \Sigma^*$; this list for h_i should satisfy that each symbol of Σ appears as the left side of some rule in h_i .
- $\omega \in \Sigma^*$ is the axiom.

Some special cases are:

- $t = 1$: we have a *0L system*.
- $\forall 1 \leq i \leq t \forall a \in \Sigma: n_{i,a} = 1$: *deterministic TOL systems*, or *DTOL* for short; in other words, each h_i is a homomorphism.

G defines a derivation relation \Rightarrow by $x \Rightarrow y$ iff $y \in h_i(x)$ for some $1 \leq i \leq t$, where we now interpret h_i as substitution mapping. The language generated by G is

$$L(G) = \{w \in \Sigma^* \mid \omega \Rightarrow^* w\}$$

with \Rightarrow^* denoting, as usual, the reflexive and transitive closure of \Rightarrow .

Possible variants are the following ones:

- Given some “terminal alphabet” Δ , one might consider the *extended language* $E(G, \Delta) = L(G) \cap \Delta^*$.
- Given some coding function $c : \Sigma \rightarrow \Delta$, the language $c(L(G)) = \{c(w) \mid w \in L(G)\}$ can be investigated.
- $A(G) = \{w \in L(G) \mid \{w\} = \{u \mid w \Rightarrow u\}\}$ is the *adult language* generated by G .

The corresponding language classes will be denoted by ET0L, CT0L, and AT0L, respectively. Again, variants like ‘‘A0L’’ denoting the adult languages definable by 0L systems, can be considered.

Let us explain these variants by means of an example:

Example 1. Consider the language

$$L = \{a^n b^n c^n \mid n \geq 1\}.$$

$$\begin{aligned} A &\rightarrow AA', & A' &\rightarrow A' \\ B &\rightarrow BB', & B' &\rightarrow B' \\ C &\rightarrow CC', & C' &\rightarrow C' \end{aligned}$$

are the rules of a 0L system with axiom ABC which, together with the coding $A, A' \mapsto a$, $B, B' \mapsto b$, and $C, C' \mapsto c$ describes L . More precisely, for the n -step derivation, we can easily see that

$$ABC \Rightarrow^n A(A')^n B(B')^n C(C')^n$$

from which the claim follows.

Taking these same rules plus the rules

$$\begin{aligned} A &\rightarrow a, & A' &\rightarrow a, & a &\rightarrow F \\ B &\rightarrow b, & B' &\rightarrow b, & b &\rightarrow F \\ C &\rightarrow c, & C' &\rightarrow c, & c &\rightarrow F \\ F &\rightarrow F \end{aligned}$$

result in an E0L system (with axiom ABC) which also generates L when taking $\{a, b, c\}$ as terminal alphabet. Observe the technique of *synchronization by failure symbol F* which we used here and which is quite usual when designing ET0L systems.

The reader is encouraged to try to find an A0L system or, if (s)he does not succeed, an AT0L system for L . This should give him/her some intuition on how the adult mechanism works.

Let us end this subsection on different variants of Lindenmayer systems by introducing *fragmentation* (Finnish: *Jakautua*) more formally: Let $G = (\Sigma, H, \omega)$ be a T0L system. Let $q \in \Sigma$ be a special symbol with only production $q \rightarrow q$ having q as left-hand side. Then,

$$J(G, q) := \{v \in (\Sigma \setminus \{q\})^* \mid \omega \xrightarrow{*} x, (x = v \vee x = uqvqw)\}.$$

(G, q) is also called a *JT0L system*. Obviously, it is also possible to consider fragmentation as operation on languages $L \subseteq \Sigma^*$:

$$J(L, q) := \{v \in (\Sigma \setminus \{q\})^* \mid \exists x \in L (x = v \vee x = uqvqw)\}.$$

Then, $J(G, q) = J(L(G), q)$.

Example 2. Consider the J0L system (G, q) with

$$G = (\{a, b, q\}, h, aba) \quad \text{and} \\ h = \{a \rightarrow a, b \rightarrow abaqaba, q \rightarrow q\}.$$

By induction, it is easy to see that

$$J(G, q) = \{aba^n, a^nba \mid n \geq 1\} :$$

In one derivation step, we have

$$aba \Rightarrow aabaqabaa,$$

so that aba , $aaba$ and $abaa$ lie in $J(G, q)$. If aba^n lies in $J(G, q)$, then aba^{n+1} lies in $J(G, q)$, as well:

$$aba^n \Rightarrow aabaqabaa^n.$$

Similarly, if a^nba lies in $J(G, q)$, then $a^{n+1}ba$ lies in $J(G, q)$. By induction, the inclusion \subseteq follows. The other direction is easily seen by induction along the same lines, since no other words are producible by G .

Up to now, we only considered *interactionless* Lindenmayer systems. Going back to the original motivation for introducing Lindenmayer systems, this means that we disallow any communication between neighbouring cells, which is biologically not a reasonable assumption. In $(i, j)L$ systems (or IL systems if the numbers i and j are arbitrary), to the contrary, the i symbols “to the left” and the j symbols “to the right” of the symbol which has to be replaced are taken into account. If there are not $i > 0$ symbols to the left (e. g., when considering the leftmost symbol), a special symbol $\#$ is considered as being read. Similarly, “missing” right neighbours are handled. Let us consider a concrete example:

Example 3. Let us consider the DIL system with axiom ad and the following rules:

$$\begin{array}{c|cccc} & a & b & c & d \\ \hline \# & c & b & a & d \\ a & a & b & a & d \\ b & a & b & a & d \\ c & b & c & a & ad \\ d & a & b & a & d \end{array}$$

The derivation proceeds as follows:

$$ad \Rightarrow cd \Rightarrow aad \Rightarrow cad \Rightarrow abd \Rightarrow cbd \Rightarrow acd \Rightarrow caad \Rightarrow abaad$$

It can be observed that this system, also known as *Gabor’s sloth* (since it was created by Gabor Herman) grows at a logarithmic scale.

As regards different notions of partial parallelism, we have to redefine the derivation relation accordingly. In the following, $|x|_a$ gives the number of occurrences of a in x , generalizing the well-known notation $|x|$ yielding the length of x . Formally, all systems discussed below can be specified like a TOL system $G = (\Sigma, H, \omega)$, $H = \{h_1, \dots, h_t\}$. Naturally, the rewriting relation is defined in a different manner.

Bharat (T0B) systems $x \Rightarrow y$ iff $\exists 1 \leq i \leq t \exists a \in \Sigma$ such that all occurrences of a in x are replaced by some word in $h_i(a)$ to obtain y from x ;

k -limited (k ITOL) systems $x \Rightarrow y$ iff $\exists 1 \leq i \leq t \forall a \in \Sigma: \min\{|x|_a, k\}$ occurrences of a in x are replaced by some word in $h_i(a)$ to obtain y from x ;

uniformly k -limited (ukITOL) systems $x \Rightarrow y$ iff $\exists 1 \leq i \leq t: \min\{|x|, k\}$ symbols in x are replaced according to h_i to obtain y from x .

Example 4. Let us consider the following concrete system in more detail:

$$G = (\{a, b\}, \{\{a \rightarrow aa, b \rightarrow ab\}\}, abb)$$

Let us look at some derivation sequences when interpreting G as OL system or as some form of partial parallel grammar.

OL system $abb \Rightarrow aaabab \Rightarrow a^6aba^2ab$.

OB system $abb \Rightarrow aabb \Rightarrow aaabab \Rightarrow a^3abaab$

110L system $abb \Rightarrow aaabb \Rightarrow aaaabab \Rightarrow a^4ba^3b$

210L system $abb \Rightarrow aaabab \Rightarrow a^5abaab$

u210L system $abb \Rightarrow aabab \Rightarrow a^4bab \Rightarrow a^6bab$

It would be a good exercise to describe the languages generated by G formally in set notation. Observe the different “growth patterns.”

Some other mechanisms of partial parallel rewriting—which are sometimes also considered as forms of regulated rewriting—, can be presented in a different form as follows:

A *scattered context grammar* is a construct

$$G = (\Sigma, P, S, \Delta), \quad \Delta \subset \Sigma, S \in N := \Sigma \setminus \Delta.$$

Here, $P = \{p_1, \dots, p_t\}$ is a set of rules with

$$p_i = (A_1, \dots, A_{n_i}) \rightarrow (w_1, \dots, w_{n_i})$$

with $A_j \in N$ and $w_j \in \Sigma^*$ for $1 \leq j \leq n_i$. The rewrite relation is defined as follows: $x \Rightarrow y$ iff there is an i , $1 \leq i \leq t$ and if there are words $u_0, \dots, u_t \in \Sigma^*$ such that

$$x = u_0 A_1 u_1 \dots A_{n_i} u_{n_i} \quad \text{and} \quad y = u_0 w_1 u_1 \dots w_{n_i} u_{n_i}. \quad (1)$$

Then, the language generated by G is again denoted by

$$L(G) = \{w \in \Delta^* \mid S \xRightarrow{*} w\}.$$

Two variants (which are severe restrictions as to the generative power of scattered context grammars) are important, since they “only” generate semilinear languages and can be characterized by many other mechanisms. The family of scattered context languages is denoted as \mathcal{LSC} (or as $\mathcal{L}(\text{SC} - \varepsilon)$ if erasing rules are disallowed).

1. If, in addition, $u_0, \dots, u_{n_i} \in \Delta^*$ in Eq. 1, then we have an *absolutely parallel grammar*. These grammars can be characterized by regulated grammars with the so-called finite index restriction imposed, as well as by a certain variant of grammar systems with prescribed teams, see [71, 172] and Theorem 2.
2. If, with exception of the start rules, each n_i equals a specific n , then (in principle) we arrive at *simple matrix grammars*. Traditionally, their rewrite relation is defined slightly differently: each rule has to be performed in a leftmost fashion; to make this restriction meaningful, one has to require moreover that the nonterminal alphabet is split into n disjoint sub-alphabet, each “responsible” for a certain area in the grammar, see [99, 162]. However, to our knowledge, all important results would stay the same if we consider the variant of simple matrix grammars we defined by introducing it as a special case of scattered context grammars, also confer [164].

Of some importance are the (even more special cases) of

- *linear simple matrix languages*—where each rule, with exception of the start rules—is a “list of linear rules,” and of
- *right-linear simple matrix languages* (or *equal matrix languages*)—where each rule, with exception of the start rules—is a “list of right-linear rules.”

Note that in these two special cases, the different interpretations of simple matrix grammars coincide. In the literature, these grammars were discussed in [63, 64, 144, 163, 205].

As seen in [186], the so-called *n-parallel grammars* (and all the corresponding special cases discussed in a series of papers by Rosebrugh and Wood [185–187, 194, 245]) are in turn special cases of simple matrix grammars; basically, the “communication” between the different “branches” in a derivation of a simple matrix grammars (enabled by the start rule) is not possible with *n-parallel grammars*; the only “regulation” is by means of the “synchronization feature” inherent to the definition of a simple matrix grammar derivation step.

Let us explain the mechanisms again by means of examples. More precisely, we discuss three examples which are usually considered as the key examples of linguistically relevant languages. Observe that it is possible to describe all these languages by means of the simplest partial parallel grammar mechanism we just described, namely equal matrix languages. This comes by not much of a surprise, since already this language family forms a nontrivial subfamily of the family of tree-adjoined grammars (TAG) which is usually considered to be of a considerable linguistic relevance, see [64, 109, 108].

Example 5. Let

$$G_i = (\{S, S_1, S_2, S'_1, S'_2, a, b\}, P_i, S, \{a, b\}),$$

where P_1 contains the following rules:

$$\begin{aligned} (S) &\rightarrow (S_1 S_2 S'_1) \\ (S_1, S_2, S'_1) &\rightarrow (aS_1, bS_2, aS'_1) \\ (S_1, S_2, S'_1) &\rightarrow (a, b, a), \end{aligned}$$

P_2 contains the following rules:

$$\begin{aligned} (S) &\rightarrow (S_1 S_2) \\ (S_1, S_2) &\rightarrow (aS_1, aS_2) \\ (S_1, S_2) &\rightarrow (bS_1, bS_2) \\ (S_1, S_2) &\rightarrow (a, a) \\ (S_1, S_2) &\rightarrow (b, b) \end{aligned}$$

and P_3 contains the following rules:

$$\begin{aligned} (S) &\rightarrow (S_1 S_2) \\ (S_1, S_2) &\rightarrow (aS_1, aS_2) \\ (S_1, S_2) &\rightarrow (S'_1, S'_2) \\ (S'_1, S'_2) &\rightarrow (bS'_1, bS'_2) \\ (S'_1, S'_2) &\rightarrow (\varepsilon, \varepsilon). \end{aligned}$$

Then, the generated languages are the following ones:

$$\begin{aligned} L(G_1) &= \{a^n b^n a^n \mid n \geq 1\} \\ L(G_2) &= \{ww \mid w \in \{a, b\}^+\} \\ L(G_3) &= \{a^n b^m a^n b^m \mid n, m \geq 0\} \end{aligned}$$

We can state the following small result without proof, since it immediately follows by our definitions. It is interesting here because it shows that the examples above also apply to absolutely parallel grammars.

LEMMA 1. *Each linear simple matrix grammar is an absolutely parallel grammar.*

A technical remark: This lemma somehow explains why the examples proving the strictness of the “finite index hierarchy” for regulated grammars and the examples showing the strictness of the language hierarchy induced by allowing (only) n rules in a matrix of some linear simple matrix language are basically the same; in fact, this observation could be helpful in a combined presentation of these results.

Finally, we will need some knowledge about language families discussed in detail in a lecture on regulated rewriting. We only “sketch” the corresponding definitions here:

- An *ordered grammar* is a context-free grammar together with a partial order on its rule set. A rule is only applicable to some sentential form if it is applicable in the ordinary sense known from context-free grammars and if no other “greater” rule is applicable (in the context-free sense) to that sentential form; the yield of an application of a rule is as known from Chomsky grammars. The corresponding language families are written as $\mathcal{L}(\text{O}, \text{CF}[-\varepsilon])$, where $-\varepsilon$ indicates that erasing rules are disallowed.
- A *programmed grammar* has—as rules—fragments of goto-programs of the following form:

$$\left(\ell : A \rightarrow w, \begin{cases} \text{if } A \text{ occurs in } \xi, & \text{then apply rule and goto } \sigma \\ \text{if } A \text{ does not occur in } \xi, & \text{then goto } \phi \end{cases} \right)$$

Here, ℓ is some label, ξ is the current sentential form, σ and ϕ are sets of labels of the given programmed grammar, called *success field* and *failure field*, respectively. For a label set Λ , goto Λ means that in the next derivation step, only rules labelled with some $\ell \in \Lambda$ can be applied. The corresponding language families are written as $\mathcal{L}(\text{P}, \text{CF}[-\varepsilon], \text{ac})$, where $-\varepsilon$ indicates that erasing rules are disallowed. Important special cases are:

- If all failure fields are empty, then we have a grammar *without appearance checking* whose language family is denoted as $\mathcal{L}(P, CF[-\varepsilon])$.
- If in each rule the success field and the failure field are identical, we have a grammar *with unconditional transfer* whose language family is denoted as $\mathcal{L}(P, CF[-\varepsilon], ut)$.

It is also possible to consider ordered as well as programmed grammars with other rule sets than context-free rules. Since we will not consider them here, we exclude them from the definition above. Nonetheless, from the viewpoint of parallel grammars, combinations of techniques from regulated rewriting with parallel grammars are quite interesting. These kinds of grammars have been discussed for various variants of parallel grammars. A (surely incomplete) list of papers is [195, 37, 77, 85, 235–238].

In order to get him/herself familiar with these mechanisms, the reader is invited to prove the following (easy) result:

THEOREM 1. $\mathcal{L}(O, CF-\varepsilon) \subseteq \mathcal{L}(P, CF-\varepsilon, ut)$, $\mathcal{L}(O, CF) \subseteq \mathcal{L}(P, CF, ut)$.

In fact, the inclusions are proper, which can be seen by the recursiveness of ordered languages, see [60], while there are non-recursive programmed languages with unconditional transfer, see Theorem 22 below.

Although we will indicate below that programmed grammars with unconditional transfer can be characterized via 1-limited Lindenmayer systems, it would be instructive for the reader to directly prove that each ordered grammar can be simulated by a 1-limited ETOL system. A simulation by k -limited ETOL systems will be explained in Theorem 24.

An interesting variants are (regulated) context-free grammars with the finite index restriction. A grammar is said to be of *index* k if each word possesses a derivation δ in which each sentential form of δ has at most k occurrences of nonterminals. A language is of *index* k if it is generatable by a grammar of index k . A language is of *finite index* if it is of index k for some k . Without proof, we mention the following relation, see [71, 172, 199, 200]:

THEOREM 2. *The family of absolutely parallel languages can be characterized as the family of programmed languages of finite index and as the family of ordered languages of finite index.*

5. Selected (mathematical) results

This section contains some overview of results on parallel grammars. Most results on L systems can be found already in [198]. The “classical questions” were also discussed in the papers introducing different forms of partial parallel grammars, as [202, 231, 232] in the limited case. A good web-site for limited Lindenmayer systems is the homepage of D. Wätjen, see Site 20.

5.1. A TOPICAL OVERVIEW

As usually done in formal language theory, the following questions were attacked in the case of parallel grammars, as well:

- hierarchy questions,
- decidability questions, and
- closure property questions.

Furthermore, the following types of questions appear to be of general interest (and have been subject of study in parallel grammars):

- combinatorial properties,
- descriptive complexity and normal forms, and
- learning / inductive inference.

In fact, as we will explain in the following, all these properties are somehow interrelated, so that it is hard to focus on one aspect only. Nonetheless, we will restrict ourselves in the following to hierarchy questions, making deliberate use of other results when needed.

HIERARCHY QUESTIONS

What is the “power” of a concrete grammatical mechanism compared to other, possibly better known language classes? Most effort has been spent here on comparing

- parallel grammar mechanisms with the Chomsky hierarchy,
- parallel grammars with suitable types of regulated grammars,
- the different Lindenmayer system types *inter alia*, and
- partial parallel grammars with “similar” types of Lindenmayer systems.

As basic tools, inclusion relations are mostly shown by direct simulations (which sometimes rely on certain normal forms of the mechanisms), while strictness of certain inclusions can be either shown via certain combinatorial properties of language classes in question (like the well-known pumping lemmas for regular and context-free languages) or by the absence/presence of certain decidability properties, although one has to be very careful when making use of the latter tool, see [59, 62].

DECIDABILITY QUESTIONS AND COMPLEXITY

The “classical” decidability questions are:

membership This question has two variants: The *fixed membership* question is: given a language L of a certain type \mathcal{L} , is this language recursive, i. e., is there a Turing machine which may, given any word w , decide whether $w \in L$ holds or not. In a sense, this question is also a “hierarchy question,” asking whether or not \mathcal{L} is included in the class of recursive languages.

The second variant is the *general* or *uniform membership* question: given a grammar G of a certain grammar type \mathcal{G} and a word w , is $w \in L(G)$ or not?

Note that while the first type of question is really about languages and can hence be used for inferring hierarchy relations (for example, if we know that all languages in \mathcal{L} are recursive and that some languages from \mathcal{L}' are non-recursive, than we can conclude that \mathcal{L}' is not contained in \mathcal{L}), the second type of question cannot be used for this purpose. Nevertheless, answering the second type of question affirmatively in the sense that there exists an algorithm for solving the general membership problem, this immediately settles the status of the first type of question affirmatively as well: each language of the type in question is recursive, and this is also the usual proof strategy for the first question, but we will learn to know in this paper also another (indirect) proof in one case, see Theorem 26.

emptiness Is the language given by a grammar of a certain type empty or not? Note that for stating complexity results more easily, there often the negation of this problem (i. e., non-emptiness) is considered.

finiteness Is the language given by a grammar of a certain type finite or not?

equivalence Are the languages of two given grammars the same?

Other questions are: Is the language given by a grammar of a certain type of type 1,2, or 3 in Chomky's sense ? This is, more generally, the so-called *\mathcal{L} -ness* question: Is the language given by a grammar of a certain type belonging to \mathcal{L} ?

As we will see, there are also decidability questions rather specific to parallel grammars in connection with growth properties.

CLOSURE PROPERTIES

Typical questions are: Is the language class under consideration closed under the Boolean operations or under more language-theoretic operations like closure under star, concatenation, homomorphism, inverse homomorphism or intersection with regular sets. A language class closed under the last mentioned five operations and under union is called an *abstract family of languages* (AFL). A language class which is not closed under any of the six AFL operations is called an *anti-AFL*.

We now come to two issues which are also of general interest in formal language theory, before turning to things which are more special to parallel grammars.

COMBINATORIAL PROPERTIES

This kind of properties comprise pumping and interchange lemmas as well-known for regular, linear and context-free languages (in fact, there are also similar properties known for partial parallel grammars, see e. g. [234]), as well as several properties of ETOL and EDTOL languages (which are usually harder to formulate). For proving these properties, normal forms are often useful. Since combinatorial properties are logically of the form "Each language of language class \mathcal{L} satisfies ...", they are very useful for providing non-examples, hence showing the strictness of inclusions or non-comparability results.

DESCRIPTIVE COMPLEXITY ISSUES AND NORMAL FORMS

Normal forms, in general, provide a way of having, if not a unique representation, then at least a standardized representation of a language of a certain language class. Well-known examples in this respect are the Chomsky normal form for context-free grammars.

Normal forms are often helpful for establishing simulation results. The proof of combinatorial results also often depends on them. Some learning models explicitly require the use of normal forms.

A related question are questions of so-called descriptive complexity. For example, is it possible to generate each context-free language with just three nonterminals? This question has a negative answer, but

the analogous question concerning the *nonterminal complexity* of scattered context languages or programmed languages has an affirmative answer, see [65, 82, 146]. If rewriting of terminal symbols is permitted, the notion of *active symbol complexity* is to be studied instead (for details concerning this notion, we refer to [19, 198, 233]). Another question concerns the restriction of the number of rules or tables. In a sense, also the question whether or not deterministic systems are as powerful as nondeterministic ones is a question of descriptive complexity, especially, if the *degree of nondeterminism* is quantified, see [16, 17, 39]. For mechanisms involving context, also the “degree of context-sensing” can be quantified, see [74, 73] in the case of scattered context.

LEARNING/INDUCTIVE INFERENCE

The main problem in this area can be described as follows: is it possible to derive a correct (or at least approximative) description of a language of a certain class when seeing the members of the language one by one (and being given possibly additional side information). Depending on the type of side information, how the information is provided (Is the learner only “passive” or can (s)he take an active part, asking questions or making experiments?), on what is exactly required to accept the answer of a learner (Are “errors” tolerated?), and on how “certainly” a learner is required to learn “successfully,” a whole multitude of learning scenarios have been discussed in the literature.

GROWTH FUNCTIONS

This is a topic special to parallel grammars. With each DIL system G , we can associate a function g which tells the length of the word derived after n derivation steps. This *growth function* has been studied extensively, and also related notions concerning more general forms of Lindenmayer systems, see [226]. In this connection, also new types of decidability questions arise, e. g.: is the growth function of a given DOL system a polynomial function?

5.2. SOME IDEAS ON NORMAL FORMS AND DESCRIPTIVE COMPLEXITY

We are going to present here two results on ETOL systems:

LEMMA 2. *For every ETOL system, we can construct an equivalent ETOL system whose terminal symbols are only trivially rewritten, i. e., by rules of the form $a \rightarrow a$.*

For the (not difficult) proof of the lemma, we refer to [198].

THEOREM 3. *Every ETOL language is generatable by an ETOL system with only two tables.*

Proof. If L is generated by an ETOL system $G = (\Sigma, H, \omega, \Delta)$ which obeys the normal form of Lemma 2 with tables

$$H = \{h_1, \dots, h_t\},$$

then let $[A, i]$ for $A \in \Sigma \setminus \Delta$ and for $1 \leq i \leq t$ be a new alphabet. For $a \in \Delta$, let $[a, i]$ be an alternative writing for a . Let

$$\Sigma' = \{[A, i] \mid A \in \Sigma, 1 \leq i \leq t\}.$$

For a word $x = \xi_1 \dots \xi_m$, let

$$[x, i] := [\xi_1, i] \dots [\xi_m, i].$$

There are two tables in the simulating ETOL system $G' = (\Sigma', H', [\omega, 1], \Delta)$:

- one simulation table containing, for all $1 \leq i \leq t$, a rule $[A, i] \rightarrow [w, i]$ iff $A \rightarrow w \in h_i$, and
- one dispatcher table with rules $[A, i] \rightarrow [A, (i \bmod t) + 1]$ for all $1 \leq i \leq t$ and $A \in \Sigma$.

As usual here and in many similar subsequent situations, we leave the detailed induction proof showing the correctness of the construction to the reader.

Alternatively, we can phrase the last theorem as follows: ETOL systems have a *synchronization degree* of two.

We only state a corresponding result for limited ETOL systems, see [244] in combination with [234].

THEOREM 4. *The synchronization degree of kLETO L systems is at most three and at least two.*

Still, there are lots of open questions and unexplored areas concerning questions of descriptonal complexity, see RP XX.

5.3. REMARKS ON CLOSURE PROPERTIES

One of the mathematically remarkable facts on Lindenmayer systems is their lack of closure properties, which is actually due to the “pure rewriting” which arises—as observed above—“naturally” with these systems.

In order to back our results, we start with two simple observations:

LEMMA 3. *Each singleton language can be generated by a (uniformly (limited)) PDOL system.*

Proof. Take the word in question as axiom and introduce rules $a \rightarrow a$ for each symbol occurring in the word.

LEMMA 4. *$\{a, aa\}$ is no (uniformly (limited)) OL language.*

Proof. If it were a OL language, then either a or aa must be the axiom. If aa is the axiom, then, in order to obtain a , the system must contain a rule $a \rightarrow \varepsilon$. But this would allow also the introduction of the empty word, a contradiction.⁴ If a is the axiom, we would have the rule $a \rightarrow aa$ in our rule set, so that the generated language would include at least $\{a^{2^n} \mid n \geq 0\}$ in the case of OL systems (for (uniformly) limited systems, a similar contradiction is derived).

THEOREM 5. *$\mathcal{L}(OL)$ is an anti-AFL. The statement is also true for systems with determinism, propagation or more than one table. Analogous results hold for (uniformly) limited systems.*

We will only prove non-closure of OL languages under four operations by making use of the preceding two lemmas in order to communicate the flavour of this kind of results:

union $\{a\} \cup \{aa\} = \{a, aa\} \notin \mathcal{L}(OL)$

catenation $\{a\} \cdot \{a, \varepsilon\} = \{a, aa\} \notin \mathcal{L}(OL)$

homomorphism $h : a, b \mapsto a \rightsquigarrow h(\{a, bb\}) = \{a, aa\} \notin \mathcal{L}(OL)$

intersection with regular sets $\{a^{2^n} \mid n \geq 0\} \cap \{a, aa\} = \{a, aa\} \notin \mathcal{L}(OL)$

On the other hand, when Lindenmayer systems with extensions are considered, we observe the same positive closure properties as for other language families like the ones from the Chomsky hierarchy. Since this kind of proofs is based on well-known techniques, we will omit them.

THEOREM 6. *$\mathcal{L}(ETOL)$ forms an AFL.*

THEOREM 7. *$\mathcal{L}(EOL)$ is closed under all AFL operations except from inverse homomorphisms.*

⁴ When we consider $\{\varepsilon, a, aa\}$ and $\{a, aa\}$ as being “the same” language, we can take $\{a, aaa\}$ as example.

5.4. MORE ON GROWTH FUNCTIONS

This subsection exposes some of the results found in the “discussion” of the fifth section in the chapter on Lindenmayer systems in [197, Vol.1].

Given a DIL system $G = (\Sigma, h, \omega)$, the *DIL growth function* $g_G : n \mapsto |w_n|$ for $\omega \Rightarrow^n w_n$ gives the length of the word derived after n steps. Due to the nature of rewriting in Lindenmayer systems, we can immediately state:

LEMMA 5. *No DIL growth function grows faster than exponential.*

What kinds of functions can be realized as DIL growth functions?

1. exponential growth: consider the D0L system with rule $a \rightarrow a^2$;
2. polynomial growth: $a \rightarrow ab, b \rightarrow b$ generate, starting with a, ab^n ;
3. logarithmic growth: see Example 3.

We will now study D0L growth functions in more detail, especially under the aspects whether all types of growth observable for DIL systems can occur here, too.

Obviously, for interactionless Lindenmayer systems, g_G does not depend on the structure given by the sequence of letters of the axiom and in the right-hand sides of the rules, it only matters how many symbols of which type appear there. This information can be nicely stored in a vector (the *Parikh (row) vector* of the axiom) and in a matrix (the *growth matrix* of the rules). Let us explain these notions less formally by means of an example:

Example 6. Consider the D0L system

$$G = (\{a, b, c\}, \{a \rightarrow abc^2, b \rightarrow bc^2, c \rightarrow c\}, a).$$

The growth matrix is:

$$M_G := \begin{pmatrix} 1 & 1 & 2 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{pmatrix}$$

The first row of M_G is the Parikh (row) vector of abc^2 , the second row the Parikh vector of bc^2 , and the third row is the Parikh vector of the right-hand side of the rule $c \rightarrow c$. What is the matrix of Parikh vectors of the words derivable from a, b , and c after two derivation steps? As the reader might convince himself or herself, this is just the matrix

$$M_G^2 = \begin{pmatrix} 1 & 2 & 6 \\ 0 & 1 & 4 \\ 0 & 0 & 1 \end{pmatrix}$$

This means that the Parikh vector of the word derivable from the axiom after two derivation steps can be computed by multiplying the Parikh vector of the axiom with M_G^2 . Let us check for this example:

$$a \Rightarrow abcc \Rightarrow abccbcccc \quad \text{and} \quad (1 \ 0 \ 0) M_G^2 = (1 \ 2 \ 6)$$

We see that we cannot only use vectors and matrices for representing growth patterns, but we also can use matrix algebra to compute the growth function for certain arguments.

Moreover, we can use now classical results of matrix theory, e. g., that every matrix satisfies its own characteristic equation (Cayley-Hamilton). In particular, this means that

$$M_G^n = c_1 M_G^{n-1} + c_2 M_G^{n-2} + \dots + c_n M_G^0,$$

from which we can derive the following recursion for the growth function g_G :

$$g_G(i+n) = c_1 g_G(i+n-1) + c_2 g_G(i+n-2) + \dots + c_n g_G(i)$$

for all $i \geq 0$. This way, it is often possible to express the function g_G explicitly. Let us reconsider our example:

Example 7. Let us compute M_G^3 for system G from Example 6. We get

$$M_G^3 = \begin{pmatrix} 1 & 3 & 12 \\ 0 & 1 & 6 \\ 0 & 0 & 1 \end{pmatrix} = c_1 \begin{pmatrix} 1 & 2 & 6 \\ 0 & 1 & 4 \\ 0 & 0 & 1 \end{pmatrix} + c_2 \begin{pmatrix} 1 & 1 & 2 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{pmatrix} + c_3 \mathbb{I},$$

where \mathbb{I} denotes the identity matrix. A little algebra reveals:

$$c_1 = 3, \quad c_2 = -3, \quad \text{and} \quad c_3 = 1.$$

This means that

$$g_G(i+3) = 3g_G(i+2) - 3g_G(i+1) + g_G(i)$$

with initial values

$$g_G(0) = 1, \quad g_G(1) = 4, \quad \text{and} \quad g_G(2) = 9.$$

As it might be guessed already from this sample sequence, we can further conclude that $g_G(i) = i^2$ in general by verifying:

$$(i+3)^2 = i^2 + 6i + 9 = 3(i+2)^2 - 3(i+1)^2 + i^2.$$

This little piece of algebra has the following consequence:

LEMMA 6. *Assume that g is a D0L growth function with arbitrarily long intervals on which it is constant, i. e., more formally:*

$$\forall k \exists i : g(i) = g(i+1) = \dots = g(i+k).$$

Then, g is ultimately constant.

This proves that there is no D0L growth function growing logarithmically like Gabor's sloth:

COROLLARY 1. *There are DIL growth functions which are not D0L growth functions.*

The mentioned algebraic formulation can be also used to attack some natural decidability questions:

1. Growth equivalence for D0L systems: Given two D0L systems G_1 and G_2 , do their growth functions g_{G_1} and g_{G_2} coincide?

THEOREM 8. *Growth equivalence for D0L systems is decidable.*

Proof. Let n_i be the alphabet cardinality of $G_i = (\Sigma_i, h_i, \omega_i)$, i. e., $n_i = |\Sigma_i|$. Then, as a further consequence of the Hamilton-Cayley Theorem, $g_{G_1} = g_{G_2}$ iff $\forall 0 \leq i \leq n_1 + n_2 (g_{G_1}(i) = g_{G_2}(i))$.

2. Polynomiality problem for D0L systems: Given a D0L system, decide whether its growth function is a polynomial.

THEOREM 9. *The polynomiality problem for D0L systems is decidable.*

Idea: A rule $a \rightarrow w$ with more than one occurrence of a in w implies exponential growth. "More indirect" situations are detectable by a "stage construction" as known from the elimination procedure for erasing rules in context-free grammars.

3. Momentary stagnation: A D0L system G is said to *stagnate momentarily* if there exists a t such that $g_G(t) = g_G(t+1)$. It is open whether the following problem is decidable or not: determine whether a given D0L system stagnates momentarily or not. Equivalently, this problem can be posed purely algebraically: Decide, given an $n \times n$ matrix M with (possibly negative) integer entries, whether or not there exist a t such that a zero appears as entry in the right upper corner of M^t . More precisely, this re-formulation is decidable when $n = 2$, but the problem is open when $n \geq 3$.

5.5. SOME “TYPICAL” DECIDABILITY RESULTS

Already in the preceding subsection, we listed some decidability results typical in the area of Lindenmayer systems. Here, we continue this list. Note that (some of) these problems make also sense for other parallel rewriting mechanisms, but, to our knowledge, these questions weren't attacked for systems other than the (by now) classical Lindenmayer ones. Proofs or hints to the literature can be found in the chapter on L systems in [197, Vol.1].

Given two DIL systems G_1 and G_2 , $G_i = (\Sigma_i, h_i, \omega_i)$, the question to determine whether or not the infinite sequences

$$\omega_1, h_1^1(\omega_1), h_1^2(\omega_1), h_1^3(\omega_1), \dots$$

and

$$\omega_2, h_2^1(\omega_2), h_2^2(\omega_2), h_2^3(\omega_2), \dots$$

coincide or not is the *sequence equivalence problem*.

THEOREM 10. *Both the sequence equivalence and the language equivalence problems are decidable for DOL systems.*

Define, for any Lindenmayer system G with axiom ω , the *sequence cardinality function*

$$d_G : n \mapsto |\{w \mid \omega \Rightarrow^n w\}|.$$

Given two systems G_1 and G_2 , the *sequence cardinality problem* ask two determine if $d_{G_1} = d_{G_2}$.

THEOREM 11. *Sequence cardinality is undecidable when given two DTOL or two OL systems.*

A Lindenmayer system is *derivation slender* iff there exists a constant c such that $d_G(n) \leq c$ for all $n \geq 0$.

THEOREM 12. *Derivation slenderness is decidable for DOL systems.*

Intriguingly, the seemingly related slenderness problem is open for DOL systems. Recall that a language L is called *slender* if there exists a constant c such that, for each $n \geq 0$, there are at most c words of length n in L .

We end this section with listing some (un-)decidability results which are in a sense not so “typical” for Lindenmayer systems.

THEOREM 13. *Language equivalence is undecidable for POL systems. The context-freeness, regularity and OL-ness problems are undecidable for EOL systems.*

Next, we restrict ourselves to unary 0L systems, also called UL systems. Here, things tend to be easier, since—by interpreting words as unary numbers—tools from number theory come at hand.

THEOREM 14. *Language equivalence between TUL and UL systems, as well as between TUL systems and regular grammars, is decidable. Regularity and UL-ness is decidable for TUL languages, as the TUL-ness for regular languages.*

5.6. DETAILS ON HIERARCHY RELATIONS

5.6.0.1. *Parallel versus sequential derivations* We will study how context-free languages relate to E0L, lE0L and A0L languages.

LEMMA 7. *Any context-free grammar can be simulated by an (l)E0L system.*

Proof. The idea is quite simple and we will encounter it in many situations when we like to circumvent the enforced parallel derivation: we only have to add all rules $a \rightarrow a$ to the rule set of the given context-free grammar.

Together with Example 1, we can deduce:

THEOREM 15. $\mathcal{L}(CF) \subsetneq \mathcal{L}(E0L) \cap \mathcal{L}(lE0L)$.

Here, it is the place to pose a first small project, which could also be part of a larger one (of RP *XX*):

Research Paper Proposal No. I: Call a symbol x *to be used in parallel* if $x \rightarrow x$ is not a rule of the considered E0L system or the only one containing x as left-hand side. How “large” is the class $\mathcal{L}(E0L)_k$ of E0L languages which are generatable by systems where *at most* k symbols are to be used in parallel? Does this definition of a *degree of parallelism* entail an infinite hierarchy? Note that by definition it is clear that

$$\mathcal{L}(CF) = \mathcal{L}(E0L)_0 \subseteq \mathcal{L}(E0L)_1 \subseteq \mathcal{L}(E0L)_2 \subseteq \dots \subseteq \mathcal{L}(E0L)$$

□

Without full proof, we further state the following rather surprising result:

THEOREM 16. $\mathcal{L}(CF) = \mathcal{L}(A0L)$.

The proof direction \supseteq is rather tricky and omitted. To understand why A0L systems are not more powerful than context-free grammars,

the reader might wish to see where the constructions of Example 1 fails. The reader should think for a moment why the construction given in Lemma 7 does not always work in this situation. Correct guess: the context-free grammar could contain useless rules so that words not consisting solely of terminals could become “stable.” Referring to the well-known technique for eliminating useless rules (which is again sort of normal form for context-free grammars) makes the previous argument applicable.

Finally, we mention another interesting result in this context without proof:

THEOREM 17. $\mathcal{L}(ETOL) = \mathcal{L}(ATOL)$.

We pose the examination of different kinds of adult languages as research project, see RP II. A recent paper in this area is [157].

5.6.0.2. *Lindenmayer systems and ordered grammars* Although the following result is also proved in the chapter on Lindenmayer systems in [197], we reproduce the idea of the proof, since it shows how different language theoretic results must come together in order to obtain such type of result.

THEOREM 18. $\mathcal{L}(ETOL) \subsetneq \mathcal{L}(O, CF - \varepsilon)$.

In the course of the proof, we have to make use of the following results which we state without proof. Observe that these results can be classified as a normal form result, as a closure property result and as a combinatorial result.

THEOREM 19. $\mathcal{L}(ETOL) = \mathcal{L}(EPTOL)$.

LEMMA 8. $\mathcal{L}(O, CF - \varepsilon)$ is closed under union.

From a combinatorial property of ETOL systems detailed in [198], one can deduce:

COROLLARY 2. $L = \{(ab^m)^n c^n \mid m \geq n \geq 1\} \notin \mathcal{L}(ETOL)$

Observe the dependence of the “inner loop” on the “outer loop” in the example language. This is typical for known non-ETOL languages.

Based on the mentioned three results, we can prove Theorem 18:

Let $L \subseteq \Delta^*$ be an ETOL language. Let L be decomposed as

$$L = \bigcup_{a \in \Delta} aL_a \cup L_F$$

where

$$L_a = \{w \in \Delta^+ \mid aw \in L\}$$

is basically the left derivative of L under a and L_F is some finite language. The two theorems 19 and 6 show that each L_a is in fact an EPTOL language.

We are going to show that each language aL_a is an ordered language, which, together with Lemma 8 proves the desired result.

Let L_a be described by an ETOL system $G_a = (\Sigma, H, \omega, \Delta)$ with $H = \{h_1, \dots, h_t\}$. Let Σ' be an alphabet of primed symbols from Σ . We interpret $'$ also as homomorphism, which means that $' : A \mapsto A'$. Let S, F, A be three new symbols, the start symbol, the failure symbol and a symbol which will finally generate the leftmost a of aL_a . The nonterminal alphabet of the simulating ordered grammar $G'_a = (N, P, S, \Delta, <)$ is given by

$$N := \Sigma' \cup \{S, F, A\} \cup ((\Sigma \cup \{A\}) \times \{k \mid 1 \leq k \leq t\}).$$

The last part of the alphabet contains symbols which are able to keep track of the table which is currently simulated.

We now describe the simulating rules together with their use in the simulation.

$$S \rightarrow (A, k)\omega' \quad \text{for } 1 \leq k \leq t$$

is a set of start rules. A simulation of table k ($1 \leq k \leq t$) is done by sequentialisation, requiring—as usual—a marking and a real application phase. The marking is realized by the following rules:

$$B' \rightarrow (B, k) \quad \left\langle \begin{array}{l} (A, s) \rightarrow F \quad \text{for } 1 \leq s \leq t, s \neq k \\ A \rightarrow F \end{array} \right.$$

Then, the actual application of table h_k is simulated by

$$(B, k) \rightarrow w' \quad \left\langle \begin{array}{l} C' \rightarrow F \quad \text{for } C \in \Sigma \\ A \rightarrow F \end{array} \right.$$

In order to “switch” between the simulation of different tables, an application of a “dispatcher rule” can be intercalated after every full simulation phase of one table application:

$$(A, k) \rightarrow (A, s) \quad < \quad (B, r) \rightarrow F \quad \text{for } 1 \leq k, r, s \leq t, B \in \Sigma.$$

Alternatively to starting a new table application, it could be chosen to terminate. This is accomplished by the following rules:

$$(A, k) \rightarrow A \quad \left\langle \begin{array}{l} (B, k) \rightarrow F \quad \text{for } B \in \Sigma \\ B' \rightarrow F \quad \text{for } B \in \Sigma \setminus \Delta \end{array} \right.$$

$$\begin{aligned}
b' \rightarrow b &< (A, k) \rightarrow F \quad \text{for } b \in \Delta, 1 \leq k \leq t \\
A \rightarrow a &< b' \rightarrow F \quad \text{for } b \in \Delta
\end{aligned}$$

5.6.0.3. *Limited Lindenmayer systems* In the following, we present three results, which originally appeared in [231, 36, 77, 61] (in that order):

THEOREM 20. *Each k LETOL language is also a 1LETOL language*

THEOREM 21. *Each k LETOL language is a programmed language with unconditional transfer and each programmed language with unconditional transfer is a 1LETOL language. A similar statement is true for languages generatable by systems/grammars disallowing erasing rules.*

THEOREM 22. *There are non-recursive k LDL languages.*

The latter theorem is particularly interesting, since programmed grammars with unconditional transfer have a decidable emptiness problem, see [189].

Theorem 20 can be shown in the following way: basically, we have k additional “marking tables” which ensure that after their application we have, if possible, (at least) k different versions of symbols are present. To this end, we have, for each symbol A , marked symbols $A[i, j]$ with $1 \leq i, j \leq k$.

More precisely, marking table M_i has the following rules: $A \rightarrow A[i, i]$ and $A[i-1, j] \rightarrow A[i, j]$ for each symbol A and $j < i$; all other (marked) symbols are sent to a special failure symbol F .

Then, for each original table h , there is a simulating table h' containing rule $A[k, j] \rightarrow w$ if $A \rightarrow w$ is present in h , as well as $A \rightarrow A$.

The problem of a possible “shortcut” by using, e. g., table M_k (instead of M_1) immediately after the use of a simulation table can be circumvented by using an additional special symbol for marking the “state” of the simulation. Then, a special termination table must be added.

This is the reason why—with the given proof—Theorem 20 is not valid in the propagating case.

Observe that the previous proof uses two frequent techniques in this area: *sequentialisation* of rule applications and *explicit state information*. These two techniques will be also used in the proof of Theorem 21:

For simulating 1LETOL systems with programmed grammars with unconditional transfer, we introduce, for each table h , one simulation

loop which enables, for each symbol a , some rule $a \rightarrow w$ of h to be simulated by using $a' \rightarrow w$. Moreover, there is a general “priming loop” where, for each symbol a , there is a rule $a \rightarrow a'$. Assuming a certain normal form for 11ETOL systems (namely, only “nonterminal” symbols are “non-constantly” replaced, see [231]), this basically shows the sequentialisation construction of J. Dassow [36]. Observe that we need not show how to simulate k 1ETOL systems by programmed grammars in general, since this follows by Theorem 20.

For the other direction, we make use of explicit state information stored in a special symbol. Then, for each rule $(p : A \rightarrow w, \gamma)$ of a programmed grammar, we introduce a table with rules $p \rightarrow q$ for each $q \in \gamma$ and $A \rightarrow w$, as well as a couple of failure rules. The “state symbol” is erased only when a terminal string is derived (by means of a special termination table).

In this place, let us prove (one part of) the following generalization of Theorem 20, which supplements a theorem previously shown by D. Wätjen [231]:

THEOREM 23. *Consider two natural numbers k_1 and k_2 . Then,*

- $(k_1 k_2)$ 1EPTOL \subseteq k_1 1EPTOL and
- $(k_1 k_2)$ 1ETOL \subseteq k_1 1ETOL.

Proof. We only give the proof for the first assertion. The second one is shown in [231]. Let some $(k_1 k_2)$ 1EPTOL system be specified by $G = (\Sigma, H, \omega, \Delta)$. For each $A \in \Sigma$ and each $1 \leq i, j \leq k_2$, we have symbols A , A_i and $A[i, j]$ (as well as the failure symbol F) in the alphabet Σ' of the simulating k_1 1EPTOL system G' . In order to simulate an application of some table $h \in H$, G' has to apply a sequence of tables described by the regular expression

$$T_0^* M_1 T_1^* M_2 T_2^* \dots M_{k_2-1} T_{k_2-1}^* M_{k_2} h'^*$$

We describe the corresponding tables in the following.

- T_0 has as only rules *not* leading to the failure symbol $A \rightarrow A_1$ and $A_1 \rightarrow A_1$ for each symbol $A \in \Sigma$.
- M_i (for $i = 1, \dots, k_2$) contains, for each $A \in \Sigma$, $A_i \rightarrow A[i, i]$ and $A[i-1, j] \rightarrow A[i, j]$ for $j < i$ as only non-failure rules.
- T_i (for $i = 1, \dots, k_2 - 1$) has as non-failure rules $A_i \rightarrow A_{i+1}$, $A_{i+1} \rightarrow A_{i+1}$, $A[i, j] \rightarrow A[i, j]$ for each $A \in \Sigma$ and $j \leq i$.
- h' contains $A[k_2, j] \rightarrow w$ for $j \leq k_2$ if $A \rightarrow w \in h$. Moreover, conversion rules $A_{k_2} \rightarrow A$ and $A \rightarrow A$ for $A \in \Sigma$ are supplied. All other rules are leading to the failure symbol.

Observe that a similar idea of “distributing the state information” can be used to show that 1LEPT0L systems can simulate programmed grammars with unconditional transfer without erasing rules: for each symbol A (be it terminal or nonterminal) of the programmed grammar G and for each label p , we introduce symbols $[A, p]$ and (A, p) . An application of a rule $p : A \rightarrow w, \sigma = \phi, w = a_1 \dots a_m$, is simulated by a table application of $h_{p,q}$ (with $q \in \sigma$) containing

$$[A, p] \rightarrow (a_1, q) \dots (a_m, q)$$

as the only non-failure rules besides $[B, p] \rightarrow (B, p)$ for $B \neq A$.

Then, a table $t_{p \rightarrow q}$ containing $[X, p] \rightarrow (X, q)$ and $(X, q) \rightarrow (X, q)$ for each grammar symbol X of G as only non-failure rules is exhaustively applied.

Finally, apply exhaustively t_q with $(X, q) \rightarrow [X, q]$ and $[X, q] \rightarrow [X, q]$ as non-failure rules for each grammar symbol X of G .

The (easy) specification of a start and a termination table is left to the reader.

Let us now prove Theorem 22 (similar as done in [57, 61]): In order to prove this sort of result, it is often important to choose the “right” computability model. We will use a variant of register machines in the following. In this model, each register is capable of storing a non-negative integer. Registers are labelled by positive integers. The input of a program for computing a function from positive integers to positive integers is stored in the first register, and the output is expected to be found in the second register.

Example 8. An example of a register machine program (RMP) is given now:

$$\begin{aligned} L_1 & : a_1 \\ L_2 & : a_2 \\ L_3 & : JZ_1 L_7 \\ L_4 & : s_1 \\ L_5 & : a_2 \\ L_6 & : JNZ_1 L_4 \\ L_7 & : \text{END} \end{aligned}$$

So, an RMP consists of a sequence of labelled commands

- for incrementing and decrementing⁵ register numbered i (by the commands a_i and s_i , respectively),

⁵ In RMP, a modified decrement is used: if a register containing zero is decremented, it will contain also zero afterwards.

- for jumping if register numbered i is zero or not (by JZ_i and JNZ_i , respectively),
- for indicating the END of an RMP.

Example 8 computes $x + 2$ for each input x ; if the start were at L_3 , the identity function would be computed.

If an RMP uses at most r registers and ℓ labels, we call it an r -RMP.

DEFINITION 1. Let $k \geq 1$. A *klTOL machine* is given by $M = (\Sigma, \{h_1, \dots, h_t\}, \{\sigma, x, y, R\})$, where Σ and $\{h_1, \dots, h_t\}$ are the total alphabet and the set of tables, respectively. σ, x, y, R are special symbols in Σ . We say that M computes the function $f : \mathbb{N} \dashrightarrow \mathbb{N}$ iff the corresponding *klTOL system* $G_{M,n} = (\Sigma, \{h_1, \dots, h_t\}, x^{kn}R\sigma, k)$ with axiom $x^{kn}R\sigma$ generates a word of the form $y^{km}\sigma$ if and only if $m = f(n)$. Especially, there is at most one word in $\{y\}^*\{\sigma\} \cap L(G_{M,n})$.

LEMMA 9. For any computable function $f : \mathbb{N} \dashrightarrow \mathbb{N}$ and any $k \geq 1$, there exists a *klTOL machine* computing f .

Proof. $f : \mathbb{N} \dashrightarrow \mathbb{N}$ can be described by an (r, ℓ) -RMP P . We describe a simulating *klTOL machine* $M = (\Sigma, H, \{\sigma, x, y, R\})$ with

$$\Sigma = \{\sigma, F, R, S, A_1, \dots, A_r, y, C_1, \dots, C_r\} \cup L \cup L',$$

where we identify x with A_1 . F is a failure symbol; in the following, we list only productions which do not lead to F . $L = \{L_1, \dots, L_\ell\}$ is the set of labels controlling the simulation of the program P . L' is a set of primed version of the labels in L . Without loss of generality, we can assume that every jump statement in P is actually a sequence of two complementary statements: $JZ_i \text{ lab}_1$ and $JNZ_i \text{ lab}_2$ which carry the same register index i .

The set of tables H consists of

- one initialization table h_I containing $\sigma \rightarrow \sigma, x \rightarrow x$ and $R \rightarrow C_1 \cdots C_r L_1$;
- two termination tables h_T containing $\sigma \rightarrow \sigma, A_i \rightarrow \varepsilon$ for $i \neq 2, A_2 \rightarrow y, L_\ell \rightarrow S, S \rightarrow S, y \rightarrow y, C_i \rightarrow C_i$ for $i \leq r$; and h'_T consisting of $\sigma \rightarrow \sigma, C_i \rightarrow \varepsilon$ for $i \leq r, S \rightarrow \varepsilon$, and $y \rightarrow y$. This way, the result is transferred from register number 2 into a sequence of y 's.
- for any label L_j , there corresponds one or two simulation tables. We explain them below.
- $L_s : a_i : L_s \rightarrow L_{s+1}, C_i \rightarrow A_i^k C_i, C_j \rightarrow C_j$ for $j \neq i, A_j \rightarrow A_j$ for $1 \leq j \leq r, \sigma \rightarrow \sigma$.

- $L_s : s_i: L_s \rightarrow L_{s+1}, A_i \rightarrow \varepsilon, C_j \rightarrow C_j$ for $1 \leq j \leq r, A_j \rightarrow A_j$ for $j \neq i, \sigma \rightarrow \sigma$.
- $L_s : \text{JZ}_i L_t: L_s \rightarrow L_t, C_j \rightarrow C_j$ for $1 \leq j \leq r, A_j \rightarrow A_j$ for $j \neq i, \sigma \rightarrow \sigma$.
- $L_s : \text{JNZ}_i L_t:$ There are two simulating tables here:
 - $t_1: L_s \rightarrow L'_s, A_i \rightarrow \sigma A_i, C_j \rightarrow C_j$ for $1 \leq j \leq r, A_j \rightarrow A_j$ for $j \neq i, \sigma \rightarrow \sigma$.
 - $t_2: L'_s \rightarrow L_t, \sigma \rightarrow \varepsilon, C_j \rightarrow C_j$ and $A_j \rightarrow A_j$ for $1 \leq j \leq r$.

In the tables above, we only included productions not leading to the failure symbol F . Such failure productions must be added for each nonterminal not appearing as the left-hand side of some production of the considered table.

Observe that we do not (and probably cannot) bother about the sequence in which the symbols occur in a string derived via M . Nevertheless, the correctness of the construction is easily seen when observing the special role of σ as a *success witness*⁶: the critical point is namely to check for the presence of a certain symbol which certifies the non-zero status of a register (when simulating the JNZ command); in the case of an erroneous jump simulation, the symbol σ will be erased and will never be introduced again.

With the help of Lemma 9, Theorem 22 can be easily proven: Let M be a $k\text{ITOL}$ machine computing a function f with nonrecursive range. Consider the $k\text{ITOL}$ system $G_{M,n}$ as constructed above. The construction ensures that $G_{M,n}$ is indeed deterministic. The $k\text{IDTOL}$ system G we are going to construct has the same tables as $G_{M,n}$ except the initialization table h_I which is replaced by two tables $h_{I,1}$ consisting of $R \rightarrow A_1^k R$ and $A_1 \rightarrow A_1$, and $h_{I,2}$ consisting of $R \rightarrow C_1 \cdots C_r L_1 \sigma$ and $A_1 \rightarrow A_1$. The axiom of G is R . Obviously, $y^{km} \sigma \in L(G) \iff m \in f(\mathbb{N})$. If $L(G)$ were recursive, the range of f would be recursive, too: a contradiction.

As an addendum to [231] (where it was proven that $k\text{IETOL}$ systems can simulate ETOL systems) which shows (again) that $k\text{IETOL}$ systems are quite powerful, we prove:

THEOREM 24. *For any $k \geq 1, \mathcal{L}(O, CF) \subsetneq \mathcal{L}(k\text{IETOL})$.*

Proof.

We only sketch the crucial simulation: The $k\text{IETOL}$ system has as alphabet—besides the terminal and nonterminal symbols of the ordered

⁶ A technique originally used by Stotskii in [211].

grammar G , primed version of terminal symbols as well as special (new) symbols σ , χ , α and F . The priming operation will be seen as morphism keeping other symbols than terminal symbols of G unchanged.

We start with axiom σS , where S is the start symbol of the ordered grammar and σ is a special symbol indicating “simulation mode.” Each rule $p : A \rightarrow w$ of G is simulated by two tables:

1. One table h_p containing $A \rightarrow \alpha^k w'$, $A \rightarrow A$, as well as $B \rightarrow F$ for each left-hand side B of any rule greater than $A \rightarrow w$ in G . Moreover, h_p contains rules $X \rightarrow X$ for the other nonterminals and rules $b' \rightarrow b'$ for the terminals b . The simulation mode marker σ is turned into the checking mode marker by applying $\sigma \rightarrow \chi$; while all terminal symbols, α and χ (as well as F) is sent to the failure symbol F .
2. Another table h'_p as as only non-failure rules: $\chi \rightarrow \sigma$ and $\alpha \rightarrow \varepsilon$, as well as $X' \rightarrow X'$ for any nonterminal and terminal of G .

h'_p checks that when applying h_p , rule $A \rightarrow \alpha w'$ was applied at most once. This is important since an application of a rule $A \rightarrow w$ might introduce a symbol which prevents a second application of $A \rightarrow w$ in a row in the ordered grammar.

A terminating table t containing rules $b' \rightarrow b'$ and $b \rightarrow b$ for terminals b and $\sigma \rightarrow \varepsilon$ allows to stop the simulation. The introduction of primed versions of terminal symbols was necessary in order to prevent premature applications of the terminating table (followed by “uncontrolled” use of some tables h_p .)

Unfortunately, the above simulation does not carry over to the case of disallowing erasing rules. However, observe that in the case of $k = 1$, the introduction of σ , χ and α is unnecessary. This means that essentially no new nonterminal are introduced (and again erased) in this case, which shows that this simulation also works in the case when disallowing erasing rules when $k = 1$.

5.6.0.4. *Uniformly limited systems* We briefly mention some results and open problems concerning hierarchy results on uniform limitation.

By combining results of K. Salomaa on k -parallel context-free grammars⁷ with Wätjen’s theorem proving the equivalence with uniformly k -limited EOL systems (see [202, 232]), we can infer:

THEOREM 25. *The language families of k -uniformly limited EOL systems form an infinite hierarchy:*

$$\mathcal{L}(CF) = \mathcal{L}(1ulEOL) \subsetneq \mathcal{L}(2ulEOL) \subsetneq \mathcal{L}(3ulEOL) \subsetneq \dots$$

⁷ not to be confused with the notion of n -parallel grammars due to D. Wood

In contrast to limited systems, uniformly limited systems are related to programmed grammars without appearance checking:

THEOREM 26. *For all $k \geq 1$, we have:*

$$\begin{aligned}\mathcal{L}(kulETOL) &\subseteq \mathcal{L}(P, CF) \\ \mathcal{L}(kulEPTOL) &\subseteq \mathcal{L}(P, CF - \varepsilon)\end{aligned}$$

The proof idea of this result, which appeared in [67], is interesting on its own, since it contains (possibly) non-algorithmic elements in the erasing case: namely, the assumption that, given a *kulETOL* system $G = (\Sigma, H, \omega, \Delta)$ with terminal set Δ , we can compute the finite set

$$L' := L(G) \cap \{w \in \Delta^* \mid |w| \leq k\}.$$

Consider now a slight variant of uniform k -limitation (called *exact uniform k -limitation*): $x \Rightarrow_{ex} y$ if y is obtained from x by replacing exactly k symbols in x . If $L_{ex}(H)$ denotes the language generated in this way by a *kulETOL* system H , then it is not hard to see that

$$L(G) = L_{ex}(G) \cup \bigcup_{w \in L'} L_{ex}(G[w])$$

where $G[w]$ is the system with axiom w (instead of ω).

Now, we can observe that each *exact* uniformly limited system can be simulated by a programmed grammar without appearance checking by sequentialisation. Due to the closure of $\mathcal{L}(P, CF)$ under union, the theorem follows.

Moreover, since $\mathcal{L}(P, CF)$ contains only recursive languages, we are inclined to believe that L' can be computed algorithmically, but, unfortunately, we have no proof for this property.

Furthermore, the strictness of the inclusions in the preceding theorem is unknown. A variant of uniformly limited systems where the inclusion of the above theorem is indeed an equality was discussed in [58].

Due to the existence of appropriate combinatorial lemmas, we can state (without proof):

THEOREM 27.

- For each $k \geq 1$, $\mathcal{L}(klEOL) \subsetneq \mathcal{L}(klETOL)$, and
- For each $k \geq 1$, $\mathcal{L}(klEOL) \subsetneq \mathcal{L}(klETOL)$, and

Similar results hold for propagating and deterministic systems, as well.

5.7. A RESULT ON SCATTERED CONTEXT GRAMMARS

We start with proving that scattered context grammars are indeed universal computational devices, because, again, it turns out to be important for the conciseness of the proof to choose the “right” computational model. More precisely, we rely on the following result which can be found in any introduction to formal language theory (if not, have a look on the proof of the undecidability of Post’s correspondence problem):

THEOREM 28. *Every recursively enumerable language can be represented as the homomorphic image of the intersection of two context-free languages.*

THEOREM 29. $\mathcal{L}(RE) = \mathcal{L}(SC)$.

Proof. For the direction \supseteq , we appeal to Church’s thesis (a direct proof is tedious but not complicated).

For the other inclusion, we make use of Theorem 28. So, let G_1 and G_2 be two context-free grammars,

$$G_i = (\Sigma_i, P_i, S_i, \Delta), \quad i = 1, 2, \quad \Sigma_1 \cap \Sigma_2 = \Delta.$$

Let $h : \Delta \rightarrow T$ be a homomorphism with $\Delta \cap T = \emptyset$. Define as total alphabet of the simulating scattered context grammar $G = (\Sigma, P, S, \Delta)$

$$\Sigma := \Sigma_1 \cup \Sigma_2 \cup \{S, \$\} \cup T,$$

where S and $\$$ are two new symbols. Observe that the symbols from Δ are now interpreted as nonterminal symbols. The rules of the scattered context grammar are the following ones:

- $(S) \rightarrow (\$S_1\$S_2\$)$ as start rule;
- $(A) \rightarrow w$ for each $A \rightarrow w \in P_1 \cup P_2$; these rules allow to simulate the given two context-free grammars;
- $(\$, A, \$, \$, A, \$) \rightarrow (h(A), \$, \$, \varepsilon, \$, \$)$ for each $A \in \Delta$; this way, a string $\$w\$\$w\$\$$ with $w \in \Delta^*$ will be transformed into $h(w)\$\4 ;
- $(\$, \$, \$, \$) \rightarrow (\varepsilon, \varepsilon, \varepsilon, \varepsilon)$.

In this connection, let us mention the following inclusion result whose strictness is a long-standing open question:

LEMMA 10. $\mathcal{L}(SC - \varepsilon) \subseteq \mathcal{L}(CS)$.

Furthermore, let us mention that the proof of Theorem 29 shows that every language representable as the morphic image of the intersection of two context-free languages of finite index is also an absolutely parallel language. It would be interesting to see whether the other direction holds true as well. Here, Theorem 2 could be helpful.

6. Selected (open technical) questions

In view of what has been done so far, already these abridged introduction raises some question of certain interest, although each of them is probably not covering a whole thesis as far as foreseeable; probably more advanced questions are contained in Section 9 for the more mathematically minded students.

Research Paper Proposal No. II: Adult languages are meant to model “maturity” in grown-up organisms whose growth pattern is modelled by means of a Lindenmayer system $G = (\Sigma, H, \omega)$. Even when considering yourself, you can easily observe that the adulthood condition formulated by A. Walker is not actually covering “reality,” where small changes can be observed also in adult organisms. By using a suitable bound on the measure of “closeness” between strings representing subsequent states of the development of a body, a reasonable generalization of the original notion of an adult language is obtainable, e. g.,

$$A_{abs,\delta}(G) = \{ w \in \Sigma^* \mid \forall v \in \Sigma^* : w \Rightarrow_G v \rightsquigarrow \mu(w, v) \leq \delta \}$$

for bounding the absolute deviation between subsequent states and

$$A_{rel,\delta}(G) = \{ w \in \Sigma^* \mid \forall v \in \Sigma^* : w \Rightarrow_G v \rightsquigarrow \mu(w, v) \leq \delta \max(|w|, |v|) \}$$

for bounding the relative deviation between subsequent states. Observe that, for each system G , $A(G) = A_{abs,0}(G) = A_{rel,0}(G)$ holds. So, it seems to be worthwhile studying the corresponding language classes $\mathcal{A}_{abs,\delta}$ and $\mathcal{A}_{rel,\delta}$ for various values of δ more thoroughly.

In a variant of this project, one could allow/consider also the corresponding classes obtained by partially parallel grammars. This way, a larger project could emerge.

Note that other modifications of the notion of adult language have been considered in [117]. \square

Given the good biological motivation of fragmentation, it might be interesting to study this mechanism also in connection with language description mechanisms different from L systems.

Research Paper Proposal No. III: Investigate the effect of the fragmentation operator applied to (uniformly) limited or Bharat systems (or other parallel grammars). In particular, it would be interesting to see whether results like

$$\mathcal{L}(\text{EJOL}) = \mathcal{L}(\text{EOL})$$

transfer to other cases. \square

Research Paper Proposal No. IV: Investigate the effect of codings (or homomorphisms, weak codings, k -limited homomorphisms, . . .) on (uniformly) limited systems. \square

Probably, both small projects together might also serve as a larger project on “effects of operators on pure partial parallel languages.”

We already discussed in the section introducing the definition of simple matrix languages that there exist, in a sense, two different notions of this grammar mechanism. Therefore, the following small research project can be stated rather briefly:

Research Paper Proposal No. V: Investigate and compare the two notions of simple matrix grammars with context-free cores. \square

7. Modelling

Since their very beginnings, automata and grammars were defined taking ideas from nature. The most prominent examples here are probably

- the ideas of McCulloch and Pitt [145] which led to the consideration of neural networks,
- the investigations of von Neumann [23] laying the foundations of the theory of cellular automata,⁸
- and the models proposed by Lindenmayer [137] for simple filamentous organisms which initiated the theory of Lindenmayer systems.

In more recent times, we only mention here the emerging theories of DNA computing [197, Vol. 2, Chapter 7] and membrane computing, motivated by biological phenomena, and quantum computing, also see [167, 171], as well as Site 18. Due to the inherent parallelisms found in nature, all these model are in fact models of parallel computation.

⁸ More information regarding cellular automata can be found on the Site 1; in fact, cellular automata and Lindenmayer systems are quite popular computational models, so that many WWW sites can be found which deal with them.

A detailed recent account of various (parallel) models of computation can be found in [24].

Research Project Proposal No. VI: In comparison with the huge amount of literature generated in this way the impact of language theory on biology and other sciences seems to be rather small. One of the reasons might be that language theory tends to present its results in a rather abstract, mathematical fashion, while biologists prefer seeing concrete results and examples. So, work *relating abstract ideas with biological data* is highly welcome and would probably re-intensify the collaboration of language theorists and biologists. For example, observe that the first papers on Lindenmayer systems were actually published in biology journals [137–139], but later on almost all papers appeared in journals dedicated to Theoretical Computer Science, see the list of references in [198]. A notable exception is a recent sequence of papers applying Lindenmayer systems to model the developments of forests, see [133, 132]. Of course, also the papers of P. Prusinkiewicz and his collaborators deserved to be mentioned here, most of them appearing recently in Computer Graphics Conferences and Journals. In view of potential research, interesting examples of articles are: [43, 176, 177, 181]. In this respect, the interview of A. Lindenmayer published in [113] is very interesting, since it discusses the biological (ir)relevance of certain concepts introduced in the theory of Lindenmayer systems.

Related questions—but pointing to a more mathematical project—are discussed in Project *IX*. □

Research Project Proposal No. VII: Another related source of inspiration for defining grammatical mechanism are the social sciences, or, more general, human behaviour or human interaction. Historically, Turing’s paper [219]—where he defines a model of computation (nowadays known as Turing machine) by abstracting the behaviour of a “calculator” (who was in those times a human being with certain mathematical skills)—can be seen as a first attempt in this direction.

Later on, the definition of cooperating distributed grammar systems [197, Vol.2, Chapter 4] or, as a recent survey containing (new) research directions, [112], and of colonies [115, 116, 114], where largely motivated by social theories. This remark includes other more special issues like the formation of teams as modelled in [168].⁹

Again, the impact of all these theories on social sciences seems to be rather limited, with the notable exception of applications of grammar systems in linguistics, see [107, 143]. So, it would be very beneficial—both from the point of view of language theory and from the point

⁹ Much material on grammar systems can be found online at the Site 4.

of view of humanities—to see a back-flow of results or insights from (abstract) formal language theory to concrete problems in humanities, best if concerning areas where ideas motivating the formal language considerations were taken from. □

Possibly more in the sense of applications but on modelling, but nonetheless quite related to the previous two project suggestions, we like to mention here the following research proposal:

Research Project Proposal No. VIII: Both Lindenmayer systems and CD grammar systems were invented to model certain phenomena in *real* life. On the other hand, there is the nowadays growing area of *artificial* life (with conferences explicitly dedicated to this area). It would be interesting to see whether grammatical mechanisms as the mentioned ones could establish a mathematical backbone in this area. Note that some computer graphics applications—as mentioned in RP *VI*—can be viewed as a step in this direction, also see [178]. More can be found at Site 16 □

The behaviour of cooperating distributed grammar systems might be described best as concurrent or distributed (as opposed to [synchronized] massively parallel computation). But since with the advent of the internet concurrency and parallelism tend to be used as synonyms by practitioners—some of them even coined the new notion of *grid computing* for talking about internet-based coarse-grained parallelism or concurrency issues, see [78]—, cooperating distributed grammar systems, colonies etc. can be listed as parallel grammars, as well. Networks of language processors can be seen as a step in this direction, which might also involve explicit parallel derivation steps, see [31].

Research Project Proposal No. IX: Parallel communicating grammar systems (PCGS) were somehow invented to model phenomena encountered in synchronized parallel processing. Unfortunately again, papers showing the real applicability of this sort of grammatical mechanism within the original area are scarce. A recent exception in this direction is [111].

Aho and others used parallel finite automata to model communication protocols by automata, see [3, 2]. It would be therefore interesting to see applications of PCGS in areas like protocol specification and the semantics of parallel programming languages. □

8. Applications

Here, we discuss applications of parallel grammars which can be either found in the literature or which we think are viable future projects.

8.1. WHAT HAS BEEN DONE

In actual fact, Lindenmayer systems can be viewed—besides finite automata and context-free grammars—as the most useful “inventions” of the whole field of formal languages. Notably, they have been applied in computer graphics and developmental biology, as can be seen in [197, Vol.3, Chapter 9] or, more in depth, in [180].

Let us briefly review the essentials of most computer graphics applications of Lindenmayer systems, which are, in fact, not exactly restricted to parallel grammars. Site 2 is an example of the multitude of nice “small” web-pages on this popular topic. More tutorial pages can be found in the collection listed on Site 6. Especially worth mentioning is the on-line tutorial of P. Prusinkiewicz at Site 17.

- We observe the derivation process of a certain grammar G working with an alphabet containing, among other symbols, f , F , $+$ and $-$.
- A generatable string containing f , F , $+$ and $-$ is sequentially interpreted by a so-called “turtle” which is basically a pen equipped with a “direction”; the special symbols are interpreted as commands signifying:

Drawing If a turtle reads an F , it draws a line of unit length by “walking” in the current direction.

Skipping The turtle interprets f as: move the cursor (representing the turtle) forward by one unit without drawing.

Turning right On seeing a $+$, the turtle turns right by δ degrees;

Turning left analogously, a left turn by δ degrees is done when “seeing” input symbol $-$.

Other “reserved words” for the turtle are in use. For the ease of presentation, we restrict ourselves to the above cases.

- Often, the sequence of pictures drawn by a turtle (corresponding to the derivation sequence in question) is of interest, especially when we are interested in “fractal properties” of the pictures which show up if we continue interpreting sentential forms. Then, an appropriate rescaling of the pictures in the sequence becomes an issue.

We illustrate the turtle graphic in cooperation with Lindenmayer systems by means of a simple example.

This sort of graphics is very popular: it is one of the basic features of the programming language LOGO designed to teach children the basics of programming, as described in [1]. In fact, the pictures above were created using the following simple interpreter for turtle graphics in Postscript written by D. Lutterkort; in fact, it contains also commands for starting and ending branches (abbreviated by X and Y due to Postscript restrictions) and for filling areas.

```

/Angle 60 def
/scalefactor 1.0 def
/fontscalefactor 1 def
/InitSize {
  /UpperX 0 def /LowerX 0 def
  /UpperY 0 def /LowerY 0 def
} def
/AddBox {
  Measure
  {
    matrix currentmatrix OldCTM setmatrix pathbbox
    5 -1 roll setmatrix
    dup UpperY gt {/UpperY exch def} { pop } ifelse
    dup UpperX gt {/UpperX exch def} { pop } ifelse
    dup LowerY lt {/LowerY exch def} { pop } ifelse
    dup LowerX lt {/LowerX exch def} { pop } ifelse
  } if
} def
%-- Define Turtle procedures in own dictionary
/TurDict 50 dict def
/Fill { stroke } def
/NP { currentpoint currentpoint gsave newpath moveto } def
  % start new path
/EP { grestore moveto } def    % end of path
TurDict begin
/F { scalefactor 0 rlineto } def
/f { scalefactor 0 rmoveto } def
/+ { Angle rotate } def
/- { Angle neg rotate } def
/X { NP } def    % start branch
/Y { AddBox stroke EP } def % end branch
/S { /fontscalefactor fontscaletfactor 3 div def
  /scalefactor scalefactor 3 div def } def
/R { /fontscalefactor fontscaletfactor 3 mul def
  /scalefactor scalefactor 3 mul def } def

```

```

% '{' == \173 , '}' == \175
(\173) { NP } def % start ‘filled path’
(\175) { AddBox fill EP } def % end ‘filled path’
end
%--- interpreter procedures
/Work 1 string def
% int 1pret --- interpret 1 character as command
/1pret {
  Work 0 3 -1 roll put
  Work dup
  TurDict exch
  known {TurDict exch get exec} {pop} ifelse
} def
% string spret --- Interpret whole turtle-string
/spret { { 1pret } forall } def
% fname fpret --- Interpret turtle-string from file
/fpret {
  (r) file
  /InFile exch def
  { InFile read
    not {exit} if % -- EOF -> end of loop
    1pret
  } loop
  InFile closefile
} def
%--- Interpret a Turtle picture with origin in left lower corner
% obj proc XSize YSize Pret -- Draw object at screen of size XSize x YSize
/Pret {
  /XSize exch def /YSize exch def /Proc exch def
  /OldCTM matrix currentmatrix def
  gsave %--- Calculate size of picture
  InitSize
  /Measure true def
  nulldevice
  OldCTM setmatrix
  newpath 0 0 moveto
  dup Proc AddBox
  grestore
  /Measure false def
  gsave
  XSize UpperX LowerX
  sub dup 0 eq { pop pop 1e13 } { div } ifelse % XSize/(UpperX-LowerX)
  YSize UpperY LowerY

```

```

sub dup 0 eq { pop pop 1e13 } { div } ifelse
2 copy gt {exch} if pop %-- Set step as minimum of both values
/StepLength exch def
UpperX LowerX sub StepLength mul /XSize exch def
UpperY LowerY sub StepLength mul /YSize exch def
LowerX UpperX LowerX sub dup 0 eq
  { pop pop 1 } { div } ifelse XSize mul neg
LowerY UpperY LowerY sub dup 0 eq
  { pop pop 1 } { div } ifelse YSize mul neg
translate StepLength dup scale
newpath 0 0 moveto
Proc stroke
/OldCTM OldCTM currentmatrix def
grestore
} def
% obj scal FPret --- Draw object at (50,300) enlarged by factor scal
/FPret {
  gsave /fpret cvx exch dup dup
  currentlinewidth exch div setlinewidth
  Pret
  grestore
} def
/SPret {
  gsave /sprex cvx exch dup dup
  currentlinewidth exch div setlinewidth
  Pret
  grestore
} def
%-- auxiliary procedures
/kill { showpage clear grestoreall } def
/tur { (turtle.ps) run } def
/org { gsave 50 300 translate } def

```

In fact, the above procedures implement also other “usual” commands of turtle graphic, like “brackets” for indicating branches (for “postscript reasons”, the left and right brackets are X and Y , respectively. In this way, it is easy to create .eps-files for inclusion in Latex documents. For example, the “basic Koch curve” (the first iteration) is

```

%!
%%BoundingBox: 0 0 500 144.4
/Angle -60 def
(F-F++F-F) 500 SPret

```

and the third iteration is:

```

%!
%%BoundingBox: 0 0 500 144.4
/Angle -60 def
(F-F++F-F-F-F++F-F++F-F++F-F-F-F++F-F-F-F++F-F-F-F++F-F++F
-F++F-F-F-F++F-F++F-F++F-F-F-F++F-F++F-F++F-F-F-F++F-F-F-F
++F-F-F-F++F-F++F-F++F-F-F-F++F-F) 500 SPret

```

Putting the command line in the turtle interpreter file (at the end) already gives nice pictures viewable with ghostview.¹⁰

Observe that the turtle commands were generated by the D0L system with the following simple rules: $F \rightarrow F - F + +F - F$, $+ \rightarrow +$ and $- \rightarrow -$.

Research Project Proposal No. X: As pointed out in [197, Page 590], there is a “discrepancy between the studies on the theory of L-systems and the needs of biological modeling.” We continue quoting: “Most theoretical results are pertinent to non-parametric 0L-systems operating on non-branching strings without geometric interpretation. . . . In contrast, . . . L-system models of biological phenomena often involve parameters, endogenous and exogenous interactions, and geometric features of the modeled structures. We hope that the further development of L-system theory will bridge this gap.”

There is also a sort of linguistic motivation behind discussing especially parametric L systems more systematically from a theoretical point of view: they form a natural L system analogue to attribute grammars known as extending context-free grammars. Although being quite “natural,” a systematic theoretic research on attributed L systems, or more general, on attributed parallel grammars—following the spirit of Knuth [120, 121] in the sequential case (also refer to [42] and the literature quoted therein)—is completely lacking. This is the more surprising when recognizing that (preliminary) work on parametric L systems started at about the same time as research on attribute grammars [140].

On the other hand, any sort of theoretical result on this enhanced forms of Lindenmayer systems might have impacts on the use of these systems as modelling tools. Let us mention that in [76], parallel grammars with so-called valuations were considered, which is a nowadays classical form of regulation originating in works of Păun, see [165], which are interpretable as a form of attributed parallel grammars. In fact, the paper [76] can be a good starting point for writing the first

¹⁰ If possible, try to get hold of some classical original research papers. Reading these classics of the masters will largely improve on your paper writing skills. In this “fractal area,” Koch’s paper [122] is such a classics.

one or two papers on this study subject, since it lists a whole number of open, mathematically interesting questions. \square

Sometimes, applications show up which are quite surprising at first glance. At a second glance, this proves that a good theory always serves to some purpose, although that purpose was certainly not intended by the “inventors” of the theory. One example of such a “non-standard” application is described in the following.

Probably since L systems are quite popular for describing fractal-like recursive structures by a turtle interpretation of the generated strings, they were used by P. Moscato and his colleagues in order to describe particular “hard” instances for the well-known Traveling Salesman Problem, see [142, 151, 160]. Moreover, due to the recursive structure of the constructed instances, they could prove optimality of certain tours in a class of graphs containing arbitrarily large graphs. This way, it is possible to measure the performance of heuristical algorithms for the Euclidean Traveling Salesman Problem against optimal solutions, so that it is possible to get good benchmark examples.

Research Project Proposal No. XI: Is it possible to find other NP-hard problems where it is possible to construct “benchmark examples” by using parallel grammars and proving optimality of certain solutions by exploiting the recursive nature of definition of parallel grammars? Here, the main task is to find other interpretations of L systems (possibly different from the turtle graphics approach) or to use parallel grammars not restricted to generating strings. Here, we point especially to the works of R. Freund (concerning array grammars) and of H.-J. Kreowski concerning collage grammars, see [79, 119]. \square

Let us mention that C. G. Nevill-Manning and I. H. Witten [154, 155] used L systems to generate “simple” benchmark examples for data compression algorithms.

A similar “unorthodox” application of L systems was reported in [4], where neural network structures were examined by using L systems. Independently, L systems were used in connection with neural nets in [83] and in a more conventional way for generating dendrite structures in [11], also see Site 15, as well as the report [44].

The question of graphical interpretations of L systems arises again in the next proposal.

Research Project Proposal No. XII: By interpreting L system developments graphically in the turtle sense as described above, using appropriate rescalings, we may come to certain fractal limit objects. Such objects can also be described by mutually recursive function systems,

as explained in [33, 149, 158, 159], see also the review MR 1 411 891 (in the AMS Mathematical Reviews) for historical details.

In this rather indirect way, fractal parameters like the Hausdorff dimension of the described fractal can be computed under certain conditions.

It would be nice to come up with direct methods for determining the Hausdorff dimension (and maybe also an appropriate Hausdorff measure) for a given L system. Also, the higher dimensional case is open.

Interestingly, there is an alternative graphical interpretation of stings: special letters are used here for each “absolute” direction like W, N, E, S for westwards, northwards, eastwards, and southwards, respectively. Here, many things seem to be easier, see [41].

Another (yet unexplored) interpretation is via number systems; they arise rather naturally in the context of Lindenmayer systems as explained in the sixth section of the chapter on L systems in [197, Vol.1]. Fractal properties for an interpretation were computed in [89, 90].

A further link to parallel rewriting in a broader sense is offered by the works of Peitgen, Saupe and Takahashi on fractal properties of cellular automata, see [96, 175, 216].

A further related research project is discussed in RP *XIV*. \square

Research Project Proposal No. XIII: As shown in a series of papers on applying partially parallel array grammars, these are quite successful for pattern recognition purposes, see [68, 70, 81].

It is quite imaginable that partial parallelism can be successfully applied in other classification tasks. For example, detecting similarities in trees in parallel could be used for supporting internet browsing or finding illegal copies of software code, even after making obvious modifications like renaming variables. Partial parallelism or (probably equivalently) regulated rewriting can be used to keep track of non-local information.

Also in this context, techniques from grammatical inference may be useful, see *XXIII*.

Moreover, this research project (coming from the praxis of computation) can be related to project *X* by the possibility of studying parallel attribute grammars, where the attributes can be used to describe certain aspects of the patterns. \square

Without knowing whether this points to another interesting research direction, let us mention that L systems have also been applied to produce “artificial music,” see Site 11 and, as some tutorial, Site 12. Moreover, L systems have been envisaged by J. F. McGowan as a

tool for describing certain replacement operations for nanosystems, see Site 13.

8.2. WHAT NEEDS TO BE DONE

In the next research proposal, we discuss possible applications of parallel grammars to linguistics. Unfortunately, there hasn't been much attention of the linguistic community in developing or using parallel grammars for specifying natural languages, although there are parallel grammars which can model many linguistically interesting languages, as explained in [64] for the case of simple matrix languages. (An exception from this statement is possibly the area of contextual grammars, which has been completely left out of the considerations of this paper up to now. More information can be found in [166, 169] and in the corresponding chapters of the Handbook of Formal Languages.)

Research Project Proposal No. XIV: The issue of *parsing* has been considerably neglected in the study of parallel grammars. This is of course important if these concepts should be applied to model linguistic phenomena, which seems to be possible as well as reasonable in many ways. For example, observe that our brain is obviously working in a sort of parallel fashion, so that we might also assume that human syntax analysis of natural languages is actually performed in a parallel way.

Closely related and important to this issue is the question of *ambiguity* in parallel grammars in a broad sense. Note that an ambiguous grammar offers various ways of "reading" or "interpreting" a given "sentence," which is largely unwanted if certain semantics is connected to each such interpretation.

Another issue in this respect is a systematic study of *semantics*. Connections to attribute grammars were already mentioned in RP X. Let us mention here that programmed grammars of finite index have been employed to model certain features of the database query language DATALOG, see [46], which is interesting in our context due to the mentioned characterization by absolutely parallel grammars, see Theorem 2.

To develop such a "parsing program," it is also important to give good automata characterizations, see RP XXII. \square

One of the intriguing things about fractal geometry and its probably most popular offspring, namely *iterated function systems (IFS)* [13], is the possibility to employ this theory as a data compression methodology, as explained for instance in [203]. In a nutshell, such a data compression algorithm assumes a certain fractal nature in the picture to be encoded and analyses the picture based on this assumption, seeking for an approximating codification in terms of an iterated function

system (which indeed is difficult, see [201]). Then, the hopefully few parameters of the IFS describing the given picture are transmitted instead of the whole bitmap.

Research Project Proposal No. XV: Due to the known close relations between iterated function systems and Lindenmayer systems, see RP *XII*, it is tempting to try a similar approach based on Lindenmayer systems or other types of parallel grammars for obtaining compression algorithms. Since the basic problem consists in obtaining a grammatical description from a certain form of example(s), this sort of *inverse problem* is obviously related to the problem of inferring parallel grammars, see RP *XXIII*.

Interestingly, one could also try to use parallel grammars for compressing text data. Hints in this direction can be found in [135, 154, 155]. \square

Research Project Proposal No. XVI: MPEG-4 is the new standard for films etc. as recently described in the MPEG group. (Currently discussed is MPEG-7, which is giving sort of enhancement of MPEG-4.) An excellent introduction to this subject can be found at the Site 8, especially, we point to the overview paper available at Site 9.

This upcoming standard includes many interesting parallel features. It might be interesting to *model* these features by means of parallel grammars. In this way, it would be possible to give a solid semantical foundation of the *script* concept, and it is quite imaginable that some sort of grammars can be used for specification purposes. \square

Research Project Proposal No. XVII: As explained in [197, Vol.2, Chapter 11], there have been various approaches to define *cryptosystems* by means of (problems arising in) formal language theory. Some of them, admittedly not the most successful ones due to some observations of L. Kari, were based on Lindenmayer systems, i. e., completely parallel rewriting.

To our knowledge, there were no attempts to use restricted forms of parallelism for defining cryptosystems. In this respect, the lack of a thorough study of computational complexity issues (see RP *XVII*) has its obvious bearing, since cryptography is the major realm of practical applications of complexity theory. \square

9. Mathematical challenges

Up to now, we mostly talked about (possible) applications of parallel grammars to various scientific areas. In fact, we think that finding

(more) substantial applications will be appreciated both by formal language theorists and by practitioners.

Nonetheless, it is also possible to treat this area as a pure mathematical subject. In fact, this has been done by formal language theorists most of the time. Intriguingly enough and despite of the efforts of a whole generation of language theorists, there are still a lot of (probably hard) open problems, as well as a plethora of largely unexplored areas.

In this section, we will have a glimpse at some of these white spots on our landscape. As usual when exploring new territory, it is in general unclear what things can be found during the examination.

Research Project Proposal No. XVIII: Although several issues, notably regarding the computational complexity of Lindenmayer systems, have been analyzed (see, e. g., [198, Section VI.4]; more recent papers are: [35]), this area is largely untouched in terms of grammars with restricted parallelism. In view of the large number of different rewriting mechanisms, this offers a huge number of concrete mathematical problems to solve. For each mechanism, the complexity of the nonemptiness problem and the fixed and variable membership problem—to mention only the three most important decidability questions—needs to be settled.

Note that in some cases upperbounds are known, sometimes disguised as language theoretic inclusion relations. As a concrete example, it is known [239] that k -limited ED0L languages are context-sensitive, which means that the fixed membership problem is in the complexity class PSPACE, but the question whether this problem is hard for PSPACE was not studied.

Furthermore, it might be helpful to take the known interrelations with various kinds of regulated grammars into account, since in that area more complexity results are known.

Related questions can be raised on the scale of undecidability degrees, see RP *XIX*. □

In fact, there are also issues in the computational complexity of L systems which are still open after all these years. Let us mention one of these in the following:

Research Paper Proposal No. XIX: As pointed to in [198, p.315], one of the classical open problems is whether or not each EDT0L language can be recognized in deterministic time $O(n^k)$ for some *fixed* value of k . In known constructions, the value of k depends on the number of nonterminals in the individual EDT0L system.

We are aware of recent attempts to employ techniques from the emerging area of *parameterized complexity* (see [47] or Site 14) to tackle this problem.¹¹

More generally speaking, it would be nice to classify certain language theoretic decision problems according to their parameterized complexity. Coming back to the original sample question, it would be interesting to know whether EDTOL membership, parameterized by the number of nonterminals, is fixed-parameter tractable, i. e., solvable in time $O(f(k)p(n))$, where f is some arbitrary function and p is some polynomial. \square

Although the above concrete question is classified as small research project, the more general question of investigating formal language problems according to the set-up offered by parameterized complexity is a largely yet untouched area.

Research Project Proposal No. XX: The more powerful a grammatical mechanism becomes, the “more undecidable” it becomes, i. e., more and more questions which can be asked about the grammars or the languages are undecidable. Since parallel grammars are in general quite powerful mechanisms, this general observation applies to them in particular.

Even if a certain question is undecidable, it can be asked “how undecidable” this question is. Interestingly, this sort of question is barely asked in formal language theory, but it is very important in recursion theory, which has developed the corresponding notions.

Nevertheless, this question can be important to typical questions of formal language theory, as well, e. g., regarding hierarchy relations, see Section 5.1. Namely, if we would know that e. g., the (general) prefix problem of one language class \mathcal{C} is of a smaller decidability degree than the same problem for another class \mathcal{C}' , then we could conclude that \mathcal{C}' cannot be contained in \mathcal{C} . Only few of these questions have been answered yet, see [75]. So, this is indeed a very interesting largely open theoretical research area which is not restricted to parallel grammars; in fact, these methods might also help settle several old open questions in other areas like regulated rewriting. Other papers considering undecidability questions for limited systems are [57, 61, 210, 241]. Also for languages based on patterns, (un)decidability questions have been discussed [106]. \square

Besides the computational complexity there is another important kind of complexity considered in relation with grammars, namely descriptonal complexity. This area discusses how economical certain “re-

¹¹ This idea was communicated to us by K.-J. Lange.

sources” in a grammar may be used without losing descriptiveness. Such considerations can become very important in practice, especially if we think about applications as sketched in RP *XV*.

Research Project Proposal No. XXI: Besides scattered context grammars [146] and Lindenmayer systems [198], only scarce results are known for other mechanisms.

Some of them are contained in Theorem 4 and others in [233]. Even the mentioned theorem does not exactly determine the synchronization degree of limited systems.

So, there is ample room of study, especially concerning the variety of partial parallel grammar mechanisms presented in this paper.

We already mentioned the study of the degree of parallelism (RP *I*) as descriptiveness complexity measure adequate for parallel rewriting mechanisms. \square

Research Paper Proposal No. XXII: As the reader may have noticed, we spread a number of open problems throughout the text of the paper. Since we mainly focussed on interrelations between language families, most of these questions are basically language hierarchy questions. This alone gives a lot of small and precise (but probably difficult) problems to think about. When browsing through the literature, the reader will find more and more of these open questions, which somehow contradicts the general dictum that formal language theory has already solved all its problems. \square

Research Project Proposal No. XXIII: In classical formal language theory, the classical language classes all have (at least) two different characterizations: an automaton based one and a grammar based one. Often, there are other characterizations based on logic, see [212] for (subclasses of) the regular languages and [134] in the case of context-free languages, or on expressions, as the well-known regular expressions for regular languages and the less known context-free expressions [95] for the context-free languages.

Unfortunately, for the case of languages defined by parallel grammars, mostly only this grammar characterization is known, or the automata models are very clumsy, as in the case of Lindenmayer systems [198]. For many applications, in the sequential case especially automaton based and expression-like characterizations have to be proven to be quite valuable. For instance, most parser of (subclasses of) context-free languages can be viewed as pushdown automata. Expressions are usually considered to be a good way for specifying regular languages.

It is therefore an interesting question to find natural alternative characterizations for languages generated by certain types of parallel

grammars. Here, the notion of *accepting grammar* might provide a useful tool [18, 67] \square

Research Project Proposal No. XXIV: A number of formal frameworks have been developed to model the intuitive notion of *learning from examples*; the most popular frameworks of *grammatical inference* are the following:

- *Identification in the limit* [92] assumes that the (possibly infinite) language to be learnt is enumerated to the learner by means of positive and negative examples completely in the sense that “in the end,” all words over the terminal alphabet show up. The learner utters a stream of hypotheses (grammars, automata or whatever kind of language description is decided upon) as an answer to the stream of examples, and this stream should ultimately become constant; in other words, the learner changes its hypothesis no more, and this hypothesis should (of course) describe the enumerated language. Many variants have been considered, the most popular being *identification in the limit from positive samples* [9, 92] where only positive examples are given to the learner, and *identification in polynomial time and data*, see [98].
- In the *query learning* scenario, the learner may ask question about the language to be learnt, and these questions are assumed to be answered by a so-called teacher, see the original work of D. Angluin [10]. Here, restricting the learning time to a polynomial in the size of the expected learning result is important to avoid trivial solutions.
- *Probably approximately correct (PAC) learning* was introduced by L. G. Valiant [220] in order to incorporate the possibility of making errors and learning only in a certain approximate sense. A different popular model—related by motivation—is *error correcting grammatical inference (ECGI)*, see [141, 225] and the literature quoted therein.

There are a couple of papers dealing with the inference of Lindenmayer systems, see e. g. [196, 147, 110, 248]. There have been also more empirical approaches to this inference problem. For example, [126] demonstrates the possibility of discovering the rewrite rule for Lindenmayer systems and the state transition rules for cellular automata by means of genetic programming. This approach is also taken in [105, 102–104, 124, 123, 125, 161], also see Site 3.

Keeping in mind that Lindenmayer systems were originally intended to model real organism development, which means that it is of basic

interest to know or infer possible formalizations (in terms of Lindenmayer systems) for “explaining” observed biological data, the literature (not completely listed above) is surprisingly scarce.

Even “worse” seems to be the situation within other types of parallel grammars. Actually, trying to give a complete list of references would result in the following papers:

- There is one paper dealing with the inference of parallel communicating grammar systems (in a variant of Gold’s model) with a peculiar mode of communication between different grammar components [66].
- As explained above, simple matrix grammars can be viewed as a simple form of parallel grammars. Simple matrix grammars with regular and even linear components have been proven to be learnable with queries, see [63, 215].

Admittedly, there is one “positive” example, namely (variants of) pattern languages. Here, the situation is really different: there is a huge amount of literature covering different aspects of learnability of these languages, starting with the ground-breaking paper [8] of D. Angluin.

Hence, there is plenty of room for proving new, interesting and useful results concerning the inference of parallel grammars. \square

The interested reader can find more information on grammatical inference on the corresponding “official” web-site 5.

10. Miscellaneous

As already mentioned in the introductory section, we are well aware of the fact that this overview is biased, possibly neglecting several interesting research directions. We like to mention some of them at least in the following.

Research Project Proposal No. XXV: From the very beginning, L systems were not only considered to work on 1-dimensional but also on multi-dimensional structures. From the viewpoint of biology, this is more than justifiable. None-withstanding, in comparison with the huge amount of research done for the 1-dimensional case, the corresponding results are only sparse and somewhat scattered. A *systematic* study of this area, including the consideration of different forms of restricted parallelism and possible comparisons with automata-based approaches like cellular automata, array grammars (see, e. g., [69, 80, 150, 230, 247]) or finite-state recognizability [88, 87] and others [14, 15, 48, 52, 100],

where the corresponding theory seems to be better developed, would be desirable.

As starting point, the interested reader is referred to [198, Section VI.5] and (without claiming any completeness of the reference list) [79, 97, 152, 153, 173]. \square

Another topic which has been neglected nearly completely is cellular automata, although since their introduction by J. von Neumann, many papers appeared which related them to the theory of formal languages in many respects. Readers interested in this topic are invited to read, for “older” accounts, [21, 27, 50, 174, 207–209], and for more recent developments [22, 118, 217, 218]. Here, we see the need of bringing together this “automata-oriented” approach with the more “grammar-oriented” theory around Lindenmayer systems. This could also help with Topic *XXIII*.

Then, we mostly concentrated on *string* rewriting. So, we left out, e.g., the whole area of graph rewriting. Also there, parallel derivation strategies has been introduced, see [51] as a recent example.

Finally, the whole area of combinatorics on words in relation with Lindenmayer systems remained unmentioned up to this point. Recent papers in the area include [26, 84].

Acknowledgements

This paper was inspired by many discussions we had with various colleagues all over the years. This means that it is nearly impossible to acknowledge each such contribution individually. We therefore refrain from giving any individual credits but rather like to thank all members of the formal language community (and beyond) for always providing a stimulating and interesting environment.

List of web-sites

Throughout the paper, we provided several links to web pages we found interesting and informative. Due to the nature of the internet, this sort of information tends to be outdated soon, since often web-sites move or simply disappear. Nevertheless, the web nowadays offers lots of good information, and students are encouraged to use these resources appropriately. We now list all web-sites we are referring to.

Moreover, due to the connections between Lindenmayer systems and computer graphics, many nice web-pages can be found “for beginners.” One example is Site 19.

It would be nice if you, dear reader, could communicate me any changes, updates and new interesting sites on the topic.

1. <http://liinwww.ira.uka.de/ca/>
2. http://www.biologie.uni-hamburg.de/b-online/e28_3/lsys.html
3. <http://www.grammatical-evolution.org/pubs.html>
4. <http://www.sztaki.hu/mms/>
5. www.cs.iastate.edu/~honavar/gi/gi.html
6. <http://www2.informatik.uni-erlangen.de/~jacob/LSystems/LSystems-Tutorials.html>
7. <http://pixie.oum.ox.ac.uk/L-Breeder/>
8. <http://mpeg.telecomitalialab.com/standards/mpeg-4/mpeg-4.htm>
9. http://mpeg.telecomitalialab.com/documents/from_mpeg-1_to_mpeg-21.htm
10. <ftp://ftp.ing.unlp.edu.ar/pub/papers/memetic/>
11. <http://www.abc.net.au/science/news/stories/s210368.htm>
12. <http://www.csu.edu.au/ci/vol03/mccorm/mccorm.html>
This site is referring to the internet-based journal "Complexity International," vol. 3, 1996.
13. <http://www.jmcgowan.com/nlsystem.PDF>
14. web.cs.mun.ca/~harold/W_hier/W_hier.html
15. <http://www.krasnow.gmu.edu/L-Neuron/> and
<http://www.nimh.nih.gov/neuroinformatics/ascolig.cfm>
16. <http://pages.cpsc.ucalgary.ca/~pwp/> and
<http://www.cpsc.ucalgary.ca/projects/bmv/papers/index.html>
17. <http://www.cpsc.ucalgary.ca/projects/bmv/vmm/title.html>
18. <http://dna.bio.disco.unimib.it/psystems/>
19. <http://vortex.bd.psu.edu/~jpp/reu2000.html>
20. <http://www.iti.cs.tu-bs.de/TI-INFO/waetjen>
21. <http://www.cs.uu.nl/research/techreps/RUU-CS-84-11.html>

References

1. H. Abelson and A. A. diSessa. *Turtle Geometry*. Cambridge: M.I.T. Press, 1980.
2. A. V. Aho, J. D. Ullman, A. D. Wyner, and M. Yannakakis. Bounds on the size and transmission rate of communication protocols. *Computers and Mathematics with Applications*, 8(3):205–214, 1982.
3. A. V. Aho, J. D. Ullman, and M. Yannakakis. Modeling communication protocols by automata. In *Proc. 20th Symp. Foundations Comput. Sci.*, pages 267–273, October 1979.
4. I. Aho, H. Kemppainen, K. Koskimies, E. Mäkinen, and T. Niemi. Searching neural network structures with L systems and genetic algorithms. *International Journal of Computer Mathematics*, 73(1):55–75, 1999.
5. V. Aladyev. τ_n -grammars and their generated languages (in Russian). *Transactions of the Academy of Sciences of Estonia; Eest NSV Teaduste Akadeemia toimetised / Biologiline seeria*, 23(1):67–87, 1974.
6. V. Aladyev. On the equivalence of τ_m -grammars and $Sb(n)$ -grammars (in Russian). *Commentationes Mathematicae Universitatis Carolinae*, 15:717–726, 1974.
7. V. Aladyev. Operations on languages generated by τ_n -grammars (in Russian). *Commentationes Mathematicae Universitatis Carolinae*, 15:211–220, 1974.
8. D. Angluin. Finding patterns common to a set of strings. *Journal of Computer and System Sciences*, 21:46–62, 1980.
9. D. Angluin. Inductive inference of formal languages from positive data. *Information and Control (now Information and Computation)*, 45:117–135, 1980.
10. D. Angluin. Learning regular sets from queries and counterexamples. *Information and Computation (formerly Information and Control)*, 75:87–106, 1987.
11. G. Ascoli and J. L. Krichmar. L-neuron: a modeling tool for the efficient generation and parsimonious description of dendritic morphology. *Neurocomputing*, 32–33:1003–1011, 2000.
12. T. Bălănescu, M. Gheorghe, and Gh. Păun. Three variants of apical growth; filamentous systems. *International Journal of Computer Mathematics*, 22:227–238, 1987.
13. M. F. Barnsley. *Fractals Everywhere*. Boston: Academic Press, 1988.
14. M. Blum and C. Hewitt. Automata on a two-dimensional tape. In *IEEE Symposium on Switching and Automata Theory*, pages 155–160, 1967.
15. M. Blum and W. J. Sakoda. On the capability of finite automata in 2 and 3 dimensional space. In *Proc. 18th Ann. Symp. on Foundations in Computer Science*, pages 147–161, 1977.
16. H. Bordihn. Pure languages and the degree of nondeterminism. *J. Inf. Process. Cybern. EIK (formerly Elektron. Inf.verarb. Kybern.)*, 28(5):231–240, 1992.
17. H. Bordihn and J. Dassow. A note on the degree of nondeterminism. Technical Report Preprint FIN-IRB-7/93, Technische Universität “Otto von Guericke” Magdeburg, July 1993.
18. H. Bordihn, H. Fernau, and M. Holzer. On accepting pure Lindenmayer systems. *Fundamenta Informaticae*, 38(4):365–375, 1999.
19. H. Bordihn and M. Holzer. On the number of active symbols in L and CD grammar systems. *Journal of Automata, Languages and Combinatorics*, 6(4):411–426, 2001.

20. F. J. Brandenburg. On the height of syntactical graphs. In Peter Deussen, editor, *Proceedings of the 5th GI-Conference on Theoretical Computer Science*, volume 104 of *LNCS*, pages 13–21, Karlsruhe, FRG, March 1981. Springer.
21. W. Bucher and K. Čulik II. On real time and linear time cellular automata. *RAIRO Informatique théorique*, 18:307–325, 1984.
22. Th. Buchholz, A. Klein, and M. Kutrib. One guess one-way cellular arrays. In Luboš Brim, Jozef Gruska, and Jiří Zlatuška, editors, *Mathematical Foundations of Computer Science MFCS 1998*, number 1450 in *LNCS*, pages 807–815. Springer, 1998.
23. A. Burks. *The Theory of Self-Reproducing Automata*, chapter J. v. Neumann: The University of Illinois Press, 1966.
24. C. S. Calude, Gh. Păun, and M. Tataram. A glimpse into natural computing. Report CDMTCS-117, Centre for Discrete Mathematics and Theoretical Computer Science, University of Auckland, Auckland, New Zealand, December 1999.
25. J. W. Carlyle, S. A. Greibach, and A. Paz. A two-dimensional generating system modeling growth by binary cell division (preliminary report). In *XV. Annual Conference on Switching and Automata Theory*, pages 1–12, 1974.
26. J. Cassaigne and J. Karhumäki. Toeplitz words, generalized periodicity and periodically iterated morphisms. *Eur. J. Comb.*, 18:497–510, 1997.
27. Ch. Choffrut and K. Čulik II. On real-time cellular automata and trellis automata. *Acta Informatica*, 21:393–407, 1984.
28. C. R. Cook and P. S.-P. Wang. A Chomsky hierarchy of isotonic array grammars and languages. *Computer Graphics and Image Processing*, 8:144–152, 1978.
29. A. Corradini and F. Drewes. (Cyclic) Term graph rewriting is adequate for rational parallel term rewriting. Technical Report TR-97-14, Dipartimento di Informatica, July 16 1997. Sun, 17 Aug 1997 16:29:22 GMT.
30. E. Csuhaj-Varjú, J. Dassow, J. Kelemen, and Gh. Păun. *Grammar Systems: A Grammatical Approach to Distribution and Cooperation*. London: Gordon and Breach, 1994.
31. E. Csuhaj-Varjú and A. Salomaa. Networks of language processors: Parallel communicating systems. *EATCS Bulletin*, 66:122–138, 1998.
32. E. Csuhaj-Varjú and Gy. Vaszil. Parallel communicating grammar systems with incomplete information communication. In W. Kuich, G. Rozenberg, and A. Salomaa, editors, *Developments in Language Theory. Preproceedings of the 5th International Conference, DLT 2001, Wien, Austria, July 16-21*, volume 2295 of *LNCS*, pages 381–392. Springer, 2002.
33. K. Čulik, II and S. Dube. L-systems and mutually recursive function systems. *Acta Informatica*, 30:279–302, 1993.
34. K. Čulik, II and A. Lindenmayer. Parallel graph generating and graph recurrence systems for multicellular development. *International Journal of General Systems*, 3:53–66, 1976.
35. C. Damm, M. Holzer, K.-J. Lange, and P. Rossmanith. Deterministic OL languages are of very low complexity: DOL is in AC^0 . In G. Rozenberg and A. Salomaa, editors, *Developments in Language Theory (Turku, 1993)*, pages 305–313. Singapore: World Scientific, 1994.
36. J. Dassow. A remark on limited OL systems. *J. Inf. Process. Cybern. EIK (formerly Elektron. Inf.verarb. Kybern.)*, 24(6):287–291, 1988.
37. J. Dassow and U. Fest. On regulated L systems. *Rostock, Mathematisches Kolloquium*, 25:99–118, 1984.

38. J. Dassow, Gh. Păun, and A. Salomaa. Grammars based on patterns. *International Journal of Foundations of Computer Science*, 4:1–14, 1993.
39. J. Dassow and D. Wätjen. On the relations between the degree of synchronization and the degree of nondeterminism in k -limited and uniformly k -limited TOL systems. *International Journal of Computer Mathematics*, 35:69–82, 1990.
40. M. de Does and A. Lindenmayer. *Algorithms for the generation and drawing of maps representing cell clones*, volume 153 of *LNCS*, pages 39–57. Berlin: Springer, 1983.
41. F. M. Dekking. Recurrent sets. *Advances in Mathematics*, 44:78–104, 1982.
42. P. Deransart and M. Jourdan, editors. *Attribute Grammars and their Applications*, volume 461 of *Lecture Notes in Computer Science*. Springer, September 1990.
43. O. Deussen, P. Hanrahan, B. Lintermann, R. Mech, M. Pharr, and P. Prusinkiewicz. Realistic modeling and rendering of plant ecosystems. In *SIGGRAPH '98, Proceedings of the 25th Annual Conference on Computer Graphics*, pages 275–286. ACM, 1998.
44. R. DeVaul and B. McCormick. Neuron developmental modeling and structural representation: The statistical model; an introduction to the n++ language, an open stochastic L-system. Technical report, Scientific Visualization Laboratory, Department of Computer Science, Texas A&M University, College Station, TX 778433112, July 1997.
45. V. Diekert and G. Rozenberg, editors. *Book of Traces*. World Scientific, Singapore, 1995.
46. G. Dong. Grammar tools and characterizations. In *Proceedings of the Eleventh ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'92)*, pages 81–90. ACM, June 1992.
47. R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer, 1999.
48. F. Drewes. Language-theoretic and algorithmic properties of d -dimensional collages and patterns in a grid. *Journal of Computer and System Sciences*, 53:33–60, 1996.
49. F. Drewes et al. Generating self-affine fractals by collage grammars. *Theoretical Computer Science*, 145:159–187, 1995.
50. C. R. Dyer. One-way bounded cellular automata. *Information and Computation*, 44:261–281, 1980.
51. R. Echahed and J.-C. Janodet. Parallel admissible graph rewriting. In J. L. Fiadeiro, editor, *WADT'98*, volume 1589 of *LNCS*, pages 122–138. Springer, 1999.
52. E. M. Ehlers and S. H. von Solms. Using random context structure grammars to represent chemical structures. *Information Processing Letters*, 30(3):159–166, February 1989.
53. D. Ermel. k -limitierte IL-systeme. Studienarbeit, Technische Universität Braunschweig, 2001. See [Studienarbeiten/Erme1.pdf](#) at Site 20.
54. D. Ermel and D. Wätjen. Limited IL systems. *Journal of Automata, Languages and Combinatorics*, 2002. To appear.
55. H. Fernau. Ersetzungssysteme mit eingeschränkter Parallelität. In *Workshop über Parallelverarbeitung (Informatik-Bericht 91/1)*, pages 41–58. TU Clausthal: Informatik Colloquium Lessach (Österreich) 1990, 1991.
56. H. Fernau. On function-limited Lindenmayer systems. *J. Inf. Process. Cybern. EIK (formerly Elektron. Inf.verarb. Kybern.)*, 27(1):21–53, 1991.

57. H. Fernau. Membership for 1-limited ETOL languages is not decidable. *J. Inf. Process. Cybern. EIK (formerly Elektron. Inf.verarb. Kybern.)*, 30(4):191–211, 1994.
58. H. Fernau. A note on uniformly limited ETOL systems with unique interpretation. *Information Processing Letters*, 54:199–204, 1995.
59. H. Fernau. A predicate for separating language classes. *EATCS Bulletin*, 56:96–97, June 1995.
60. H. Fernau. Closure properties of ordered languages. *EATCS Bulletin*, 58:159–162, February 1996.
61. H. Fernau. Membership for k -limited ETOL languages is not decidable. *Journal of Automata, Languages and Combinatorics*, 1:243–245, 1996.
62. H. Fernau. On grammar and language families. *Fundamenta Informaticae*, 25(1):17–34, 1996.
63. H. Fernau. Efficient learning of some linear matrix languages. In T. Asano et al., editors, *COCOON'99*, volume 1627 of *LNCS*, pages 221–230. Springer, 1999.
64. H. Fernau. Even linear simple matrix languages: formal language aspects. In C.S. Calude, M.J. Dinneen, and S. Sburlan, editors, *Discrete Mathematics and Theoretical Computer Science , DMTCS'01*, pages 81–96, 2001.
65. H. Fernau. Nonterminal complexity of programmed grammars. In M. Margenstern and Y. Rogozhin, editors, *Machines, Computations, and Universality; 3rd MCU*, volume 2055 of *LNCS*, pages 202–213, 2001.
66. H. Fernau. Parallel communicating grammar systems with terminal transmission. *Acta Informatica*, 37:511–540, 2001.
67. H. Fernau and H. Bordihn. Remarks on accepting parallel systems. *International Journal of Computer Mathematics*, 56:51–67, 1995.
68. H. Fernau and R. Freund. Bounded parallelism in array grammars used for character recognition. In P. Perner, P. Wang, and A. Rosenfeld, editors, *Advances in Structural and Syntactical Pattern Recognition (Proceedings of the SSPR'96)*, volume 1121 of *LNCS*, pages 40–49. Berlin: Springer, 1996.
69. H. Fernau and R. Freund. Accepting array grammars with control mechanisms. In Gh. Păun and A. Salomaa, editors, *New Trends in Formal Languages*, volume 1218 of *LNCS*, pages 95–118. Berlin: Springer, 1997.
70. H. Fernau, R. Freund, and M. Holzer. Character recognition with k -head finite array automata. In A. Amin et al., editors, *Proceedings of SSPR'98*, volume 1451 of *LNCS*, pages 282–291, 1998.
71. H. Fernau and M. Holzer. Regulated finite index language families collapse. Technical Report WSI-96-16, Universität Tübingen (Germany), Wilhelm-Schickard-Institut für Informatik, 1996.
72. H. Fernau and M. Holzer. Conditional context-free languages of finite index. In Gh. Păun and A. Salomaa, editors, *New Trends in Formal Languages*, volume 1218 of *LNCS*, pages 10–26. Springer, 1997.
73. H. Fernau and A. Meduna. A simultaneous reduction of several measures of descriptonal complexity in scattered context grammars. Submitted for publication.
74. H. Fernau and A. Meduna. On the degree of scattered context-sensitivity. *Theoretical Computer Science*, To appear, 2002.
75. H. Fernau and F. Stephan. Characterizations of recursively enumerable languages by programmed grammars with unconditional transfer. *Journal of Automata, Languages and Combinatorics*, 4(2):117–142, 1999.

76. H. Fernau and R. Stiebe. Valences in Lindenmayer systems. *Fundamenta Informaticae*, 45:329–358, 2001.
77. H. Fernau and D. Wätjen. Remarks on regulated limited ETOL systems and regulated context-free grammars. *Theoretical Computer Science*, 194(1–2):35–55, 1998.
78. I. Foster and C. Kesselman, editors. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, San Francisco, CA, 1999.
79. R. Freund. Aspects of n -dimensional Lindenmayer systems. In G. Rozenberg and A. Salomaa, editors, *Developments in Language Theory; At The Crossroads of Mathematics, Computer Science and Biology (Turku, Finland, 12–15 July 1993)*, pages 250–261. Singapore: World Scientific, 1994.
80. R. Freund. Control mechanisms on $\#$ -context-free array grammars. In *Mathematical Aspects of Natural and Formal Languages*, pages 97–137. Singapore: World Scientific, 1994.
81. R. Freund, M. Neubauer, M. Summerer, St. Gruber, J. Schaffer, and R. Swoboda. A hybrid system for the recognition of hand-written characters. In F. J. Ferri, J. M. Inesta, A. Amin, and P. Pudil, editors, *Advances in Pattern Recognition, Joint IAPR International Workshops SSPR 2000 and SPR 2000*, volume 1876 of *LNCS*. Springer, 2000.
82. R. Freund and Gh. Păun. On the number of non-terminal symbols in graph-controlled, programmed and matrix grammars. In M. Margenstern and Y. Rogozhin, editors, *Machines, Computations, and Universality; 3rd MCU*, volume 2055 of *LNCS*, pages 214–225, 2001.
83. R. Freund and F. Tafill. Modelling Kohonen networks by attributed parallel array systems. In *Conference on Neural Networks and Artificial Intelligence, International Symposium on Substance Identification Technologies (Innsbruck, 1993)*, October 1993.
84. A. E. Frid. On the frequency of factors in a D0L word. *Journal of Automata, Languages and Combinatorics*, 3(1):29–41, 1998.
85. S. Gärtner. A remark on the regulation of k1ETOL systems. *Information Processing Letters*, 48:83–85, 1993.
86. S. Gärtner. On partition limited 0L systems. In *Developments in Language Theory II; at the crossroads of mathematics, computer science and biology*, pages 230–236, 1996.
87. D. Giammarresi et al. Monadic second-order logic over rectangular pictures and recognizability by tiling systems. *Information and Computation (formerly Information and Control)*, 125:32–45, 1996.
88. D. Giammarresi and A. Restivo. Two-dimensional finite state recognizability. *Fundamenta Informaticae*, 25:399–422, 1996.
89. W. J. Gilbert. Fractal geometry derived from complex bases. *The Mathematical Intelligencer*, 4:78–86, 1982.
90. W. J. Gilbert. The fractal dimension of sets derived from complex bases. *Canadian Math. Bull.*, 29:495–500, 1986.
91. J. S. Golan, A. Mateescu, and D. Vaida. Semirings and parallel composition of processes. *Journal of Automata, Languages and Combinatorics*, 1:199–217, 1996.
92. E. M. Gold. Language identification in the limit. *Information and Control (now Information and Computation)*, 10:447–474, 1967.
93. J. Gonczarowski and M. K. Warmuth. Scattered versus context-sensitive rewriting. *Acta Informatica*, 27:81–95, 1989.

94. S. Greibach and J. Hopcroft. Scattered context grammars. *Journal of Computer and System Sciences*, 3:233–247, 1969.
95. J. Gruska. A characterization of context-free languages. *Journal of Computer and System Sciences*, 5:353–364, 1971.
96. F. v. Haeseler, H.-O. Peitgen, and G. Skordev. Linear cellular automata, substitutions, hierarchical iterated function systems and attractors. In J. L. Encarnação et al., editors, *Fractal Geometry and Computer Graphics*, Beiträge zur Graphischen Datenverarbeitung des ZGDV (Darmstadt), pages 3–23. Springer, 1992.
97. P. Hartmann. Implementation of parallel replacement systems for cellular hypergraphs. *Journal of Automata, Languages and Combinatorics*, 1(2):129–146, 1996.
98. C. de la Higuera. Characteristic sets for polynomial grammatical inference. *Machine Learning*, 27(2):125–138, 1997.
99. O. Ibarra. Simple matrix languages. *Information and Control (now Information and Computation)*, 17:359–394, 1970.
100. K. Inoue and I. Takanami. A survey of two-dimensional automata theory. In J. Dassow and J. Kelemen, editors, *Machines, Languages, and Complexity IMYCS'88*, volume 381 of *LNCS*, pages 72–91, 1988.
101. M. Ito, L. Kari, and G. Thierrin. Insertion and deletion closure of languages. *Theoretical Computer Science*, 183:3–19, 1997.
102. C. Jacob. Genetic L-system programming. In *PPSN III - Parallel Problem Solving from Nature, International Conference on Evolutionary Computation*, volume 866 of *LNCS*, pages 334–343. Springer, 1994.
103. C. Jacob. Modeling growth with L-systems & Mathematica. *Mathematica in Education and Research*, 4:12–19, 1994.
104. C. Jacob. Genetic L-system programming: Breeding and evolving artificial flowers with Mathematica. In *IMS'95, Proc. First International Mathematica Symposium*, pages 215–222. Computational Mechanics Publications, Southampton, UK, 1995.
105. C. Jacob and J. Rehder. Evolution of neural net architectures by a hierarchical grammar-based genetic system. In R. Albrecht and N. Steele, editors, *ICANN'93, Proc. International Conference on Artificial Neural Networks and Genetic Algorithms*, pages 72–79, 1993.
106. T. Jiang, A. Salomaa, K. Salomaa, and S. Yu. Decision problems for patterns. *Journal of Computer and System Sciences*, 50:53–63, 1995.
107. M. D. Jiménez-López. *Grammar Systems: A Formal Language Theoretic Framework for Linguistics and Cultural Evolution*. PhD thesis, University Rovira i Virgili, Tarragona, Spain, 2000.
108. A. K. Joshi. How much context-sensitivity is necessary for characterizing structural descriptions – tree adjoining grammars. In D. Dowty, L. Karttunen, and A. Zwicky, editors, *Natural Language Processing – Theoretical, Computational and Psychological Perspectives*, pages 206–250. Cambridge University Press, 1985.
109. A. K. Joshi, L. S. Levy, and M. Takahashi. Tree adjunct grammars. *Journal of Computer and System Sciences*, 10:133–163, 1975.
110. H. Jürgensen and A. Lindenmayer. Inference algorithms for developmental systems with cell lineages. *Bulletin of Mathematical Biology*, 49(1):93–123, 1987.
111. L. Kari, H. Lutfiyya, Gh. Păun, and C. Martin-Vide. Bringing PC grammar systems closer to Hoare's CSP. In Gh. Păun and A. Salomaa, editors, *Gram-*

- mathematical Models of Multi-Agent Systems*, pages 64–86. Gordon and Breach, 1999.
112. J. Kelemen and A. Kelemenová. Grammar systems: where are we? And where do we go from here? *Journal of Automata, Languages and Combinatorics*, 5(1):5–12, 2000. Special issue: Selected papers of the MFCS'98 Satellite Workshop on Grammar Systems, Brno, 1998. Guest editors: J. Kelemen and A. Kelemenová.
 113. J. Kelemen, A. Kelemenová, and A. Lindenmayer. Answers by a. lindenmayer to questions of j. kelemen and a. kelemenová. *EATCS Bulletin*, 23:185–198, 1984.
 114. J. Kelemen, A. Kelemenová, C. Martín Vide, and V. Mitrana. Colonies with limited activation of components. *Theoretical Computer Science*, 244:289–298, 2000.
 115. A. Kelemenová and E. Csuhaj-Varjú. Languages of colonies. *Theoretical Computer Science*, 134(1):119–130, 1994. Selected papers of the Second International Colloquium on Words, Languages and Combinatorics, Kyoto, 1992.
 116. A. Kelemenová and J. Kelemen. From colonies to eco(grammar)systems: an overview. In J. Karhumäki, H. A. Maurer, and G. Rozenberg, editors, *Results and Trends in Theoretical Computer Science*, volume 812 of *Lecture Notes in Theoretical Computer Science*, pages 213–231. Springer, Berlin, 1994.
 117. A. Kelemenová and P. Vajgel. Productions in stable 0L systems. In G. Rozenberg and A. Salomaa, editors, *Developments in Language Theory (Turku, 1993)*, pages 102–110. Singapore: World Scientific, 1994.
 118. A. Klein and M. Kutrib. A time hierarchy for bounded one-way cellular automata. In Jiří Sgall, Aleš Pultr, and Petr Kolman, editors, *Mathematical Foundations of Computer Science MFCS 2001*, volume 2136 of *LNCS*, pages 439–450. Springer, 2001. Long version to appear in *Theoretical Computer Science*.
 119. R. Klempien-Hinrichs, H.-J. Kreowski, and S. Taubenberger. Correct translation of mutually recursive function systems into TOL collage grammars. In G. Ciobanu and Gh. Păun, editors, *Fundamentals of Computation Theory: FCT'99*, volume 1684 of *LNCS*, pages 350–361, 1999.
 120. D. E. Knuth. Semantics of context-free languages. *Mathematical Systems Theory*, 2:127–145, 1968.
 121. D. E. Knuth. Correction: Semantics of context-free languages. *Mathematical Systems Theory*, 5(1):95–96, 1971.
 122. H. von Koch. Une méthode géométrique élémentaire pour l'étude de certaines questions de la théorie de courbes planes. *Acta Mathematica*, 30:145–174, 1905.
 123. G. Kókai, Z. Tóth, and R. Ványi. Evolving artificial trees described by parametric L-systems. In *IEEE Canadian Conference on Electrical & Computer Engineering*, pages 1722–1728, 1999.
 124. G. Kókai, Z. Tóth, and R. Ványi. Modelling blood vessels of the eye with parametric L-systems using evolutionary algorithms. In W. Horn, Y. Shahar, G. Lindberg, S. Andreassen, and J. Wyatt, editors, *Artificial Intelligence in Medicine AIMDM'99*, volume 1620 of *LNCS/LNAI*, pages 433–442. Springer, 1999.
 125. G. Kókai, R. Ványi, and Z. Tóth. Parametric L-system description of the retina with combined evolutionary operators. In W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R.E. Smith, edi-

- tors, *Proceedings of the Genetic and Evolutionary Computation Conference — GECCO-1999*, pages –. Morgan Kaufmann Publishers, 1999.
126. J. R. Koza. Discovery of rewrite rules in Lindenmayer systems and state transition rules in cellular automata via genetic programming. In *Symposium on Pattern Formation (SPF-93)*, Claremont, California, USA, 1993.
 127. M. Kudlek. Languages defined by Indian parallel systems. In G. Rozenberg and A. Salomaa, editors, *The Book of L*, pages 233–244. Berlin: Springer, 1985.
 128. M. Kudlek. Dead word languages and adult languages. *Pure Mathematics and Applications, Ser. A*, 1(3/4):343–355, 1990.
 129. M. Kudlek. Mix operation with catenation and shuffle. In S. Bozapalidis, editor, *Proc. of the 3rd Int. Conf. Developments in Language Theory*, pages 387–398. Aristotle University of Thessaloniki, 1997.
 130. M. Kudlek. On mix operation. In Gh. Păun and A. Salomaa, editors, *New Trends in Formal Languages*, volume 1218 of *LNCS*, pages 430–439. Berlin: Springer, 1997.
 131. M. Kudlek and A. Mateescu. On distributed catenation. *Theoretical Computer Science*, 180:341–352, 1997.
 132. W. Kurth. Towards universality of growth grammars: models of Bell, Pagès, and Takenaka revisited. *Ann. For. Sci.*, 57:543–554, 2000.
 133. W. Kurth and B. Sloboda. Tree and stand architecture described by formal grammars. *J. Forest Sci.*, 45:16–30 & 53–63, 1999.
 134. C. Lautemann, T. Schwentick, and D. Thérien. Logics for context-free languages. In *CSL: 8th Workshop on Computer Science Logic*, volume 933 of *LNCS*, pages 205–216. Springer, 1994.
 135. E. Lehman and A. Shelat. Approximations algorithms for grammar-based compression. In *Thirteenth Annual Symposium on Discrete Algorithms (SODA'02)*, 2002.
 136. M. K. Levitina. On some grammars with rules of global replacement (in Russian). *Scientific-technical information (Nauchno-tehnicheskaya informacii)*, Series 2, (3):32–36, 1972.
 137. A. Lindenmayer. Mathematical models for cellular interactions in development I. Filaments with one-sided inputs. *Journal of Theoretical Biology*, 18:280–299, 1968.
 138. A. Lindenmayer. Mathematical models for cellular interactions in development II. Simple and branching filaments with two-sided inputs. *Journal of Theoretical Biology*, 18:300–315, 1968.
 139. A. Lindenmayer. Developmental systems without cellular interactions, their languages and grammars. *Journal of Theoretical Biology*, 30:455–484, 1971.
 140. A. Lindenmayer. *Adding continuous components to L-systems*, volume 15 of *LNCS*, pages 53–68. Berlin: Springer, 1974.
 141. D. López and I. Piñaga. Syntactic pattern recognition by error correcting analysis on tree automata. In F. J. Ferri et al., editors, *Advances in Pattern Recognition, Joint IAPR International Workshops SSPR+SPR'2000*, volume 1876 of *LNCS*, pages 133–142, 2000.
 142. A. Mariano, P. Moscato, and M. G. Norman. Using L-systems to generate arbitrarily large instances of the Euclidean traveling salesman problem with known optimal tours. In *Anales del XXVII Simposio Brasileiro de Pesquisa Operacional*, 1995. See Site 10, file `final.ps.Z`.
 143. C. Martin-Vide. Natural language understanding: a new challenge for grammar systems. *Acta Cybernetica*, 12:461–472, 1996.

144. A. Mateescu. Special families of matrix languages and decidable problems. *Acta Cybernetica*, 10:45–52, 1991.
145. W. S. McCulloch. *Embodiments of the Mind*, chapter McCulloch and Pitt: A logical calculus of the ideas immanent in nervous activity, pages 19–39. MIT Press, 1965, originally the article appeared in 1943.
146. A. Meduna. Generative power of three-nonterminal scattered context grammars. *Theoretical Computer Science*, 246:279–284, 2000.
147. B. Meltzer and D. Michie, editors. *Machine Intelligence*, volume 7, chapter G. T. Herman, A. D. Walker: The syntactic inference problem applied to biological systems, pages 341–356. Edinburgh at the University Press, 1972.
148. V. Mitrana, Gh. Păun, G. Rozenberg, and A. Salomaa. Pattern systems. *Theoretical Computer Science*, 154:183–201, 1996.
149. M. Morcrette. Sur l'équivalence de descriptions de figures itérées. *Theoretical Computer Science*, 165:325–354, 1996.
150. K. Morita, Y. Yamamoto, and K. Sugata. The complexity of some decision problems about two-dimensional array grammars. *Information Sciences*, 30:241–262, 1983.
151. P. Moscato and M. G. Norman. An analysis of the performance of traveling salesman heuristics on infinite-size fractal instances in the Euclidean plane. See Site 10, file `lssystem7.ps.Z`, October 1994.
152. Ph. Narbel. *Limits and boundaries of word and tiling substitutions*. PhD thesis, Université Paris 7, 1993.
153. Ph. Narbel. Geometrical OL-systems coming from tilings of the plane. In G. Rozenberg and A. Salomaa, editors, *Developments in Language Theory (Turku, 1993)*, pages 237–249. Singapore: World Scientific, 1994.
154. C. G. Nevill-Manning and I. H. Witten. Compression and explanation using hierarchical grammars. *The Computer Journal*, 40(2/3):103–116, 1997.
155. C. G. Nevill-Manning and I. H. Witten. Online and offline heuristics for inferring hierarchies of repetitions in sequences. *Proc. IEEE*, to appear.
156. N. Nirmal and K. Krithivasan. Filamentous systems with apical growth. *International Journal of Computer Mathematics*, 12:203–215, 1983.
157. T. Y. Nishida. Sizes of context-free languages generated by context-free grammars and stable and recurrent OL systems. *Journal of Automata, Languages and Combinatorics*, 6(4):507–518, 2001.
158. M. Nolle. Ein Vergleich zwischen L-Systemen, iterierten Funktionensystemen und ähnlichen Methoden zur Erzeugung von Fraktalen. Studienarbeit, Universität Karlsruhe (TH), 1992.
159. M. Nolle. *Comparison of different methods for generating fractals*, pages 163–173. Basel: Gordon and Breach Science Publishers, 1994.
160. M. G. Norman and P. Moscato. The Euclidean traveling salesman problem and a space-filling curve. *Chaos, Solitons and Fractals*, 6:389–397, 1995. See Site 10, file `peano.ps.Z`.
161. G. Ochoa. On genetic algorithms and Lindenmayer systems. In A. Eiben, T. Baeck, M. Schoenauer, and H. P. Schwefel, editors, *PPSN V - Parallel Problem Solving from Nature*, volume 1498 of *LNCS*, pages 335–344. Springer, 1998.
162. A. Pascu and Gh. Păun. On simple matrix grammars. *Bull[etin] Math. de la Soc. Sci. Math. [de la R. S.] de Roumanie*, 20:333–340, 1976.
163. Gh. Păun. Linear simple matrix languages. *Elektronische Informationsverarbeitung und Kybernetik (jetzt J. Inf. Process. Cybern. EIK)*, 14:377–384, 1978.

164. Gh. Păun. On simple matrix languages versus scattered context languages. *RAIRO Informatique théorique*, 16(3):245–253, 1982.
165. Gh. Păun. Valences: increasing the power of grammars, transducers, grammar systems. *EATCS Bulletin*, 48:143–156, 1992.
166. Gh. Păun. *Marcus contextual grammar*. Studies in Linguistics and Philosophy. Dordrecht: Kluwer Academic Publishers, 1997.
167. Gh. Păun and C. S. Calude. *Computing with Cells & Atoms*. Taylor & Francis, 2001.
168. Gh. Păun and G. Rozenberg. Prescribed teams of grammars. *Acta Informatica*, 31:525–537, 1994.
169. Gh. Păun, G. Rozenberg, and A. Salomaa. Contextual grammars: Parallelism and blocking of derivation. *Fundamenta Informaticae*, 25:381–397, 1996.
170. Gh. Păun, G. Rozenberg, and A. Salomaa. Pattern grammars. *Journal of Automata, Languages and Combinatorics*, 1(3):219–235, 1996.
171. Gh. Păun, G. Rozenberg, and A. Salomaa. *DNA Computing: New Computing Paradigms*. Springer, 1998.
172. Gh. Păun and A. Salomaa, editors. *Grammatical Models of Multi-Agent Systems*, chapter H. Fernau, R. Freund, and M. Holzer: Regulated array grammars of finite index, pages 157–181 (Part I) and 284–296 (Part II). London: Gordon and Breach, 1999.
173. A. Paz. *Multidimensional parallel rewriting systems*, pages 509–515. North-Holland, 1976.
174. J. Pecht. On the real-time recognition of formal languages in cellular automata. *Acta Cybernetica*, 6:33–53, 1983.
175. H.-O. Peitgen, A. Rodenhausen, and G. Skordev. Self-affine curves and sequential machines. *Real Analysis Exchange*, 22(2):446–491, 1996–97.
176. P. Prusinkiewicz. A look at the visual modeling of plants using L-systems. In R. Hofestädt, Th. Lengauer, M. Löffler, and D. Schomburg, editors, *Bioinformatics, German Conference on Bioinformatics, Leipzig, Germany, September 30 - October 2, 1996, Selected Papers; German Conference on Bioinformatics*, volume 1278 of *LNCS*, pages 11–29. Springer, 1997.
177. P. Prusinkiewicz. Simulation modeling of plants and plant ecosystems. *Communications of the ACM*, 43:84–93, 2000.
178. P. Prusinkiewicz, M. Hammel, R. Měch, and J. Hanan. The artificial life of plants. In *Artificial Life for Graphics, Animation, and Virtual Reality*, volume 7 of *SIGGRAPH'95 Course Notes*, pages 1–38. ACM, ACM Press, 1995.
179. P. Prusinkiewicz and L. Kari. Subapical bracketed L-systems. In J. Curry, H. Ehrig, G. Engels, and G. Rozenberg, editors, *Grammars and Their Application to Computer Science*, volume 1073 of *LNCS*, pages 550–564, 1996.
180. P. Prusinkiewicz and A. Lindenmayer. *The Algorithmic Beauty of Plants*. New York: Springer, 1990.
181. P. Prusinkiewicz, L. Mündermann, R. Karwowski, and B. Lane. The use of positional information in the modeling of plants. In *SIGGRAPH 2001, Proceedings of the 28th Annual Conference on Computer Graphics*, pages 289–300. ACM, 2001.
182. V. Rajlich. Absolutely parallel grammars and two-way deterministic finite state transducers. In *Third Annual ACM Symp. Theory Computing*, pages 132–137. ACM, 1971.
183. V. Rajlich. Absolutely parallel grammars and two-way deterministic finite state transducers. *Journal of Computer and System Sciences*, 6:324–342, 1972.

184. K. Reinhardt. A parallel contextfree derivation hierarchy. In *Fundamentals of Computation Theory FCT*, volume 1684 of *LNCS*, pages 441–450. Springer, 1999.
185. R. D. Rosebrugh and D. Wood. A characterization theorem for n -parallel right linear languages. *Journal of Computer and System Sciences*, 7:579–582, 1973.
186. R. D. Rosebrugh and D. Wood. Image theorems for simple matrix languages and n -parallel languages. *Mathematical Systems Theory*, 8:150–155, 1974.
187. R. D. Rosebrugh and D. Wood. Restricted parallelism and right linear grammars. *Utilitas Mathematica*, 7:151–186, 1975.
188. A. L. Rosenberg. On multi-head finite automata. *IBM Journal of Research and Development*, 10:388–394, 1966.
189. D. J. Rosenkrantz. Programmed grammars and classes of formal languages. *Journal of the ACM*, 16(1):107–131, 1969.
190. G. Rozenberg. TOL systems and languages. *Information and Control (now Information and Computation)*, 23:357–381, 1973.
191. G. Rozenberg, K. Ruohonen, and A. Salomaa. Developmental systems with fragmentation. *International Journal of Computer Mathematics*, 5:177–191, 1976.
192. G. Rozenberg and A. Salomaa, editors. *L Systems*, volume 15 of *LNCS*, chapter A. Walker: Adult languages of L systems and the Chomsky hierarchy, pages 201–215. Berlin: Springer, 1974.
193. G. Rozenberg and A. Salomaa, editors. *L Systems*, volume 15 of *LNCS*, chapter S. Skyum: Nonterminals and codings defining variations of 0L-systems, pages 244–249. Berlin: Springer, 1974.
194. G. Rozenberg and A. Salomaa, editors. *L Systems*, volume 15 of *LNCS*, chapter D. Wood: Bounded parallelism and regular languages, pages 292–301. Berlin: Springer, 1974.
195. G. Rozenberg and A. Salomaa, editors. *L Systems*, volume 15 of *LNCS*, chapter Čulik, II, K. and J. Opatrný: Context in parallel rewriting, pages 230–243. Berlin: Springer, 1974.
196. G. Rozenberg and A. Salomaa, editors. *L Systems*, volume 15 of *LNCS*, chapter P. G. Doucet: The syntactic inference problem for D0L-sequences, pages 146–161. Berlin: Springer, 1974.
197. G. Rozenberg and A. Salomaa, editors. *Handbook of Formal Languages (3 volumes)*. Springer, 1997.
198. G. Rozenberg and A. Salomaa. *The Mathematical Theory of L Systems*. Academic Press, 1980.
199. G. Rozenberg and D. Vermeir. On ET0L systems of finite index. *Information and Control (now Information and Computation)*, 38:103–133, 1978.
200. G. Rozenberg and D. Vermeir. On the effect of the finite index restriction on several families of grammars. *Information and Control (now Information and Computation)*, 39:284–302, 1978.
201. M. Ruhl and H. Hartenstein. Optimal fractal coding is NP-hard. In J. A. Storer and M. Cohn, editors, *DCC'97 Data Compression Conference*. IEEE, 1997.
202. K. Salomaa. Hierarchy of k -context-free languages. *International Journal of Computer Mathematics*, 26:69–90, 193–205, 1989.
203. K. Sayood. *Introduction to Data Compression*. Morgan Kaufmann, 1996.

204. M. Seemann. Multiple-limited ET0L systems. In S. Bozapalidis, editor, *Proc. of the 3rd Int. Conf. Developments in Language Theory*, pages 377–385. Aristotle University of Thessaloniki, 1997.
205. R. Siromoney. On equal matrix languages. *Information and Control (now Information and Computation)*, 14:133–151, 1969.
206. R. Siromoney and K. Krithivasan. Parallel context-free languages. *Information and Control (now Information and Computation)*, 24:155–162, 1974.
207. A. R. Smith III. Two-dimensional formal languages and pattern recognition by cellular automata. In *12th Ann. IEEE Symposium on Switching and Automata Theory*, pages 144–152. IEEE, 1971.
208. A. R. Smith III. Real-time language recognition by one-dimensional cellular automata. *Journal of Computer and System Sciences*, 6:233–253, 1972.
209. A. R. Smith III. Introduction to and survey of polyautomata theory. In A. Lindenmayer and G. Rozenberg, editors, *Automata, Languages, Development*, pages 405–422. North-Holland, Amsterdam, 1976.
210. H. Spilker and D. Wätjen. Some undecidability results concerning k -limited 0L systems. *Fundamenta Informaticae*, 26:23–30, 1996.
211. E. D. Stotskii. Formal grammars and constraints on derivation. *Problemy peredachi informacii; translated: Problems of Information Transmission*, 7(1):69–81, 1971 (Translation 1973).
212. H. Straubing. *Finite automata, formal logic, and circuit complexity*. Boston, Basel, Berlin: Birkhäuser, 1994.
213. I. H. Sudborough. On tape-bounded complexity classes and multi-head finite automata. *Journal of Computer and System Sciences*, 10(1):62–76, 1975.
214. Suppes, Henkin, Joja, and Moisil, editors. *Logic, Methodology, and Philosophy of Science*, volume 4, chapter A. Lindenmayer: Cellular automata, formal languages and developmental systems, pages 677–691. Amsterdam: North Holland, 1973.
215. Y. Takada. Learning even equal matrix languages based on control sets. In A. Nakamura et al., editors, *Parallel Image Analysis, ICPIA'92*, volume 652 of *LNCS*, pages 274–289. Springer, 1992.
216. S. Takahashi. Self-similarity of linear cellular automata. *Journal of Computer and System Sciences*, 44:114–140, 1992.
217. V. Terrier. On real time one-way cellular array. *Theoretical Computer Science*, 141:331–335, 1995.
218. V. Terrier. Language not recognizable in real time by one-way cellular automata. *Theoretical Computer Science*, 156(1-2):281–287, 1996.
219. A. M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proc. London Math. Soc.*, 2(42):230–265, 1936. (Corrections on volume 2(43):544–546).
220. L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27:1134–1142, 1984.
221. Gy. Vaszil. Various communications in parallel communicating grammar systems. *Acta Cybernetica*, 13:173–196, 1997.
222. Gy. Vaszil. Communication in parallel communicating Lindenmayer systems. *Grammars*, 1(3):255–270, 1999. Special Issue on Grammar Systems. Guest editor: J. Kelemen.
223. Gy. Vaszil. On parallel communicating Lindenmayer systems. In Gh. Păun and A. Salomaa, editors, *Grammatical Models of Multi-Agent Systems*, vol-

- ume 8 of *Topics in Computer Mathematics*, pages 99–112. Gordon and Breach, Amsterdam, 1999.
224. Gy. Vaszil. On the generative capacity of parallel communicating extended Lindenmayer systems. In C. Martín-Vide and V. Mitrana, editors, *Grammars and Automata for String Processing: From Mathematics and Computer Science to Biology, and Back*. Taylor and Francis, (to appear).
 225. E. Vidal and M. J. Castro. Classification of banded chromosomes using error-correcting grammatical inference (ECGI) and multilayer perceptron (MLP). In *VIIth National Symposium on Pattern Recognition and Image Analysis*, pages 31–36. Centre de Visió per Computador, 1997.
 226. P. M. B. Vitányi. Lindenmayer systems: Structure, languages, and growth functions. Technical Report 96, Amsterdam: Mathematisch Centrum, 1980.
 227. P. M. B. Vitányi. Development, growth and time. In G. Rozenberg and A. Salomaa, editors, *The Book of L*, pages 431–444. Berlin: Springer, 1985.
 228. R. Vollmar. *Algorithmen in Zellularautomaten*, volume 48 of *LAMM*. Stuttgart: Teubner, 1979.
 229. P. S.-P. Wang. Parallel context-free array grammar normal forms. *Computer Graphics and Image Processing*, 15:296–300, 1981.
 230. P. S.-P. Wang. A new hierarchy of two-dimensional array languages. *Information Processing Letters*, 15:223–226, 1982.
 231. D. Wätjen. k -limited 0L systems and languages. *J. Inf. Process. Cybern. EIK (formerly Elektron. Inf.verarb. Kybern.)*, 24(6):267–285, 1988.
 232. D. Wätjen. On k -uniformly-limited T0L systems and languages. *J. Inf. Process. Cybern. EIK (formerly Elektron. Inf.verarb. Kybern.)*, 26(4):229–238, 1990.
 233. D. Wätjen. Restriction of active symbols in k -limited ET0L systems and a normal form theorem. *International Journal of Computer Mathematics*, 40:47–62, 1991.
 234. D. Wätjen. A weak iteration theorem for k -limited E0L systems. *J. Inf. Process. Cybern. EIK (formerly Elektron. Inf.verarb. Kybern.)*, 28(1):37–40, 1992.
 235. D. Wätjen. Regulation of k -limited ET0L systems. *International Journal of Computer Mathematics*, 47:29–41, 1993.
 236. D. Wätjen. Regulation of uniformly k -limited T0L systems. *J. Inf. Process. Cybern. EIK (formerly Elektron. Inf.verarb. Kybern.)*, 30(3):169–187, 1994.
 237. D. Wätjen. On regularly controlled k -limited T0L systems. *International Journal of Computer Mathematics*, 55:57–66, 1995.
 238. D. Wätjen. Regulations of uniformly k -limited ET0L systems and their relations to controlled context-free grammars. *Journal of Automata, Languages and Combinatorics*, 1(1):55–74, 1996.
 239. D. Wätjen. k -limited ED0L languages are context-sensitive. *EATCS Bulletin*, 61:89–91, 1997.
 240. D. Wätjen. Teams of limited and uniformly limited 0L systems. *Journal of Automata, Languages and Combinatorics*, 4(1):31–57, 1999.
 241. D. Wätjen. Undecidability results for uniformly k -limited 0L systems. *Journal of Automata, Languages and Combinatorics*, 5(2):159–167, 2000.
 242. D. Wätjen. Parallel communicating limited and uniformly limited 0L systems. *Theoretical Computer Science*, 255(1–2):163–191, 2001.
 243. D. Wätjen and D. Ostrovsky. Function-limited 0L systems revisited. *Journal of Automata, Languages and Combinatorics*, 6(1):97–114, 2001.

244. D. Wätjen and E. Unruh. On the degree of synchronization of k LETOL systems. *Information Processing Letters*, 29:87–89, 1988.
245. D. Wood. n -linear simple matrix languages and n -parallel linear languages. *Rev. Roumaine Math. Pures Appl.*, 22:403–412, 1977.
246. Th. Worsch. *Komplexitätstheoretische Untersuchungen an myopischen Polyautomaten*. Dissertation, TU Braunschweig, 1990.
247. Y. Yamamoto, K. Morita, and K. Sugata. Context-sensitivity of two-dimensional regular array grammars. *International Journal of Pattern Recognition and Artificial Intelligence*, 3:295–319, 1989.
248. T. Yokomori. Inductive inference of OL languages. In *Lindenmayer Systems*, pages 115–132. Springer, 1992.

