Parallel Grammars

A Phenomenological Approach

Henning Fernau
Universität Trier
fernau@informatik.uni-trier.de

Overview

- Phenomena / literature pointers
- Some definitions
- A flavor of results
- Possible projects
  - technical / mathematical
  - models / applications

Literature

- G. Rozenberg and A. Salomaa, editors. *Handbook of Formal Languages (3 volumes)*. Springer, 1997.
  In particular: Vol. 1, Chap. 5 & Vol. 3, Chap. 9

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

- P. Prusinkiewicz and A. Lindenmayer. *The Algorithmic Beauty of Plants*. New York: Springer, 1990.

These classical sources mostly deal with complete parallelism.

Many papers concerning partial parallelism.

Article on this course:

- H. Fernau. Parallel grammars: a phenomenology. *GRAMMARS*, vol. 6 (2003), pp. 25–87.

Phenomena

AIM: A (formal language) model of parallel computation

More specifically: a grammar model

Parallel execution of operations

Basic operation in grammars:

one derivation step $\leadsto$ parallel derivation step

Details to come ;-)

A. Lindenmayer I

starting point: modeling algae growth

filamentous organisms are strings (of cells)

Read the masters:

A. Lindenmayer. Mathematical models for cellular interactions in development I. Filaments with one-sided inputs. *Journal of Theoretical Biology*, 18:280–299, 1968.

A. Lindenmayer II Basic ideas of Lindenmayer's model:

1. Each cell is always in one of a finite number of states.

2. Cells may interact with their immediate (at most two) neighbors.

3. Cells may assume a new state depending on their own and their neighbors' states.

4. For an external observer, cells "process" in parallel;

5. these observations are undertaken after discrete time steps.

A. Lindenmayer III Mathematical translation of Lindenmayer's model:

1. finite-state model

2. transition function / relation takes at most three arguments

3. output of transition function / relation is again a (new) state

4. transition is applied to each cell (state) in parallel;

5. development over time modeled by repetitions of the application of the transition function / relation

A. Lindenmayer IV Is this all of Lindenmayer's model? NO

1. cells may multiply / divide

2. cells may die

$\rightsquigarrow$

grammatical model rather than automaton model (cellular automata)

if a cell is then a character of a string, then dying means becoming the empty word

A. Lindenmayer V Extensions of Lindenmayer's model

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.

A. Lindenmayer. *Adding continuous components to L-systems*, IN: volume 15 of *LNCS*, pages 53–68. Berlin: Springer, 1974.

A. Lindenmayer VI Modelling higher organisms

- growth only occurs near the boundary
- L systems with apical growth
- cellular automata (J von Neumann 1950..)
- (array grammars) [mostly viewed in sequential variants]

Literature

A. Burks. *The Theory of Self-Reproducing Automata* contains a chapter of J. v. Neumann (a reprint of earlier work?). The University of Illinois Press, 1966.

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.

V. Aladyev. Operations on languages generated by τ_n-grammars (in Russian). *Commentationes Mathematicae Universitatis Carolinae*, 15:211–220, 1974.

V. Aladyev. On the equivalence of τ_m-grammars and Sb(n)-grammars (in Russian). *Commentationes Mathematicae Universitatis Carolinae*, 15:717–726, 1974.

N. Nirmal and K. Krithivasan. Filamentous systems with apical growth. *International Journal of Computer Mathematics*, 12:203–215, 1983.

A. Lindenmayer VII Modelling higher organisms *more realistically*

- 2-3 dimensions:

  A. Paz. *Multidimensional parallel rewriting systems*, IN: Automata, Languages, Development, pages 509–515. North-Holland, 1976.

- computer graphics rendering: Prusinkiewicz

A shift of interests I Mathematics / Computer Science influence

Mathematical beauty

$\rightsquigarrow$ study of L systems without interaction

A. Lindenmayer. Developmental systems without cellular interactions, their languages and grammars. *Journal of Theoretical Biology*, 30:455–484, 1971.

A shift of interests II Biology influence

interest in the development of organisms (over time)

$\rightsquigarrow$

- study of growth patterns: PhD theses of P. G. Doucet and of P. M. B. Vitányi, the latter one with the title: Lindenmayer systems: structure, languages, and growth functions.

- partial parallel derivations, ecogrammar systems might avoid exponential / polynomial growth

Partial parallelism Literature overview (partially chronological)

- multihead finite automata / TMs

- finite index restriction $\rightsquigarrow$ absolutely parallel grammars

  V. Rajlich. Absolutely parallel grammars and two-way deterministic finite state transducers. *Journal of Computer and System Sciences*, 6:324–342, 1972.

- these can be generalized: scattered context grammars

  S. Greibach and J. Hopcroft. Scattered context grammars. *Journal of Computer and System Sciences*, 3:233–247, 1969.

J. Gonczarowski and M. K. Warmuth. Scattered versus context-sensitive rewriting. *Acta Informatica*, 27:81–95, 1989.

many papers of A. Meduna

- Simple matrix grammars

  O. Ibarra. Simple matrix languages. *Information and Control (now Information and Computation)*, 17:359–394, 1970.

  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.

- n-parallel automata and grammars

  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.

Image theorems for simple matrix languages and n-parallel languages. *Mathematical Systems Theory*, 8:150–155, 1974.

Restricted parallelism and right linear grammars. *Utilitas Mathematica*, 7:151–186, 1975.

- Indian parallelism / Bharat systems

R. Siromoney and K. Krithivasan. Parallel context-free languages. *Information and Control (now Information and Computation)*, 24:155–162, 1974.

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.

- (uniformly) limited Lindenmayer systems

  D. Wätjen. k-limited 0L systems and languages. *J. Inf. Process. Cybern. EIK (formerly Elektron. Inf.verarb. Kybern.)*, 24(6):267–285, 1988.

  K. Salomaa. Hierarchy of k-context-free languages. *International Journal of Computer Mathematics*, 26:69–90,193–205, 1989.

  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.

- parallel communicating grammar systems

  $\rightsquigarrow$ Lecture on grammar systems!

- (Pattern languages)

D. Angluin. Finding patterns common to a set of strings. *Journal of Computer and System Sciences*, 21:46–62, 1980.

- Concepts can be combined:
  - Russian parallel grammars

    M. K. Levitina. On some grammars with rules of global replacement (in Russian). *Scientific-technical information (Nauchno-tehnichescaya informacii), Series 2*, (3):32–36, 1972.

  - PCGS with partial parallelism

    D. Wätjen. Parallel communicating limited and uniformly limited 0L systems. *Theoretical Computer Science*, 255(1–2):163–191, 2001.

Features to be modeled

- (partial) parallelism (see above)

- synchronization
  - (limited) L systems
  - PCGS
  - trace languages

    V. Diekert and G. Rozenberg, editors. *Book of Traces*. World Scientific, Singapore, 1995.

- communication
  - neighborhood patterns in cellular automata
  - PCGS modes
- Communicating distributed grammar systems CDGS (Csuhaj-Varjú)

Parallel side tracks

- parallelism in operators
  (Kudlek / Mateescu; traces)
- picture processing
  - collage grammars (Kreowski/Habel/Drewes)
  - array grammars (Freund/Wang)
  - iterated function systems
- term rewriting
- "parallelism degrees" in context-free grammars (Brandenburg/Reinhardt)

Phenomena translation (The Lindenmayer Alphabet)

	$A \to \varepsilon$	erasing rule	cell death
P		**P**ropagating	cells don't die
	$A \to A$	chain rule	state change
(G)	$A \to BC$	growing rule	cell division
I	$XAY \to XBY$	context-sensing	**I**nteraction
J		(Finnish: **J**akautua)	fragmentation
T		**T**ables	environment conditions
A		**A**dult mechanism	only grown-ups count
C		**C**odings	non-observable features

Further Phenomena

Signals play an eminent role in CAs.

D eterminism $\rightsquigarrow$ DL systems

(a special terminal alphabet $\rightsquigarrow$ E xtension)

Overview

- Phenomena / literature pointers
- Some definitions
- A flavor of results
- Possible projects
  - technical / mathematical
  - models / applications

T0L systems

An *interactionless Lindenmayer system with tables*, for short a *T0L system*, is given by a triple $G = (\Sigma, H, \omega)$:

- Σ: alphabet.
- $H = \{h_1, \ldots, h_t\}$
  h_i: a finite substitution i. e.,
  $h_i : a \mapsto \{w_1, \ldots, w_{n_{i,a}}\}$.
- $\omega \in \Sigma^*$: axiom.

Some special cases are:

- $t = 1$: *0L system*.
- $\forall 1 \leq i \leq t \ \forall a \in \Sigma$: $n_{i,a} = 1$: *deterministic* T0L systems, or DT0L 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 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 $c : \Sigma \to \Delta$, investigate the language $c(L(G)) = \{\, c(w) \mid w \in L(G) \,\}$.

- $A(G) = \{\, w \in L(G) \mid \{w\} = \{u \mid w \Rightarrow u\}\,\}$ 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.

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

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

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

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

Taking these same rules plus the rules

$$\begin{array}{lll} A \to a, & A' \to a, & a \to F \\ B \to b, & B' \to b, & b \to F \\ C \to c, & C' \to c, & c \to F \\ F \to F & & \end{array}$$

result in an E0L system (with axiom ABC) which also generates L when taking $\{a, b, c\}$ as terminal alphabet.

synchronization by failure symbol F: useful when designing ET0L systems.

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

(How) can you prove that L is generated by the given C/E0L system ?

Can L be generated by an A0L system ?

If not, by an AT0L system?

Partial parallelism Formally, all systems can be specified like a T0L system $G = (\Sigma, H, \omega)$, $H = \{h_1, \ldots, h_t\}$. We redefine "$\Rightarrow$".

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

k-limited (kIT0L) 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; ($|x|_a$ is the number of occurrences of a in x.)

uniformly k-limited (ukIT0L) 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.

Another set of examples

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

0L system $abb \Rightarrow aaabab \Rightarrow a^6aba^2ab.$

0B system $abb \Rightarrow aabb \Rightarrow aaabab \Rightarrow a^6ba^2b$

1I0L system $abb \Rightarrow aaabb \Rightarrow aaaabab \Rightarrow a^4baaab$

2I0L system $abb \Rightarrow aaabab \Rightarrow a^5abaab$ (compare 0L / 1I0L)

u2I0L system $abb \Rightarrow aabab \Rightarrow aaaabab \Rightarrow a^6bab$

Observe the different "growth patterns."
What is $L(G)$ in each case?

Some further mechanisms

A *scattered context grammar* is a construct

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

$P = \{p_1, \ldots, p_t\}$, where

$$p_i = (A_1, \ldots, A_{n_i}) \to (w_1, \ldots, w_{n_i})$$

with $A_j \in N$ and $w_j \in \Sigma^*$ for $1 \leq j \leq n_i$.

$x \Rightarrow y$ iff there is an i, $1 \leq i \leq t$ and if there are words $u_0, \ldots, u_t \in \Sigma^*$ such that

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

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

Special cases

1. If in addition, $u_0, \ldots, u_{n_i} \in \Delta^*$ in Eq. (1), then the grammar is *absolutely parallel*.

2. If, with exception of the start rules, each n_i equals a specific n, then (in principle) we arrive at *simple matrix grammars*.

   Caveat: interpretation different in literature (leftmost);
   basic results are preserved.

   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."

In these two special cases, the different interpretations of simple matrix grammars coincide.

The so-called n*-parallel grammars* (and all the corresponding special cases discussed in a series of papers by Rosebrugh and Wood) 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.

Linguistic relevant examples

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

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:

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

Some sequential mechanisms

- 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 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 $\mathcal{L}(\mathrm{O}, \mathrm{CF}[-\varepsilon])$, where $-\varepsilon$ indicates that erasing rules are forbidden.

- A *programmed grammar* has—as rules—fragments of goto-programs of the following form:

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

  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.
  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 $\mathcal{L}(\mathrm{P}, \mathrm{CF}[-\varepsilon], \mathrm{ac})$.

Important special cases are:

– If all failure fields are empty, then we have a grammar *without appearance checking* whose language family is denoted as $\mathcal{L}(\mathrm{P}, \mathrm{CF}[-\varepsilon])$.

– 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 $\mathcal{L}(\mathrm{P}, \mathrm{CF}[-\varepsilon], \mathrm{ut})$.

A simple important relation

Theorem 1 $\mathcal{L}(O, CF[-\varepsilon]) \subsetneq \mathcal{L}(P, CF[-\varepsilon], ut)$.

Proof (Sketch): Let $A_1, \ldots, A_n$ be the left-hand sides of rules which are the rules that are "greater" than a specific rule $A \to w$. If F is a special new failure symbol, then the sequence ("matrix"!)

$$(A_1 \to F, \ldots, A_n \to F, A \to w)$$

can simulate an application of $A \to w$.

Since ordered languages are recursive, but not all ut-languages, the strictness follows.

A little exercise inbetween. . .

Write up the definitions in a formal way.

Prove the (weak) inclusion relation more formally.

Overview

- Phenomena / literature pointers
- Some definitions
- A flavor of results
- Possible projects
  - technical / mathematical
  - models / applications

A topical overview

The following list is a typical to-do list for formal language classes:

hierarchy questions,

decidability questions,

closure property questions,

combinatorial properties,

descriptional complexity and normal forms, and

learning / inductive inference.

(growth functions)

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 shown

either 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.

Decidability questions and complexity

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?

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*.

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), as well as several properties of ET0L and EDT0L 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.

Descriptional 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 descriptional 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. If rewriting of terminal symbols is permitted, the notion of *active symbol complexity* is to be studied instead.

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 descriptional complexity, especially, if the *degree of nondeterminism* is quantified.

For mechanisms involving context, also the "degree of context-sensing" can be quantified.

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 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.

In this connection, also new types of decidability questions arise, e. g.: is the growth function of a given D0L system a polynomial function?

Normal forms for ET0L systems

Lemma 2 *For every ET0L system, we can construct an equivalent ET0L system whose terminal symbols are only trivially rewritten, i. e., by rules of the form* $a \rightarrow a$.

Proof: Exercise! **Q.E.D.**

Theorem 3 *Every ET0L language is generatable by an ET0L system with only two tables.*

Proof: (Sketch) If L is generated by an ET0L system $G = (\Sigma, H, \omega, \Delta)$ which obeys the normal form of Lemma 2 with tables

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

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

$$\Sigma' = \{[A, i] \mid A \in \Sigma, 1 \leq i \leq m\}.$$

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 ET0L system $G' = (\Sigma', H', [\omega, 1], \Delta)$:

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

Q.E.D.

Alternatively, we can phrase the last theorem as follows:
ET0L systems have a *synchronization degree* of two.

We only state a corresponding result for limited ET0L systems.

Theorem 4 *The synchronization degree of klET0L systems is at most three and at least two.*

Can you find a proof?

Remarks on closure properties

lack of (positive) closure properties due to the "pure rewriting"

Lemma 5 *Each singleton language can be generated by a (uniformly (limited)) PD0L system.*

Proof: Take the word as axiom and introduce rules $a \to a$. **Q.E.D.**

Lemma 6 *$\{a, aa\}$ is no (uniformly (limited)) 0L language.*

Proof: If it were a 0L 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 $a \longrightarrow \varepsilon$.

But this would allow to generate ε, contradiction.*

If a is the axiom, we would have the rule $a \longrightarrow aa$ in our rule set, so that the generated language would include at least $\{a^{2^n} \mid n \geq 0\}$ in the case of 0L systems (similar for (uniformly) limited systems). **Q.E.D.**

Theorem 7 *$\mathcal{L}(0L)$ 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.*

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

We will only prove non-closure of 0L languages under four operations by making use of the preceding two lemmas in order to communicate the flavour of this kind of results:

union $\{a\} \cup \{aa\} = \{a, aa\} \notin \mathcal{L}(\mathsf{0L})$

catenation $\{a\} \cdot \{a, \varepsilon\} = \{a, aa\} \notin \mathcal{L}(\mathsf{0L})$

homomorphism $h : a, b \mapsto a \rightsquigarrow h(\{a, bb\}) = \{a, aa\} \notin \mathcal{L}(\mathsf{0L})$

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

On the other hand, when Lindenmayer systems with extensions are considered, we observe positive closure properties:

Theorem 8 *$\mathcal{L}(ET0L)$ forms an AFL.*

Theorem 9 *$\mathcal{L}(E0L)$ is closed under all AFL operations except from inverse homomorphisms.*

Parallel versus sequential derivations

How do CF languages relate to E0L, IE0L and A0L languages?

Lemma 10 *Any CF grammar can be simulated by an (I)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: sequentialization; we only add all rules $a \to a$ to the rule set of the given CF grammar. **Q.E.D.**

With the example $\{a^n b^n a^n \mid n \geq 1\}$ seen above, we can deduce:

Theorem 11 $\mathcal{L}(CF) \subsetneq \mathcal{L}(E0L) \cap \mathcal{L}(IE0L)$.

A new measure of descriptional complexity

Let $G = (\Sigma, H, \omega)$ be a T0L system. The *static(ally measured) degree of parallelism* of a table $h \in H$ is defined by

$$\pi^{st}(h) = \#\{\, a \in \Sigma \mid a \to a \notin h \,\}.$$

Correspondingly, for G we set

$$\pi^{st}(G) = \max\{\, \pi^{st}(h) \mid h \in H \,\}.$$

For a language L in $\mathcal{L}$(T0L), we define

$$\pi^{st}(L) = \min\{\, \pi^{st}(G) \mid G \text{ is an T0L system and } L = L(G) \,\}.$$

Adult languages

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

Theorem 12 $\mathcal{L}(CF) = \mathcal{L}(A0L)$.

The proof direction $\supseteq$ is rather tricky and omitted.

To understand why A0L systems are not more powerful than CFG, try to understand where the construction of the example $\{a^n b^n a^n \mid n \geq 1\}$ fails.

Think for a moment why the construction given in Lemma 10 does not always work in the "adult" situation.

Correct guess: malicious words.
The CF 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 CFL) makes the previous argument applicable.

Finally, we mention another interesting result in this context:

Theorem 13 $\mathcal{L}(ET0L) = \mathcal{L}(AT0L)$.

Lindenmayer systems and ordered grammars

Theorem 14 $\mathcal{L}(ET0L) \subsetneq \mathcal{L}(O, CF - \varepsilon)$.

In the proof, we make use of the following results.

Theorem 15 $\mathcal{L}(ET0L) = \mathcal{L}(EPT0L)$.

Lemma 16 $\mathcal{L}(O, CF - \varepsilon)$ *is closed under union.*

From a combinatorial property of ET0L systems, one can deduce:

Corollary 17 $L = \{(ab^m)^n c^n \mid m \geq n \geq 1\} \notin \mathcal{L}(ET0L)$

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 14:
Proof: Let $L \subseteq \Delta^*$ be an ET0L language. Let L be decomposed as

$$L = \bigcup_{a \in \Delta} aL_a \cup L_F \qquad \text{where} \qquad L_a = \{w \in \Delta^+ \mid aw \in L\}$$

is basically the left derivative of L under a and L_F is some finite language. The two theorems 15 and 8 show that each L_a is in fact an EPT0L language.

We are going to show that each language aL_a is an ordered language, which, together with Lemma 16 proves the desired result.

Let L_a by described by an ET0L system $G_a = (\Sigma, H, \omega, \Delta)$ with $H = \{h_1, \ldots, h_t\}$. Let Σ', Σ'' be alphabets of primed (double-primed) symbols from Σ. We interpret $'$ and $''$ also as homomorphisms, 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 simulating ordered grammar $G'_a = (N, P, S, \Delta, <)$:

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

The last bunch of symbols keeps track of the currently simulated table.

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

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

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

marking I: $B' \to (B,k) \quad \left\langle \begin{array}{ll} (A,s) \to F & \text{for } 1 \le s \le t, s \ne k \\ A \to F & \end{array} \right.$

the actual application of table h_k: $(B,k) \to w'' \quad \left\langle \begin{array}{ll} C' \to F & \text{for } C \in \Sigma \\ A \to F & \end{array} \right.$

marking IO: $B'' \to B' \quad \left\langle \begin{array}{ll} (C,k) \to F & \text{for } C \in \Sigma, 1 \le k \le t \\ A \to F & \end{array} \right.$

dispatcher rule: $(A,k) \to (A,s) \quad < \quad (B,r) \to F \quad \text{for } 1 \le k,r,s \le t, B \in \Sigma.$

termination rules: $(A,k) \to A \quad \left\langle \begin{array}{ll} (B,k) \to F & \text{for } B \in \Sigma \\ B' \to F & \text{for } B \in \Sigma \setminus \Delta \end{array} \right.$

$b' \to b \quad < \quad (A,k) \to F \quad \text{for } b \in \Delta, 1 \le k \le t$

$A \to a \quad < \quad b' \to F \quad \text{for } b \in \Delta$ **Q.E.D.**

What would have to be shown to actually produce a proof?

Limited Lindenmayer systems

In the following, we present three results:

Theorem 18 *Each klET0L language is also a 1lET0L language.*

Theorem 19 *Each klET0L language is a programmed language with unconditional transfer and each programmed language with unconditional transfer is a 1lET0L language.*
A similar statement is true for languages generatable by systems/grammars without erasing rules.

Theorem 20 *There are non-recursive klDT0L languages.*

Theorem 18 can be shown in the following way:

Proof: (Sketch) Idea: sequentialisation by marking when simulating G with G'.
For each symbol A of G, introduce marked symbols $A[i,j]$ with $1 \leq i,j \leq k$.

Marking table M_i, $1 \leq i \leq k$, has the following rules:
$A \to A[i,i]$ and $A[i-1,j] \to A[i,j]$ for each symbol A and $j < i$; all other (marked) symbols go to the failure symbol F.

For each original table h, there is a simulating table h' containing rule $A[k,j] \to w$ if $A \to w$ is present in h, as well as $A \to A$. (All other symbols go to F.) **Q.E.D.**

Is the sketch working?
Check: $L(G) \subseteq L(G')$? & $L(G') \subseteq L(G)$?
Claim: If simulation starts with M_1, then it has to cycle through $M_2, M_3, \ldots, M_k, h'$.

Two frequent techniques in this area: *sequentialisation* of rule applications and *explicit state information*.

Proof: (of Theorem 19)
For simulating 1lET0L systems with programmed grammars with ut, we introduce, for each table h, one simulation loop which enables, for each symbol a, some rule $a \to w$ of h to be simulated by using $a' \to w$. Moreover, there is a general priming loop where, for each symbol a, there is a rule $a \to a'$.
Assuming a certain normal form for 1lET0L systems due to D. Wätjen (namely, only "nonterminal" symbols are "non-constantly" replaced), this basically shows the sequentialisation construction of J. Dassow.
Observe: no need to show how to simulate klET0L systems by programmed grammars in general due to Theorem 18.

For the other direction, we use explicit state information stored in a state symbol.

For each rule $(p : A \to w, \gamma)$ of a programmed grammar, we introduce a table with non-failure rules $p \to q$ for each $q \in \gamma$ and $A \to w$.

The state symbol is erased only when a terminal string is derived (termination table). **Q.E.D.**

Theorem 18 can be generalized:

Theorem 21 *Consider two natural numbers k_1 and k_2. Then,*

- (k_1k_2)*lEPT0L* $\subseteq k_1$*lEPT0L and*
- (k_1k_2)*lET0L* $\subseteq k_1$*lET0L.*

Proof: We only give the proof for the first assertion (propagating case).
Let $G = (\Sigma, H, \omega, \Delta)$ be some (k_1k_2)lEPT0L system.
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_1lEPT0L system G'.
A table $h \in H$ is simulated by G' by applying a sequence of tables described by the regular expression

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

Details of the tables:

- T_0 has as non-failure rules $A \to A_1$ and $A_1 \to A_1$ for each symbol $A \in \Sigma$.
- M_i (for $i = 1, \ldots, k_2$) contains, for each $A \in \Sigma$, $A_i \to A[i,i]$ and $A[i-1,j] \to A[i,j]$ for $j < i$ as only non-failure rules.
- T_i (for $i = 1, \ldots, k_2 - 1$) has as non-failure rules $A_i \to A_{i+1}$, $A_{i+1} \to A_{i+1}$, $A[i,j] \to A[i,j]$ for each $A \in \Sigma$ and $j \leq i$.
- h' contains $A[k_2, j] \to w$ for $j \leq k_2$ if $A \to w \in h$. Moreover, conversion rules $A_{k_2} \to A$ and $A \to A$ for $A \in \Sigma$ are supplied. All other rules lead to the failure symbol.

Q.E.D.

Observation 22 *A similar idea of distributing the state information can be used to show that 1IEPT0L systems can simulate programmed grammars with unconditional transfer without erasing rules.*

Exercise: Provide the details!

Let us now prove Theorem 20: Problem: to choose the "right" computability model. We will use a variant of register machines in the following.

Each register is capable of storing a non-negative integer.

Registers are labelled by positive integers.

The input is stored in the first register.

The output is expected to be found in the second register.

Example 23 *An example of a register machine program (RMP):*

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

Example 23 computes $x + 2$ for each input x; if the start were at L_3, the identity function would be computed.

RMP consists of a sequence of labelled commands

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

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

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

Let $k \geq 1$. A *kIT0L machine* is given by

$$M = (\Sigma, \{h_1, \ldots, h_t\}, \{\sigma, x, y, R\}, k),$$

where Σ and $\{h_1, \ldots, h_t\}$ 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 $f : \mathbb{N}_0 \multimap \mathbb{N}_0$ iff the corresponding kIT0L system $G_{M,n} = (\Sigma, \{h_1, \ldots, h_t\}, x^{kn}R\sigma, k)$ 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 24 *For any computable function $f : \mathbb{N}_0 \multimap \mathbb{N}_0$ and any $k \geq 1$, there exists a kIT0L machine computing f.*

Proof: $f : \mathbb{N}_0 \multimap \mathbb{N}_0$ can be described by an (r, ℓ)-RMP P. We describe a simulating kIT0L machine $M = (\Sigma, H, \{\sigma, x, y, R\})$ with

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

F is a failure symbol; we list only productions which do not lead to F.
$L = \{L_1, \ldots, L_\ell\}$ is the set of labels controlling the simulation of the program P.
L' is a set of primed version of the labels in L.
We can assume that every jump statement in P is actually a sequence of two complementary statements: JZ_i lab_1 and JNZ_i lab_2 which carry the same register index i. This avoids implicit jumps.

The set of tables H consists of

- one initialization table h_I containing $\sigma \to \sigma$, $x \to x$ and $R \to C_1 \cdots C_r L_1$;

- two termination tables
  h_T with $\sigma \to \sigma$, $A_i \to \varepsilon$ for $i \neq 2$, $A_2 \to y$, $L_\ell \to S$, $S \to S$, $y \to y$, $C_i \to C_i$ $(i \leq r)$;
  and h'_T with $\sigma \to \sigma$, $C_i \to \varepsilon$ for $i \leq r$, $S \to \varepsilon$, and $y \to y$.
  This way, the result is transferred from register 2 into a sequence of y's.

- for any label L_j, there corresponds one or two simulation tables.
- $L_s : a_i$: $L_s \to L_{s+1}$, $C_i \to A_i^k C_i$, $C_j \to C_j$ for $j \neq i$, $A_j \to A_j$ for $1 \leq j \leq r$, $\sigma \to \sigma$.
- $L_s : s_i$: $L_s \to L_{s+1}$, $A_i \to \varepsilon$, $C_j \to C_j$ for $1 \leq j \leq r$, $A_j \to A_j$ for $j \neq i$, $\sigma \to \sigma$.
- $L_s : \mathsf{JZ}_i L_t$: $L_s \to L_t$, $C_j \to C_j$ for $1 \leq j \leq r$, $A_j \to A_j$ for $j \neq i$, $\sigma \to \sigma$.
- $L_s : \mathsf{JNZ}_i L_t$: There are two simulating tables here:
  - t_1: $L_s \to L'_s$, $A_i \to \sigma A_i$, $C_j \to C_j$ for $1 \leq j \leq r$, $A_j \to A_j$ for $j \neq i$, $\sigma \to \sigma$.
  - t_2: $L'_s \to L_t$, $\sigma \to \varepsilon$, $C_j \to C_j$ and $A_j \to A_j$ for $1 \leq j \leq r$.

Observe: we do not (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 observing the special role of σ as a *success witness*. **Q.E.D.**

With the help of Lemma 24, Theorem 20 can be shown: Exercise!

Theorem 25 *For any $k \geq 1$, $\mathcal{L}(O, CF) \subsetneq \mathcal{L}(k\mathit{IET0L})$.*

Proof: (Sketch of crucial simulation)
Alphabet of kIET0L system G' that simulates ordered grammar $G = (N, P, S, \Delta, <)$:

$$\Sigma = N \cup \Delta \cup \Delta' \cup \{\sigma, \chi, \alpha, F\}.$$

Priming is seen as a morphism keeping fixed symbols others than terminals.
Axiom of G': σS, σ indicates simulation mode.
Each rule $p : A \to w$ of G is simulated by two tables:

1. h_p containing $A \to \alpha^k w'$, $A \to A$, as well as $B \to F$ for each left-hand side B of any rule greater than $A \to w$ in G.
   Moreover, h_p contains rules $X \to X$ for the other nonterminals and rules $b' \to b'$ for the terminals b.
   $\sigma \to \chi$ introduces the checking mode marker;
   all other symbols are sent to the failure symbol F.

2. h'_p having only the following non-failure rules:
   $\chi \to \sigma$ and $\alpha \to \varepsilon$, as well as $X' \to X'$ for any nonterminal and terminal X of G.

h'_p checks that when applying h_p, rule $A \to \alpha^k w'$ was applied at most once.

Termination table t contains non-failure rules $b' \to b'$ and $b \to b$ for terminals b and $\sigma \to \varepsilon$ to stop the simulation. **Q.E.D.**

Explain: Purpose of primed terminals?!
Why simulation of ordered rule in two tables?!
Unfortunately, the above simulation does not carry over to the case of disallowing erasing rules in general. Why?

Uniformly limited systems

Theorem 26 *(Salomaa / Wätjen) The language families of k-uniformly limited E0L systems form an infinite hierarchy:*

$$\mathcal{L}(CF) = \mathcal{L}(1ulE0L) \subsetneq \mathcal{L}(2ulE0L) \subsetneq \mathcal{L}(3ulE0L) \subsetneq \cdots .$$

Theorem 27 *For all $k \geq 1$, we have:*

$$\begin{aligned} \mathcal{L}(kulET0L) &\subseteq \mathcal{L}(P, CF) \\ \mathcal{L}(kulEPT0L) &\subseteq \mathcal{L}(P, CF - \varepsilon) \end{aligned}$$

Proof: (Idea contains (possibly) non-algorithmic elements in the erasing case!) Given a $k$ulET0L system $G = (\Sigma, H, \omega, \Delta)$, we can compute the finite set

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

Consider now a slight variant of uniform k-limitation—*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 $k$ulET0L 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).

Each *exact* uniformly limited system can be simulated by a programmed grammar without appearance checking by sequentialisation. $\mathcal{L}(\mathrm{P}, \mathrm{CF})$ is closed under union, $\rightsquigarrow \checkmark$.

Q.E.D.

Open: (1) Can L' be computed algorithmically?
(2) Furthermore, the strictness of the inclusions in the preceding theorem is unknown.

Appropriate combinatorial lemmas $\rightsquigarrow$

Theorem 28

- *For each* $k \geq 1$, $\mathcal{L}(kulE0L) \subsetneq \mathcal{L}(kulET0L)$, *and*

- *for each* $k \geq 1$, $\mathcal{L}(klE0L) \subsetneq \mathcal{L}(klET0L)$.

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

A result on scattered context grammars

Aim: Pove universality result. Problem: to choose the "right" computational model. Here, we rely on the following result:

Theorem 29 *Every recursively enumerable language can be represented as the homomorphic image of the intersection of two context-free languages.*

Hint: have a look on the proof of the undecidability of Post's correspondence problem

Theorem 30 $\mathcal{L}(RE) = \mathcal{L}(SC)$.

Proof: $\supseteq$: ✓ (Church's thesis)

$\subseteq$: Use Theorem 29. G_1 and G_2 be CFGs, $G_i = (\Sigma_i, P_i, S_i, \Delta), i = 1, 2, \Sigma_1 \cap \Sigma_2 = \Delta$. Let $h : \Delta \to T$ be a homomorphism with $\Delta \cap T = \emptyset$. Define as total alphabet of the simulating scattered context grammar $G = (\Sigma, P, S, \Delta)$

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

S and \$ being new symbols. Note: Δ are nonterminal symbols!
Rules of G:

- $(S) \to (\$S_1\$\$S_2\$)$ as start rule;
- $(A) \to w$ for each $A \to w \in P_1 \cup P_2$; (to simulate G_1 and G_2)
- $(\$, A, \$, \$, A, \$) \to (h(A), \$, \$, \varepsilon, \$, \$)$ for each $A \in \Delta$;
  this way, a string $\$w\$\$w\$$ with $w \in \Delta^*$ will be transformed into $h(w)\$^4$;
- $(\$, \$, \$, \$) \to (\varepsilon, \varepsilon, \varepsilon, \varepsilon)$.

Q.E.D.

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

Lemma 31 $\mathcal{L}(SC - \varepsilon) \subseteq \mathcal{L}(CS)$.

Corollary 32 *Every language representable as the morphic image of the intersection of two context-free languages of finite index is also an absolutely parallel language.*

Back to Lindenmayer Classical topics:

- Coding versus extensions
- Fragmentation
- Growth patterns / Interaction
- Decidability

Coding versus extensions

Theorem 33 $\mathcal{L}(E0L) = \mathcal{L}(C0L)$.

Proof: (Rough idea) $\supseteq$: reconsider $\{a^n b^n c^n \mid n \geq 1\}$ example (*synchronization by failure symbol*)

$\subseteq$: more tricky. . .

Q.E.D.

Fragmentation

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 $q \to q$ having q as left-hand side.

$$J(G, q) := \{v \in (\Sigma \setminus \{q\})^* \mid \exists u, w \in \Sigma^* : q\omega q \stackrel{*}{\Rightarrow} uqvqw\}.$$

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

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

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

Example 34 *Consider the J0L system* (G, q) *with*

$$G = (\{a, b, q\}, h, aba) \quad \textit{and}$$

$$h = \{a \to a, b \to abaqaba, q \to q\}.$$

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

Proof: 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 $a^n ba$ 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; no other words are producible by G. **Q.E.D.**

Exercise: Investigate the J0L language generated by

$$G = (\{a, b, c, q\}, \{a \to aqa, b \to ba, c \to qb, q \to q\}, abc).$$

Theorem 35 $\mathcal{L}(EJ0L) = \mathcal{L}(E0L)$.

Growth patterns / Interaction

In (i, j)*L systems* (or *IL systems* if the numbers i and j are arbitrary), 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.

Example 36 *Consider the DIL system with axiom* ad *and the rules:*

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

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, *grows at a logarithmic scale.*

Given a DIL system $G = (\Sigma, h, \omega)$, the *DIL growth function*

$$g_G : n \mapsto |w_n| \quad \text{for} \quad \omega \Rightarrow^n w_n$$

gives the length of the word derived after n steps.

Lemma 37 *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 $a \to a^2$;

2. polynomial growth: $a \to ab, b \to b$ generate, starting with a, ab^n;

3. logarithmic growth: see Example 36.

Growth patterns without interaction

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 stored in a vector (the *Parikh (row) vector* of the axiom) and in a matrix (the *growth matrix* of the rules).

Example 38 *Consider the D0L system*

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

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 $c \to c$.

What is the matrix of Parikh vectors of the words derivable from a, b, *and* c *after two derivation steps? 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*:*

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

Hence, 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 Theorem). In particular, this means that

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

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

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

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

Example 39 *Let us compute M_G^3 for system G from Example 38. 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:

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

This means that

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

with initial values

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

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

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

Then, g is ultimately constant*.*

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

Corollary 41 *There are DIL growth functions which are not D0L growth functions.*

Decidability

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 42 *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 $\forall 0 \leq i \leq n_1 + n_2 (g_{G_1}(i) = g_{G_2}(i))$. **Q.E.D.**

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

   Theorem 43 *The polynomiality problem for D0L systems is decidable.*

   Idea: A rule $a \longrightarrow w$ 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 $n \geq 3$.

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

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

and

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

coincide or not is the *sequence equivalence problem*.

Theorem 44 *Both the sequence equivalence and the language equivalence problems are decidable for D0L 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 45 *Sequence cardinality is undecidable when given two DT0L or two 0L 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 46 *Derivation slenderness is decidable for D0L systems.*

Intriguingly, the seemingly related slenderness problem is open for D0L 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 47 *Language equivalence is undecidable for P0L systems. The context-freeness, regularity and 0L-ness problems are undecidable for E0L systems.*

UL systems are unary 0L systems. Here, things tend to be easier, since—by interpreting words as unary numbers—tools from number theory come at hand.

Theorem 48 *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.*

Overview

- Phenomena / literature pointers
- Some definitions
- A flavor of results
- Possible projects
  - technical / mathematical
  - models / applications

Research Paper Proposal No. I: Adult languages are meant to model "maturity" in grown-up organisms whose growth pattern is modelled by means of a Lindenmayer system $G = (\Sigma, H, \omega)$. You can easily observe that the adultness condition formulated by A. Walker is not actually covering "reality," where small changes can be observed also in adult organisms.

Based on a measure μ of "closeness" between strings, 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: 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. □

Research Paper Proposal No. II: 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

$$\mathcal{L}(\mathrm{EJ0L}) = \mathcal{L}(\mathrm{E0L})$$

transfer to other cases. □

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

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

We already discussed above 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. IV: Investigate and compare the two notions of simple matrix grammars with context-free cores. □

Modelling

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

- the ideas of McCulloch and Pitt (1943) which led to the consideration of neural networks,
- the investigations of von Neumann (1950s) laying the foundations of the theory of cellular automata,
- and the models proposed by Lindenmayer for simple filamentous organisms which initiated the theory of Lindenmayer systems.

In more recent times, DNA computing and membrane computing, motivated by biological phenomena, and quantum computing.

Due to the inherent parallelisms found in nature, all these model are in fact models of parallel computation.

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.

Research Project Proposal No. V: 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.

Observe that the first papers on Lindenmayer systems were actually published in biology journals, but later on almost all papers appeared in journals dedicated to Theoretical Computer Science, see the references in (Rozenberg & Salomaa, 1980).

A notable exception is a recent sequence of papers applying Lindenmayer systems to model the developments of forests by W. Kurth, e.g., Towards universality of growth grammars: models of Bell, Pagès, and Takenaka revisited. *Ann. For. Sci.*, 57:543–554, 2000.

Of course, also the papers of P. Prusinkiewicz and his collaborators deserved to mentioned here, most of them appearing in Computer Graphics Conferences and Journals.

In this respect, the interview of A. Lindenmayer published in *EATCS Bulletin*, 23:185–198, 1984, 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 the project on parametric L systems. □

Research Project Proposal No. VI: 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: 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) —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 and of colonies where largely motivated by social theories. This remark includes other more special issues like the formation of teams.

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 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.

So, it would be very beneficial—both from the point of view of language theory and from the point 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. □

Research Project Proposal No. VII: Both L systems and CDGS 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 the RP on biomodeling—can be viewed as a step in this direction. □

Research Project Proposal No. VIII: Parallel communicating grammar systems (PCGS) were somehow invented to model phenomena encountered in synchronized parallel processing. But what about the real applicability of this approach?

An exception in this direction is 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, *Grammatical Models of Multi-Agent Systems*, pages 64–86. Gordon and Breach, 1999.

Aho and others used parallel finite automata to model communication protocols by automata, see 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.

It would be therefore interesting to see applications of PCGS in areas like protocol specification and the semantics of parallel programming languages. □

Applications

(NOT in the "original" area)

+ computer graphics

* pattern recognition

+ fractal geometry

* benchmark generation

- parsing / ambiguity

- data compression

- MPEG scripts

Computer graphics

- We observe the derivation process of a certain grammar G working with an alphabet containing, among other symbols: f, F, $+$, $-$.

- A 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 F: the turtle draws a unit length line, moving in the current direction.

  Skipping f: similarly, without drawing.

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

Turning left analogously, a left turn by δ degrees is indicated by $-$.

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.

Example 49 *The D0L system with the following simple rules:* $F \to F - F + + F - F$, $+ \to +$ *and* $- \to -$ *generates the well-known* Koch-curve.

$F - F + + F - F$ *is the string after the first iteration, while* $F - F + + F - F - F - F + + F - F + + F - F + + F - F - F - F + + F - F - F - F + + F - F - F - F + + F - F + + F - F + + F - F - F - F + + F - F + + F - F + + F - F - F - F + + F - F + + F - F + + F - F - F - F + + F - F - F - F + + F - F - F - F + + F - F + + F - F + + F - F - F - F + + F - F$ *is the result of the third iteration.*

Research Project Proposal No. IX: 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 works of Čulik, Dube, Morcrette and Nolle.

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 F. M. Dekking. Recurrent sets. *Advances in Mathematics*, 44:78–104, 1982.

Another (yet unexplored) interpretation is via number systems; (see the chapter on L systems in "the Handbook").

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.

A further related research project $\rightsquigarrow$ RP on compression. $\square$

Parametric / attributed L systems

Research Project Proposal No. X: As pointed out in "the Handbook", 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 in the sequential case—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 (Lindenmayer 1974).

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 (Fernau & Stiebe 2001), parallel grammars with so-called valuations were considered, which is a nowadays classical form of regulation originating in works of (Păun 1982) which are interpretable as a form of attributed parallel grammars.

The mentioned can be a good starting point for writing the first one or two papers on this study subject, since it lists a whole number of concrete open questions. □

Research Project Proposal No. XI: As shown in a series of papers on applying partially parallel array grammars, these are quite successful for pattern recognition purposes, see, e.g.,

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.

Possibly, partial parallelelism 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.

Moreover, this research project (coming from the praxis of computation) can be related to the study of parallel attribute grammars, where the attributes can be used to described certain aspects of the patterns. □

Unorthodox applications I: Benchmarking

L systems are quite popular for describing fractal-like recursive structures by a turtle interpretation of the generated strings. Hence, they were used in order to describe particular hard instances for the Traveling Salesman Problem (TSP), see, e.g.,

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.

Due to the recursive structure of the constructed instances, they could prove optimality 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 TSP against optimal solutions, so that it is possible to get good benchmark examples.

Research Project Proposal No. XII: Find other NP-hard problems where it is possible to construct benchmark examples by using parallel grammars! Prove 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. □

Unorthodox applications II: Data compression

C. G. Nevill-Manning and I. H. Witten used L systems to generate simple benchmark examples for data compression algorithms.

C. G. Nevill-Manning and I. H. Witten. Compression and explanation using hierarchical grammars. *The Computer Journal*, 40(2/3):103–116, 1997.

Unorthodox applications III: Neural networks

A similar "unorthodox" application of L systems was reported in 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. There, neural network structures were examined by using L systems.

Independently, L systems were used in connection with neural nets in 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. and in a more conventional way for generating dendrite structures in 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.

Unorthodox applications IV: Miscellaneous

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 `http://www.abc.net.au/science/news/stories/s210368.htm` and, as some tutorial, `http://www.csu.edu.au/ci/vol03/mccorm/mccorm.html`.

Moreover, L systems have been envisaged by J. F. McGowan as a tool for describing certain replacement operations for nanosystems, see `http://www.jmcgowan.com/nlsystem.PDF`

Unexplored applications: linguistics

Research Project Proposal No. XIII: 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. Let us mention here that programmed grammars of finite index have been employed to model certain features of the database query language DATALOG, see 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., which is interesting in our context due to the mentioned characterization by absolutely parallel grammars.

To develop such a "parsing program," it is also important to give good automata characterizations, see RP on characterization. □

Unexplored applications: compression

One of the intriguing things about fractal geometry and its probably most popular offspring, namely *iterated function systems (IFS)*, is the possibility to employ this theory as a data compression methodology.

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. Then, the hopefully few parameters of the IFS describing the given picture are transmitted instead of the whole bitmap.

Research Project Proposal No. XIV: Due to the known close relations between iterated function systems and Lindenmayer systems, it is tempting to try a similar approach based on L 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, i.e., a grammatical inference problem.

Interestingly, one could also try to use parallel grammars for compressing text data. Hints in this direction can be found, e.g., in:

E. Lehman and A. Shelat. Approximations algorithms for grammar-based compression. In *Thirteenth Annual Symposium on Discrete Algorithms (SODA'02)*, 2002.

C. G. Nevill-Manning and I. H. Witten. Compression and explanation using hierarchical grammars. *The Computer Journal*, 40(2/3):103–116, 1997. □

Unexplored applications: MPEG

Research Project Proposal No. XV: MPEG-4 is the new standard for films etc. as recently described in the MPEG group. We refer to: `http://mpeg.telecomitalialab.com/standards/mpeg-4/mpeg-4.htm` and `http://mpeg.telecomitalialab.com/documents/from\_mpeg-1\_to\_mpeg-21.htm`.

This 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. □

Unexplored applications: cryptography

Research Project Proposal No. XVI: As explained in Vol. 2 of "the Handbook," 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 has its obvious bearing, since cryptography is the major realm of practical applications of complexity theory. □

Mathematical challenges: Complexity

Research Project Proposal No. XVII: Although several issues, notably regarding the computational complexity of L systems, have been analyzed, 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.

Note that in some cases upperbounds are known, sometimes disguised as language theoretic inclusion relations. As a concrete example, it is known that kIED0L 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 the recursion theoretic project below. □

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. XVIII: As pointed to in (Rozenberg & Salomaa, 1980, 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 (yet fruitless) attempts to employ techniques from the emerging area of *parameterized complexity*.

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 EDT0L 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. □

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.

Mathematical challenges: Recursion Theory

Research Project Proposal No. XIX: 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 developped the corresponding notions.

Nevertheless, this question can be important to typical questions of formal language theory, as well, e. g., regarding hierarchy relations.

Only few of these questions have been answered yet, see 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.

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. □

Mathematical challenges: Descriptional Complexity

This area discusses how economical certain "resources" in a grammar may be used without loosing descriptional power. Such considerations can become very important in practice (see RP on compression).

Research Project Proposal No. XX: Besides scattered context grammars (many papers by Meduna) and classical Lindenmayer systems, only scarce results are known for other mechanisms.

We already discussed the synchronization degree in Theorem 4. 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.

A recent paper is: H. Bordihn and H. Fernau. The degree of parallelism. In: *Descriptional Complexity of Formal Systems 7th Workshop (DCFS 2005)*. □

Mathematical challenges: Inclusions and Hierarchies

Research Paper Proposal No. XXI: We spread a number of open problems throughout the lecture. 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, you 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. □

Mathematical challenges: Characterizations

Research Project Proposal No. XXII: 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 (Straubing 1994) for (subclasses of) the regular languages and 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. in the case of CFL, or on expressions, as the well-known regular expressions for REG and the less known context-free expressions (Gruska, 1971) for CFL.

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.

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, see H. Fernau and H. Bordihn. Remarks on accepting parallel systems. *International Journal of Computer Mathematics*, 56:51–67, 1995. □

Mathematical challenges: Learning

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

- *Identification in the limit* (Gold 67) 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* where only positive examples are given to the learner, and *identification in polynomial time and data*, see C. de la Higuera. Characteristic sets for polynomial grammatical inference. *Machine Learning*, 27(2):125–138, 1997.

- 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. 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 (1984) 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)*.

There are a couple of papers dealing with the inference of Lindenmayer systems. There have been also more empirical approaches to this inference problem.

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 is surprisingly scarce.

Even "worse" seems to be the situation within other types of parallel grammars: nearly all work is left to be done.

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 groundbreaking paper D. Angluin. Finding patterns common to a set of strings. *Journal of Computer and System Sciences*, 21:46–62, 1980. □

More research ideas

- multidimensional parallel rewriting
- parallel rewriting on trees, graphs, . . .
- . . .

Good luck with your PhD!

Recent additions

To quote from `www.algorithmicbotany.org`:

Publications 2004:

Martin Fuhrer, Henrik Wann Jensen, and Przemyslaw Prusinkiewicz. Modeling hairy plants. Proceedings of Pacific Graphics 2004 pp. 217-226.

Enrico Coen, Anne-Galle Rolland-Lagan, Mark Matthews, Andrew Bangham, and Przemyslaw Prusinkiewicz. The genetics of geometry. Proceedings of the National Academy of Sciences 101 (14), pp. 4728-4735.

Przemyslaw Prusinkiewicz. Self-similarity in plants: Integrating mathematical and biological perspectives. In M. Novak (ed.), Thinking in Patterns: Fractals and Related Phenomena in Nature, pp. 103-118.

Pavol Federl and Przemyslaw Prusinkiewicz. Solving differential equations in developmental models of multicellular structures expressed using L-systems. In Proceedings of ICCS 2004, Lecture Notes in Computer Science 3037, pp. 65-72.

Pavol Federl and Przemyslaw Prusinkiewicz. Finite element model of fracture formation on growing surfaces. In Proceedings of ICCS 2004, Lecture Notes in Computer Science 3037, pp 138-145.

Przemyslaw Prusinkiewicz. Modeling plant growth and development. Current Opinion in Plant Biology 7 (1), pp 79-83.

Przemyslaw Prusinkiewicz. Art and science for life: Designing and growing virtual plants with L-systems. Acta Horticulturae 630, pp. 15-28.

Publications 2003:

Colin Smith, Przemyslaw Prusinkiewiz, and Faramarz Samavati. Local specification of surface subdivision algorithms. In Proceedings of AGTIVE 2003, Lecture Notes in Computer Science 3062, pp. 313-327.

Przemyslaw Prusinkiewicz, Faramarz Samavati, Colin Smith, and Radoslaw Karwowski. L-system description of subdivision curves. International Journal of Shape Modeling 9 (1), pp. 41-59.

Radomir Mech and Przemyslaw Prusinkiewicz. Generating subdivision curves with L-systems on a GPU. SIGGRAPH 2003 Sketches and Applications.

Charles Baker, Sheelagh Carpendale, Przemyslaw Prusinkiewicz, and Michael Surette. GeneVis: simulation and visualization of genetic networks. Journal of Information Visualization 2 (4), pp 201-217.

Lars Mündermann, Peter MacMurchy, Juraj Pivovarov, and Przemyslaw Prusinkiewicz. Modeling lobed leaves. In Proceedings of CGI 2003, pp. 60-65.

Radoslaw Karwowski and Przemyslaw Prusinkiewicz. Design and implementation of the L+C modeling language. Electronic Notes in Theoretical Computer Science 86 (2), 19 pp.

Mario Costa Sousa and Przemyslaw Prusinkiewicz. A few good lines: Suggestive drawing of 3D models. Proceedings of Eurographics 2003: Computer Graphics Forum 22 (3), pp. 381-390.

Frederic Boudon, Przemyslaw Prusinkiewicz, Pavol Federl, Christophe Godin and Radoslaw Karwowski. Interactive design of bonsai tree models. Proceedings of Eurographics 2003: Computer Graphics Forum 22 (3), pp. 591-599.

Combining different ideas

Gyrgy Vaszil: Collapsing hierarchies of parallel rewriting P systems without target conflicts (with D. Besozzi, G. Mauri, and C. Zandron), In: Membrane Computing. International Workshop WMC 2003, Tarragona, Spain, July 17-22, 2003. Revised Papers. Volume 2933 of Lecture Notes in Computer Science, edited by C. Martín-Vide, G. Mauri, Gh. Paun, G. Rozenberg, and A. Salomaa. Springer-Verlag Berlin-Heidelberg, 2004, 55-69.

The biological context

Perttunen, J. and Sievnen, R. 2005. Incorporating Lindenmayer systems for architectural development in a functional-structural tree model. Ecological Modelling 181(4): 479-491, see `http://www.sal.hut.fi/Personnel/Homepages/JariP/thesis/article_Vb.pdf`

S. Chuai-Aree, W. Jäger, H. G. Bock, S. Siripant, C. Lursinsap: PlantVR : An Algorithm for Generating Plant Shoot and Root Growth Using Applied Lindenmayer Systems. In B.Hu and M. Jaeger (eds.): Proc. of 2003' Intern. Symposium on Plant growth Modeling, simulation, visualization and their Applications , Tsinghua University Press - Springer Verlag, October 13-16, Beijing, China, pp. 55-68, 2003.

S. Chuai-Aree, W. Jger, H. G. Bock, S. Siripant: Simulation and Visualization of Plant Growth using Lindenmayer Systems, Proc. of Intern. Conf. on High Performance Scientific Computing (HPSC), March 10-14, Hanoi, Vietnam, 2003.

From `http://www.geog.ucl.ac.uk/~plewis/phd2003.html`; suggested PhD topic:
Ph3D: Modelling and monitoring the temporal dynamics of wheat using 3D dynamic vegetation models and Lindenmayer systems

Nice fractal tools

http://robotics.ee.uwa.edu.au/lgrammar/java/

Unvonventional applications:

`http://www.demo.cs.brandeis.edu/pr/evo_design/evo_design.html`

Introduction

Recent research has demonstrated the ability for the automatic design of basic shapes and the morphology and control of real physical robots using techniques inspired by biological evolution. ...

Here we claim that for automatic design systems to scale in complexity the designs they produce must be made of re-used modules. Our approach is based on the use of a generative representation as the method to encode individuals in the population. Unlike a direct representation of a design, a generative representation is an algorithm for creating a design. That is, the data being optimized by the search algorithm is itself a kind of program containing rules and program-like instructions for generating a design, with the ability to re-use parts of the program in a modular way allowing it to create more complex parts from simpler parts and re-use a component multiple times in a design.

We describe a system for creating generative specifications capable of hierarchical regularity by using Lindenmayer systems (L-systems) as the generative representation for an evolutionary algorithms. Using this system we demonstrate a system that evolves hierarchically modular tables and locomoting robots (called genobots).

G. Escuela, G. Ochoa, N. Krasnogor. (2005) Evolving L-Systems to Capture Protein Structure Native Conformations. 8th European Conference on Genetic Programming (EuroGP 2005), Lecture Notes in Computer Science 3447, pp 73-83,Springer-Verlag, Berlin.

What I missed out...

Ligia Collado-Vides, Guillermo Gómez-Alcaraz, Gerardo Rivas-Lechuga and Vinicio Gómez-Gutierrez: Simulation of the clonal growth of Bostrychia radicans (Ceramiales-Rhodophyta) using Lindenmayer systems. Biosystems, Volume 42, Issue 1 , March 1997, Pages 19-27.

Possibly, much more . . .