BRNO UNIVERSITY OF TECHNOLOGY
Faculty of Electrical Engineering and Communication
Department of Information Systems

Ing. Milan Kolka

L-SYSTEMS: NEW RESULTS AND APPLICATION

L-SYSTÉMY: NOVÉ VÝSLEDKY A APLIKACE

SHORT VERSION OF PH.D. THESIS

Study field:	Information technology
Supervisors:	Doc. RNDr. Alexander Meduna, CSc.
Opponents:	Prof. Ing. Bořivoj Melichar, DrSc. Prof. Ing. Ivo Serba, CSc. RNDr. Petr Šaloun, Ph.D.
Presentation date:	17th March 2004

KEYWORDS

L-systems, reduction, exponential density, string interpretation, graph grammar, chord interpretation, programming language

KLÍČOVÁ SLOVA

L-systémy, redukce, exponenciální hustota, interpretace řetězce, grafová gramatika, interpretace tětiv, programovací jazyk

THE DISSERTATION THESIS IS AT ONE'S PROPOSAL:

Department of Information Systems
Faculty of Information Technology
Božetěchova 2
612 00 Brno
Czech Republic

ISBN 80-214-2799-X
ISSN 1213-4198

CONTENTS

1 INTRODUCTION

The history of L-systems started in 1968 when a biologist Aristid Lindenmayer introduced his formalism for the description of development of filamentous organism. This formalism, which is called L-systems after the author, uses a final set of symbols for cells and their developmental stages. A filamentous organism is a sequence of cells and it is represented as a string of corresponding symbols. The cells can divide, die or grow. The evolution of a cell is expressed as a rule as in Chomsky grammars. Because the evolution of a metazoan organism is a simultaneous evolution of all its parts, of all its cells, the rules are applied to all symbols in the string in parallel.

In the terms of formal language theory, L-systems are parallel grammars. The importance of some results obtained during the investigation of L-systems is purely theoretical, on the other hand, some other results lead to various applications. At present, L-systems are use not only for biological models, but also in other areas. The most common use of them is for creating of developing models such as fractals in computer graphics, models of plants[13] or neural networks[33]. Some models, such as neural networks, can be expressed as graphs. Thus we can consider L-systems to be graph grammars too.

L-system have several important properties, which they are used for. In particular, it is their parallelism and fractal character of words of some their languages.

This thesis discuses L-systems from two viewpoints.

1. Theoretically, we discusses L-systems as a generators of languages. Specifically, the exponential density of languages, reduction of context-sensitive L-systems and two extensions and modifications of interactive L-systems are studied here.

2. From a practical point of view, we investigate L-systems as formalism for developing models. Interpretation of words generated by L-systems, especially interpretation as graphs with skeleton and chords, is discussed here. A programming language for L-systems is introduced here.

The PhD thesis is organized as follows. Part 1 contains the mathematical background of this work. It begins with section Notation, where the used symbols are defined. This part is divided into Chapters 1–3. Chapter 1 contains the necessary definitions and theorems concerning the formal language theory which are important for the definitions and results obtained in Chapters 4, 7 and 8. Chapter 2 defines L-systems and their extensions. It also includes theorems concerning the generating power of L-systems. This chapter forms a mathematical background for investigation in terms of both theory and application. Chapter 3 defines graphs and graph grammars that were studied.

Part 2 forms a heart of this work in terms of formal the language theory. Section 4.1 proves new theorem, which can be used to decide if a given language is exponentially dense. Corollaries demonstrating how to use this theorem are given. Specifically, we show how to prove that the classes of POL languages and pure recursively enumerable languages are exponentially dense. A reduction of ETIL-systems is studied in

Sections 4.2 and 4.3. Section 4.4 defines a controlled sequential grammar systems and proves that these systems generate the same languages as EIL-systems—that is the class of languages generated by controlled sequential grammars is equal to the class of recursively enumerable languages. Section 4.5 defines interactive L-systems with semi-free context. They are generalization of interactive L-systems and they allow to expressed the dependence of application of a production on the other symbols in the current word, regardless of if the symbols neighbour or not with the symbols being rewritten.

Part 3 overviews well-known applications of L-systems. It is formed with Chapters 5–6. Chapter 5 describes applications of L-systems in various areas. Particularly, it shows applications in the computer graphics for modeling of biological organisms. Chapter 6 introduces some generators of L-systems and high-level languages based on L-systems.

Part 4 closes this thesis. It is divided into Chapters 7 and 8. Chapter 7 formally studies interpretation of words generated with L-systems with turtle systems and discuses interpretation of L-systems as graph grammars with a skeleton and chords. New high-level programming language is introduced in Chapter 8. This language is designed in order to allow the user to describe various developing models as an L-systems and define the interpretation of the terminal string. The syntax and the semantics of the language is defined with respect to our investigation.

This thesis brings new results both in the theory and applications of L-systems. The theoretical investigation of L-systems and investigation of their application are not two separate areas. They are connected in the proposed programming language.

2 PRELIMINARIES

The present section recall some basic notions and introduces convention used henceforth. For other notions used in their formal language theory, consult [1] and for other information about L-system consult [3].

For an alphabet, Σ, Σ^* denotes the free monoid generated by Σ under operation of concatenation; its unit is called the empty word and denoted by ε. For every $\alpha \in \Sigma^*$, $|\alpha|$ denotes the length of α, $\overline{\alpha}$ denotes the reversion of α.

0-type grammar is a quadruple $G = (\Sigma, P, S, \Delta)$, where Σ is an alphabet, $\Delta \subset \Sigma$ is an alphabet of terminal symbols, $P \subseteq \Sigma^+ \times \Sigma^*$ is a finite set of productions and $S \in \Sigma - \Delta$ is the starting symbol. We define relation *derivation* $\Rightarrow \in (\Sigma^+ \times \Sigma^*)$ as follows: $\alpha \Rightarrow \beta$ if and only if $\alpha \in \Sigma^+$, $\beta \in \Sigma^*$, $\alpha = xuy$, $\beta = xwy$, $w.x, y \in \Sigma^*$, $u \in \Sigma^+$and $(u \rightarrow w) \in P$. The transitive closure of the relation of direct derivation is denoted by $\Rightarrow^+$ and the transitive and reflexive closure is denoted by $\Rightarrow^*$. The language generated by G is defined by

$$\mathcal{L}(G) = \{\alpha \in \Delta^* | S \Rightarrow^* \alpha\} \ .$$

Languages generated by 0-type grammar are called recursively enumerable. $\mathbb{RE}$ denotes the class of all recursively enumerable languages. A given language $\mathcal{L} \in$

$\mathbb{RE}$ is said to be *pure recursively enumerable* language, or *pure languag*e in short, if there exists a 0-type grammar $G = (\Sigma, P, S, \Sigma)$, that is a 0-type grammar without nonterminals, such that $\mathcal{L} = \mathcal{L}(G)$. The class of pure languages is denoted by $\mathbb{PL}$.

2-type grammars, also called context free grammar, are restricted 0-type grammars where $P \subseteq (\Sigma - \Delta) \times \Sigma^*$. Languages generated by 2-type grammars are called context-free and the class of context-free languages is denoted $\mathbb{CF}$. $\mathbb{CF}$ is a proper subset of $\mathbb{RE}$.

It can be proved (see [6]) that for every language $\mathcal{L} \in \mathbb{RE}$ there are two languages $\mathcal{L}_1, \mathcal{L}_2 \in \mathbb{CF}$ and homomorphism h such that $\mathcal{L} = h(\mathcal{L}_1 \cap \mathcal{L}_2)$. This theorem plays crucial role in some proofs in the thesis.

L-systems are parallel grammar. The basic L-system, denoted OL-system, is defined as a triple

$$G = (\Sigma, P, \omega)$$

where Σ is an alphabet, $\omega \in \Sigma^+$ is a starting word called axiom and $P \subseteq \Sigma \times \Sigma^*$ is a set of productions of the form $a \to \alpha$, where $a \in \Sigma$ and $\alpha \in \Sigma^*$. Definition of relation of direct derivation $\Rightarrow \subseteq (\Sigma^+ \times \Sigma^*)$ is different from sequential grammars: $\alpha \Rightarrow \beta$ if and only if $\alpha \in \Sigma^+$, $\beta \in \Sigma^*$, $\alpha = a_1 a_2 \dots a_k$, $\beta = \gamma_1 \gamma_2 \dots \gamma_k$, $k = |\alpha|$ and for every integer $1 \leq i \leq k$, $a_i \to \gamma_i$ is a production from P. $\Rightarrow^+$ or $\Rightarrow^*$ denote the transitive or the transitive and reflexive closure of the relation $\Rightarrow$, respectively. Language generated by 0L-systems is defined as

$$\mathcal{L}(G) = \{\alpha \in \Sigma^* | \omega \Rightarrow^* \alpha\}$$

Example 2.0.1 *Example of 0L-systems:* $G = \left(\{a, b\}, \left\{a \to b^2, b \to a^2\right\}, aba\right)$. *First derivation steps are:*

$$aba \Rightarrow bbaabb \Rightarrow aaaabbbbaaaa \Rightarrow bbbbbbbbaaaaaaaabbbbbbbb$$

$\mathcal{L}(G) = \left\{a^{2k-1}b^{2k-1}a^{2k-1}\right\} \cup \left\{b^{2k}a^{2k}b^{2k}\right\}$ *for every* $k \geq 1$.

There are various extension of the former L-systems. ETOL-system and ETIL system are important for this thesis. They are defined now.

An ETOL-system is a quadruple $G = (\Sigma, H, \omega, \Delta)$ where Σ is an alphabet, $\omega \in \Sigma^+$ is an axiom, $\Delta \subseteq \Sigma$ is a terminal alphabet of G and for each $P_i \in H$ where $1 \leq i \leq \|H\|$ $G_i = (\Sigma, P_i, \omega)$ is a E0L-system. Relation of direct derivation $\Rightarrow \subseteq \Sigma^+ \times \Sigma^*$ is defined as follows. $\alpha \Rightarrow \beta$ if and only if there is such 0L-system G_i such that $\alpha \Rightarrow_{G_i} \beta$. The language of G, denoted by $\mathcal{L}(G)$, is defined by

$$\mathcal{L}(G) = \{\alpha \in \Sigma^* | \omega \Rightarrow^* \alpha\} \cap \Delta^*.$$

Definition 2.0.2 *Let* $m, n \in \aleph_0$. *An (m,n)L-system is a triple* $G = (\Sigma, P, \omega)$, *where* Σ *is an alphabet,* $\omega \in \Sigma^+$ *is the axiom, and* P *is a mapping of the set* $\bigcup_{i=0}^{m} \Sigma^i \times \Sigma \times \bigcup_{i=0}^{n} \Sigma^i$ *into the set of all nonempty finite subsets of* Σ^*. *If a word* $\gamma \in P(\alpha, a, \beta)$ *we write* $(\alpha, a, \beta) \to \gamma$ *and call it a* production. *A word* $\alpha \in \Sigma^+$ *directly derives a word*

$\beta \in \Sigma^*$ in G, written as $\alpha \Rightarrow \beta$ if $\alpha = a_1 \dots a_k$, $\beta = \beta_1 \dots \beta_k$, $k \geq 1$ is an integer, $a_i \in \Sigma$,$\beta_i \in \Sigma^*$ and for all $i = 1, \dots, k$,

$$(a_{i-m}a_{i-m+1} \dots a_{i-1}, a_i, a_{i+1} \dots a_{i+n}) \rightarrow \beta_i \tag{2.1}$$

is a production of G. In 2.1 we define $a_j = \varepsilon$ whenever $j \leq 0$ or $j > k$.

As usual, $\Rightarrow^+$ and $\Rightarrow^*$ are transitive and transitive and reflexive closure of the relation $\Rightarrow$, respectively. The language generated by (m,n)L-systems G is defined $\mathcal{L}(G) = \{\alpha \in \Sigma^* | \omega \Rightarrow^* \alpha\}$. The class of languages generated by (m,n)L-systems for given m, n is denoted $(\mathbf{m}, \mathbf{n})\mathbb{L}$.

(m,n)L-systems are also called IL-systems (I for *interactive*). The class of languages generated by IL-systems is $\mathbb{IL} = \bigcup_{m=0,n=0}^{\infty,\infty}(\mathbf{m}, \mathbf{n})\mathbb{L}$.

(1,0)L-systems and (0,1)L-systems are called one-sided L-systems and denoted as 1L-systems. (1,1)L-systems are denoted as 2L-systems.

The previous interactive L-systems where pure, that is without nonterminal. Because all words derived from the axiom belong to $\mathcal{L}(G)$, context cannot be reduced by simulation. Thus, IL-systems defines infinitely many language classes which are either incomparable or one is a proper subset of another.

By analogy with 0L-systems, EIL-systems are IL-system with nonterminals. They are defined now.

Definition 2.0.3 *An EIL-system is a quadruple $G = (\Sigma, P, \omega, \Delta)$ where $U(G) = (\Sigma, P, \omega)$ is a IL-system (called underlaying system of G) and $\Delta \subseteq \Sigma$ is a terminal alphabet of G. The language of G is $\mathcal{L}(G) = \mathcal{L}(U((G)) \cap \Delta^*$. The class of all languages generated by EIL-systems is denoted by $\mathbb{EIL} = \bigcup_{m=0,n=0}^{\infty,\infty} \mathbb{E}(\mathbf{m}, \mathbf{n})\mathbb{L}$.*

EIL-systems without erasing productions (ε-productions) are called propagating EIL-systems and denoted by EPIL. By analogy, $\mathbb{EPIL}$ is a class of all languages generated by EPIL-systems.

Theorem 2.0.4 $\mathbb{EIL} = \mathbb{E}\mathbf{2}\mathbb{L} = \mathbb{E}(\mathbf{1}, \mathbf{0})\mathbb{L} = \mathbb{E}(\mathbf{0}, \mathbf{1})\mathbb{L} = \mathbb{RE}$.

3 THEORETICAL INVESTIGATION OF L-SYSTEMS

This chapter is a theoretical heart of this work. It includes original investigation of exponential density, reduction of context productions, transformation of L-systems and definition of L-systems as graph grammars. The organization of this chapter is as follows. Section 3.1 proves new theorem, which can be used to decide if a given language is exponentially dense. Application of the this theorem is demonstrated in several corollaries. A reduction of ETIL-systems is studied in Sections 3.2 and 3.3. Section 3.4 defines a controlled sequential grammar systems and proves that these systems generate the same languages as EIL-systems—that is the class of languages generated by controlled sequential grammars is equal to the class of recursively enumerable languages. Section 3.5 defines interactive L-systems with semi-free context as a generalization of interactive L-systems.

3.1 Exponential Density

Let $\mathcal{L}$ be a language. $\mathcal{L}$ is called *exponentially dense* if there exist positive constants c_1 and c_2 having following property: for any $n \geq 1$ there exists a string $x \in \mathcal{L}$ such that $c_1 e^{(n-1)c_2} \leq |x| < c_1 e^{nc_2}$.

The class of all exponentially dense languages is denoted by $\mathbb{ED}$.

The exponential density can be used in proofs of that some languages are out of a language familly. Let $\mathcal{L}$ be a language and $\mathbb{CL}$ be a class of languages. If $\mathcal{L} \notin \mathbb{ED}$ and $\mathbb{CL} \subseteq \mathbb{ED}$, then $\mathcal{L}$ cannot be in $\mathbb{CL}$. E. g. Since language $\mathcal{L} = a^{2^{2^n}}$ is not exponentially dense and all infinite ETOL languages are exponentially dense (see [7]), $\mathcal{L}$ cannot be an ETOL language.

3.1.1 Results

The PhD thesis proves following theorem.

Theorem 3.1.1 *Let L be a infinitive language, $\mathcal{H} \subseteq \mathcal{L}$, and $R \subseteq \mathcal{L} \times \mathcal{L}$. If there exists a constant $d > 1$ such that for every $v \in \mathcal{L}$ there is $w \in \mathcal{H}$ such that $v\, R^*\, w$ with $|v| < |w| \leq d\,|v|$, then every language $\mathcal{M} \supseteq \mathcal{H}$ is exponentially dense.*

This theorem can be used to prove a given language is exponentially dense. Example of corollaries of this theorems are

Corollary 3.1.2 *Every infinite OL language is exponential dense.*

Corollary 3.1.3 *Every infinitive pure recursively enumerable language is exponential dense.*

Corollary 3.1.4 *$\mathbb{PL} \subset \mathbb{RE}$.*

Corollary 3.1.5 *Every infinite language $\mathcal{L}$, such that $\mathcal{L} \in \mathbb{ETOL}$, is exponentially dense.*

Corollary 3.1.6 *There are exponentially dense context-sensitive languages that are not ETOL languages.*

All corollaries are proved in the thesis. Corollary 3.1.5 was proved by Meduna in [7]. The proof presented in the thesis is a simplification of the original proof using Theorem 3.1.1.

3.2 Modified Version of EIL systems and Their Reduction

This paper introduces a modified version of EIL-systems. This modification consists in allowing context-free productions to be used inside of sentential forms. The present paper proves that these modified EIL-systems with no more than 12 context-sensitive productions are as powerful as ordinary EIL-systems. That is they define the family of type-0 languages (see [3, page 286]).

3.2.1 Definition

($mE(m,n)L$) This modification is based on [13].

Let m, n be two non-negative integers, Σ be an alphabet, $\Delta \subseteq \Sigma$ be an alphabet of terminals and $\omega \in \Sigma^*$. A modified interactive L-system, mE(m,n)L-system in short, is a quadruple

$$G = (\Sigma, P, \omega, \Delta)\,,$$

where ω is the axiom of G and P is a finite set of productions of form

$$(\alpha, a, \beta) \to \gamma\,,$$

where $a \in \Sigma$, $\alpha \in \Sigma^i$ for some $i \in \{0, 1, \ldots, m\}$, $\beta \in \Sigma^j$ for some $j \in \{0, 1, \ldots, n\}$, $\gamma \in \Sigma^*$.

Let $x = a_1 a_2 \ldots a_k$ where $a_h \in \Sigma$ for any $1 \leq h \leq k$, $k = |x|$, $y = \gamma_1\gamma_2 \ldots \gamma_k$, where $\gamma_h \in \Sigma^*$ for any $1 \leq h \leq k$. Write $x \Rightarrow y$ if for every a_h for $h = 1, \ldots, k$ there is a production of the form $(\alpha_h, a_h, \beta_h) \to \gamma_h$ in P where $\alpha_h \in suffix(a_1 \ldots a_{h-1})$ such that $|\alpha_h| \leq m$ and $\beta_h \in prefix(a_{h+1} \ldots a_k)$ such that $|\beta_h| \leq n$.

The language of G, denoted by $\mathcal{L}(G)$, is defined by

$$\mathcal{L}(G) = \{\alpha \in \Sigma^* | \omega \Rightarrow^* \alpha\} \cap \Delta^*\,.$$

Let $p \in 0, \ldots, m$ and $q \in 0, \ldots, n$. The number of productions of the form

$$(\alpha, a, \beta) \to \gamma\,,$$

form P, where $a \in \Sigma$, $\alpha \in \Sigma^p$, $\beta \in \Sigma^q$, $\gamma \in \Sigma^*$ is denoted $\#_{p,q}P$. Obviously $\sum_{i=0}^{m}\sum_{j=0}^{n} \#_{i,j}P = \|P\|$.

If $((\varepsilon, a, \varepsilon) \to \gamma) \in P$, where $a \in \Sigma$ and $\gamma \in \Sigma^*$, we write simply $a \to \gamma$ instead of $(\varepsilon, a, \varepsilon) \to \gamma$.

mET(m,n)L-systems are mE(m,n)L-systems with one or more sets of production. They are analogy to ETOL-system.

3.2.2 Main Results

Theorem 3.2.1 *For every EIL-system $G = (\Sigma, P, \omega, \Delta)$ there exist an equivalent mE(1,0)L-system $G_m = (\Sigma_m, P_m, \omega_m, \Delta)$ such that $\mathcal{L}(G) = \mathcal{L}(G_m)$.*

Theorem 3.2.2 *For every mE(1,0)L-system $G = (\Sigma, P, \omega, \Delta)$ there exists a mET(1,1)L-system $G' = (\Sigma', H', \omega, \Delta)$ where $H = \{h_1, h_2, h_3, h_4\}$ and $\sum_{i=1}^{4}(\#_{1,0}h_i + \#_{0,1}h_i) = 12$, that is H contains only 12 context sensitive productions, such that $\mathcal{L}(G) = \mathcal{L}(G')$.*

Both theorems are proves in the thesis.

3.3 Complexity of ETIL-systems

This section proves that the class of ETIL-systems with only six context productions is equal to $\mathbb{RE}$. The proof is inspired by [12] and [11].

Theorem 3.3.1 *For every recursively enumerable language $\mathcal{L}$ there exists a ETIL-system $G = (\Sigma', H, \omega', \Delta')$ such that H contains no more than six context productions.*

This theorem is proved in the thesis.

3.4 Controlled Grammar System for L-systems

3.4.1 Definitions

Definition 3.4.1 *A* controller *C is a sextuple $C = (Q, \Sigma, \Pi, Q_0, t, o)$ where Q is a set of states, Σ is a set of input symbols (input alphabet), Π is a set of output symbols, called output alphabet, $Q_0 \subseteq Q$ is a set of starting states, $t : (Q \times \Sigma) \to Q$ is a transition function and $o : (Q \times \Sigma) \to \Pi$ is an output function. Q can be infinite.*

Definition 3.4.2 *A* controlled grammar system *is a tuple $\Gamma = (\Sigma, \Delta, \omega, C, P_1, \ldots, P_k)$ where Σ is an alphabet, $\Delta \subset \Sigma$ is an alphabet of terminal symbols, $\omega \in \Sigma^+$ is a starting word (axiom), $C = (Q, \Sigma - \Delta, \{1, \ldots, k\}, Q_0, t, o)$ is a controller and $G_i = (\Sigma, P_i, S, \Delta)$ for $i = 1 \ldots k$ are context-free grammars providing left-most derivations only, $k \in \aleph$.*

By analogy with automata define configuration.

Let $\Gamma = (\Sigma, \Delta, \omega, C, P_1, \ldots, P_k)$ be a controlled grammar and

$$C = (Q, \Sigma - \Delta, \{1, \ldots, k\}, Q_0, t, o)$$

be a controller. We say that a couple $(\alpha, q) \in \Sigma^* \times Q$ where α is a word and q is a current state, is a configuration *of controlled grammar.*

Relation of direct derivation $\Rightarrow \in (\Sigma^* \times Q) \times (\Sigma^* \times Q)$ is defined with analogy to transition in automata. $(\alpha, q_i) \Rightarrow (\beta, q_{i+1})$ if

1. *$A \in \Sigma - \Delta$ is the leftmost nonterminal symbol,*
2. *$\alpha \Rightarrow_{G_{o(q_i, A)}} \beta$*[1] *and*
3. *$q_{i+1} = t(q_i, A)$.*

The language generated by controlled grammar system is defined as

$$\mathcal{L}(\Gamma) = \{\alpha \in \Delta^* | (\omega, q_0) \Rightarrow^* (\alpha, q_k)\}$$

where $q_0 \in Q_0$ and $q_k \in Q$.

3.4.2 Main results

Theorem 3.4.3 *Let $\mathcal{L} \in \mathbb{RE}$. Then there exists a controlled grammar system Γ such that $\mathcal{L}(\Gamma) = \mathcal{L}$.*

[1] As usual, notation $\alpha \Rightarrow_{G_x} \beta$ means α directly derives β in grammar G_x. q_i is a current state of the controller, $x = o(q_i, A)$ is a value of the output function for current state and the leftmost nonterminal symbol. The output function select grammar that is used to derive α.

3.5 Interactive L-systems with Semi-free Context

Definition 3.5.1 *An interactive L-systems with semi-free context (SFCIL-system in short) is an octad*

$$G = (\Sigma, Q, P, \omega, \Delta, \lambda, \rho, \heartsuit)$$

where Σ is an alphabet, $\Delta \subseteq \Sigma$ is a terminal alphabet, $\omega \in \Sigma^+$ is the axiom, Q is a, possibly infinite, set of messages, $\heartsuit \in Q$ is an empty message, $\lambda, \rho : \Sigma \times Q \to Q$ are flow functions, P is a finite set of production of the form $a \to \alpha, \Pi(l, r)$ where $a \in \Sigma$, $\alpha \in \Sigma^*$ and $\Pi(l, r)$ is a predicate with $l, r \in Q$.

Let $x = a_1 a_2 \dots a_n$ be a word, where $n = |x|$ and $a_i \in \Sigma$ for every $i = 1 \dots n$. Define $l_1 = \heartsuit$, $r_n = \heartsuit$, $l_{i+1} = \rho(a_i, l_i)$, $r_i = \lambda(a_{i+1}, r_{i+1})$ for every $i = 1 \dots n-1$. Word x directly derives word $y = \alpha_1 \alpha_2 \dots \alpha_n$, where $\alpha_i \in \Sigma^*$ for every $i = 1 \dots n$, if for every $i = 1 \dots n$ there is a production $(a_i \to \alpha_i, \Pi(l_i, r_i))$ such that $\Pi(l_i, r_i)$ is true. We write $x \Rightarrow y$.

The language generated by SFCIL-system G is defined as

$$\mathcal{L}(G) = \{x \in \Delta^* | \omega \Rightarrow_G^* x\}$$

The class of languages generated by SFCIL-system is denoted $\mathbb{SFCIL}$.

3.5.1 Results

Theorem 3.5.2 $\mathbb{SFCIL} = \mathbb{RE}$.

SFCIL-systems are suitable for such models based on L-system where some part directly depends on some other part and these parts do not neighbours. Such as models are models of a group of organisms.

4 APPLICATIONS OF L-SYSTEMS

Przemyslaw Prusinkiewicz and his group have studied application of L-system in computer graphics, especially for modeling of plants [13, 14]. Their investigation comes out investigation of interpretation of L-systems in computer graphics, modular structure of plants, recognition of the fractal character of plants and fractal character of L-systems, such as locally catenative formulas in [3]. L-systems are used as a language describing developing models, where each word that belongs to the language, is treated as a formal description of a developmental stage of the model.

Although the generative power of EIL-systems is equal to the power of Turing machines, i. e. $\mathbb{EIL} = \mathbb{RE}$, various extension of L-systems were introduced. These extensions increase the range of phenomena that can be modeled using L-system. Such as extensions are parametric L-systems (see [13]) or open L-system (see [15]).

Words generated by an (parametric) L-systems are interpreted by a logo-style[39] turtle in computer graphics. Turtle interpretation was introduced by Szilard and Quinton in [16] and extended by Prusinkiewicz (see [14, 13]) and Hannan (in [17]). A turtle is a object that interprets modules and creates the target object according to the module being interpreted and a current state of the turtle.

Although L-system were originally introduced as formalism for description of the development of metazoan organisms, they are used in various areas, today. Application of L-systems as grammars describing parallel generation of graphs was mentioned in [33, 32], program LMuse [42] uses L-system to generate a fractal music. We mention here application of L-system for generation of fractal music and for symbolic computation. Goel and Shen extended L-systems in [22] for symbolic computation.

5 INVESTIGATION OF INTERPRETATION OF L-SYSTEM

L-systems are usually process in two stages. During first stage is a final word derived from the axiom in a given number of steps. In the second stage, this word is consider to be a program, where each symbol is a command of an interpret. This section studies the second stage of L-system interpretation. It introduces turtle system, discuses pipelining in processing of L-systems and studies interpretation of chords.

5.1 Abstract Model of Turtle

The word generated by an L-system are interpreted so called turtle. The studied papers use the turtle but there is no formal definition of it. This thesis defines the turtle by analogy with automata. . The turtle is defined as a pushdown automaton (or transducer) with infinitive state space rather then finite set of states.

Definition 5.1.1 *A* turtle *is a septuple*

$$T = (Q, \Sigma, \Gamma, \delta, q_0, C, \rho)$$

where Q is a state space *of the turtle, Σ is an L-system alphabet, Γ is a pushdown alphabet with $\spadesuit \in \Gamma$, $\delta : (Q \times \Sigma \times \Gamma) \rightarrow (Q \times \Gamma^*)$ is a transition function, $q_0 \in Q$ is a start state, C is a output set of state-independent commands and $\rho : (Q \times \Sigma \times \Gamma) \rightarrow C^*$ is the output function. Note that the state space and pushdown alphabet can be, unlike pushdown automata, infinite. Because Q and Γ can be infinitive the transition function must be defined by expressions, like real functions.*

The state space and the transition function depend on the purpose of the L-system.

A *configuration* of the turtle and *relation* $\vdash$ are defined by analogy with pushdown automaton.

Definition 5.1.2 *Let $T = (Q, \Sigma, \Gamma, \delta, q_0, C, \rho)$ be a turtle. We say T* properly accepts *a word $\alpha \in \Sigma^*$ if $(\spadesuit, s, \alpha) \vdash^* (\spadesuit, f, \varepsilon)$ and $f \in Q$.*

The class of all languages accepted by push-down automata is equal to Class $\mathbb{CF}$, the class of languages accepted by finite automata is even only $\mathbb{REG}$. Here, we describe interpretation of L-systems with push-down automata. But the classes of languages generated by L-systems are either incomparable with $\mathbb{CF}$, or $\mathbb{CF}$ is their proper subset. For example, $\mathbb{OL}$ is incomparable with $\mathbb{CF}$, $\mathbb{CF} \subset \mathbb{ETOL} \subset \mathbb{EIL} = \mathbb{RE}$. We want to process languages that are not in $\mathbb{CF}$ with push-down automata. Is not it a contradiction? Yes, it can look like. But we do not want to accept languages generated by

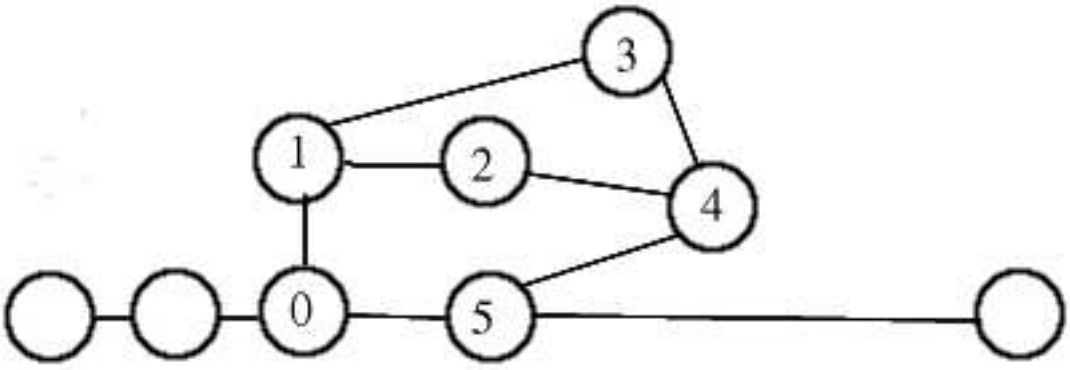

Figure 5.1: Graph representation of word
$EE[(0)E[(1)E](2)E](3)E\{(2)E\{(3)E\}(4)E\}(5)E$

L-system with a push-down automata. We obtain a word α as a result of a derivation of the axiom in a given L-system G. It ensures $\alpha \in \mathcal{L}(G)$. The sequence of symbols in α is treated as a sequence of actions needed to create the target object. Push-down automata are not used as recognizers, but as state-controlled interprets.

The thesis discusses interpretation of words by single turtle and defines a turtle system for parallel interpretation of words generated by an L-system.

5.2 Generation of General Continuous Multigraphs

Remark that only trees can be generated with bracket L-system. A tree $F_s(V, E, W)$ can be considered to be a skeleton of some continuous graph $F_c(V, E', W')$, where $E \subseteq E'$ and $W \subseteq W'$. Graph F_c is obtained from F_s by adding chords to skeleton F_s[28, page 103].

The definition of a graph is extended with node labels. A labeled graph is a quadruple $F = (V, E, W, \lambda)$ where V, E, W are set of nodes, set of edges and a morphisms assigning source and target node to every edge, respectively. $\lambda : \aleph \to V$ is a morphism such that it assign a node $n \in V$ to each label $l \in \aleph$. If some $l \in \aleph$ is not a label of any node then $\lambda(l)$ is not defined.

Extend bracket L-systems by chord support. Add symbols {,} into Σ. The target word will be parametric and will be generated by a parametric L-system. Every node, where more than two edges meet (these nodes are denoted by [) and leaves (denoted by]) will be indexed with unique number as a parameter. This number is a label of such noes. A chord will be expressed as a substring $\{(i)t\}(j)$, where i and j are labels of nodes. An interpret labels last node with j and create a subgraph F_x denoted by word t such that root of F_x will be node $\lambda(i)$ and last leaf of F_x will be node $\lambda(j)$. Remark word t does not represent an edge—the chord—but a graph, that replaces the chord. Such graph that replaces the chord is called an *expanded chord*.

Example 5.2.1 *Word $EE[(0)E[(1)E](2)E](3)E\{(2)E\{(3)E\}(4)E\}(5)E$ represents a graph that is shown at Figure 5.1*

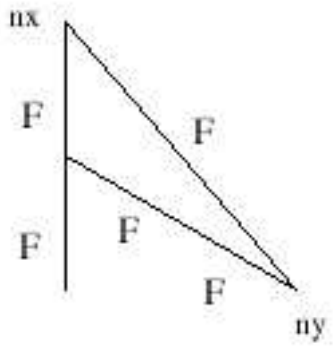

Figure 5.2: Graphical representation of string
$F[++FF](x)F\{(x)F\}(y)$

The thesis defines a turtle such that it generates general continuous graphs. The thesis also contains a proof that any continuous graph can be expressed as a string of the form above.

5.3 Interpretation of Chords

A chord is an edge connecting two nodes created by two (separate) turtle, say t_x and t_y. We say a chord connects turtles t_x and t_y. A connection of turtles t_x and t_y is a drawing of the expanded chord. Interpretation of a expanded chord is a interpretation of a substring $\{(x)u\}(y)$.

If we look at a graph from topological point of view, we do not take care of interpretation of edges and their graphical representation. For example there are infinitely many drawings of any (multi)graph.

A problem can occur when a string is interpreted by a turtle system. Consider substring $\{(y)u\}(x)$ and turtles t_x, t_y such that state of t_x is a just the state at the node n_x and state of t_y is a just the state at the node n_y. Intuitively, substring u should be interpreted to connect nodes n_x and n_y by a curve. It can be impossible to obtain this curve by standard interpretation of u in some cases. For example, consider $\alpha = F[++FF](x)F\{(x)F\}(y)$ and $q_0 = (\overrightarrow{s_{p0}}, \overrightarrow{s_{h0}}, s_l, s_a) = ((0,0),(0,1),1,\frac{\pi}{3})$. Graphical representation of the string is shown at figure 5.2. Turtle t_y reaches the point n_y and its state is $s_y = \left(\left(\sqrt{3},0\right), \left(\frac{\sqrt{3}}{2}, -\frac{1}{2}\right), 1, \frac{\pi}{3}\right)$, the main turtle reaches point n_x and its state is $s_x = \left((0,2),(0,1),1,\frac{\pi}{3}\right)$. A chord must be drawn between points n_x and n_y. If we directly interpret the chord code from the state s_x, the point n_y cannot be reached. Furthermore the distance of these points $\|n_x - n_y\| = \sqrt{\left(\sqrt{3},-2\right)\left(\sqrt{3},-2\right)} = \sqrt{13} > 1$ and command F draws only unit-length lines.

One can see that the state of turtle $t_x$drawing the chord must be modified to ensure that this turtle meats the $t_y$in point n_{y}. Define morphism $T \subseteq (Q \times Q \times \Sigma^*) \times Q$ such that $(\spadesuit, T(s_x, s_y, u), u) \vdash^{|u|} = (\spadesuit, s_e, \varepsilon) \wedge s_{e1} = s_{y1}$ where u is a string representing

the chord and the index 1 means the first component of a state, that is the position of the turtle. Informally, morphism T modifies the state of turtle t_x to ensure the position of turtle t_x after interpretation of u is equal to the position of turtle t_y.

5.4 Example of Transformation in 2D

Have a look to two dimensional L-systems and 2D transformation. A point (x, y) is usually expressed as a three-element vector $(x, y, 1)$ (see [25]). Call it *extended coordinates*.Transformation of extended coordinates is defined by transformation matrix

$$T = \begin{pmatrix} a_{11} & a_{12} & 0 \\ a_{21} & a_{22} & 0 \\ m_x & m_y & 1 \end{pmatrix}$$

where (m_x, m_y) is a translation vector and sub matrix $\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$ is a rotation matrix. An image $(x', y', 1) = (x, y, 1)T$. Drawing of a chord is completed by turtle t_x from its current state s_x. This drawing will be transformed to fix it between nodes n_x and n_y. This transformation uses transformation matrix T which have to fulfill following conditions:

$\overrightarrow{p_1}T = \overrightarrow{p_1}$

$\overrightarrow{p_2'}T = \overrightarrow{p_2}$

where $\overrightarrow{p_1}$ is extended coordinate of n_x, $\overrightarrow{p_2}$ is extended coordinate of n_y and $\overrightarrow{p_2'}$ is extended coordinate of point which is reached by t_x after interpretation of a chord. To ensure definiteness of transformation matrix T, three points which does not lie in a line and their images are needed. We need another point $\overrightarrow{p_4}$ and its image $\overrightarrow{p_5}$ such that $\overrightarrow{p_4}T = \overrightarrow{p_5}$. Furthermore it is expected the transformation is "homogeneous" in the sense of scale. It means that for every three points with extended coordinates $\overrightarrow{r_1}$, $\overrightarrow{r_2}$ and $\overrightarrow{r_3}$ and their images $\overrightarrow{r_1}'$, $\overrightarrow{r_2'}$ and $\overrightarrow{r_3'}$

$$(\|\overrightarrow{r_1} - \overrightarrow{r_2}\| = n) \wedge (\|\overrightarrow{r_1} - \overrightarrow{r_3}\| = m) \Leftrightarrow \left(\left\|\overrightarrow{r_1'} - \overrightarrow{r_2'}\right\| = kn\right) \wedge \left(\left\|\overrightarrow{r_1'} - \overrightarrow{r_3'}\right\| = km\right)$$

where n, m, k are real numbers.

Vectors $\overrightarrow{p_3}$ and $\overrightarrow{p_3'}$ can be defined as follows:

$\overrightarrow{p_3} = \overrightarrow{p_1} + \perp(\overrightarrow{p_2} - \overrightarrow{p_1})$

$\overrightarrow{p_3'} = \overrightarrow{p_1} + \perp\left(\overrightarrow{p_2'} - \overrightarrow{p_1}\right)$

where $\perp\overrightarrow{s} = \perp(s_1, s_2, 0) = (s_2, -s_1, 0)$. To explain the above, $\perp\overrightarrow{s}$ is a vector which is orthogonal to $\overrightarrow{s}$ and has the same length as $\overrightarrow{s}$.

Matrix T is a solution of equalationequationequation

$$\begin{pmatrix} \overrightarrow{p_1} \\ \overrightarrow{p_2'} \\ \overrightarrow{p_3'} \end{pmatrix} T = \begin{pmatrix} \overrightarrow{p_1} \\ \overrightarrow{p_2} \\ \overrightarrow{p_3} \end{pmatrix}$$

6 PROGRAMMING LANGUAGE BASED ON L-SYSTEM

A new programming language is designed on the basis of investigation of L-systems from the viewpoints of the theory and interpretation. The language is a semi-functional programming language with weak type system.

The syntax of the language allows the programmer use two approaches. The model can be based on interactive L-system with semi-free context (see 3.5) or as controlled grammar system (3.4). The second approach allows to pipeline generation of the model and its interpretation.

The language uses identifiers rather then one-letter symbols. Productions are consider to be functions and the syntax of definition of „productions" and definition of terminal is unique, in opposite to other generators, like [34, 35]. All functions can have parameters, so it is easy to created a model based on parametric L-systems [13]. The set of terminals is not fixed, as it is in common free generators of L-systems. The user can define new terminals or modified behaviour of the built-ones.

The generated parametric word is interpreted by a turtle system. This turtle system uses built in state space which can be extended with any number of new dimensions. Furthermore, the proposed language supports branched L-systems with chords, defined in section 5.2.

Because SFCIL-systems and controlled grammar systems define the class of recursively enumerable languages, the interpret uses an extensible state space and an extensible set of terminals and the branched L-systems with chords are supported in the language, the proposed language can be used in various applications.

CURRICULUM VITAE

Name Milan Kolka

Date of birth 23^{rd} May 1975

Place of birth Havlíčkův Brod, Czech republic

Residence Zikova 26, Brno

E-mail milan@milankolka.cz

Occupation Civil service (since October 2003)

1994 Grammar School Chotěboř, district Havlíčkův Brod

1999 Brno University of Technology, Faculty of Electrical Engineering and Computer Science. Master degree at Information technology and computer science (Ing.)

1999-2001 Doctoral study at Department of Biomedical Engineering at Brno University of Technology, Faculty of Electrical Engineering and Computer Science. Research at segmentation of ultrasound images. Not completed

2001-2003 Doctoral study at Faculty of Information Technology at Brno University of Technology. Research at reduction and application of L-systems

1997-1999 EZ Praha, communication software for power station Ledvice

1999 Moravia translations Brno, software tester

2001-2002 ApS Brno, instructor of Java

since 2003 Civil service at Faculty of Law, Brno

1. Kolka Milan: Multiturtle L-system Interpretation, In: Proceedings of 6th International Conference ISIM 03 Information System Implementation and Modeling, Ostrava, CZ, MARQ, 2003, p. 179-186, ISBN 80-85988-84-4
2. Kolka Milan: Using CSP for Modeling of an Environment in a Programming Language Based on L-systems, In: Proceeding of 9th Conference and Competition Student EEICT 2003, Brno, CZ, FEKT VUT, 2003, p. 593-597, ISBN 80-214-2379-X
3. KOLKA, M. Spojování želv v jazyce založeném na L-systémech. Elektrorevue - Internetový časopis (http://www.elektrorevue.cz), ISSN 1213-1539, 2001, roč. 2001, č. 1,
4. KOLKA, M., PROVAZNÍK, I. Connecting of Turtles in a Programming Language Based on L-Systems In Proceedings of 7th Conference Student FEI 2001. Konference a soutěž STUDENT FEI 2001. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a informatiky, 2001, s. 244 - 248, ISBN 80-214-1860-5

5. Kolka, M., Provazník I: Programming Language Based on L-systems, Volume of the papers written by students and postgraduate students, VI. (2000), pages 64 to 66, Faculty of Electrical Engineering and Computer Science BUT, Brno, 2000

ABSTRACT

Lindenmayer systems, L-systems in short, where introduced in 1968 as a formalism for description of evolution of metazanan organism. L-system are parallel grammars, from the point of view of formal language theory. This PhD thesis studies L-systems from from two points of view. Theoretical part of the thesis deals with exponential density of languages and reduction of interactive (e.i. context-sensitive) L-system. New theorem for proofs of exponential density of languages is proved here and several corollaries of this theorem are also included. The work proves that the class of table interactive L-systems with at most six context production and the class of modified interactive L-system with at most twelve context productions are equal to the class of recursively enumerable languages.

Application part of the thesis extends bracket L-systems by chord support. Such as L-systems can be used to create general continous graphs defined as skeleton with chords. The PhD thesis also discusses interpretation of chords. The bracket L-systems with chords can be used to create objects modeled as graphs with fixed skeleton, such as model of leaves or models of trees in the wind.

The closing part of the thesis connects the theretical investigation of L-systems and investigation of their interpretation. New programming language based on L-systems is defined here as a result of our investigations.

ABSTRACT

Lindenmayerovy systémy, zkráceně označované L-systémy, byly poprvé představeny v roce 1968 jako nástroj pro popis vývoje mnohobuněčných organismů. Z hlediska teorie formálních jazyků jsou L-systémy paralelními gramatikami. Tato disertační práce je studuje ze dvou hledisek.

Teoretická část práce pojednává o exponenciální hustotě formálních jazyků a redukci interaktivních nebo-li kontextových L-systémů. Práce dokazuje nový teorém pro dokazování exponenciální hustoty jazyků včetně několika příkladů jeho aplikace. Dále je v disertační práci dokázáno, že třída ETIL-systémů s nejvýše šesti kontextovými pravidly a třída modifikovaných ETIL-systémů s nejvýše dvanácti kontextovými pravidly jsou rovny třídě rekurzívně vyčíslitelných jazyků.

Aplikační část disertační práce rozšiřuje závorkové L-systémy o podporu tětiv. Díky tomu mohou být L-systémy použity pro generování obecných spojitých grafů definovaných jako kostra s tětivy. Interpretace tětiv je též předmětem této práce. Takto rozšířené závorkové L-systémy s podporou tětiv mohou být použity pro vytváření objektů modelovaných jako grafy s pevnou kostrou, jako jsou například modely listů nebo modely stromů ohýbajících se ve větru.

Závěrečná část práce spojuje výsledky výzkumu L-systémů z matematického hlediska a z hlediska jejich intepretace. Výsledkem je nový programovací jazyk založený na L-systémech.

BIBLIOGRAPHY

[1] Wood, D.: Theory of Computation, Harper & Row, Publishers, Inc, New York, 1987

[2] Vitányi, P.M.B: Lindenmayer Systems: Structure, Languages, and Growths Functions, Vrije Universiteit te Amsterdam, Mathematisch centrum, Amstedam, 1978

[3] Rozenberg, G., Salomaa, A.: The Mathematical Theory of L-systems, Academic press, New York, 1980

[4] Meduna, A., Kolář, D: Homogeneous grammars with a reduced number of non-context-free productions, Information Processing Letters, Vol. 81 (2002)

[5] Wood, D.: Grammar and L Forms: An Introduction, Lecture Notes in Computer Science, Springer-Verlag, New York, 1980

[6] Salomaa, A.: Formal languages. Academic press, London, 1973.

[7] Meduna, A.: A Note on Exponential Density of ETOL Languages, Kybernetika, Volume 22 (1986), number 6, pages 514 through 516.

[8] Meduna, A., Švec, M.: Forbiding ETOL Grammars. Theoretical Computer Science, Vol. 2003, No. 54, Paris, FR, p. 256–276, ISSN 0303-3975

[9] Meduna, A., Kolář, D.: Homogenous grammars with a reduced number of non-context-free productions. Information Processing Letters 81 (2002), Elsevier, 2002, pages 253–257

[10] Meduna, A.: Scattered Context Grammars Need Only Three Context Productions to Generate any Recursively Enumerable Language. Unpublished manuscript, 2002

[11] Meduna, A.: Generative Power of Three-Nonterminal Scattered Context Grammars, Theoretical Computer Science, Amsterdam, NL, 2000, p. 625-631, ISSN 0304-3975

[12] Fernau, H., Meduna, A.: Note On the degree of scatter context-sensitivity. Theoretical Computer Science 290 (2003), pages 2121–2124.

[13] Prusinkiewicz, P., Hammel, M., Hanan, J., Měch R.: L-systems: From the Theory to Visual Model of Plans, Proceeding of the 2nd CSIRO Symposium on Computational Challenges in Life Sciences, 1996, HTTP://WWW.CPSC.UCALGARY.CA/REDIRECT/BMV/PAPERS/ L-SYS.CSIRO96.HTML

[14] Prusinkiewicz, P., Lindenmayer A.: The Algorithmic Beauty of Plants, Springer-Verlag, New York, 1990

[15] Měch, R., Prusinkiewicz, P.: Visual Models of Plants Interactiong with Their Environment, Proceedings of SIGGRAPH 96 (New Orleans, Lousiana, August 4–9, 1996). In Computer Graphics Proceeding, Annual Conference Series, 1996, ACM SIGGRAPH, pages 397–410, HTTP://WWW.CPSC.UCALGARY.CA/REDIRECT/BMV/PAPERS/ ENVIRO.SIG96.HTML

[16] Szilard, A. L., Quinton, R. E.: An Interpretation for DOL systems by Computer Graphics, In: The Science Tarrapin, 4:8–13, 1979

[17] Hanan, J.S.: Parametric L-systems and their application to the modelling and visualization of plants, Ph. D. thesis, University of Regina, June 1992

[18] Prusinkiewicz, P.: Modeling and Visualization of Biological Structures, Proceeeding of Graphics Interface ’93, Toronto, Ontario, May 1993, pages 128–137

[19] Hammel M. S., Prusinkiewicz P., Wyvill B, Modelling Compound Leaves Using Implicit Contours, In Visual computing: integrating computer graphics with computer vision, pages 119-212. Springer-verlag, 1992. Proceedings of Computer Graphics International ’92 (Tokyo, Japan, 22-26 June 1992)

[20] Prusinkiwicz, P., Hammel, M. S., Mjolsness, E.: Animation of Plant Development, Computer Graphics Proceedings, Annual Conference Series, 1993, SIGGRAPH 93, Anatheim, California, 1993, pages 351–360

[21] Kelemenová, A.: Complexity of L-systems, Theme and Etudes, in Rozenberg, G., Salomaa, A.: The Book of L., Springer-Verlag, Berlin, 1986, pages 179–191

[22] Goel, N. S., Shen, B.: Symbolic computation Using L-systems II: Extension, Applied mathematics and Computation 69, Elsevier Science Inc., 1995, pages 227–240

[23] Nový, V.: Lindemayerovy systémy v počítačové grafice. Master thesis at Department of Mathematics and Education of Computer Science, The Faculty of Math and Physics, Charles University, Praha, 1995

[24] Szemla, L.: Růstový generátor 3D modelů přírodních dřevin pro virtuální realitu Master thesis at Department of Computer Science and Engineering, Faculty of Electrical Engineering and Computer Science, BUT Brno, 2000. (In Czech)

[25] Žára, J., Beneš, B., Felkel, P.: Moderní počítačová grafika, Computer Press, Praha, 1998, pages 211–238 (in Czech)

[26] Kolka, M., Provazník I: Programming Language Based on L-systems, Volume of the written by students and postgraduate students, VI. (2000), pages 64 to 66, Faculty of Electrical Engineering and Computer Science BUT, Brno, 2000

[27] Kolář, Štěpánková: Logika, algebry, grafy, SNTL—nakladatelství technické literatury, Praha 1989. (In Czech)

[28] Kolář, J., Kroha P., Grafy, cvičení, České vysoké učení technické v Praze, Fakulta elektrotechnická, Praha, 1990 (in Czech)

[29] Demel, J., Grafy, matematika pro vysoké školy technické, SNTL Praha, 1988 (in Czech)

[30] Demel, J.: Grafy a jejich applikace, Academia, Praha, 2002 (in Czech)

[31] Habel, A.: Hyperedge Replacement: Grammars and Languages, Lecture Notes in Computer Science 643, Springer–Verlag, Berlin, 1992

[32] Lindenmayer, A., Rozenberg, G.: Parallel Generation of Maps: Developmental Systems for Cell Layers, In Graph grammars and their application to computer science, International Workshop on Graph Grammars and their Applications, pages 301–316, 1978

[33] Boers, E.J.W: Using L-systems as Graph Grammars: G2L-systems. Presented at the Fifth International Workshop on Graphs Grammars and Their Application to Computer science, Williambusburg, Virginia, 1994. Technical Report 95-30, Computer Science Department, Leiden University. FTP://FTP.WI.LEIDENUNIV.NL/PUB/CS/TECHNICALREPORTS/ 1995/TR95-30.PS.GZ

[34] Bílek, R.: Interpretační překladač stochastických atributovaných L-systémů. Diploma thesis at Department of Computer Science and Engineering, Faculty of Electrical Engineering and Computer Science, BUT Brno, 2001. (In Czech)

[35] Goel , N. S., Rozehnal, I.: A High-level Language for L-systems and Its Applications, In Lindenmayer Systems: impacts on theoretical computer science, computer graphics and developmental biology, Springer, Berlin, 1992, ISBN 3-540-55320-7, pages 231–251

[36] Virius, M.: Programovací jazyky C/C++, GComp, Praha, 1992

[37] Hudak, P., Peterson, J., Fasel J., A Gentle Introduction To Haskell Version 98, June 2000HTTP://WWW.HASKELL.ORG/TUTORIAL/

[38] Mathworks: Matlab - Programming and Data, original online manual HTTP://WWW.MATHWORKS.COM/ACCESS/HELPDESK/HELP/TECHDOC/MATLAB-PROG/MATLAB-PROG.SHTML.

[39] What is Logo?, Logo Foundation HTTP://EL.MEDIA.MIT.EDU/ LOGO-FOUNDATION/LOGO/INDEX.HTML

[40] Mirilovič, M.: Fracint Tutoriál, FEI TU, Košice. Last update in July, 2001. HTTP://ALIFE.TUKE.SK/PROJEKTY/FRACINT/(In Slovak)

[41] McWorter, W. : Fracint L-system Tutorial, January 1997 HTTP://SPANKY.TRIUMF.CA/WWW/FRACTINT/LSYS/TUTOR.HTML

[42] LMuse: L-systems to music, HTTP://WWW.GEOCITIES.COM/ATHENS/ACADEMY/8764/LMUSE/LMUSE.HTML

[43] LParser home page HTTP://HOME.WANADOO.NL/LAURENS.LAPRE/LPARSER.HTM

[44] Communicating Sequential Processes for JavaTM (JCSP), University of Kent at Cantebury , HTTP://WWW.CS.UKC.AC.UK/PROJECTS/OFA/JCSP/

[45] LindaUsers's Guide and Reference Manual, Scientific Computing Associates, HTTP://WWW.LINDASPACES.COM/DOWNLOADS/PARADISEMANUAL.PDF

[46] Paradise User's Guide and Reference Manual, Scientific Computing Associates, HTTP://WWW.LINDASPACES.COM/DOWNLOADS/PARADISEMANUAL.PDF