BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FORMAL SYSTEMS BASED UPON AUTOMATA AND GRAMMARS

FORMÁLNÍ SYSTÉMY AUTOMATŮ A GRAMATIK

EXTENDED THESIS ABSTRACT
TEZE DISERTAČNÍ PRÁCE

AUTHOR MARTIN ČERMÁK
AUTOR PRÁCE

SUPERVISOR prof. RNDr. ALEXANDER MEDUNA, CSc.
VEDOUCÍ PRÁCE

BRNO 2012

Contents

Chapter 1

Introduction

In the seventh century before Christ, Egyptians believed they are the oldest nation in the world. The former king, Psantek I., wanted to confirm this assumption. The confirmation was based on the idea that children, who cannot learn to speak from adults, will use innate human language. That language was supposed to be Egyptian. For this purpose, Psantek I. took two children from a poor family and let them to grow up in care of a shepherd in an environment, where nobody was allowed to speak with these children. Although the test ultimately failed, it brings us testimony that already in old Egypt, people somehow felt the importance of languages (the whole story you can see in *The story of psychology* by *Morton Hunt*).

In 1921, *Ludwig Wittgenstein* published a philosophical work (*Logisch-philosophische Abhandlung*) containing claim that says "The limits of my language mean the limits of my world". In the computer science, this claim is doubly true. Languages are a way how people express information and ideas in terms of computer science or information technology. In essence, any task or problem, which a computer scientist is able to describe, can be described by a language. The language represents a problem and all sentences belonging into this language are its solutions.

Fact about the limitation by languages led to the birth of a new research area referred to as *theory of formal languages* studying languages from a mathematical point of view. The main initiator was linguist *Noam Chomsky*, who, in the late fifties, introduced hierarchy of formal languages given by four types of language generators. By this work, Noam Chomsky inspired many mathematicians and computer scientists so they began to extend this fundamental hierarchy by adding new models for language definition. Because the theory of formal languages examines the languages from the precise mathematical viewpoint, its results are significant for many areas in information technology. Models, which are studied by the theory, are used in compilers, mathematical linguistics, bioinformatics, especially genetics and simulation of natural biology processes, artificial intelligence, computer graphics, computer networks, and others.

The classical formal language theory uses three approaches to define formal languages: Grammatical approach, where the languages are generated by *grammars*, automata approach, where the languages are recognized by *automata*, algebraic approach, where the languages are defined by some *language operations.*

To be more precise, in the grammatical approach, a grammar generates its language by application of *derivation steps* replacing sequences of symbols by other sequences according to its prescribed rules. The symbols can be *terminal* or *nonterminal*, and the sequences of these symbols are called *strings.* In a single derivation step, the grammar, by application of its rule, replaces a part of string by some other string. Any string, which contains no nonterminal symbol and which can be generated from a start nonterminal by application of a sequence of derivation steps, belongs to the language of the grammar. The language of the grammar is represented by the set of such generated strings.

While a grammar generates language, an automaton represents formal algorithm by which the automaton can recognize correctly made sequences of symbols belonging into the language the automaton defines. More specifically, an automaton has string written on its input tape. By application of prescribed rules, it processes the string symbol by symbol and changes its current state to determine whether the string belongs to the language represented by the automaton. If so, the string is accepted by the automaton. The set of all strings accepted by the automaton is the language that the automaton defines.

1.1 Objectives of the Thesis

All models, investigated in the theory of formal languages, are designed to reflect needs of given information technology. Today, when a task distribution, parallel and cooperation process are extremely popular, the main attention is focused on controlled models and systems of models. The necessity of efficient data processing, computer networks, parallel architectures, parallel processing, and nature motivated computing devices justify studying of these approaches in terms of the theory of formal models, where the mechanisms representing these approaches are called *systems of formal models.* The main motivation for investigation of systems lies in a possibility to distribute a task into several smaller tasks, which are easier to solve and easier to describe. These tasks can be solved sequentially or in parallel, and usually, due a communication, the cooperating models are more efficient than the models themselves. The thesis concentrates on these modern approaches and brings new, or generalized, formal mechanisms and results into the theory. More specifically, this thesis mainly deals with systems of automata and grammars and studies their properties.

My thesis, at first, continues with studying of sequential grammar systems, known as *cooperating distributed grammar systems* (shortly *CD grammar systems*). These were introduced in the late eighties as a model for blackboard problem solving. The main idea standing behind the CD grammar systems is in a cooperation of well-known simple grammars working on a shared string under a cooperation protocol. Unfortunately, the increased efficiency, obtained from the cooperation, is given by higher degree of ambiguity and non-determinism, what is unpleasant for a practical purpose. The thesis introduces several restrictions limiting the ambiguity or non-determinism, and investigates their effect on the systems.

The further investigation builds on the work of Lukáš and Meduna, who, in 2006, introduced a new variant of parallel grammar systems named as *multi-generating grammar systems.* In contrast with classic widely studied *parallel communicating grammar systems*, where included grammars are used as supporting elements and the language of a parallel grammar system is generated by one predetermined grammar, these new systems take into account strings from all their grammars. The final strings are obtained from all generated strings by a string operation. The thesis introduces two versions of automata counterpart to these grammar systems and proves their equivalence. Thereafter, the investigated systems are generalized and a fundamental hierarchy of these systems is established. Finally, the thesis suggests systems based on mentioned approaches as a direct translator of natural languages and parser of languages generated by a specific type of controlled grammars.

Chapter 2

Notation and Basic Definitions

In this work, we assume the reader is familiar with the formal language theory (see [54]) and the basic aspects of computational linguistics (see [61]).

For a set, Q, $|Q|$ denotes the cardinality of Q. Let $K \subset \mathbb{N}_0$ is a final set. Then, $max(K) = k$, where $k \in K$ and for all $h \in K$, $k \geq h$; and $min(K) = l$, where $l \in K$ and for all $h \in K$, $l \leq h$. Furthermore, let $(X, \geq)$ is an ordered set and $A \subseteq X$. We say that $x \in X$ is an upper and lower bound of A, if for all $a \in A$, $a \leq x$ and $x \leq a$, respectively. The least upper bound is called *supremum*, written as $sup(A)$. Conversely, the greatest lower bound is known as *infimum*, denoted $inf(A)$.

For an alphabet, V, V^* represents the free monoid generated by V (under the operation concatenation). The identity of V^* is denoted by ε. Set $V^+ = V^* - \{\varepsilon\}$; algebraically, V^+ is thus the free semigroup generated by V. For every string $w \in V^*$, $|w|$ denotes the length of w, $(w)^R$ denotes the mirror image of w, and for $A \in V$, $occur(A, w)$ denotes the number of occurrences of A in w. For $a, b \in \mathbb{Z}$, function $max(a, b)$ returns the greater value from a and b.

A *finite automaton*, FA, is a quintuple $M = (Q, \Sigma, \delta, q_0, F)$, where Q is a finite set of states; Σ is an alphabet; $q_0 \in Q$ is the initial state; δ is a finite set of transition rules of the form $qa \to p$, where $p, q \in Q$, and $a \in \Sigma \cup \{\varepsilon\}$; and $F \subseteq Q$ is a set of final states. A configuration of M is any string from $Q\Sigma^*$. For any configuration qay, where $a \in \Sigma$, $y \in \Sigma^*$, $q \in Q$, and any $r = qa \to p \in \delta$, M makes a move from configuration qay to configuration py according to r, written as $qay \Rightarrow py[r]$, or simply $qay \Rightarrow py$. $\Rightarrow^*$ and $\Rightarrow^+$ represent transitive-reflexive and transitive closure of $\Rightarrow$, respectively. If $w \in \Sigma^*$ and $q_0w \Rightarrow^* f$, where $f \in F$, then w is accepted by M and $q_0w \Rightarrow^* f$ is an acceptance of w in M. The language of M is defined as $L(M) = \{w|\ w \in \Sigma^*, q_0w \Rightarrow^* f$ is an acceptance of $w\}$.

A *partially blind k-counter automaton*, k-PBCA, is finite automaton $M = (Q, \Sigma, \delta, q_0, F)$ with k integers $v = (v_1, \ldots, v_k)$ in $\mathbb{N}_0^k$ as an additional storage. Transition rules in δ are of the form $pa \to qt$, where $p, q \in Q$, $a \in \Sigma \cup \{\varepsilon\}$, and $t \in \mathbb{Z}^k$. As a configuration of k-PBCA we understand any string from $Q\Sigma^*\mathbb{N}_0^k$. Let $\chi_1 = paw(v_1, \ldots, v_k)$ and $\chi_2 = qw(v'_1, \ldots, v'_k)$ be two configurations of M and $r = pa \to q(t_1, \ldots, t_k) \in \delta$, where $(v_1 + t_1, \ldots, v_k + t_k) = (v'_1, \ldots, v'_k)$. Then, M makes a move from configuration χ_1 to χ_2 according to r, written as $\chi_1 \Rightarrow \chi_2[r]$, or simply $\chi_1 \Rightarrow \chi_2$. $\Rightarrow^*$ and $\Rightarrow^+$ represent transitive-reflexive and transitive closure of $\Rightarrow$, respectively. The language of M is defined as $L(M) = \{w|\ w \in \Sigma^*, q_0w(0, \ldots, 0) \Rightarrow^* f(0, \ldots, 0), f \in F\}$.

A *pushdown automaton*, PDA, is a septuple $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, where Q is a finite set of states; Σ is an alphabet; $q_0 \in Q$ is the initial state, Γ is a pushdown alphabet; δ is a finite set of transition rules of the form $Zqa \to \gamma p$, where $p, q \in Q$, $Z \in \Gamma$, and $a \in \Sigma \cup \{\varepsilon\}$; $\gamma \in \Gamma^*$; $Z_0 \in \Gamma$ is the initial pushdown symbol; and $F \subseteq Q$ is a set of final states. A configuration of M is any string from $\Gamma^* Q \Sigma^*$. For any configuration $xAqay$, where $x \in \Gamma^*$, $y \in \Sigma^*$, $q \in Q$, and any $r = Aqa \to \gamma p \in \delta$, M makes a move from configuration $xAqay$ to configuration $x\gamma py$ according to r, written as $xAqay \Rightarrow x\gamma py[r]$, or simply $xAqay \Rightarrow x\gamma py$. $\Rightarrow^*$ and $\Rightarrow^+$ represent transitive-reflexive and transitive closure of $\Rightarrow$, respectively. If $w \in \Sigma^*$ and $Z_0 q_0 w \Rightarrow^* f$, where $f \in F$, then w is accepted by M and $Z_0 q_0 w \Rightarrow^* f$ is an acceptance of w in M. The language of M is defined as $L(M) = \{w |\ w \in \Sigma^*, Z_0 q_0 w \Rightarrow^* f$ is an acceptance of $w\}$.

A *k-turn PDA* is a PDA in which the length of the pushdown tape alternatively increases and decreases at most k-times during any sweep of the pushdown automaton.

A *context-free grammar*, CFG, is quadruple $G = (N, T, P, S)$, where N and T are disjoint alphabets of nonterminal and terminal symbols, respectively; $S \in N$ is the start symbol of G; and P is a finite set of grammar rules of the form $A \to \alpha$, where $A \in N$, and $\alpha \in (N \cup T)^*$. Furthermore, if $\alpha \in T^* N T^*$, we say that the grammar is *linear*, LNG for short, and if $\alpha \in TN$, we say that the grammar is *right-linear*, RLNG for short. A sentential form of G is any string from $(N \cup T)^*$. Let $u, v \in (N \cup T)^*$ and $r = A \to \alpha \in P$. Then, G makes a derivation step from u to v according to r, written as $uAv \Rightarrow u\alpha v[r]$, or simply $uAv \Rightarrow u\alpha v$. Let $\Rightarrow^*$ and $\Rightarrow^+$ denote transitive-reflexive and transitive closure of $\Rightarrow$. The language of G is defined as $L(G) = \{w |\ S \Rightarrow^* w, w \in T^*\}$.

A *phrase-structure grammar* is a quadruple $G = (N, T, S, P)$, where N and T are alphabets such that $N \cap T = \emptyset$, $S \in N$, and P is a finite set of productions of the form $\alpha \to \beta$, where $\alpha \in N^+$ and $\beta \in (N \cup T)^*$. If $\alpha \to \beta \in P$, $u = x_0 \alpha x_1$, and $v = x_0 \beta x_1$, where $x_0, x_1 \in V^*$, then $u \Rightarrow v\ [\alpha \to \beta]$ in G or, simply, $u \Rightarrow v$. Let $\Rightarrow^+$ and $\Rightarrow^*$ denote the transitive closure of $\Rightarrow$ and the transitive-reflexive closure of $\Rightarrow$, respectively. The *language of* G is denoted by $L(G)$ and defined as $L(G) = \{w \in T^* |\ S \Rightarrow^* w\}$.

A *programmed grammar* (see [34]) is a septuple $G = (N, T, S, P, \Lambda, \sigma, \phi)$, where

- N and T are alphabets such that $N \cap T = \emptyset$,
- $S \in N$,
- P is a finite set of productions of the form $A \to \beta$, where $A \in N$ and Λ is a finite set of labels for the productions in P.
- Λ can be interpreted as a function which outputs a production when being given a label,
- σ and ϕ are functions from Λ into the 2^Λ.

For $(x, r_1), (y, r_2) \in (N \cup T)^* \times \Lambda$ and $\Lambda(r_1) = (\alpha \to \beta)$, we write $(x, r_1) \Rightarrow (y, r_2)$ iff either $x = x_1 \alpha x_2$, $y = x_1 \beta x_2$ and $r_2 \in \sigma(r_1)$, or $x = y$, and rule $\alpha \to \beta$ is not applicable to x, and $r_2 \in \phi(r_1)$.

The *language of* G is denoted by $L(G)$ and defined as $L(G) = \{w |\ w \in T^*, (S, r_1) \Rightarrow^* (w, r_2)$, for some $r_1, r_2 \in \Lambda\}$. Let $\mathscr{L}(\mathrm{P, ac})$ denote the class of languages generated by programmed grammars. If $\phi(r) = \emptyset$, for each $r \in \Lambda$, we are led to the class $\mathscr{L}(\mathrm{P})$.

Let G be a programmed grammar. For a derivation $D : S = w_1 \Rightarrow w_2 \Rightarrow \ldots \Rightarrow w_n = w$, $w \in T^*$, of G, $ind(D, G) = max(\{occur(w_i, N) |\ 1 \leq i \leq n\})$, and for $w \in T^*$, $ind(w, G) = min(\{ind(D, G) |\ D$ is a derivation of w in $G\})$. The *index of* G is $ind(G) = sup(\{ind(w, G) |\ w \in L(G)\})$. For a language L in the class $\mathscr{L}(P)$ generated by programmed grammars,

$ind(L) = inf(\{ind(G) | \ L(G) = L\}$. For the class $\mathscr{L}(P)$, $\mathscr{L}_n(P) = \{L| \ L \in \mathscr{L}(P)$ and $ind(L) \leq n$, for $n \geq 1\}$ (see [34]).

A *matrix grammar*, MAT, is a pair $H = (G, C)$, where $G = (N, T, P, S)$ is a context-free grammar and $C \subset P^*$ is a finite set of strings denoted as matrices. A sentential form of H is any string from $(N \cup T)^*$. Let u, v be two sentential forms. Then, we say that H makes a derivation step from u to v according to r, written as $u \Rightarrow v[m]$, or simply $u \Rightarrow v$, if $m = p_1 \dots p_m \in C$ and there are $v_0, \dots, v_m$, where $v_0 = u$, $v_m = v$, and $v_0 \Rightarrow v_1[p_1] \Rightarrow \dots \Rightarrow v_m[p_m]$ in G. Let $\Rightarrow^*$ and $\Rightarrow^+$ denote transitive-reflexive and transitive closure of $\Rightarrow$. The language of H is defined as $L(H) = \{w| \ S \Rightarrow w_1[m_1] \Rightarrow \dots \Rightarrow w_n[m_n], w_n = w, m_1, \dots, m_n \in C, w \in T^*, n \geq 0\}$. The class of languages generated by matrix grammars is denoted by $\mathscr{L}(MAT)$.

The classes of regular languages, linear languages, context-free languages, context-sensitive languages, and recursively enumerable languages are denoted by **REG**, **LIN**, **CF**, **CS**, and **RE**, respectively.

Chapter 3

State of the Art

Unlike the classic formal languages and automata theory, which studies models accepting or generating language by one automaton or grammar, a modern computer science aims to distribute this computation. The main reasons follow from necessities and possibilities of computer networks, distributed databases, parallel processors, etc., which give us new terms such as *distribution*, *communication*, *concurrency*, and *parallelism.*

A formal system is defined as a set of formal models working together under a specified protocol. Such systems have many advantages. For example, they allow to model distribution, the generative or accepting power of used models usually increases, the (descriptional) complexity of a language decreases, there is a possibility of parallel cooperation, etc.

The main role in the theory of formal systems is played by cooperation protocols and used formal models. This chapter considers four basic classes of systems of formal models: *sequential grammar systems*, *parallel grammar systems*, *sequential automata systems*, and *parallel automata systems.*

3.1 Cooperating Distributed Grammar System

A *cooperating distributed grammar system*, *CD grammar system* for short, was first introduced in [57] related to two-level grammars. Several years later, by investigation of this system in relation with multi-agent systems and blackboard problem solving architectures in [18], studies of CD grammar systems became an intense research area.

A CD grammar system consists of finite number of grammars, called *components.* These symbolize agents. The common sentential form, which the agents sequentially modify according to a mode given by a certain protocol, represents the current state of the problem to be solved. The authors of [18] considered five modes under which agents work: $*$*-mode* – the active agent works as long as it wants; t*-mode* – the active agent works as long as it is able to work; and, $\geq k, \leq k$, and $= k$ *modes* correspond to a time limitation of agents activity, when the active agent has to make i steps for $i \geq k, i \leq k$, and $i = k$, respectively. If a terminal string is generated, the problem is solved (see definitions 3.1 through 3.4 specifying CD grammar systems in terms of formal languages).

Definition 3.1 (Cooperating distributed grammar system)
A *cooperating distributed grammar system*, a *CD grammar system* for short, is an $(n+3)$-tuple $\Gamma = (N, T, S, P_1, \ldots, P_n)$, where N, T are alphabets such that $N \cap T = \emptyset$, $V = N \cup T$, $S \in N$, and $G_i = (N, T, P_i, S), 1 \leq i \leq n$, is a context-free grammar.

Definition 3.2 (Mode of derivation in CD grammar systems)
Let $\Gamma = (N, T, S, P_1, \ldots, P_n)$ be a CD grammar system.

- For every $i = 1, 2, \ldots, n$, *terminating derivation* by ith component, written as $\Rightarrow^t_{P_i}$, is defined as

$$x \Rightarrow^t_{P_i} y \text{ iff } x \Rightarrow^*_{P_i} y \text{ and there is no } z \in \Sigma^* \text{ such that } y \Rightarrow_{P_i} z.$$

- For every $i = 1, 2, \ldots, n$, *k-steps derivation* by ith component, written as $\Rightarrow^{=k}_{P_i}$, is defined as

$$x \Rightarrow^{=k}_{P_i} y \text{ iff there are } x_1, \ldots, x_{k+1} \text{ and for every } j = 1, \ldots, k,\ x_j \Rightarrow_{P_i} x_{j+1}.$$

- For every $i = 1, 2, \ldots, n$, *at most k-steps derivation* by ith component, written as $\Rightarrow^{\leq k}_{P_i}$, is defined as

$$x \Rightarrow^{\leq k}_{P_i} y \text{ iff } x \Rightarrow^{=k'}_{P_i} y \text{ for some } k' \leq k.$$

- For every $i = 1, 2, \ldots, n$, *at least k-steps derivation* by ith component, written as $\Rightarrow^{\geq k}_{P_i}$, is defined as

$$x \Rightarrow^{\geq k}_{P_i} y \text{ iff } x \Rightarrow^{=k'}_{P_i} y \text{ for some } k' \geq k.$$

Definition 3.3 (Language generated by a CD grammar system)
Let $\Gamma = (N, T, S, P_1, \ldots, P_n)$ be a CD grammar system and $f \in D$ be a mode of derivation, where $D = \{*, t\} \cup \{\leq k, = k, \geq k \mid k \in \mathbb{N}\}$. Then, the *language generated by* Γ, $L_f(\Gamma)$, is

$$L_f(\Gamma) = \{\omega \in T^* \mid S \Rightarrow^f_{P_{i_1}} \omega_1 \Rightarrow^f_{P_{i_2}} \ldots \Rightarrow^f_{P_{i_m}} \omega_m = \omega, m \geq 1, 1 \leq j \leq m, 1 \leq i_j \leq n\}.$$

Definition 3.4 (Classes of languages generated by CD grammar systems)
The classes of languages generated by CD grammar systems we denote by $\mathscr{L}(\mathrm{CD}, n, f)$, where $f \in \{*, t\} \cup \{= k, \leq k, \geq k \mid k \in \mathbb{N}\}$, and $n \in \mathbb{N} \cup \{\infty\}$ is the number of components.

By the following theorems, we summarize selected basic results regarding the power of CD grammar systems.

Theorem 3.5 $\mathbf{CF} = \mathscr{L}(\mathrm{CD}, 1, t) = \mathscr{L}(\mathrm{CD}, 2, t) \subset \mathscr{L}(\mathrm{CD}, 3, t) = \mathscr{L}(\mathrm{CD}, \infty, t) \subset \mathbf{CS}$.

Theorem 3.6 If $f \in \{= 1, \geq 1, *\} \cup \{\leq k \mid k \geq 1\}$, then $\mathscr{L}(\mathrm{CD}, \infty, f) = \mathbf{CF}$.

Theorem 3.7 $\mathbf{CF} = \mathscr{L}(\mathrm{CD}, 1, f) \subset \mathscr{L}(\mathrm{CD}, 2, f) \subseteq \mathscr{L}(\mathrm{CD}, r, f) \subseteq \mathscr{L}(\mathrm{CD}, \infty, f) \subseteq \mathscr{L}(\mathrm{MAT})$, for all $f \in \{= k, \geq k \mid k \geq 2\}$ and $r \geq 3$.

Other Variants of CD Grammar Systems

The standard CD grammar systems, defined above, use only conditions saying when the enabled component can, or has to, stop working on a sentential form. Selection of component for work is non-deterministic. However, in [21], [19], [27], [5], etc., you can find discussions about many variants of CD grammar systems with several approaches how to select working components.

As a natural extension of CD grammar systems, Mitrana and Păun introduced a *hybrid cooperating distributed grammar systems* in [62] and [68]. In contrast with CD grammar systems, where all components work in the same mode, these systems consists of components working in different modes.

The generative power of CD grammar systems can be increased by teams. This idea was introduced and has been firstly investigated in [44]. Formally, a *CD grammar system with teams* is defined as a tuple $\Gamma = (N, T, S, P_1, \ldots, P_n, R_1, \ldots, R_m)$, where $\Gamma = (N, T, S, P_1, \ldots, P_n)$ is an ordinary CD grammar system and $R_i \subseteq \{P_1, \ldots, P_n\}$ is a team, for all $i = 1, \ldots, m$. At one moment, components from a team simultaneously rewrite corresponding part of a shared sentential form. Precisely, $x \Rightarrow_{R_i} y$ iff $x = x_1A_1x_2 \ldots x_sA_sx_{s+1}$, $y = x_1y_1x_2 \ldots x_sy_sx_{s+1}$, for all $j = 1, \ldots, s+1$, $x_j \in (N \cup T)^*$, and for all $k = 1, \ldots, s$, $A_k \to y_k \in P \in R_i$. For this one step derivation, k-steps derivation, at most k-steps derivation, at least k-steps derivation, and derivation of any number of steps are defined as usual. Only terminating derivation has three variants, where the active team stops working if the team as a whole cannot perform any further step, no component can apply any of its rules, or at least one component cannot rewrite any symbol of the current sentential form (see [44, 37, 69]).

Besides mentioned variants, many others appear in the literature from the introduction of ordinary CD grammar systems in [57] and [18] up to these days, e.g. CD grammar systems with external storage (see [31, 76, 32, 35]), CD grammar systems consisting of different components (see [78, 39, 49, 23]), hierarchical systems (see [2]), deterministic systems (see [59]), etc.

3.2 Parallel Communicating Grammar Systems

Parallel communicating grammar systems, *PC grammar systems* for short, were introduced in [70]. These systems consist of a finite number of grammars (components), which work on their own sentential form. The components are synchronized and make derivation steps concurrently. During derivation, the communication is performed through special *query* symbols. Whenever at least one component generates a query symbol, all components suspend generating and the grammar system makes a *communication step*—that is, for every component in the system, each occurrence of a query symbol in its sentential form is replaced by the sentential form of the component to which the query symbol is pointing to. One component of the system is called *master* and the language of the master is the language of PC grammar system.

Similarly as in the case of CD grammar systems, the theory of formal languages studies different variants of PC grammar systems, such as

- returning PC grammar systems, where each component that has sent its sentential form to another starts from the start nonterminal;
- centralized PC grammar systems, where only the master can generate query symbols;
- non-synchronized PC grammar systems, where all components include rules of the form $A \to A$ for every nonterminal symbol A;
- PC grammar system with communication by commands, where each component has a control language and in a certain situation all components send their current sentential

form to other components owning a control language to which the sentential form belongs to;

- PC grammar systems with languages given by concatenation of all strings over terminal symbols after the end of generation;
- PC grammar systems using query strings instead of query symbols, where the communication steps is done after at least one component generates a query string pointing to another component;
- PC grammar systems, where components make different number of steps;
- and many others, see [21], [20], [42], [72], etc.

Probably the most important features of parallel communicating grammar systems are *communication protocol* and *types of used components* together with the way they work. Further important feature is *synchronization*. Habitually, the synchronization of components is done by an universal clock (each component make one derivation step in each time unit), but others synchronization mechanisms are also studied (see [21], [67], [25]). Two of the most natural variants are synchronization by rules, which can be applied simultaneously, and synchronization by nonterminals, which can be rewritten at the same time unit. Both these approaches Lukáš and Meduna used in [55] and [46], where they have investigated multi-generative grammar systems.

Multi-Generative Grammar Systems

Multi-generative grammar systems are a variant of parallel communicating grammar systems, where the communication is provided only by synchronization. This synchronization restricts either rules, which can be used for each common derivation step, or nonterminals, which can be simultaneously rewritten. For successful generation, all components have to produce sentences at the same time. Lukáš and Meduna have considered three types of languages defined by multi-generative grammar systems—languages consisting of all sentences produced by all components, languages consisting of concatenations of all sentences produced by all components, and languages consisting of sentences produced by the first component of a multi-generating grammar system.

Definition 3.8 (Multi-generative nonterminal synchronized grammar system)
A *multi-generative nonterminal synchronized grammar system*, GN, is an $(n+1)$-tuple

$$\Gamma = (G_1, \dots, G_n, Q), \text{ where}$$

- $G_i = (N_i, T_i, P_i, S_i)$ is a context-free grammar, for all $i = 1, \dots, n$,
- Q is a finite set of control n-tuples of the form $(A_1, \dots, A_n)$, where $A_i \in N_i$ for all $i = 1, \dots, n$.

Definition 3.9 (Multi-generative rule synchronized grammar system)
A *multi n-generative rule synchronized grammar system*, GR, is an $(n+1)$-tuple

$$\Gamma = (G_1, \dots, G_n, Q), \text{ where}$$

- $G_i = (N_i, T_i, P_i, S_i)$ is a context-free grammar for all $i = 1, \dots, n$,

- Q is a finite set of control n-tuples of the form $(r_1, \ldots, r_n)$, where $r_i \in P_i$ for all $i = 1, \ldots, n$.

Definition 3.10 (Multi-sentential form)
Let $\Gamma = (G_1, \ldots, G_n, Q)$ be either GN or GR. Then a *multi-sentential form* is an n-tuple $\chi = (x_1, \ldots, x_n)$, where $x_i \in (T_i \cup N_i)^*$ for all $i = 1, \ldots, n$.

Definition 3.11 (Derivation step in GN)
Let $\Gamma = (G_1, \ldots, G_n, Q)$ be a GN, let $\chi = (u_1A_1v_1, \ldots, u_nA_nv_n)$, $\chi' = (u_1x_1v_1, \ldots, u_nx_nv_n)$, be two multi-sentential forms, where $A_i \in N_i, u_i, v_i, x_i \in (N_i \cup T_i)^*$ for all $i = 1, \ldots, n$. Let $A_i \to x_i \in P_i$ for all $i = 1, \ldots, n$, and $(A_1, \ldots, A_n) \in Q$. Then, χ *directly derives* χ', written as $\chi \Rightarrow \chi'$.

Definition 3.12 (Derivation step in GR)
Let $\Gamma = (G_1, \ldots, G_n, Q)$ be a GR, let $\chi = (u_1A_1v_1, \ldots, u_nA_nv_n)$, $\chi' = (u_1x_1v_1, \ldots, u_nx_nv_n)$, be two multi-sentential forms, where $A_i \in N_i, u_i, v_i, x_i \in (N_i \cup T_i)^*$ for all $i = 1, \ldots, n$. Let $r_i{:}A_i \to x_i \in P_i$ for all $i = 1, \ldots, n$, and $(r_1, \ldots, r_n) \in Q$. Then, χ *directly derives* χ', written as $\chi \Rightarrow \chi'$.

Definition 3.13 (Multi-language generated by GN and GR)
Let $\Gamma = (G_1, \ldots, G_n, Q)$ be either GN or GR. Then, the *n-language generated by* Γ, n-$L(\Gamma)$, is

$$n\text{-}L(\Gamma) = \{(w_1, \ldots, w_n) |\ (S_1, \ldots, S_n) \Rightarrow^* (w_1, \ldots, w_n),\ w_i \in T_i^* \text{ for all } i = 1, \ldots, n\}.$$

Definition 3.14 (Languages of GN and GR)
Let $\Gamma = (G_1, \ldots, G_n, Q)$ be either GN or GR. Then, we define

- *the language generated by* Γ *in union mode*, $L_\cup(\Gamma))$, as

$$L_\cup(\Gamma) = \bigcup_{i=1}^{n} \{w_i |\ (w_1, \ldots, w_n) \in n\text{-}L(\Gamma)\},$$

- *the language generated by* Γ *in concatenation mode*, $L_\bullet(\Gamma)$, as

$$L_\bullet(\Gamma) = \{w_1 \ldots w_n |\ (w_1, \ldots, w_n) \in n\text{-}L(\Gamma)\},$$

- *the language generated by* Γ *in first-component-selection mode*, $L_1(\Gamma))$, as

$$L_1(\Gamma) = \{w_1 |\ (w_1, \ldots, w_n) \in n\text{-}L(\Gamma)\}.$$

Definition 3.15 (Canonical multi-generative grammar systems)
We say, that GN and GR are *canonical* if all the components of GN and GR can make only the leftmost derivations, i.e. only the leftmost nonterminal can be rewritten in each sentential form. Canonical multi-generative rule synchronized grammar systems and canonical multi-generative nonterminal synchronized grammar systems are denoted by CGR and CGN, respectively.

$$\begin{tikzcd}
\mathscr{L}(k\text{-}^f\mathrm{CGN}) \arrow[rr, leftrightarrow] & & \mathscr{L}(k\text{-}^f\mathrm{CGR}) \arrow[r, leftrightarrow] & \mathbf{RE} & & & \\
& & & \mathbf{CS} \arrow[u] & & & \\
\mathscr{L}(\mathrm{NPC}, \mathbf{CF}) \arrow[rrrrrr] & & & & & & \mathscr{L}(\mathrm{PC}, \mathbf{CF}) \arrow[uulll] \\
& \mathscr{L}(\mathrm{NPC}, \mathbf{REG}) \arrow[rrrr] & & & & \mathscr{L}(\mathrm{PC}, \mathbf{REG}) & \\
& \mathscr{L}(k\text{-}^f\mathrm{GN}) \arrow[r, leftrightarrow] & \mathscr{L}(k\text{-}^f\mathrm{GR}) \arrow[rr, leftrightarrow] & & \mathrm{MAT} \arrow[uuuul] & & \\
\mathscr{L}(\mathrm{NCPC}, \mathbf{CF}) \arrow[uuu] \arrow[uuuurrr] & & & & & & \mathscr{L}(\mathrm{CPC}, \mathbf{CF}) \arrow[uuu] \arrow[uuuulll] \\
& & & \mathbf{CF} \arrow[uuuuu] \arrow[uur] & & & \\
& \mathscr{L}(\mathrm{NCPC}, \mathbf{REG}) \arrow[uuuu] \arrow[uul] & & & \mathscr{L}(\mathrm{CPC}, \mathbf{REG}) \arrow[uuu] \arrow[uuuur] \arrow[uurr] & & \\
& & & \mathbf{LIN} \arrow[uu] \arrow[uuuuurr] & & & \\
& & & \mathbf{REG} \arrow[u] \arrow[uull] \arrow[uur] & & &
\end{tikzcd}$$

Figure 3.1: Hierarchy of languages (it is considered that $k \geq 2$ and $f \in \{\cup, \bullet, 1\}$)

Convention 3.16 If there is an attention on the number of components in a multi-generative grammar system, we use terms n-generative grammar system, n-GN, n-GR, n-CGR, n-CGN, sentential n-form, and n-language, for some positive integer n, rather than multi-generative grammar system, GN, GR, CGN, CGR, multi-sentential form, and multi-language, respectively.

Definition 3.17 (Classes of n-GN, n-CGN, n-GR, and n-CGR n-languages)
Let $X \in \{\mathrm{GN}, \mathrm{CGN}, \mathrm{GR}, \mathrm{CGR}\}$. The class of n-languages of n-X, $\mathscr{L}({}_nX)$, is defined as $\mathscr{L}({}_nX) = \{n\text{-}L \mid n\text{-}L$ is an n-language generated by $n\text{-}X\}$.

Definition 3.18 (Classes of n-GN, n-GR, n-CGN, and n-CGR languages)
Let $X \in \{\mathrm{GN}, \mathrm{CGN}, \mathrm{GR}, \mathrm{CGR}\}$ and $f \in \{\cup, \bullet, 1\}$. The class of languages generated by an n-X in f-mode, $\mathscr{L}(n\text{-}^f X)$, is defined as $\mathscr{L}(n\text{-}^f X) = \{L \mid L$ is a language generated in the f-mode by $n\text{-}X\}$.

Let's say that $\mathscr{L}(X\mathrm{PC}, Y)$ with $X \in \{\varepsilon, \mathrm{C}, \mathrm{N}, \mathrm{NC}\}$ and $Y \in \{\mathbf{REG}, \mathbf{CF}\}$ denote the classes of languages generated by XPC grammar systems with unlimited number of components, where N and C before PC say that PC grammar systems are non-returning and centralized, respectively, and furthermore, $Y = \mathbf{REG}$ and $Y = \mathbf{CF}$ mean that the components of the systems are regular grammars and context-free grammars, respectively. In Figure 3.1 you can see several important relationships between the classes of languages defined by parallel grammar systems. The results are taken from [72], [55], and [46].

3.3 Automata Systems

Automata and automata systems are used in many areas of computer science. One can find them in computer networks, formal analysis and verifications, pattern matching, parallel computers, DNA computing, artificial intelligence, etc. In this section, we briefly outline several important cooperating models in terms of theory of formal languages.

A *multiprocesor automaton* is based upon finite automata, called *processors*. These processors are coordinated by a central arbiter determining which processor is to become active or inactive (an inactive processor preserves its configuration) at a given step. The only informations that the arbiter has for decision about automata activities are the current state of each automaton and number of steps proceeding by active automata (see [7]).

Similar system to the multiprocessor automata allows to share information about current states of processors. In such system, each automaton makes a move with respect of the current input symbol and states of all automata. If we reduce all these automata to one with multiple reading head, we make equivalent model called *multi-head automaton* (see [71]).

In relation to automata, [29] has firstly investigated an idea to apply strategies akin to those that cooperating distributed grammar systems use. For this purpose, Mitrana and Dassow introduced special types of *multi-stack pushdown automata.* However, they do not form the automata counterpart of CD grammar systems. This was introduced and has been studied in [24] under the name *distributed pushdown automata system.*

A *distributed pushdown automata system* contains a shared one-way input tape, one reading head, and finite number of components having their own pushdown and finite sets of states. At any moment, only one component is active. According to a cooperation protocol, the active component must perform k, at least k, at most k, for $k \geq 1$, or it must work as long as it is able to perform a move.

Parallel communicating automata systems have been investigated both with finite automata and pushdown automata as components. The first variant, *parallel communicating finite automata system*, was introduced by Martín-Vide, Mateescu, and Mitrana in [48]. Finite automata in such systems work independently but on a request, they communicate by states to each other. More precisely, the finite automata are entitled to request the current state of any other component. In [48] has been discussed several variants, where contacted automaton after communication is/is not returned to the initial state (*returning/non-returning parallel communication automata systems*), or, only one automaton has/all automata have the right to ask the current state from the others (*centralized/non-centralized parallel communication automata systems*). By application of these strategies on pushdown automata, the investigation was continued in [22], where the attention is focused especially on communication by stacks, i.e. on request an asked automaton send the content of its pushdown to requesting automata which push it on their pushdowns).

In the same way as in the case of grammar systems discussed above, you can find many other variants of automata systems in the literature (see [74, 75, 65, 58, 66, 26]). Generally, we can say that the theory of formal languages reflects the approaches used in grammar systems into automata systems and studies the accepting power of given systems in relation to component represented by automata working in many different ways.

3.4 New Definitions and Selected Results

3.4.1 Restrictions on CD Grammar Systems

Formal language theory has investigated various left restrictions placed on derivations in grammars working in a context-free way. In ordinary context-free grammars, these restrictions have no effect on the generative power. In terms of regulated context-free grammars, the formal language theory has introduced a broad variety of leftmost derivation restrictions, many of which change their generative power (see [3, 6, 28, 30, 33, 36, 38, 41, 50, 51, 52, 53]). In terms of grammars working in a context-sensitive way, significantly fewer left derivation restrictions have been discussed in the language theory. Indirectly, this theory has placed some restrictions on the productions so the resulting grammars make only derivations in a left way (see [3, 6]). This theory also directly restricted derivations in the strictly leftmost way so the rewritten symbols are preceded only by terminals in the sentential form during every derivation step (see [50]). In essence, all these restrictions result in decreasing the generative power to the power of context-free grammars (see page 198 in [73]). This section generalizes the discussion of this topic by investigating regularly controlled cooperating distributed grammar systems (see Chapter 4 in [73]) whose components are phrase-structure grammars restricted in some new ways.

Now, we define the restrictions on derivations in phrase-structure grammars. In the following, we consider V as the total alphabet of $G = (N, T, P, S)$, i.e. $V = N \cup T$. *Derivation-restriction of type I:* Let $l \in \mathbb{N}$ and let $G = (N, T, P, S)$ be a phrase-structure grammar. If there is $\alpha \to \beta \in P$, $u = x_0\alpha x_1$, and $v = x_0\beta x_1$, where $x_0 \in T^*N^*$, $x_1 \in V^*$, and $\text{occur}(x_0\alpha, N) \leq l$, then $u\ {}_l\Leftrightarrow\ v\ [\alpha \to \beta]$ in G, or simply $u\ {}_l\Leftrightarrow\ v$.

The k-fold product of ${}_l\Leftrightarrow$, where $k \geq 0$, is denoted by ${}_l\Leftrightarrow^k$. The reflexive-transitive closure and transitive closure of ${}_l\Leftrightarrow$ are denoted by ${}_l\Leftrightarrow^*$ and ${}_l\Leftrightarrow^+$, respectively.

Derivation-restrictions of type II and III Let $m, h \in \mathbb{N}$. $W(m)$ denotes the set of all strings $x \in V^*$ satisfying 1 given next. $W(m, h)$ denotes the set of all strings $x \in V^*$ satisfying 1 and 2.

1. $x \in (T^*N^*)^mT^*$;

2. $(y \in \text{sub}(x)$ and $|y| > h)$ implies $\text{alph}(y) \cap T \neq \emptyset$.

Let $u \in V^*N^+V^*$, $v \in V^*$, and $u \Rightarrow v$. Then, $u\ {}^h_m\Leftrightarrow\ v$ in G, if $u, v \in W(m, h)$; and if $u, v \in W(m)$, $u\ {}_m\Leftrightarrow\ v$ in G.

The k-fold product of ${}^h_m\Leftrightarrow$ and ${}_m\Leftrightarrow$ are denoted by ${}^h_m\Leftrightarrow^k$ and ${}_m\Leftrightarrow^k$, respectively, where $k \geq 0$. The reflexive-transitive closure and transitive closure of ${}^h_m\Leftrightarrow$ are denoted by ${}^h_m\Leftrightarrow^*$ and ${}^h_m\Leftrightarrow^+$, respectively; and the reflexive-transitive closure and transitive closure of ${}^h_m\Leftrightarrow$ and ${}_m\Leftrightarrow$ are denoted by ${}_m\Leftrightarrow^*$ and ${}_m\Leftrightarrow^+$, respectively.

Convention 3.19 Let $\Gamma = (N, T, S, P_1, \ldots, P_n)$ be a CD grammar system with phrase-structure grammars as its components and $V = N \cup T$ be the total alphabet of Γ. Furthermore, let $u \in V^*N^+V^*$, $v \in V^*$, $k \geq 0$. Then, we write $u\ {}_l\Leftrightarrow^k_{P_i}\ v$, $u\ {}^h_m\Leftrightarrow^k_{P_i}\ v$, and $u\ {}_m\Leftrightarrow^k_{P_i}\ v$ to denote that $u\ {}_l\Leftrightarrow^k\ v$, $u\ {}^h_m\Leftrightarrow^k\ v$, and $u\ {}_m\Leftrightarrow^k\ v$, respectively, was performed by P_i. Analogously, we write $u\ {}_l\Leftrightarrow^*_{P_i}\ v$, $u\ {}^h_m\Leftrightarrow^*_{P_i}\ v$, $u\ {}_m\Leftrightarrow^*_{P_i}\ v$, $u\ {}_l\Leftrightarrow^+_{P_i}\ v$, $u\ {}^h_m\Leftrightarrow^+_{P_i}\ v$, $u\ {}_m\Leftrightarrow^+_{P_i}\ v$, $u\ {}^h_m\Leftrightarrow^t_{P_i}\ v$, and $u\ {}_m\Leftrightarrow^t_{P_i}\ v$.

Let $\Gamma = (N, T, S, P_1, \ldots, P_n)$ be a CD grammar system with phrase-structure grammars as its component and C be a control language. Then, ${}_lL^C(\Gamma) = \{w \in T^* |\ S\ {}_l\Leftrightarrow^t_{P_{i_1}}$

$w_1 \; {}_{l}{\diamond\!\!\Rightarrow}^{t}_{P_{i_2}} \cdots {}_{l}{\diamond\!\!\Rightarrow}^{t}_{P_{i_p}} \; w_p = w, p \geq 1, 1 \leq i_j \leq n, 1 \leq j \leq p, \; i_1 i_2 \ldots i_p \in C\}, {}_{\mathrm{N}}L^C(\Gamma, m, h) = \{w \in T^* \mid S \; {}^{h}_{m}{\circ\!\!\Rightarrow}^{t}_{P_{i_1}} \; w_1 \; {}^{h}_{m}{\circ\!\!\Rightarrow}^{t}_{P_{i_2}} \cdots {}^{h}_{m}{\circ\!\!\Rightarrow}^{t}_{P_{i_p}} \; w_p = w, p \geq 1, 1 \leq i_j \leq n, 1 \leq j \leq p, \; i_1 i_2 \ldots i_p \in C\}, {}_{\mathrm{N}}L^C(\Gamma, m) = \{w \in T^* \mid S \; {}_{m}{\circ\!\!\Rightarrow}^{t}_{P_{i_1}} \; w_1 \; {}_{m}{\circ\!\!\Rightarrow}^{t}_{P_{i_2}} \cdots {}_{m}{\circ\!\!\Rightarrow}^{t}_{P_{i_p}} \; w_p = w, p \geq 1, 1 \leq i_j \leq n, 1 \leq j \leq p, \; i_1 i_2 \ldots i_p \in C\}$.

Let $l, m, h \in \mathbb{N}$ and let $\Gamma = (N, T, S, P_1, \ldots, P_n)$ be a CD grammar system with phrase-structure grammars. We define the following classes of languages.

$$\mathscr{L}({}_{\mathrm{l}}\mathrm{CD}^{\mathbf{REG}}) = \{{}_{\mathrm{l}}L^C(\Gamma) \mid C \in \mathbf{REG}\}$$

$$\mathscr{L}({}_{\mathrm{N}}\mathrm{CD}^{\mathbf{REG}}(m, h)) = \{{}_{\mathrm{N}}L^C(\Gamma, m, h) \mid C \in \mathbf{REG}\}$$

$$\mathscr{L}({}_{\mathrm{N}}\mathrm{CD}^{\mathbf{REG}}(m)) = \{{}_{\mathrm{N}}L^C(\Gamma, m) \mid C \in \mathbf{REG}\}$$

For these classes, the following theorems are established.

Theorem 3.20 Let $l \in \mathbb{N}$. Then, $\mathbf{CF} = \mathscr{L}({}_{\mathrm{l}}\mathrm{CD}^{\mathbf{REG}})$.

Theorem 3.21 $\mathbf{RE} = \mathscr{L}({}_{\mathrm{N}}\mathrm{CD}^{\mathbf{REG}}(1))$.

Theorem 3.22 $\mathscr{L}_m(\mathrm{P}) = \mathscr{L}({}_{\mathrm{N}}\mathrm{CD}^{\mathbf{REG}}(m, h))$, for any $m \geq 1$ and $h \geq 1$.

3.4.2 Parallel Systems of Formal Models

In my thesis, we introduce two n-accepting restricted pushdown automata systems representing automata counterpart of multi-generative grammar systems (see Section 3.2). First, we define *n-accepting state-restricted pushdown automata systems*. By using prescribed n-state sequences, the restrictions of these systems determines which of the components perform a move and which of them do not. Second, we define *n-accepting move-restricted pushdown automata systems*, where the restriction precisely determines which transition rule can be used in each of the n components. Both of these systems define sets of n-tuples of strings (n-languages).

After that, we generalize the theory of n-languages and discussed *hybrid canonical rule-synchronized n-generative grammar systems* and *hybrid n-accepting move-restricted automata systems*, where components with different generative and accepting power can be used in one grammar and automata system, respectively. More specifically, we investigate grammar systems, which combine right-linear grammars, linear grammars, and context-free grammars; and automata systems, which combine finite automata, 1-turn pushdown automata, and pushdown automata in one instance.

A *hybrid canonical rule-synchronized n-generative grammar system*, $\mathrm{HCGR}^{(t_1, \ldots, t_n)}$ for short, is an $n+1$-tuple $\Gamma = (G_1, \ldots, G_n, Q)$, where

- $G_i = (N_i, T_i, P_i, S_i)$ is a right-linear, linear, or context-free grammar for every $i = 1, \ldots, n$,
- Q is a finite set of n-tuples of the form $(r_1, \ldots, r_n)$, where $r_i \in P_i$ for every $i = 1, \ldots, n$, and
- for all $i = 1, \ldots, n$, $t_i \in \{\mathrm{RLNG}, \mathrm{LNG}, \mathrm{CFG}\}$ denotes type of ith component.

A *sentential n-form* of $\mathrm{HCGR}^{(t_1,\dots,t_n)}$ is an n-tuple $\chi = (x_1, \dots, x_n)$, where $x_i \in (N_i \cup T_i)^*$ for all $i = 1, \dots, n$.

Consider sentential n-forms, $\chi = (u_1A_1v_1, \dots, u_nA_nv_n)$ and $\chi' = (u_1x_1v_1, \dots, u_nx_nv_n)$ with

- $A_i \in N_i$,
- $u_i \in T^*$,
- $v_i, x_i \in (N \cup T)^*$,
- $r_i = A_i \to x_i \in P_i$, for all $i = 1, \dots, n$, and
- $(r_1, \dots, r_n) \in Q$.

Then, $\chi \Rightarrow \chi'$, and $\Rightarrow^*$ and $\Rightarrow^+$ are its reflexive-transitive and transitive closure, respectively.

The *n-language of* Γ is defined as $n\text{-}L(\Gamma) = \{(w_1, \dots, w_n) | \ (S_1, \dots, S_n) \Rightarrow^* (w_1, \dots, w_n), w_i \in T_i^*, \text{ for all } 1 \leq i \leq n\}$.

A hybrid *n-accepting move-restricted automata system*, denoted $\mathrm{HMAS}^{(t_1,\dots,t_n)}$, is defined as an $n+1$-tuple $\vartheta = (M_1 \dots, M_n, \Psi)$ with M_i as a finite or (1-turn) pushdown automaton for all $i = 1, \dots, n$, and with Ψ as a finite set of n-tuples of the form $(r_1, \dots, r_n)$, where for every $j = 1, \dots, n$, $r_j \in \delta_j$ in M_j. Furthermore, for all $i = 1, \dots, n$, $t_i \in \{\text{FA, 1-turn PDA, PDA}\}$ indicates the type of ith automaton.

An n-configuration is defined as an n-tuple $\chi = (x_1, \dots, x_n)$, where for all $i = 1, \dots, n$, x_i is a configuration of M_i. Let $\chi = (x_1, \dots, x_n)$ and $\chi' = (x'_1, \dots, x'_n)$ be two n-configurations, where for all $i = 1, \dots, n$, $x_i \Rightarrow x'_i \, [r_i]$ in M_i, and $(r_1, \dots, r_n) \in \Psi$, then ϑ makes computation steps from n-configuration χ to n-configuration χ', denoted $\chi \Rightarrow \chi'$, and in the standard way, $\Rightarrow^*$ and $\Rightarrow^+$ denote the reflexive-transitive and the transitive closure of $\Rightarrow$, respectively.

Let $\chi_0 = (x_1\omega_1, \dots, x_n\omega_n)$ be the start and $\chi_f = (q_1, \dots, q_n)$ be a final n-configuration of $\mathrm{HMAS}^{(t_1 \dots, t_n)}$, where for all $i = 1, \dots, n$, ω_i is the input string of M_i and q_i is state of M_i. The n-language of $\mathrm{HMAS}^{(t_1,\dots,t_n)}$ is defined as $n\text{-}L(\vartheta) = \{(\omega_1, \dots, \omega_n) | \ \chi_0 \Rightarrow^* \chi_f$ and for every $i = 1, \dots, n$, M_i accepts$\}$.

In a special case, where all components are of type X, we write ${}_nX$ instead of $(X, \dots, X)$. If there is no attention on the number and type of components, we write HMAS and HCGR rather than $\mathrm{HMAS}^{(t_1,\dots,t_n)}$ and $\mathrm{HCGR}^{(t_1,\dots,t_n)}$, respectively.

$\mathscr{L}(\mathrm{HMAS}^{(t_1,\dots,t_n)})$ and $\mathscr{L}(\mathrm{HCGR}^{(t_1,\dots,t_n)})$ are the classes of n-languages accepted by $\mathrm{HMAS}^{(t_1,\dots,t_n)}$ and n-languages generated by $\mathrm{HCGR}^{(t_1,\dots,t_n)}$, respectively.

The basic hierarchy of such systems is given by Figure 3.2.

3.4.3 Rule-Restricted Transducers

In formal language theory, there exist two basic translation-method categories. The first category contains interprets and compilers, which first analyse an input string in the source language and, after that, they generate a corresponding output string in the target language (see [1], [60], [64], [43], or [77]). The second category is composed of language-translation systems or, more briefly, transducers. Frequently, these trasducers consist of several components, including various automata and grammars, some of which read their input strings while others produce their output strings (see [4], [40], [63], and [79]).

$$\begin{tikzcd}
 & \mathscr{L}(\mathrm{HMAS}^{n\,\mathrm{PDA}}) \arrow[d, leftrightarrow] & & \\
 & \mathscr{L}(\mathrm{HMAS}^{(\mathrm{FA},\dots,\mathrm{FA},\text{1-turn PDA},\text{1-turn PDA})}) \arrow[dl, leftrightarrow] & & \\
\mathscr{L}(\mathrm{HMAS}^{(\mathrm{FA},\dots,\mathrm{FA},\mathrm{PDA},\mathrm{PDA})}) \arrow[rrr, leftrightarrow] & & & \mathscr{L}(\mathrm{HCGR}^{(\mathrm{RLNG},\dots,\mathrm{RLNG},\mathrm{CFG},\mathrm{CFG})}) \\
 & \mathscr{L}(\mathrm{HMAS}^{(\mathrm{FA},\dots,\mathrm{FA},\mathrm{PDA})}) \arrow[ul] \arrow[rr, leftrightarrow] & & \mathscr{L}(\mathrm{HCGR}^{(\mathrm{RLNG},\dots,\mathrm{RLNG},\mathrm{CFG})}) \\
 & & \mathscr{L}(\mathrm{HCGR}^{n\,\mathrm{LNG}}) \arrow[uuul] \arrow[ur] & \\
 & \mathscr{L}(\mathrm{HMAS}^{n\,\mathrm{FA}}) \arrow[uu] \arrow[rr, leftrightarrow] & & \mathscr{L}(\mathrm{HCGR}^{n\,\mathrm{RLNG}}) \arrow[ul]
\end{tikzcd}$$

Figure 3.2: Hierarchy of n-languages for $n \geq 2$

Although transducers represent language-translation devices, language theory often views them as language-defining devices and investigates the language family resulting from them. In essence, it studies their accepting power consisting in determining the language families accepted by the transducer components that read their input strings. Alternatively, it establishes their generative power that determines the language family generated by the components that produce their strings. The thesis contributes to this vivid investigation trend in formal language theory.

In this section, we introduce three new variants of transducer, referred to as rule-restricted transducer, based upon a finite automaton and a context-free grammar. In addition, a restriction set controls the rules which can be simultaneously used by the automaton and by the grammar.

An *rule-restricted transducer*, RT for short, is a triplet $\Gamma = (M, G, \Psi)$, where $M = (Q, \Sigma, \delta, q_0, F)$ is a finite automaton, $G = (N, T, P, S)$ is a context-free grammar, and Ψ is a finite set of pairs of the form (r_1, r_2), where r_1 and r_2 are rules from δ and P, respectively.

A 2*-configuration* of RT is a pair $\chi = (x, y)$, where $x \in Q\Sigma^*$ and $y \in (N \cup T)^*$. Consider two 2-configurations, $\chi = (pav_1, uAv_2)$ and $\chi' = (qv_1, uxv_2)$ with $A \in N$, $u, v_2, x \in (N \cup T)^*$, $v_1 \in \Sigma^*$, $a \in \Sigma \cup \{\varepsilon\}$, and $p, q \in Q$. If $pav_1 \Rightarrow qv_1[r_1]$ in M, $uAv_2 \Rightarrow uxv_2[r_2]$ in G, and $(r_1, r_2) \in \Psi$, then Γ makes a computation step from χ' to χ', written as $\chi \Rightarrow \chi'$. In the standard way, $\Rightarrow^*$ and $\Rightarrow^+$ are transitive-reflexive and transitive closure of $\Rightarrow$, respectively.

The 2*-language of* Γ, 2-$L(\Gamma)$, is 2-$L(\Gamma) = \{(w_1, w_2) | \ (q_0w_1,\ S) \Rightarrow^* (f, w_2),\ w_1 \in \Sigma^*,\ w_2 \in T^*, \text{ and } f \in F\}$. From the 2-language we can define two languages:

- $L(\Gamma)_1 = \{w_1 | \ (w_1, w_2) \in 2\text{-}L(\Gamma)\}$, and
- $L(\Gamma)_2 = \{w_2 | \ (w_1, w_2) \in 2\text{-}L(\Gamma)\}$.

By $\mathscr{L}(RT)$, $\mathscr{L}(RT)_1$, and $\mathscr{L}(RT)_2$, the classes of 2-languages of RTs, languages accepted by M in RTs, and languages generated by G in RTs, respectively, are understood. The generative and accepting power are given by the following theorems.

Theorem 3.23 $\mathscr{L}(RT)_2 = \mathscr{L}(MAT)$.

Theorem 3.24 $\mathscr{L}(RT)_1 = \bigcup_{k=1}^{\infty} \mathscr{L}(k\text{-}PBCA)$.

Although the investigated system is relatively powerful, in defiance of weakness of models they are used, non-deterministic selections of nonterminals to be rewritten can be relatively problematic from the practical point of view. Therefore, the effect of a restriction, in the form of leftmost derivations placed on the grammar in RTs, has been examined.

Let $\Gamma = (M, G, \Psi)$ be an RT with $M = (Q, \Sigma, \delta, q_0, F)$ and $G = (N, T, P, S)$. Furthermore, let $\chi = (pav_1, uAv_2)$ and $\chi' = (qv_1, uxv_2)$ be two 2-configurations, where $A \in N$, $v_2, x \in (N \cup T)^*$, $u \in T^*$, $v_1 \in \Sigma^*$, $a \in \Sigma \cup \{\varepsilon\}$, and $p, q \in Q$. Γ makes a computation step from χ to χ', written as $\chi \Rightarrow_{lm} \chi'$, if and only if $pav_1 \Rightarrow qv_1[r_1]$ in M, $uAv_2 \Rightarrow uxv_2[r_2]$ in G, and $(r_1, r_2) \in \Psi$. In the standard way, $\Rightarrow_{lm}^*$ and $\Rightarrow_{lm}^+$ are transitive-reflexive and transitive closure of $\Rightarrow_{lm}$, respectively.

The 2-*language of* Γ with G generating in the leftmost way, denoted by 2-$L_{lm}(\Gamma)$, is defined as 2-$L_{lm}(\Gamma) = \{(w_1, w_2) |\ (q_0w_1, S) \Rightarrow_{lm}^* (f, w_2),\ w_1 \in \Sigma^*,\ w_2 \in T^*, \text{ and } f \in F\}$; we call Γ as *leftmost restricted RT*; and we define the languages given from 2-$L_{lm}(\Gamma)$ as $L_{lm}(\Gamma)_1 = \{w_1 |\ (w_1, w_2) \in$ 2-$L_{lm}(\Gamma)\}$ and $L_{lm}(\Gamma)_2 = \{w_2 |\ (w_1, w_2) \in$ 2-$L_{lm}(\Gamma)\}$. By $\mathscr{L}(RT_{lm})$, $\mathscr{L}(RT_{lm})_1$, and $\mathscr{L}(RT_{lm})_2$, we understand the classes of 2-languages of leftmost restricted RTs, languages accepted by M in leftmost restricted RTs, and languages generated by G in leftmost restricted RTs, respectively. The leftmost restriction effects the generative and accepting power as the following theorem says.

Theorem 3.25 $\mathscr{L}(RT_{lm})_2 = \mathbf{CF}$ and $\mathscr{L}(RT_{lm})_1 = \mathbf{CF}$.

Unfortunately, the price for the leftmost restriction, placed on derivations in the context-free grammar, is relatively high and both accepting and generative ability of RT with the restriction decreases to **CF**.

In the thesis, RTs have been extended with the possibility to prefer a rule over another—that is, the restriction sets contain triplets of rules (instead of pairs of rules), where the first rule is a rule of FA, the second rule is a main rule of CFG, and the third rule is an alternative rule of CFG, which is used only if the main rule is not applicable.

An *RT with appearance checking*, RT$_{\text{ac}}$ for short, is a triplet $\Gamma = (M, G, \Psi)$, where $M = (Q, \Sigma, \delta, q_0, F)$ is a finite automaton, $G = (N, T, P, S)$ is a context-free grammar, and Ψ is a finite set of triplets of the form (r_1, r_2, r_3) such that $r_1 \in \delta$ and $r_2, r_3 \in P$.

Let $\chi = (pav_1, uAv_2)$ and $\chi' = (qv_1, uxv_2)$, where $A \in N$, $v_2, x, u \in (N \cup T)^*$, $v_1 \in \Sigma^*$, $a \in \Sigma \cup \{\varepsilon\}$, and $p, q \in Q$, be two 2-configurations. Γ makes a computation step from χ to χ', written as $\chi \Rightarrow \chi'$, if and only if for some $(r_1, r_2, r_3) \in \Psi$, $pav_1 \Rightarrow qv_1[r_1]$ in M, and either

- $uAv_2 \Rightarrow uxv_2[r_2]$ in G, or
- $uAv_2 \Rightarrow uxv_2[r_3]$ in G and r_2 is not applicable on uAv_2 in G.

The 2-language 2-$L(\Gamma)$ and languages $L(\Gamma)_1, L(\Gamma)_2$ are defined in the same way as usual. The classes of languages defined by the first and the second component in the system are denoted by $\mathscr{L}(RT_{ac})_1$ and $\mathscr{L}(RT_{ac})_2$, respectively. The power of the RTs with appearance checking is declared by the following theorem.

Theorem 3.26 $\mathscr{L}(RT_{ac})_2 = \mathbf{RE}$ and $\mathscr{L}(RT_{ac})_1 = \mathbf{RE}$.

3.5 Thesis Summary and Further Investigation

My PhD thesis discusses and studies formal languages and systems of formal models. Its main results are published or submitted in [16, 56, 8, 9, 10, 11, 17, 12, 14, 13, 15]. This section summaries these results.

The thesis was focused on a study of systems of formal models which plays important role in the modern information technology and computer science. Since the introduction of CD grammar systems, many other systems were studied and systems of formal models have become a vivid research area. Aim of the thesis was to further investigate properties of the systems of formal models to their better understanding. This research can be divided into several main parts.

In the first part, we continued in studying of regularly controlled CD grammar systems, where we used phrase-structure grammars as components, and introduced three new restrictions on derivations in these systems. The first restriction requires that derivation rules could be applied within the first l nonterminals, for given $l \geq 1$. Although phrase-structured grammars define all languages from **RE**, regularly controlled CD grammar systems with phrase-structure grammars as components under such restriction generate only context-free languages. One may ask, how strong the control language must be to leave the generative power unchanged. Our assumption is that linear languages are sufficient, but a rigorous proof has not yet been done. The second restriction allows to have only limited number of undivided blocks of nonterminals in each sentential form during any successful derivation. It has been proven that this restriction has no effect on the generative power of these CD grammar systems even in the case when the restriction allows only one such block. On the other hand, the restriction limiting the maximum length and number of the blocks decreases the generative power of these systems to the classes $\mathscr{L}_m(P)$ representing infinite hierarchy, with respect of m, lying between the classes of linear and context-sensitive languages. Notice that m is maximal number of blocks and $\mathbf{CF} - \mathscr{L}_m(P) \neq \emptyset$. Question whether the stronger control language effects the generative power of CD grammar systems with phrase-structure grammars subject to the third restriction is still open.

The second part deals with parallel grammar and automata systems based upon CFGs and PDAs, respectively. More specifically, we introduced two variants of n-accepting restricted pushdown automata systems, accepting n-tuples of interdependent strings, as counterparts of canonical n-generating nonterminal/rule synchronized grammar systems based upon context-free grammars. Both types of the automata systems consist of n PDAs, for $n \geq 2$, and one restriction-set. In the case of n-accepting state-restricted automata systems, the restriction-set allows to suspend and resume some automata during computation in relation to combination of current states of the PDAs. In the case of n-accepting move-restricted automata systems, the restriction-set determines which combination of transition rules used in the common computation step are permitted. We have proven that these n-accepting restricted automata systems are able to accept such n-languages that the canonical n-generating grammar systems can generate and vice versa. Furthermore, we have established fundamental hierarchy of n-languages generating/accepting by these canonical multi-generating rule synchronized grammar/n-accepting rule-restricted automata systems with different types of components. First of all, we have shown that both these systems are equivalent even if we combine RLNGs with CFGs in the grammar systems and FAs with PDAs in the automata systems. After that, we have established the hierarchy given by Figure 3.2 ($\rightarrow$ and $\leftrightarrow$ mean $\subset$ and $=$, respectively), where it can be seen, inter alia, that canonical n-generating rule synchronized grammar systems based upon linear grammars are significantly weaker than n-accepting move-restricted automata systems, with two 1-turn PDAs and $n-2$ FAs as components.

The second part of the research can be continued by better approximation of power of the state/move-restricted automata systems based upon FAs (especially in relation to string-interdependences), or by investigation of restarting and/or stateless finite and pushdown automata as the components of discussed automata systems.

In the last part, we have suggested rule-restricted systems for processing of linguistically motivated languages. In this part, we introduced three variants of rule-restricted translating systems based upon finite automaton and context-free grammar. At first, we have proven that leftmost restriction placed on derivation in the context-free grammar effects both the generative and accepting power of such systems. In addition, we introduced a rule-restricted transducer system with appearance checking, where the restriction-set Ψ is a set of 3-tuples containing one rule of the FA and two rules of the CFG. For the common computation step, the system has to use the first and second rules of a 3-tuple, if it is possible; otherwise, it can use the first and third rules from the 3-tuple. This system is able to recognize and generate any language from **RE**. Thereafter, some examples of natural language translating are given.

The investigation of processing of linguistically motivated languages continued by generalization of TC grammars that generate the language under path-based control introduced in [47]. We have considered TC grammars that generate their languages under n-path control by linear language which were introduced in [45].

We have demonstrated that for $L \in$ **n-path-TC** under assumption that L is generated by TC grammar (G, R) in which G and R are unambiguous and, furthermore, G is restricted to be LL grammar, there is parsing method working in polynomial time. This method check whether or not the paths of the derivation tree t of $x \in L(G)$ belongs to control language R in the time of building of t. Moreover, when we consider LR parser for $L \in$ **n-path-TC** under assumption that L is generated by TC grammar (G, R) in which G has bounded ambiguity (i.e. G is unambiguous or m-ambiguous) and unambiguous language $R \in$ **LIN**, there is also a parsing method working in polynomial time.

However, the open question is whether there is polynomial time parsing method

- if G is not LL,
- if G is ambiguous.

It is also of interest to quantify the worst case of the parsing complexity more precisely.

The open investigation area is represented by the transformation of n-path TC grammars into some normal forms based on Chomsky normal form of underlying context-free grammar which would lead to possibility to use parsing methods based on transformation to Chomsky normal form.

Chapter 4

Abstract

My PhD thesis continues with study of grammar and automata systems. First of all, it deals with regularly controlled CD grammar systems with phrase-structure grammars as components. Into these systems, three new derivation restriction are placed and their effect on the generative power of these systems are investigated. Thereafter, the thesis defines two automata counterparts of canonical multi-generative nonterminal and rule synchronized grammar systems, generating vectors of strings, and it shows that these investigated systems are equivalent. Furthermore, the thesis generalizes definitions of these systems and establishes fundamental hierarchy of n-languages (sets of n-tuples of strings). In relation with these mentioned systems, automaton-grammar translating systems based upon finite automaton and context-free grammar are introduced and investigated as a mechanism for direct translating. At the end, in the thesis introduced automata systems are used as the core of parse-method based upon n-path-restricted tree-controlled grammars.

Bibliography

[1] A.V. Aho. *Compilers: principles, techniques, & tools.* Pearson/Addison Wesley, 2007.

[2] M. Amin and R. Abd el Mouaty. Stratified grammar systems with simple and dynamically organized strata. *Computers and Artificial Intelligence*, 23(4):355–362, 2004.

[3] B. S. Baker. Context-sesitive grammars generating context-free languages. In M. Nivat, editor, *Automata, Languages and Programming*, pages 501–506. North-Holland, Amsterdam, 1972.

[4] M. Bál, O. Carton, C. Prieur, and J. Sakarovitch. Squaring transducers: an efficient procedure for deciding functionality and sequentiality. *Theoretical Computer Science*, 292(1):45 – 63, 2003.

[5] M. H. Beek and H. C. M. Kleijn. Petri net control for grammar systems. In *Formal and Natural Computing - Essays Dedicated to Grzegorz Rozenberg [on occasion of his 60th birthday, March 14, 2002]*, pages 220–243, London, UK, UK, 2002. Springer-Verlag.

[6] R. V. Book. Terminal context in context-sensitive grammars. *SIAM Journal of Computing*, 1:20–30, 1972.

[7] A. O. Buda. Multiprocessor automata. *Inf. Process. Lett.*, 25(4):257–261, June 1987.

[8] M. Čermák. Systems of formal models and their application. In *Proceedings of the 14th Conference Student EEICT 2008*, Volume 2, pages 164–166. Faculty of Electrical Engineering and Communication BUT, 2008.

[9] M. Čermák. Power decreasing derivation restriction in grammar systems. In *Proceedings of the 15th Conference and Competition STUDENT EEICT 2009 Volume 4*, pages 385–389. Faculty of Information Technology BUT, 2009.

[10] M. Čermák. Multilanguages and multiaccepting automata system. In *Proceedings of the 16th Conference and Competition STUDENT EEICT 2010 Volume 5*, pages 146–150. Faculty of Information Technology BUT, 2010.

[11] M. Čermák. Basic properties of n-languages. In *Proceedings of the 17th Conference and Competition STUDENT EEICT 2011 Volume 3*, pages 460–464. Faculty of Information Technology BUT, 2011.

[12] M. Čermák. Restrictions on derivations in n-generating grammar systems. In *Proceedings of the 18th Conference and Competition STUDENT EEICT 2012 Volume 5*, pages 371–375. Faculty of Information Technology BUT, 2012.

[13] M. Čermák, P. Horáček, and A. Meduna. Rule-restricted automaton-grammar transducers: Power and linguistic applications. *Mathematics for Applications*, 2012, in press.

[14] M. Čermák, J. Koutný, and A. Meduna. Parsing based on n-path tree-controlled grammars. *Theoretical and Applied Informatics*, 2011(23):213–228, 2012.

[15] M. Čermák, J. Koutný, and A. Meduna. On n-language classes hierarchy. *Acta Cybernetica*, 2012, sumbited.

[16] M. Čermák and A. Meduna. n-Accepting restricted pushdown automata systems. In *13th International Conference on Automata and Formal Languages*, pages 168–183. Computer and Automation Research Institute, Hungarian Academy of Sciences, 2011.

[17] M. Čermák and A. Meduna. n-Accepting restricted pushdown automata systems. In *7th Doctoral Workshop on Mathematical and Engineering Methods in Computer Science*, pages 110–110. Brno University of Technology, 2011.

[18] E. Csuhaj-Varjú and J. Dassow. On cooperating/distributed grammar systems. *Elektronische Informationsverarbeitung und Kybernetik*, 26(1/2):49–63, 1990.

[19] E. Csuhaj-Varjú, J. Dassow, and G. Păun. Dynamically controlled cooperating/distributed grammar systems. *Inf. Sci.*, 69(1-2):1–25, April 1993.

[20] E. Csuhaj-Varjú, J. Kelemen, and G. Păun. Grammar systems with wave-like communication. *Computers and Artificial Intelligence*, 15(5), 1996.

[21] E. Csuhaj-Varju, J. Kelemen, G. Păun, and J. Dassow, editors. *Grammar Systems: A Grammatical Approach to Distribution and Cooperation.* Gordon and Breach Science Publishers, Inc., Newark, NJ, USA, 1st edition, 1994.

[22] E. Csuhaj-Varjú, C. Martín-Vide, V. Mitrana, and G. Vaszil. Parallel communicating pushdown automata systems. *Int. J. Found. Comput. Sci.*, 11(4):633–650, 2000.

[23] E. Csuhaj-Varjú, T. Masopust, and G. Vaszil. Cooperating distributed grammar systems with permitting grammars as components. *Romanian Journal of Information Science and Technology (ROMJIST)*, 12(2):175–189, 2009.

[24] E. Csuhaj-Varjú, V. Mitrana, and G. Vaszil. Distributed pushdown automata systems: computational power. In *Proceedings of the 7th international conference on Developments in language theory*, DLT'03, pages 218–229, Berlin, Heidelberg, 2003. Springer-Verlag.

[25] E. Csuhaj-Varjú and G. Vaszil. On context-free parallel communicating grammar systems: synchronization, communication, and normal forms. *Theor. Comput. Sci.*, 255(1-2):511–538, 2001.

[26] E. Czeizler and E. Czeizler. On the power of parallel communicating watson-crick automata systems. *Theoretical Computer Science*, 358(1):142 – 147, 2006.

[27] J. Dassow. Cooperating/distributed grammar systems with hypothesis languages. *J. Exp. Theor. Artif. Intell.*, 3(1):11–16, 1991.

[28] J. Dassow, H. Fernau, and Gh. Păun. On the leftmost derivation in matrix grammars. *International Journal of Foundations of Computer Science*, 10(1):61–80, 1999.

[29] J. Dassow and V. Mitrana. Stack cooperation in multistack pushdown automata. *Journal of Computer and System Sciences*, 58(3):611 – 621, 1999.

[30] J. Dassow and Gh. Păun. *Regulated Rewriting in Formal Language Theory.* Springer, Berlin, 1989.

[31] J. Dassow and G. Păun. Cooperating/distributed grammar systems with registers: the regular case. *Found. Control Eng.*, 15:19–38, 1990.

[32] S. Dumitrescu. Characterization of RE using CD grammar systems with two registers and RL rules. In *New Trends in Formal Languages*, volume 1218 of *Lecture Notes in Computer Science*, pages 167–177. Springer Berlin / Heidelberg, 1997.

[33] H. Fernau. Regulated grammars under leftmost derivation. *Grammars*, 3(1):37–62, 2000.

[34] H. Fernau, M. Holzer, and R. Freund. External versus internal hybridization for cooperating distributed grammar systems, 1996.

[35] H. Fernau and R. Stiebe. On the expressive power of valences in cooperating distributed grammar systems. In *Computation, Cooperation, and Life*, volume 6610 of *Lecture Notes in Computer Science*, pages 90–106. Springer Berlin / Heidelberg, 2011.

[36] M. Frazier and C. D. Page. Prefix grammars: An alternative characterization of the regular languages. *Information Processing Letters*, 51(2):67–71, 1994.

[37] R. Freund and G. Păun. A variant of team cooperation in grammar systems. *j-jucs*, 1(2):105–130, 1995.

[38] S. Ginsburg and S. Greibach. Mappings which preserve context-sensitive languages. *Information and Control*, 9:563–582, 1966.

[39] F. Goldefus, T. Masopust, and A. Meduna. Left-forbidding cooperating distributed grammar systems. *Theor. Comput. Sci.*, 411(40-42):3661–3667, September 2010.

[40] E. M. Gurari and O. H. Ibarra. A note on finite-valued and finitely ambiguous transducers. *Theory of Computing Systems*, 16:61–66, 1983. 10.1007/BF01744569.

[41] T. N. Hibbard. Context-limited grammars. *Journal of the ACM*, 21:446–453, 1974.

[42] L. Ilie. Collapsing hierarchies in parallel communicating grammar systems with communication by command, 1996.

[43] O. Jirák and Z. Křivka. Design and implementation of back-end for picoblaze C compiler. In *Proceedings of the IADIS International Conference Applied Computing 2009*, pages 135–138. International Association for Development of the Information Society, 2009.

[44] L. Kari, A. Mateescu, G. Păun, and A. Salomaa. Teams in cooperating grammar systems. *Journal of Experimental & Theoretical Artificial Intelligence*, 7(4):347–359, 1995.

[45] J. Koutný, Z. Křivka, and A. Meduna. Pumping properties of path-restricted tree-controlled languages. In *7th Doctoral Workshop on Mathematical and Engineering Methods in Computer Science*, pages 61–69. Brno University of Technology, 2011.

[46] R. Lukáš and A. Meduna. Multigenerative grammar systems and matrix grammars. *Kybernetika*, 46(1):68–82, 2010.

[47] S. Marcus, C. Martín-Vide, V. Mitrana, and Gh. Păun. A new-old class of linguistically motivated regulated grammars. In *Walter Daelemans, Khalil Sima'an, Jorn Veenstra, Jakub Zavrel (Eds.): Computational Linguistics in the Netherlands 2000, Selected Papers from the Eleventh CLIN Meeting, Tilburg*, pages 111–125. Language and Computers - Studies in Practical Linguistics 37 Rodopi 2000, 2000.

[48] C. Martín-Vide, A. Mateescu, and V. Mitrana. Parallel finite automata systems communicating by states. *Int. J. Found. Comput. Sci.*, 13(5):733–749, 2002.

[49] T. Masopust. On the terminating derivation mode in cooperating distributed grammar systems with forbidding components. *International Journal of Foundations of Computer Science*, 20(2):331–340, 2009.

[50] G. Matthews. A note on symmetry in phrase structure grammars. *Information and Control*, 7:360–365, 1964.

[51] G. Matthews. Two-way languages. *Information and Control*, 10:111–119, 1967.

[52] A. Meduna. Matrix grammars under leftmost and rightmost restrictions. In Gh. Păun, editor, *Mathematical Linguistics and Related Topics*, pages 243–257. Romanian Academy of Sciences, Bucharest, 1994.

[53] A. Meduna. On the number of nonterminals in matrix grammars with leftmost derivations. In *New Trends in Formal Languages: Control, Cooperation, and Combinatorics*, pages 27–39. Springer-Verlag, 1997.

[54] A. Meduna. *Automata and Languages: Theory and Applications.* Springer, 2000.

[55] A. Meduna and R. Lukáš. Multigenerative grammar systems. *Schedae Informaticae*, 2006(15):175–188, 2006.

[56] A. Meduna, M. Čermák, and T. Masopust. Some power-decreasing derivation restrictions in grammar systems. *Schedae Informaticae*, 2010(19):23–34, 2011.

[57] R. Meersman and G. Rozenberg. *Cooperating Grammar Systems.* 1978.

[58] H. Messerschmidt and F. Otto. Strictly deterministic CD-systems of restarting automata. In Erzsébet Csuhaj-Varjú and Zoltán Ésik, editors, *Fundamentals of Computation Theory*, volume 4639 of *Lecture Notes in Computer Science*, pages 424–434. Springer Berlin / Heidelberg, 2007.

[59] V. Mihalache, V. Mitrana, T. Centre, and Computer Science. Deterministic cooperating distributed grammar systems. *Computers and AI*, 2:20, 1996.

[60] T. Mine, R. Taniguchi, and M. Amamiya. Coordinated morphological and syntactic analysis of japanese language. In *Proceedings of the 12th international joint conference on Artificial intelligence - Volume 2*, pages 1012–1017. Morgan Kaufmann Publishers Inc., 1991.

[61] R. Mitkov. *The Oxford Handbook of Computational Linguistics.* Oxford University Press, 2003.

[62] V. Mitrana. Hybrid cooperating/distributed grammar systems. *Computers and artificial intelligence*, 12(1):83–88, 1993.

[63] M. Mohri. Finite-state transducers in language and speech processing. *Comput. Linguist.*, 23(2):269–311, June 1997.

[64] S. Muchnick. *Advanced compiler design and implementation.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997.

[65] B. Nagy. On $5' \rightarrow 3'$ sensing watson-crick finite automata. In Max Garzon and Hao Yan, editors, *DNA Computing*, volume 4848 of *Lecture Notes in Computer Science*, pages 256–262. Springer Berlin / Heidelberg, 2008.

[66] F. Otto. CD-systems of restarting automata governed by explicit enable and disable conditions. In *SOFSEM 2010: Theory and Practice of Computer Science*, volume 5901 of *Lecture Notes in Computer Science*, pages 627–638. Springer Berlin / Heidelberg, 2010.

[67] G. Păun. On the synchronization in parallel communicating grammar systems. *Acta Inf.*, 30(4):351–367, 1993.

[68] G. Păun. On the generative capacity of hybrid CD grammar systems. *Elektronische Informationsverarbeitung und Kybernetik*, pages 231–244, 1994.

[69] G. Păun and G. Rozenberg. Prescribed teams of grammars. *Acta Informatica*, 31:525–537, 1994.

[70] G. Păun and L. Sântean. Parallel communicating grammar systems– the regular case. *Ann. Univ. Buc. Ser. Mat.-Inform*, 2:55–63, 1989.

[71] A. L. Rosenberg. On multi-head finite automata. *IBM J. Res. Dev.*, 10(5):388–394, September 1966.

[72] G. Rozenberg and A. Salomaa, editors. *Handbook of Formal Languages*, volume 2. Springer-Verlag, Berlin, 1997.

[73] G. Rozenberg and A. Salomaa, editors. *Handbook of Formal Languages*, volume 1. Springer-Verlag, Berlin, 1997.

[74] M. H. ter Beek, E. Csuhaj-Varjú, and V. Mitrana. Teams of pushdown automata. *Int. J. Comput. Math.*, 81(2):141–156, 2004.

[75] M. H. ter Beek and J Kleijn. Team automata satisfying compositionality. In Keijiro Araki, Stefania Gnesi, and Dino Mandrioli, editors, *FME 2003: Formal Methods*, volume 2805 of *Lecture Notes in Computer Science*, pages 381–400. Springer Berlin / Heidelberg, 2003.

[76] S. Vicolov. Cooperating/distributed grammar systems with registers: the regular case. *Computers and artificial intelligence*, 12:89–98, 1993.

[77] P. Šaloun. Parallel LR parsing. In *Proceedings of the Fifth International Scientific Conference Electronic Computers and Informatics 2002*. The University of Technology Košice, 2002.

[78] D. Wajten. On cooperating–distributed extended limited 0l systems. *International Journal of Computer Mathematics*, 63:227–244, 1997.

[79] A. Weber. On the valuedness of finite transducers. *Acta Informatica*, 27:749–780, 1990. 10.1007/BF00264285.

Curriculum Vitae

Personal Information:

Name Surname, title:	Martin Čermák, Ing.
Mailto:	icermak@fit.vutbr.cz
Date of Birth:	October 15 1982
Nationality:	Moravian
Vernacular:	Czech Republic

Education:

1998 – 2002	ISŠP – Purkyňova 97, Brno (Application of Personal Computers – Information and Database Systems), finished by leaving examination.
2002 – 2003	Palacký University Olomouc – Faculty of Science (Applied Informatics).
2003 – 2006	Faculty of Information Technology BUT, Brno – Bachelor of Information Technology.
2006 – 2008	Faculty of Information technology BUT, Brno – Master of Intelligent Systems.
Since 2008	Faculty of Information technology BUT, Brno – PhD. student of Information Systems.

Study Visits and International Conferences

- 9.5.2010 – 15.5.2010 Spain, University in Valladolid
- 28.6.2010 – 4.7.2010 France, University in La Rochelle
- 16.8.2010 – 23.8.2010 Canada, Conference DLT
- 4.9.2010 – 10.9.2010 Hungary, College of Nyíregyháza
- 5.12.2010 – 11.12.2010 France, University Paris-Est Marne-la-Vallée
- 29.5.2011 – 19.6.2011 Hungary, Hungarian academy of science in Budapest
- 16.8.2011 – 22.8.2011 Hungary, Conference AFL in Debrecen
- 10.4.2012 – 13.4.2012 Hungary, College of Nyíregyháza
- 15.4.2012 – 18.4.2012 Hungary, University Paris-Est Marne-la-Vallée
- 13.5.2012 – 19.5.2012 Spain, University in Valladolid

Projects

1. Teaching of formal language theory at the international level, FRVŠ MŠMT, FR2581/2010/G1, 2010
2. Context-free grammars and pushdown automata, MŠMT, MEB041003, 2010-2011

PUBLICATIONS:

2012 Čermák, M., Koutný, J., Meduna, A.: Parsing Based on n-Path Tree-Controlled Grammars, In: Theoretical and Applied Informatics, Vol. 2011, No. 23, 2012, Varšava, PL, p. 213-228, ISSN 1896-5334

Čermák, M.: Restrictions on Derivations in n-Generating Grammar Systems, In: Proceedings of the 18th Conference and Competition STUDENT EEICT 2012 Volume 5, Brno, CZ, FIT VUT, 2012, p. 371-375, ISBN 978-80-214-4462-1

Čermák, M., Horáček, P., Meduna, A.: Rule-Restricted Automaton-Grammar Transducers: Power and Linguistic Applications, In: Mathematics for Applications, in press

Čermák, M., Koutný, J., Meduna, A.: n-Language Classes Hierarchy, submitted

2011 Čermák, M., Meduna, A.: n-Accepting Restricted Pushdown Automata Systems, In: 13th International Conference on Automata and Formal Languages, Nyíregyháza, HU, MTA SZTAKI, 2011, p. 168-183, ISBN 978-615-5097-19-5

Čermák, M., Meduna, A.: n-Accepting Restricted Pushdown Automata Systems, 7th Doctoral Workshop on Mathematical and Engineering Methods in Computer Science, Brno, CZ, MUNI, 2011, p. 1, ISBN 978-80-214-4305-1

Čermák, M.: Basic Properties of n-Languages, In: Proceedings of the 17th Conference and Competition STUDENT EEICT 2011 Volume 3, Brno, CZ, FIT VUT, 2011, p. 460-464, ISBN 978-80-214-4273-3

Meduna, A., Čermák, M., Masopust, T.: Some Power-Decreasing Derivation Restrictions in Grammar Systems, In: Schedae Informaticae, Vol. 2010, No. 19, 2011, Krakov, PL, p. 23-34, ISSN 0860-0295

2010 Čermák, M.: Multilanguages and Multiaccepting Automata System, In: Proceedings of the 16th Conference and Competition STUDENT EEICT 2010 Volume 5, Brno, CZ, FIT VUT, 2010, p. 146-150, ISBN 978-80-214-4080-7

2009 Čermák, M.: Power Decreasing Derivation Restriction in Grammar Systems, In: Proceedings of the 15th Conference and Competition STUDENT EEICT 2009 Volume 4, Brno, CZ, FIT VUT, 2009, p. 385-389, ISBN 978-80-214-3870-5

2008 Čermák, M.: Systems of Formal Models and Their Application, In: Proceedings of the 14th Conference Student EEICT 2008, Brno, CZ, FEKT VUT, 2008, p. 164-166, ISBN 978-80-214-3615-2

2006	Čermák, M.: Syntax Analysis Based on Combination of Several Methods, In: Proceedings of the 12th Conference STUDENT EEICT 2006, Brno, CZ, FEKT VUT, 2006, p. 200-202, ISBN 80-214-3160-1

PROFESSIONAL CAREER:

September 2006 – January 2009	Duha System s.r.o., Brno – junior developer of Information systems in C# .NET and Java
January 2009 – Fabruary 2010	iNexia s.r.o., Brno – junior developer of Workflow application in C# .NET