Formal Systems Based upon Automata and Grammars

Martin Čermák*
Institute of Information Systems
Faculty of Information Technologies
Brno University of Technology
Božetěchova 2, 612 00 Brno, Czech Republic
icermak@fit.vutbr.cz

Abstract

This work is based on my PhD thesis, which continues with studying 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 restrictions 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.

Categories and Subject Descriptors

F.4.3 [**Formal Languages**]: classes defined by grammars or automata, operation on languages

Keywords

grammar, automaton, grammar system, automata system, transducer, parsing

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*) claiming *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*, and 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

*Recommended by thesis supervisor: Prof. Alexander Meduna. Defended at Faculty of Information Technologies, University of Technology in Brno on September 18, 2012

Čermák, M. Formal Systems Based upon Automata and Grammars. Information Sciences and Technologies Bulletin of the ACM Slovakia, Vol. 4, No. 4 (2012) 7-14

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.

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*.

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 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 main role in the theory of formal systems is played by cooperation protocols and used formal models. The thesis continues with study of these modern approaches and brings new, or generalized, formal mechanisms and results into the theory. More specifically, the thesis mainly deals with systems of automata and grammars and studies their properties. At first, it 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 nondeterminism, 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 Roman Lukáš and Alexander 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.

2. Preliminaries

In this paper, we assume the reader is familiar with the formal language theory (see [30]) and the basic aspects of computational linguistics (see [33]).

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 $q_0 w \Rightarrow^* f$, where $f \in F$, then w is accepted by M and $q_0 w \Rightarrow^* f$ is an acceptance of w in M. The language of M is defined as $L(M) = \{w |\ w \in \Sigma^*, q_0 w \Rightarrow^* f \text{ is an acceptance of } w\}$.

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 $Zqa \rightarrow \gamma p$, 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 $r = Aqa \rightarrow \gamma p \in \delta$, 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 $Z_0q_0w \Rightarrow^* f$, where $f \in F$, then w is accepted by M and $Z_0q_0w \Rightarrow^* f$ is an acceptance of w in M. The language of M is defined as $L(M) = \{w|\ w \in \Sigma^*, Z_0q_0w \Rightarrow^* f$ 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 $A \rightarrow \alpha$, where $A \in N$, and $\alpha \in (N \cup T)^*$. Furthermore, if $\alpha \in T^*NT^*$, 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 $r = A \rightarrow \alpha \in P$. 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 $\alpha \rightarrow \beta$, where $\alpha \in N^+$ and $\beta \in (N \cup T)^*$. If $\alpha \rightarrow \beta \in P$, $u = x_0\alpha x_1$, and $v = x_0\beta x_1$, where $x_0, x_1 \in V^*$, then $u \Rightarrow v$ $[\alpha \rightarrow \beta]$ 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 [18]) 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 $A \rightarrow \beta$, 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 $\Lambda(r_1) = (\alpha \rightarrow \beta)$, 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 $\alpha \rightarrow \beta$ 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 $\mathscr{L}$(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 $\mathscr{L}$(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 [18]).

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 $m = p_1 \ldots p_m \in C$ and there are $v_0, \ldots, v_m$, where $v_0 = u$, $v_m = v$, and $v_0 \Rightarrow v_1[p_1] \Rightarrow \ldots \Rightarrow v_m[p_m]$ in G. Let $\Rightarrow^*$ and $\Rightarrow^+$ denote transitive-reflexive and transitive closure of $\Rightarrow$. The language of H is defined as $L(H) = \{w|\ S \Rightarrow w_1[m_1] \Rightarrow \ldots \Rightarrow w_n[m_n], w_n = w, m_1, \ldots, m_n \in C, w \in T^*, n \geq 0\}$. 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.

A *canonical n-generative nonterminal-synchronized grammar system*, abbreviated by n-CGN, 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 context-free grammar for each $i = 1, \ldots, n$ and Q is finite set of n-tuples of the form $(A_1, \ldots, A_n)$, where $A_i \in N_i$ for all $i = 1, \ldots, n$. A sentential n-form of n-CGN is an n-tuple of the form $\chi = (x_1, \ldots, x_n)$, where $x_i \in (N_i \cup T_i)^*$ for all $i = 1, \ldots, n$. Let n-forms $\chi = (u_1A_1v_1, \ldots, u_nA_nv_n)$ and $\chi' = (u_1x_1v_1, \ldots, u_nx_nv_n)$ be two sentential forms, where $A_i \in N_i$, $u_i \in T^*$ and $v_i, x_i \in (N_i \cup T_i)^*$ for all $i = 1, \ldots, n$. Let $A_i \rightarrow x_i$ for all $i = 1, \ldots, n$ and $(A_1, \ldots, A_n) \in Q$. Then $\chi \Rightarrow \chi'$ and $\Rightarrow^*$ and $\Rightarrow^+$ are its transitive-reflexive and transitive closure, respectively.

A *canonical n-generative rule-synchronized grammar system* (n-CGR) 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 context-free grammar for each $i = 1, \ldots, n$ and Q is finite set of n-tuples of the form $(r_1, \ldots, r_n)$, where $r_i \in P_i$ for all $i = 1, \ldots, n$. A sentential n-form of n-CGR is an n-tuple of the form $\chi = (x_1, \ldots, x_n)$, where $x_i \in (N_i \cup T_i)^*$ for all $i = 1, \ldots, n$. Let n-forms $\chi = (u_1A_1v_1, \ldots, u_nA_nv_n)$ and $\chi' = (u_1x_1v_1, \ldots, u_nx_nv_n)$ be two sentential forms, where $A_i \in N_i$, $u_i \in T^*$ and $v_i, x_i \in (N_i \cup T_i)^*$ for all $i = 1, \ldots, n$. Let $r_i : A_i \rightarrow x_i \in P_i$ for all $i = 1, \ldots, n$ and $(r_1, \ldots, r_n) \in Q$. Then $\chi \Rightarrow \chi'$ and $\Rightarrow^*$ and $\Rightarrow^+$ are its transitive-reflexive and transitive closure, respectively.

3. New Definitions and Selected Results

3.1 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 [2, 4, 15, 16, 17, 19, 20, 22, 26, 27, 28, 29]). 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 [2, 4]). 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 [26]). In essence, all these restrictions result in decreasing the generative power to the power of context-free grammars (see page 198 in [36]). This section generalizes the discussion of this topic by investigating regularly controlled cooperating distributed grammar systems (see Chapter 4 in [36]) 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 $u \; {}_l\Rightarrow v \; [\alpha \to \beta]$ in G, or simply $u \; {}_l\Rightarrow v$.

The k-fold product of ${}_l\Rightarrow$, where $k \geq 0$, is denoted by ${}_l\Rightarrow^k$. The reflexive-transitive closure and transitive closure of ${}_l\Rightarrow$ are denoted by ${}_l\Rightarrow^*$ and ${}_l\Rightarrow^+$, 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. $x \in (T^*N^*)^m T^*$;
2. $(y \in \text{sub}(x) \text{ and } |y| > h)$ implies $\text{alph}(y) \cap T \neq \emptyset$.

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

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

Convention: 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 $u \; {}_l\Rightarrow^k_{P_i} v$, $u \; {}^h_m\!\Rightarrow^k_{P_i} v$, and $u \; {}_m\!\Rightarrow^k_{P_i} v$ to denote that $u \; {}_l\Rightarrow^k v$, $u \; {}^h_m\!\Rightarrow^k v$, and $u \; {}_m\!\Rightarrow^k v$, respectively, was performed by P_i. Analogously, we write $u \; {}_l\Rightarrow^*_{P_i} v$, $u \; {}^h_m\!\Rightarrow^*_{P_i} v$, $u \; {}_m\!\Rightarrow^*_{P_i} v$, $u \; {}_l\Rightarrow^+_{P_i} v$, $u \; {}^h_m\!\Rightarrow^+_{P_i} v$, $u \; {}_m\!\Rightarrow^+_{P_i} v$, $u \; {}^h_m\!\Rightarrow^t_{P_i} v$, and $u \; {}_m\!\Rightarrow^t_{P_i} v$.

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, ${}_1L^C(\Gamma) = \{w \in T^* |\; S \; {}_l\Rightarrow^t_{P_{i_1}} w_1 \; {}_l\Rightarrow^t_{P_{i_2}} \cdots \; {}_l\Rightarrow^t_{P_{i_p}} w_p = w, p \geq 1, 1 \leq i_j \leq n, 1 \leq j \leq p,\; i_1 i_2 \ldots i_p \in C\}$, ${}_{\text{N}}L^C(\Gamma, m, h) = \{w \in T^* |\; S \; {}^h_m\!\Rightarrow^t_{P_{i_1}} w_1 \; {}^h_m\!\Rightarrow^t_{P_{i_2}} \cdots \; {}^h_m\!\Rightarrow^t_{P_{i_p}} w_p = w, p \geq 1, 1 \leq i_j \leq n, 1 \leq j \leq p,\; i_1 i_2 \ldots i_p \in C\}$, ${}_{\text{N}}L^C(\Gamma, m) = \{w \in T^* |\; S \; {}_m\!\Rightarrow^t_{P_{i_1}} w_1 \; {}_m\!\Rightarrow^t_{P_{i_2}} \cdots \; {}_m\!\Rightarrow^t_{P_{i_p}} w_p = w, p \geq 1, 1 \leq i_j \leq n, 1 \leq j \leq p,\; i_1 i_2 \ldots i_p \in C\}$.

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.

$$\mathscr{L}({}_1\text{CD}^{\mathbf{REG}}) = \{{}_1L^C(\Gamma) |\; C \in \mathbf{REG}\}$$

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

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

For these classes, the following theorems are established.

Theorem. Let $l \in \mathbb{N}$. Then, $\mathbf{CF} = \mathscr{L}({}_1\text{CD}^{\mathbf{REG}})$.

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

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

3.2 Parallel Systems of Formal Models

In the thesis, we introduce n-accepting restricted pushdown automata systems representing automata counterpart of multi-generative grammar systems (see Section 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 discuss *hybrid n-accepting move-restricted automata systems* and *hybrid canonical rule-synchronized n-generative grammar systems*, where components with different accepting and generative power can be used in one automata and grammar 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*, shortly $\text{HCGR}^{(t_1,\ldots,t_n)}$, 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 $i = 1, \ldots, n$, $t_i \in \{\text{RLNG}, \text{LNG}, \text{CFG}\}$ denotes type of ith component.

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

Consider sentential n-forms, $\chi = (u_1 A_1 v_1, \ldots, u_n A_n v_n)$ and $\chi' = (u_1 x_1 v_1, \ldots, u_n x_n v_n)$ 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 $i = 1, \ldots, n$, and
- $(r_1, \ldots, r_n) \in Q$.

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

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

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

An n-configuration is defined as an n-tuple $\chi = (x_1, \ldots, x_n)$, where for all $i = 1, \ldots, n$, x_i is a configuration of M_i. Let $\chi = (x_1, \ldots, x_n)$ and $\chi' = (x_1', \ldots, x_n')$ be two n-configurations, where for all $i = 1, \ldots, n$, $x_i \Rightarrow x_i' [r_i]$ in M_i, and $(r_1, \ldots, r_n) \in \Psi$, 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 $\chi_0 = (x_1\omega_1, \ldots, x_n\omega_n)$ be the start and $\chi_f = (q_1, \ldots, q_n)$ be a final n-configuration of $\text{HMAS}^{(t_1,\ldots,t_n)}$, where for all $i = 1, \ldots, n$, ω_i is the input string of M_i and q_i is state of M_i. The n-language of $\text{HMAS}^{(t_1,\ldots,t_n)}$ is defined as $n\text{-}L(\vartheta) = \{(\omega_1, \ldots, \omega_n) | \ \chi_0 \Rightarrow^* \chi_f$ and for every $i = 1, \ldots, n$, M_i accepts$\}$.

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

$\mathscr{L}(\text{HMAS}^{(t_1,\ldots,t_n)})$ is the class of n-languages accepted by $\text{HMAS}^{(t_1,\ldots,t_n)}$, and $\mathscr{L}(\text{HCGR}^{(t_1,\ldots,t_n)})$ is the class of n-languages generated by $\text{HCGR}^{(t_1,\ldots,t_n)}$.

The basic hierarchy of such systems is given by Figure 1.

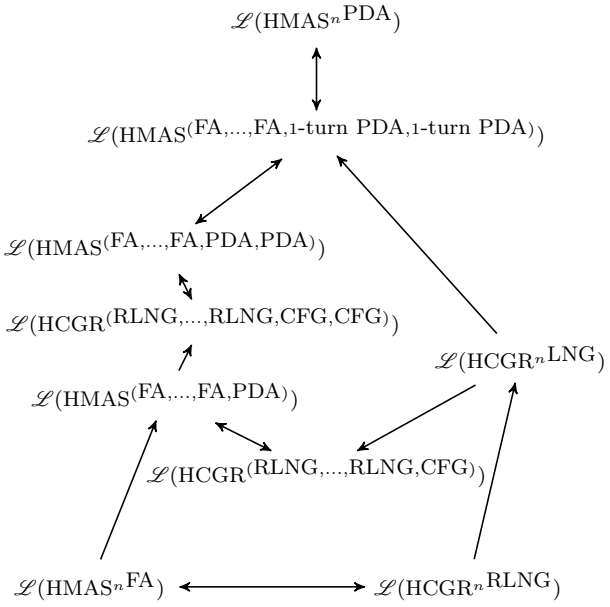

Figure 1: Hierarchy of n-languages for $n \geq 2$

3.3 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], [32], [35], [23], or [37]). 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 [3], [21], [34], and [38]).

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-$L(\Gamma) = \{(w_1, w_2) |\ (q_0 w_1, S) \Rightarrow^* (f, w_2),\ w_1 \in \Sigma^*,\ w_2 \in T^*,$ and $f \in F\}$. From the 2-language we can define two languages:

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

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

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

Although the investigated system is relatively powerful, in defiance of weakness of models they are used, nondeterministic 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-$L_{lm}(\Gamma) = \{(w_1, w_2) |\ (q_0 w_1, S) \Rightarrow_{lm}^* (f, w_2),\ w_1 \in \Sigma^*,\ w_2 \in T^*,$ 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. $\mathscr{L}(RT_{lm})_2 = \mathbf{CF}$ 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*, $\mathrm{RT_{ac}}$ 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. $\mathscr{L}(RT_{ac})_2 = \mathbf{RE}$ and $\mathscr{L}(RT_{ac})_1 = \mathbf{RE}$.

4. Thesis Summary and Further Investigation

My thesis discusses and studies formal languages and systems of formal models. Its main results are published or submitted in [13, 31, 5, 6, 7, 8, 14, 9, 11, 10, 12]. 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 1 ($\rightarrow$ 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 system based upon a finite automaton and a 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 [25]. We have considered TC grammars that generate their languages under n-path control by linear language which were introduced in [24].

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.

Acknowledgements. This work was supported by the research plan MSM0021630528, BUT FIT grant FIT-S-11-2, MŠMT grant MEB041003, and the European Regional Development Fund in the IT4Innovations Centre of Excellence project (CZ.1.05/1.1.00/02.0070).

References

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

[2] 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.

[3] 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.

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

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

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

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

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

[13] 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.

[14] 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.

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

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

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

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

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

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

[21] 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.

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

[23] 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.

[24] 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.

[25] S. Marcus, C. Martín-Vide, V. Mitrana, and G. 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.

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

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

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

[29] 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.

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

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

[32] 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.

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

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

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

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

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

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

Selected Papers by the Author

A. Meduna, M. Čermák, T. Masopust. Some Power-Decreasing Derivation Restrictions in Grammar . *Schedae Informaticae*, Vol. 2010, No. 19, 2011, Krakov, PL, p. 23–34.

M. Čermák, A. Meduna. n-Accepting Restricted Pushdown Automata Systems. In *13th International Conference on Automata and Formal Languages*, Nyíregyháza, HU, MTA SZTAKI, 2011, p. 168–183.

M. Čermák, A. Meduna. n-Accepting Restricted Pushdown Automata Systems. In *7th Doctoral Workshop on Mathematical and Engineering Methods in Computer Science*, Brno, CZ, MUNI, 2011, p. 1.

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

M. Čermák, P. Horáček, A. Meduna. Rule-restricted automaton-grammar transducers: Power and Linguistic Applications. In *Mathematics for Applications*, Brno, in press.

M. Čermák, J. Koutný, A. Meduna. On n-languages classes hierarchy. In *Acta Cybernetica*, submitted.