Schedae Informaticae Vol. 22 (2013): 47–68
doi: 10.4467/20838476SI.13.005.2090

New Grammar Systems and Their Application Perspectives

Petr Horáček, Alexander Meduna
Brno University of Technology, Faculty of Information Technology,
Božetěchova 1/2, 612 66 Brno, Czech Republic
e-mail: {*ihoracekp, meduna*}*@fit.vutbr.cz*

Abstract. This paper presents new grammar systems that describe transformations of syntactic structures. They represent two approaches: synchronous grammars and transducers. The systems consist of well-known models such as context-free grammars and finite automata. Particular attention is paid to synchronization of regulated grammars. The paper recalls formal definitions of the systems and discusses theoretical results regarding their generative and accepting power. The last part briefly introduces application perspectives in natural language translation, illustrated by examples of Czech-English translation.

Keywords: synchronous grammars, regulated grammars, transducers, computational control, generative power, natural language translation.

1. Introduction

Machine translation is one of the major tasks in natural language processing (NLP). With increasing availability of large corpora, corpus-based systems became favoured over rule-based, using statistical methods and machine-learning techniques. They mostly rely on formal models that represent local information only, such as n-gram models. However, recently, there have been attempts to improve results by incorporating syntactic information into such systems [14, 21, 4].

To do this, we need formal models that can describe syntactic structures and their transformations. In our work, we study well-known models from formal language theory (FLT), and extend them for application in NLP, particularly in translation.

Based on the principles of synchronous grammars [5], we have previously [12] proposed synchronous versions of some regulated grammars, such as matrix grammars [6] and scattered context grammars [16]. Revised definitions as well as further study of theoretical propeties of the new models can be found in [13].

Other type of models we can use are transducers [2]. Unlike synchronous grammars, which generate a pair of sentences in one derivation and thus define translation, transducers take a given input sentence and transform it into a corresponding output sentence. Frequently, these transducers consist of several components, including various automata and grammars, some of which read their input strings while others produce their output strings [19, 9]. In [17], we have introduced the rule-restricted automaton-grammar transducer and its variants, and discussed its accepting and generative power.

The present paper is organized as follows. The first section after the introduction recalls basic definitions from FLT that are used throughout this paper. The two following sections contain definitions of the new formals models (new synchronous grammars and new transducers, respectively). Each of the two sections also presents theoretical results that we have estabilished. Finally, the last section further explores the application perspectives of the new models, with focus on natural language translation. We discuss and compare their main advantages and illustrate them by examples from Czech and English.

2. Preliminiaries

We assume that the reader is familiar with the basic aspects of modern FLT [20, 15] and NLP [18, 3].

2.1. Grammars

Definition 1 (Context-free grammar) *A* context-free grammar *(CFG) is a quadruple $G = (N, T, P, S)$, where N is a finite set of* nonterminals*, T is a finite set of* terminals*, $N \cap T = \emptyset$, $P \subset N \times (N \cup T)^*$ is a finite set of* rules*, where $(u, v) \in P$ is written as $u \to v$, and $S \in N$ is the* start symbol*. Further, let $u, v \in (N \cup T)^*$ and $p = A \to x \in P$. Then, we say that uAv* directly derives *uxv according to p in G, written as $uAv \Rightarrow_G uXv\,[p]$, or simply $uAv \Rightarrow uxv$. We further define $\Rightarrow^+$ as the transitive closure and $\Rightarrow^*$ as the transitive and reflexive closure of $\Rightarrow$. The* language generated by *G, denoted by $L(G)$, is defined as $L(G) = \{w \colon w \in T^*, S \Rightarrow^* w\}$.*

Definition 2 (Matrix grammar) *A* matrix grammar *(MAT) is a pair $H = (G, M)$, where $G = (N, T, P, S)$ is a CFG and M is a finite language over P ($M \subset P^*$) – a sentence of this language is called a* matrix*. Further, for $u, v \in (N \cup T)^*$,*

$m = p_1 \dots p_n \in M$ *we define* $u \Rightarrow v\,[m]$ *in* H*, if there are strings* $x_0, \dots, x_n$ *such that* $u = x_0$*,* $v = x_n$*, and for all* $0 \leq i < n$*,* $x_i \Rightarrow x_{i+1}\,[p_{i+1}]$ *in* G*. The* language generated by H*, denoted by* $L(H)$*, is defined as* $L(H) = \{w \colon w \in T^*, S \Rightarrow^* w\}$*.*

Definition 3 (Matrix grammar with appearance checking) *A* matrix grammar with appearance checking *(*MAT_{ac}*) is a pair* $H = (G, M)$*, where* $G = (N, T, P, S)$ *is a CFG and* M *is a finite set of finite sequences (called* matrices*) of pairs* (p, t) *with* $p \in P$ *and* $t \in \{-, +\}$*. Further, for* $u, v \in (N \cup T)^*$*,* $m = (p_1, t_1) \dots (p_m, t_m) \in M$ *we define* $u \Rightarrow v\,[m]$ *in* H*, if there are strings* $x_0, \dots, x_n$ *such that* $u = x_0$*,* $v = x_n$*, and for all* $0 \leq i < n$*, either* $x_i \Rightarrow x_{i+1}\,[p_{i+1}]$ *in* G*, or* $t_{i+1} \in \{-\}$*,* $x_i = x_{i+1}$*, and* p_{i+1} *is not applicable on* x_i *in* G*. The* language generated by H*, denoted by* $L(H)$*, is defined as* $L(H) = \{w \colon w \in T^*, S \Rightarrow^* w\}$*.*

Definition 4 (Scattered context grammar) *A* scattered context grammar *(S-CG) is a quadruple* $G = (N, T, P, S)$*, where* N *is a finite set of* nonterminals*,* T *is a finite set of* terminals*,* $N \cap T = \emptyset$*,* P *is a finite set of* rules *of the form* $(A_1, \dots, A_n) \to (x_1, \dots, x_n)$*, where* $A_1, \dots, A_n \in N$*,* $x_1, \dots, x_n \in (N \cup T)^*$*, and* $S \in N$ *is the* start symbol*. Further, for* $u, v \in (N \cup T)^*$*,* $p \in P$ *we define* $u \Rightarrow v\,[p]$*, if there is a factorization of* $u = u_1 A_1 \dots u_n A_n u_{n+1}$*,* $v = u_1 x_1 \dots u_n x_n u_{n+1}$ *such that* $p = (A_1, \dots, A_n) \to (x_1, \dots, x_n)$ *and* $u_i \in (N \cup T)^*$ *for* $1 \leq i \leq n$*. The* language generated by G*, denoted by* $L(G)$*, is defined as* $L(G) = \{w \colon w \in T^*, S \Rightarrow^* w\}$*.*

2.2. Automata

Definition 5 (Finite automaton) *A* finite automaton *(FA) is a quintuple* $M = (Q, \Sigma, \delta, q_0, F)$*, where* Q *is a finite set of* states*,* Σ *is the* input 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* $qay \Rightarrow_M py\,[r]$*, 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 accepted by M*, denoted by* $L(M)$*, is defined as* $L(M) = \{w \colon w \in \Sigma^*, q_0 w \Rightarrow^* f$ *is an acceptance of* $w\}$*.*

Definition 6 (Pushdown automaton) *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 the* input alphabet*,* $q_0 \in Q$ *is the* initial state*,* Γ *is the* 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* $Z_0q_0w \Rightarrow^* f$*, 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 accepted by M *is defined as* $L(M) = \{w|\ w \in \Sigma^*, Z_0q_0w \Rightarrow^* f$ *is an acceptance of* $w\}$*.*

Definition 7 (Partially blind k-counter automaton) *A* partially blind k-counter automaton *(*k*-PBCA) is a FA* $M = (Q, \Sigma, \delta, q_0, F)$ *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 accepted by M *is defined as* $L(M) = \{w|\ w \in \Sigma^*, q_0w(0, \ldots, 0) \Rightarrow^* f(0, \ldots, 0), f \in F\}$*.*

Definition 8 (k-counter automaton) *A* k-counter automaton *(*k*-CA) is a FA* $M = (Q, \Sigma, \delta, q_0, F)$ *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* $pa \to q(t_1, \ldots, t_n)$*, where* $p, q \in Q$*,* $a \in \Sigma \cup \{\varepsilon\}$*, and* $t_i \in \{-\} \cup \mathbb{Z}$*. A* configuration *of* k*-CA is 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 configuration of* M *and* $r = pa \to q(t_1, \ldots, t_k) \in \delta$*, where the following holds: if* $t_i \in \mathbb{Z}$*, then* $v'_i = v_i + t_i$*; otherwise, it is satisfied that* $v_i, v'_i = 0$*. 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 accepted by M *is defined as* $L(M) = \{w|\ w \in \Sigma^*, q_0w(0, \ldots, 0) \Rightarrow^* f(0, \ldots, 0), f \in F\}$*.*

3. Synchronous grammars

In essence, synchronous grammars are grammars (or grammar systems) that generate pairs of sentences in one derivation, instead of single sentences (as for example in CFG). In this way, they allow us to describe translations. We can see synchronous CFG [5] as a modification of CFG where every rule has two right-hand sides. Nonterminals are linked, which means that in each derivation step, we rewrite both the selected nonterminal symbol in the input sentential form and its appropriate counterpart in the output sentential form.

In [12], we proposed synchronization based on linked rules instead of nonterminals. Informally, such synchronous grammar is a system of two grammars in which the corresponding rules share labels. For example, if we apply rule labelled 1 in the input grammar, we also have to apply rule labelled 1 in the output grammar and this makes for a single derivation step in the synchronous grammar. In other words, the input and output sentence have the same parse (a sequence of rules applied, denoted by their labels).

3.1. Definitions

Here we recall the definitions from [13].

Definition 9 (Rule-synchronized CFG) *A* rule-synchronized CFG *(RSCFG) H is a quintuple $H = (G_I, G_O, \Psi, \varphi_I, \varphi_O)$, where $G_I = (N_I, T_I, P_I, S_I)$ and $G_O = (N_O, T_O, P_O, S_O)$ are CFGs, Ψ is a set of* rule labels*, and φ_I is a function from Ψ to P_I and φ_O is a function from Ψ to P_O.*

We use the following notation (presented for input grammar G_I, analogous for output grammar G_O):

$p : A_I \to x_I$ where $p \in \Psi, A_I \to x_I \in P_I$	$\varphi_I(p) = A_I \to x_I$
$x_I \Rightarrow_{G_I} y_I\,[p]$ where $x_I, y_I \in (N \cup T)^*, p \in \Psi$	derivation step in G_I applying rule $\varphi_I(p)$
$x_I \Rightarrow^n_{G_I} y_I\,[p_1 \dots p_n]$ where $x_I, y_I \in (N \cup T)^*, p_i \in \Psi$ for $1 \leq i \leq n$	derivation in G_I applying rules $\varphi_I(p_1) \dots \varphi_I(p_n)$

Definition 10 (Translation in RSCFG) *Let $H = (G_I, G_O, \Psi, \varphi_I, \varphi_O)$ be a RSCFG.* Translation defined by *H, $T(H)$, is the set of pairs of sentences, which is defined as $T(H) = \{(w_I, w_O) \colon w_I \in T_I^*, w_O \in T_O^*, S_I \Rightarrow^*_{G_I} w_I\,[\alpha], S_O \Rightarrow^*_{G_O} w_O\,[\alpha], \alpha \in \Psi^*\}$.*

Originally [12], we considered RSCFG only as a variant of synchronous CFG. However, there is in fact a significant difference. While the latter does not increase the generative power over CFG, RSCFG does, as is shown in next subsection.

Next, to define synchronization for SCGs, we simply replace context-free rules with scattered context rules.

Definition 11 (Synchronous SCG) *A* synchronous SCG *(SSCG) H is quintuple $H = (G_I, G_O, \Psi, \varphi_I, \varphi_O)$, where $G_I = (N_I, T_I, P_I, S_I)$ and $G_O = (N_O, T_O, P_O, S_O)$ are SCGs, Ψ is a set of* rule labels*, and φ_I is a function from Ψ to P_I and φ_O is a function from Ψ to P_O. Further, the* translation defined by *H, denoted by $T(H)$, is the set of pairs of sentences, which is defined as $T(H) = \{(w_I, w_O) \colon w_I \in T_I^*, w_O \in T_O^*, S_I \Rightarrow^*_{G_I} w_I\,[\alpha], S_O \Rightarrow^*_{G_O} w_O\,[\alpha], \alpha \in \Psi^*\}$.*

Finally, with MATs, we link whole matrices rather than individual rules.

Definition 12 (Synchronous matrix grammar) *A* synchronous matrix grammar *(SMAT) H is a septuple $H = (G_I, M_I, G_O, M_O, \Psi, \varphi_I, \varphi_O)$, where (G_I, M_I) and (G_O, M_O) are MATs, Ψ is a set of* matrix labels*, and φ_I is a function from Ψ to M_I and φ_O is a function from Ψ to M_O. Further, the* translation defined by *H, denoted by $T(H)$, is the set of pairs of sentences, which is defined as $T(H) = \{(w_I, w_O) \colon w_I \in T_I^*, w_O \in T_O^*, S_I \Rightarrow^*_{(G_I, M_I)} w_I\,[\alpha], S_O \Rightarrow^*_{(G_O, M_O)} w_O\,[\alpha], \alpha \in \Psi^*\}$.*

3.2. Theoretical results

This subsection recalls the theoretical results published in [13].

Synchronous grammars define translations – that is, sets of pairs of sentences. To be able to compare their generative power with well-known models such as CFGs which define languages, we can consider their input and output language.

Definition 13 (Input and output language) *Let H be a synchronous grammar. Then, we define*
I. the input language *of H, $L_I(H)$, as $L_I(H) = \{w_I \in T_I^* \colon (w_I, w_O) \in T(H)\}$,*
II. the output language *of H, $L_O(H)$, as $L_O(H) = \{w_O \in T_O^* \colon (w_I, w_O) \in T(H)\}$.*

Example 1 Consider a RSCFG $H = (G_I, G_O, \Psi, \varphi_I, \varphi_O)$ with the following rules (nonterminals are in capitals, linked rules share the same label, S_I and S_O are the start symbols of G_I and G_O respectively):

G_I				G_O			
1:	S_I	$\to$	ABC	1:	S_O	$\to$	A
2:	A	$\to$	aA	2:	A	$\to$	B
3:	B	$\to$	bB	3:	B	$\to$	C
4:	C	$\to$	cC	4:	C	$\to$	A
5:	A	$\to$	ε	5:	A	$\to$	B'
6:	B	$\to$	ε	6:	B'	$\to$	C'
7:	C	$\to$	ε	7:	C'	$\to$	ε

An example of a derivation follows.

S_I	$\Rightarrow$	ABC	[1]	S_O	$\Rightarrow$	A	[1]
	$\Rightarrow$	$aABC$	[2]		$\Rightarrow$	B	[2]
	$\Rightarrow$	$aAbBC$	[3]		$\Rightarrow$	C	[3]
	$\Rightarrow$	$aAbBcC$	[4]		$\Rightarrow$	A	[4]
	$\Rightarrow$	$aaAbBcC$	[2]		$\Rightarrow$	B	[2]
	$\Rightarrow$	$aaAbbBcC$	[3]		$\Rightarrow$	C	[3]
	$\Rightarrow$	$aaAbbBccC$	[4]		$\Rightarrow$	A	[4]
	$\Rightarrow$	$aabbBccC$	[5]		$\Rightarrow$	B'	[5]
	$\Rightarrow$	$aabbccC$	[6]		$\Rightarrow$	C'	[6]
	$\Rightarrow$	$aabbcc$	[7]		$\Rightarrow$	ε	[7]

We can easily see that $L_I(H) = \{a^n b^n c^n \colon n \geq 0\}$, which is well known not to be a context-free language. This shows that RSCFGs are stronger than (synchronous) CFGs. Where exactly do synchronous grammars (rule-synchronized) stand in terms of generative power?

Let $\mathscr{L}(RSCFG)$, $\mathscr{L}(SMAT)$, and $\mathscr{L}(SSCG)$ denote the class of languages generated by RSCFGs, SMATs, and SSCGs, respectively, as their input language. Note that the results presented below would be the same if we considered the output language instead.

In some of the proofs below, we use a function which removes all terminals from a sentential form, formally defined as follows.

Definition 14 *Let $G = (N, T, P, S)$ be a CFG. Then, we define the function Δ over $(N \cup T)^*$ as follows:*
I. for all $w \in T^*$, $\Delta(w) = \varepsilon$
II. for all $w = x_0 A_1 x_2 A_2 \dots x_{n-1} A_n x_n$, $x_i \in T^*$, $0 \leq i \leq n$, $A_j \in N$, $1 \leq j \leq n$ for some $n \geq 1$, $\Delta(w) = A_1 A_2 \dots A_n$.

The idea here is that if we consider only context-free rules, for a given sentential form, the applicability of rules only depends on nonterminals. Therefore, we can remove terminals without affecting computational control.

Theorem 1

$$\mathscr{L}(RSCFG) = \mathscr{L}(MAT)$$

Proof. I. First, we prove that $\mathscr{L}(RSCFG) \subseteq \mathscr{L}(MAT)$. For every RSCFG $H = (G_I, G_O, \Psi, \varphi_I, \varphi_O)$, where $G_I = (N_I, T_I, P_I, S_I)$, $G_O = (N_O, T_O, P_O, S_O)$, we can construct a MAT $H' = (G, M)$, where $G = (N, T, P, S)$, such that $L(H') = L_I(H)$, as follows. Without loss of generality, assume $N_I \cap N_O = \emptyset$, $S \notin N_I \cup N_O$. Set $N = N_I \cup N_O \cup \{S\}$, $T = T_I$, $P = \{S \to S_I S_O\}$, $M = \{S \to S_I S_O\}$. For every label $p \in \Psi$, add rules p_I, p_O to P, add matrix $p_I p_O$ to M, where $p_I = \varphi_I(p)$ and $p_O = A \to x$ such that $\varphi_O(p) = A \to x'$, $x = \Delta(x')$.[1]

Basic idea. H' simulates the principle of linked rules in H by matrices. That is, for every pair of rules $(A_I \to x_I, A_O \to x_O)$ such that $\varphi_I(p) = A_I \to x_I, \varphi_O(p) = A_O \to x_O$ for some $p \in \Psi$ in H, there is a matrix $m = A_I \to x_I A_O \to \Delta(x_O)$ in H'. If, in H, $x_I \Rightarrow y_I\,[p]$ in G_I and $x_O \Rightarrow y_O\,[p]$ in G_O, then there is a derivation step $x_I\,\Delta(x_O) \Rightarrow y_I\,\Delta(y_O)\,[m]$ in H'. Note that since the rules are context-free, the presence (or absence) of terminals in a sentential form does not affect which rules we can apply. Furthermore, because the nonterminal sets N_I and N_O are disjoint, the sentential form in H' always consists of two distinct parts such that the first part corresponds to the derivation in G_I and the second part to the derivation in G_O. The complete formal proof of $L(H') = L(H)$ can be found in [13].

II. Now we have to show that $\mathscr{L}(MAT) \subseteq \mathscr{L}(RSCFG)$ holds. For every MAT $H = (G, M)$, where $G = (N, T, P, S)$, we can construct a RSCFG $H' = (G_I, G_O, \Psi, \varphi_I, \varphi_O)$, where $G_I = (N_I, T_I, P_I, S_I)$, $G_O = (N_O, T_O, P_O, S_O)$, such that $L_I(H') = L(H)$, as follows. Without loss of generality, assume $N \cap \{S_I, S_O, X\} = \emptyset$. Set $N_I = N \cup \{S_I, X\}$, $T_I = T$, $P_I = \{S_I \to SX, X \to \varepsilon\}$, $N_O = \{S_O, X\}$, $T_O = \emptyset$, $P_O = \{S_O \to X, X \to \varepsilon\}$. Set $\Psi = \{0, 1\}$, $\varphi_I = \emptyset$, $\varphi_O = \emptyset$, $\varphi_I(0) = S_I \to SX$, $\varphi_O(0) = S_O \to X$, $\varphi_I(1) = X \to \varepsilon$, $\varphi_O(1) = X \to \varepsilon$. For every matrix $m = p \in M$, where $p \in P$, add rule p to P_I, $X \to X$ to P_O, add label $\langle m \rangle$ to Ψ, and set $\varphi_I(\langle m \rangle) = p$, $\varphi_O(\langle m \rangle) = X \to X$. For every matrix $m = p_1 \dots p_n \in M$, where $n > 1$, add rules $p_1, \dots, p_n$ to P_I, add new nonterminals $\langle Xm \rangle_1, \dots, \langle Xm \rangle_{n-1}$ to N_O, add rules $X \to \langle Xm \rangle_1, \langle Xm \rangle_1 \to \langle Xm \rangle_2, \dots, \langle Xm \rangle_{n-2} \to \langle Xm \rangle_{n-1}, \langle Xm \rangle_{n-1} \to X$ to P_O, add new labels $\langle m \rangle_1, \dots, \langle m \rangle_n$ to Ψ, and set $\varphi_I(\langle m \rangle_1) = p_1$, $\varphi_O(\langle m \rangle_1) = X \to \langle Xm \rangle_1$, $\varphi_I(\langle m \rangle_i) = p_i$, $\varphi_O(\langle m \rangle_i) = \langle Xm \rangle_{i-1} \to \langle Xm \rangle_i$ for $1 < i < n$, and $\varphi_I(\langle m \rangle_n) = p_n$, $\varphi_O(\langle m \rangle_n) = \langle Xm \rangle_{n-1} \to X$.

[1] This removes all terminals from the right-hand side of the rule. Note that if we left the rule unchanged, we would obtain the concatenation of the input and the output sentence. Further, if we wanted $L(H') = L_O(H)$, we would simply modify p_I instead of p_O.

Basic idea. H' simulates matrices in H by derivation in G_O. That is, if $x \Rightarrow y\,[m]$ in H, where $m = p_1 \dots p_n$ for some n, then there is a sequence of derivation steps $X \Rightarrow \langle Xm\rangle_1\,[\langle m\rangle_1] \Rightarrow \langle Xm\rangle_2\,[\langle m\rangle_2] \Rightarrow \dots \Rightarrow \langle Xm\rangle_{n-2}\,[\langle m\rangle_{n-2}] \Rightarrow \langle Xm\rangle_{n-1}\,[\langle m\rangle_{n-1}] \Rightarrow X\,[\langle m\rangle_n]$ in G_O and $\varphi_I(\langle m\rangle_i) = p_i$ for $1 \leq i \leq n$. Now observe that in G_O constructed by the above algorithm, every nonterminal except X can only appear as the left-hand side of no more than one rule. This means that after rewriting X to $\langle Xm\rangle_1$, the only way for the derivation to proceed is the above sequence, and the entire matrix is simulated. The complete formal proof of $L(H') = L(H)$ can be found in [13].

Note that G_O constructed by the above algorithm is not only context-free, but also regular.

Theorem 2

$$\mathscr{L}(SMAT) = \mathscr{L}(MAT)$$

Proof. The inclusion $\mathscr{L}(MAT) \subseteq \mathscr{L}(SMAT)$ follows from definition. It only remains to prove that $\mathscr{L}(SMAT) \subseteq \mathscr{L}(MAT)$. For every SMAT $H = (G_I, M_I, G_O, M_O, \Psi, \varphi_I, \varphi_O)$, where $G_I = (N_I, T_I, P_I, S_I)$, $G_O = (N_O, T_O, P_O, S_O)$, we can construct a MAT $H' = (G, M)$, where $G = (N, T, P, S)$, such that $L(H') = L_I(H)$, as follows. Without loss of generality, assume $N_I \cap N_O = \emptyset$, $S \notin N_I \cup N_O$. Set $N = N_I \cup N_O \cup \{S\}$, $T = T_I$, $P = \{S \to S_I S_O\}$, $M = \{S \to S_I S_O\}$. For every label $p \in \Psi$, add rules $p_{I1}, \dots, p_{In}, p_{O1}, \dots, p_{Om}$ to P, add matrix $p_{I1} \dots p_{In} p_{O1} \dots p_{Om}$ to M, where $p_{I1} \dots p_{In} = \varphi_I(p)$ and for $1 \leq j \leq m$, $p_{Oj} = A_j \to x_j$ such that $\varphi_O(p)[j] = A_j \to x'_j$, $x_j = \Delta(x'_j)$.[2]

Basic idea. H' simulates H by combining the rules of each two linked matrices in H into a single matrix in H'. That is, for every pair of matrices (m_I, m_O) such that $m_I = \varphi_I(p), m_O = \varphi_O(p)$ for some $p \in \Psi$ in H, there is a matrix $m = m_I m'_O$ in H', where m'_O is equal to m_O with all terminals removed (formally defined above). If, in H, $x_I \Rightarrow y_I\,[p]$ in G_I and $x_O \Rightarrow y_O\,[p]$ in G_O, then there is a derivation step $x_I\,\Delta(x_O) \Rightarrow y_I\,\Delta(y_O)\,[m]$ in H'. Note that since the rules are context-free, the presence (or absence) of terminals in a sentential form does not affect which rules we can apply. Furthermore, because the nonterminal sets N_I and N_O are disjoint, the sentential form in H' always consists of two distinct parts such that the first part corresponds to the derivation in G_I and the second part to the derivation in G_O. The complete formal proof of $L(H') = L(H)$ can be found in [13].

Theorem 3

$$\mathscr{L}(SSCG) = \mathbf{RE}$$

Proof. Clearly, $\mathscr{L}(SSCG) \subseteq \mathbf{RE}$ holds. From definition, it follows that $\mathscr{L}(SCG) \subseteq \mathscr{L}(SSCG)$. Because $\mathscr{L}(SCG) = \mathbf{RE}$, $\mathbf{RE} \subseteq \mathscr{L}(SSCG)$ must also hold.

[2] Again, this removes all terminals from the right-hand side of the rules (see Theorem 1). $m[j]$ denotes the j-th rule in matrix m.

4. Transducers and hybrid systems

Transducers are language-translation devices, which often consist of several components. The components can be various types of automata, and it is also possible to include various grammars as well (hence hybrid systems). Some of these components read their input strings while other produce the corresponding output strings.

In [17], we have introduced a new type of transducer called the rule-restricted automaton-grammar transducer. It is a hybrid system consisting of a FA and a CFG. The basic idea is straightforward: we read an input sentence with a FA while generating an appropriate output sentence with a CFG. A control set determines which rules from the FA and the CFG can be used simultaneously. The computation of the system is successful if and only if the FA accepts the input string and the CFG generates a string of terminals.

4.1. Definitions

Here we recall the definitions from [17].

Definition 15 (Rule-restricted transducer) *The* rule-restricted transducer *(R-T) is a triple* $\Gamma = (M, G, \Psi)$*, where* $M = (Q, \Sigma, \delta, q_0, F)$ *is a FA,* $G = (N, T, P, S)$ *is a CFG, 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* Γ *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-$L(\Gamma) = \{(w_1, w_2) |\ (q_0w_1, S) \Rightarrow^* (f, w_2),\ w_1 \in \Sigma^*,\ w_2 \in T^*,$ *and* $f \in F\}$*. From the* 2*-language we can define two languages:*
I. $L(\Gamma)_1 = \{w_1 \colon (w_1, w_2) \in$ 2-$L(\Gamma)\}$*, and*
II. $L(\Gamma)_2 = \{w_2 \colon (w_1, w_2) \in$ 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.*

Non-deterministic selections of nonterminals to be rewritten can be relatively problematic from the practical point of view. We can place a restriction in the form of leftmost derivations in the CFG in RT.

Definition 16 (Leftmost restriction on derivation in RT) *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* $L_{lm}(\Gamma)_1 = \{w_1 |\ (w1, w2) \in$ 2-$L_{lm}(\Gamma)\}$ *and* $L_{lm}(\Gamma)_2 = \{w_2 |\ (w1, w2) \in$ 2-$L_{lm}(\Gamma)\}$.

By $\mathscr{L}(RT_{lm})$, $\mathscr{L}(RT_{lm})_1$, *and* $\mathscr{L}(RT_{lm})_2$, *we understand the following language classes, respectively:* 2*-languages of leftmost restricted RTs, languages accepted by* M *in leftmost restricted RTs, and languages generated by* G *in leftmost restricted RTs.*

We can also extend RT with the possibility to prefer a rule over another – that is, the restriction sets contain triples 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.

Definition 17 (RT with appearance checking) RT with appearance checking *(*RT_{ac}*) is a triple* $\Gamma = (M, G, \Psi)$, *where* $M = (Q, \Sigma, \delta, q_0, F)$ *is a FA,* $G = (N, T, P, S)$ *is a CFG, and* Ψ *is a finite set of triples 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* $uAv_3 \Rightarrow uxv_2\,[r_3]$ *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 in Definition 15. The classes of languages defined by the first and the second component in the system is denoted by* $\mathscr{L}(RT_{ac})_1$ *and* $\mathscr{L}(RT_{ac})_2$, *respectively.*

4.2. Theoretical results

This subsection recalls the theoretical results published in [17].

It is well-known that FA and CFG describe different classes of languages. Specifically, by FAs we can accept regular languages, whereas CFGs define the class of context-free languages. However, in Example 2 is shown that by the combination of these two models, the system is able to accept and generate even non-context-free languages.

Example 2 Consider RT $K = (M, G, \Psi)$ with

- $M = (\{1, 2, 3', 3, 4, 5', 5, 6\}, \{a, b\}, \delta, 1, \{6\})$, where

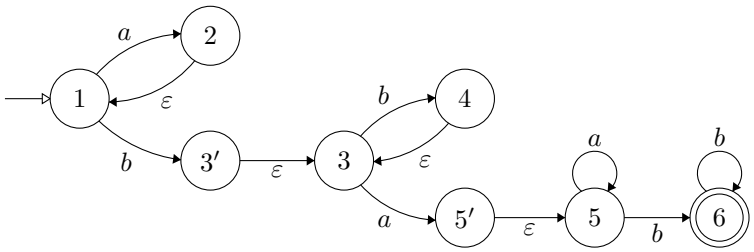

Fig. 1. Definition of FA M from Example 2

– $\delta = \{$
$p_1 = 1a \to 2$, $p_2 = 2 \to 1$, $p_3 = 1b \to 3'$, $p_4 = 3' \to 3$,
$p_5 = 3b \to 4$, $p_6 = 4 \to 3$, $p_7 = 3a \to 5'$, $p_8 = 5' \to 5$,
$p_9 = 5a \to 5$, $p_{10} = 5b \to 6$, $p_{11} = 6b \to 6\}$,

See graphical representation of M in Figure 1.

- $G = (\{S, A, B, C, D, D'\}, \{a, b\}, P, S)$, where

  – $P = \{$
  $r_1 = S \to BbD'$, $r_2 = B \to Bb$, $r_3 = D' \to D'D$,
  $r_4 = B \to aA$, $r_5 = D' \to C$, $r_6 = A \to aA$,
  $r_7 = C \to CC$, $r_8 = D \to b$, $r_9 = A \to \varepsilon$,
  $r_{10} = C \to a\}$,

- $\Psi = \{(p_1, r_1), (p_1, r_2), (p_2, r_3), (p_3, r_4), (p_4, r_5), (p_5, r_6), (p_6, r_7), (p_7, r_8), (p_8, r_9), (p_9, r_8), (p_{10}, r_{10}), (p_{11}, r_{10})\}$.

The languages of M and G are $L(M) = \{a^i b^j a^k b^l |\ j, k, l \in \mathbb{N}, i \in \mathbb{N}_0\}$ and $L(G) = \{a^i b^j a^k b^l |\ i, j, k \in \mathbb{N}, l \in \mathbb{N}_0\}$, respectively. However, 2-language of K is $L(K) = \{(a^i b^j a^i b^j, a^j b^i a^j b^i) |\ i, j \in \mathbb{N}\}$.

From the example, observe that the power of the grammar increases due to the possibility of synchronization with the automaton that can dictate sequences of usable rules in the grammar. The synchronization with the automaton enhances the generative power of the grammar up to the class of languages generated by MATs.

Theorem 4

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

Proof. I. First we prove that $\mathscr{L}(MAT) \subseteq \mathscr{L}(RT)_2$.

Consider a MAT $I = ({}_I G, {}_I C)$ and construct RT $\Gamma = ({}_\Gamma M, {}_\Gamma G, \Psi)$, such that $L(I) = L(\Gamma)_2$, as follows: Set ${}_\Gamma G = {}_I G$; construct FA ${}_\Gamma M = (Q, \Sigma, \delta, s, F)$ in the following way: Set $F, Q = \{s\}$; for every $m = p_1 \dots p_k \in {}_I C$, add $k - 1$ new states, $q_1, q_2, \dots, q_{k-1}$, into Q, k new rules, $r_1 = s \to q_1, r_2 = q_1 \to q_2, \dots, r_{k-1} = q_{k-2} \to q_{k-1}, r_k = q_{k-1} \to s$, into δ, and k new pairs, $(r_1, p_1), (r_2, p_2), \dots, (r_{k-1}, p_{k-1})$, (r_k, p_k), into Ψ.

The FA simulates matrices in I by transitions. That is, if $x_1 \to x_2[p]$ in I, where $p = p_1, \ldots, p_i$ for some $i \in \mathbb{N}$, then there is $q_1, \ldots, q_{i-1} \in Q$ such that $r_1 = s \to q_1, r_2 = q_1 \to q_2, \ldots, r_{i-1} = q_{i-2} \to q_{i-1}, r_i = q_{i-1} \to s \in \delta$ and $(r_1, p_1), \ldots, (r_i, p_i) \in \Psi$. Therefore, $(s, x_1) \to^i (s, x_2)$ in Γ. Similarly, if $(s, x_1) \to^i (s, x_2)$ in Γ, for $i \in \mathbb{N}$, and there is no $j \in \mathbb{N}$ such that $0 < j < i$ and $(s, x_1) \to^j (s, y) \to^* (s, x_2)$, there has to be $p \in {}_IC$ and $x_1 \to x_2[p]$ in I. Hence, if $(s, S) \to^* (s, w)$ in Γ, where w is a string over the set of terminals in ${}_\Gamma G$, then $S \to^* w$ in I; and, on the other hand, if $S \to^* w$ in I for a string over the set of terminals in ${}_IG$, then $(s, S) \to^* (s, w)$ in Γ. The inclusion $\mathscr{L}(MAT) \subseteq \mathscr{L}(RT)_2$ has been proven.

II. For any RT $\Gamma = ({}_\Gamma M = (Q, \Sigma, \delta, s, F), {}_\Gamma G = ({}_\Gamma N, {}_\Gamma T, {}_\Gamma P, {}_\Gamma S), \Psi)$, we can construct a MAT $O = ({}_OG, {}_OC)$ such that $L(\Gamma)_2 = L(O)$ as follows: Set ${}_OG = ({}_\Gamma N \cup \{S'\}, {}_\Gamma T, {}_OP, S')$, ${}_OP = {}_\Gamma P \cup \{p_0 = S' \to \langle s \rangle {}_\Gamma S\}$, and ${}_OC = \{p_0\}$. For each pair $(p_1, p_2) \in \Psi$ with $p_1 = qa \to r$, $q, r \in Q$, $a \in \Sigma \cup \{\varepsilon\}$, $p_2 = A \to x$, $A \in {}_\Gamma N$ and $x \in ({}_\Gamma N \cup {}_\Gamma T)^*$, add $p_1 = \langle q \rangle \to \langle r \rangle$ into ${}_OP$ and $p_1 p_2$ into ${}_OC$. Furthermore, for all $q \in F$, add $p = \langle q \rangle \to \varepsilon$ into ${}_OP$ and p into ${}_OC$.

The complete formal proof of $L(O) = L(\Gamma)_2$ can be found in [17].

On the other hand, the CFG in the RT can be exploited as an additional storage space of the FA to remember some non-negative integers. If the automaton uses the CFG in this way, the additional storage space is akin to counters in a multi-counter machine. The following lemma says that the FAs in the RTs are able to accept every language accepted by partially blind k-counter automata.

Lemma 5 *For every k-PBCA I, there is an RT $\Gamma = (M, G, \Psi)$ such that $L(I) = L(\Gamma)_1$.*

Proof. Let $I = ({}_IQ, \Sigma, {}_I\delta, q_0, F)$ be a k-PBCA for some $k \geq 1$ and construct RT $\Gamma = (M = ({}_MQ, \Sigma, {}_M\delta, q_0, F), G = (N, T, P, S), \Psi)$ as follows: Set $T = \emptyset$, $\Psi = \emptyset$, $N = \{S, A_1, \ldots, A_k\}$, $P = \{A \to \varepsilon |\ A \in N\}$, ${}_M\delta = \{f \to f |\ f \in F\}$, and ${}_MQ = {}_IQ$.

For each $pa \to q(t_1, \ldots, t_k)$ in ${}_I\delta$ and for $n = (\Sigma_{i=1}^{k} max(0, -t_i))$ add:

- $q_1, \ldots, q_n$ into ${}_MQ$;
- $r = S \to xS$, where $x \in (N - \{S\})^*$ and $occur(A_i, x) = max(0, t_i)$, for $i = 1, \ldots, k$, into P;
- $r_1 = q_0 a \to q_1, r_2 = q_1 \to q_2, \ldots, r_n = q_{n-1} \to q_n, r_{n+1} = q_n \to q$ into ${}_M\delta$ with $q_0 = p$; and $(r_{i+1}, \alpha_i \to \varepsilon)$, where $\alpha_i = A_j$ and each A_j is erased $max(0, -t_i)$-times during the sequence, into Ψ ($n = 0$ means that only $pa \to q$, $S \to xS$ and (r_1, r) are considered);
- $(f \to f, S \to \varepsilon)$ into Ψ for all $f \in F$.

The FA of the created system uses the CFG as an external storage. Each counter of the I is represented by a nonterminal. Every step from p to q that modifies counters are simulated by several steps leading from p to q and during this sequence of steps the number of occurrences of each nonterminal in the grammar is modified to be equal to the corresponding counter in I. Clearly, $L(I) = L(\Gamma)_1$.

Lemma 6 states that the CFG is helpful for the FA in RT at most with the preservation of the non-negative numbers without possibility to check their values.

Lemma 6 *For every RT* $\Gamma = (M, G, \Psi)$*, there is a k-PBCA* O *such that* $L(O) = L(\Gamma)_1$ *and* k *is the number of nonterminals in* G*.*

Proof. Let $\Gamma = (M = (Q, \Sigma, {}_M\delta, q_0, F), G = (N, T, P, S), \Psi)$ be an RT. Without any loss of generality suppose that $N = \{A_1, \ldots, A_n\}$, where $S = A_1$. The partially blind $|N|$-counter automaton $O = (Q, \Sigma, {}_O\delta, q_0, F)$ is created in the following way. For each $r_1 = pa \to q \in {}_M\delta$ and $r_2 = \alpha \to \beta \in P$ such that $(r_1, r_2) \in \Psi$, add $pa \to q(v_1, \ldots, v_{|N|})$, where $v_i = occur(A_i, \beta) - occur(A_i, \alpha)$ for all $i = 1, \ldots, |N|$.

The constructed partially blind $|N|$-counter automaton has a counter for each nonterminal from the grammar of Γ. Whenever the automaton in Γ makes a step and sentential form of grammar G is changed, O makes the same step and accordingly changes number of occurrences of nonterminals in its counters.

From Lemma 5 and Lemma 6, we can establish the following theorem.

Theorem 7

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

Proof. It directly follows from Lemma 6 and Lemma 5.

For the better illustration of the accepting and generating power of RT, let us recall that the class of languages generated by MATs is properly included in the class of **RE** languages [1, 6], and the class of languages defined by partially blind k-counter automata, with respect to number of counters, is superset of the class of **CF** languages and properly included in the class of **CS** languages [7, 8].

Unfortunately, the price for the leftmost restriction, placed on derivations in the CFG, is relatively high and both accepting and generative ability of RT with the restriction decreases to the definition of **CF** languages.

Theorem 8

$$\mathscr{L}(RT_{lm})_2 = \mathbf{CF}$$

Proof. The inclusion $\mathbf{CF} \subseteq \mathscr{L}(RT_{lm})_2$ is clear from the definition, because any time we can construct leftmost restricted RT, where the automaton M cycles with reading all possible symbols from the input or ε whilst the grammar G is generating some output string. Therefore, we only need to prove the opposite inclusion.

We know that the class of context-free languages is defined, inter alia, by non-deterministic PDA. It is therefore sufficient to prove that every language $L_{lm}(\Gamma)_2$ of RT can be accepted by a non-deterministic PDA. Consider an RT $\Gamma = ({}_\Gamma M = (Q, {}_\Gamma\Sigma, {}_\Gamma\delta, q_0, F), G = (N, T, P, S), \Psi)$ and define PDA $O = (Q, T, {}_O\Gamma, {}_O\delta, q_0, S, F)$, where ${}_O\Gamma = N \cup T$ and ${}_O\delta$ is created as follows:

- set ${}_O\delta = \emptyset$;
- for each $r_1 = A \to x \in P$ and $r_2 = pa \to q \in {}_\Gamma\delta$ such that $(r_1, r_2) \in \Psi$, add $Ap \to (x)^R q$ into ${}_O\delta$;
- for each $p \in Q$ and $a \in T$ add $apa \to p$ into ${}_O\delta$.

The complete formal proof of $L(O) = L_{lm}(\Gamma)_2$ can be found in [17].

Lemma 9 *For every language $L \in \mathbf{CF}$, there is an RT $\Gamma = (M, G, \Psi)$ such that $L_{lm}(\Gamma)_1 = L$.*

Proof. Let $I = ({}_IN, T, {}_IP, S)$ be a CFG such that $L(I) = L$. For I, we can construct a CFG $H = ({}_HN, T, {}_HP, S)$, where ${}_HN = {}_IN \cup \{\langle a\rangle |\ a \in T\}$ and ${}_HP = \{\langle a\rangle \to a |\ a \in T\} \cup \{A \to x |\ A \to x' \in {}_IP$ and x is created from x' by replacing all $a \in T$ in x' with $\langle a\rangle\}$. Surely, $L(I) = L(H)$ even if H replaces only the leftmost nonterminals in each derivation step. In addition, we construct FA $M = (\{q_0\}, T, \delta, q_0, \{q_0\})$ with $\delta = \{q_0 \to q_0\} \cup \{q_0a \to q_0 |\ a \in T\}$, and $\Psi = \{(q_0 \to q_0, A \to x) |\ A \to x \in {}_HP, A \in {}_IN\} \cup \{(q_0a \to q_0, \langle a\rangle \to a) |\ a \in T\}$.

It is easy to see that any time when H replaces nonterminals from ${}_IN$ in its sentential form, M reads no input symbol. If and only if H replaces $\langle a\rangle$ with a, where $a \in T$, then M reads a from the input. Since H works in a leftmost way, 2-$L_{lm}(\Gamma) = \{(w, w) |\ w \in L(I)$. Hence, $L_{lm}(\Gamma)_1 = L(I)$.

Similarly, we show that any RT generating outputs in the leftmost way can recognize no language out of **CF**.

Lemma 10 *Let Γ is an RT. Then, for every language $L_{lm}(\Gamma)_1$, there is a PDA O such that $L_{lm}(\Gamma)_1 = L(O)$.*

Proof. In the same way as in the proof of Theorem 4, we construct PDA O such that $L(O) = L_{lm}(\Gamma)_1$ for RT $\Gamma = (M = (Q, {}_\Gamma\Sigma, {}_\Gamma\delta, q_0, F), G = (N, T, P, S), \Psi)$. We define O as $O = (Q, {}_\Gamma\Sigma, N, {}_O\delta, q_0, S, F)$, where ${}_O\delta$ is created in the following way:

- set ${}_O\delta = \emptyset$;
- for each $r_1 = pa \to q \in {}_\Gamma\delta$ and $r_2 = A \to x \in P$ such that $(r_1, r_2) \in \Psi$, add $Apa \to (\theta(x))^R q$ into ${}_O\delta$, where $\theta(x)$ is a function from $(N \cup T)^*$ to N^* that replaces all terminal symbols in x with ε – that is, $\theta(x)$ is x without terminal symbols.

The complete formal proof of $L(O) = L_{lm}(\Gamma)_1$ can be found in [17].

Theorem 11

$$\mathscr{L}(RT_{lm})_1 = \mathbf{CF}$$

Proof. It directly follows from Lemma 9 and Lemma 10.

By the appearance checking both generative and accepting power of RT grow to the power of Turing machines.

Theorem 12

$$\mathscr{L}(RT_{ac})_2 = \mathbf{RE}$$

Proof. Since $\mathscr{L}(\mathrm{MAT_{ac}}) = \mathbf{RE}$ [6], we only need to prove that $\mathscr{L}(\mathrm{MAT_{ac}}) \subseteq \mathscr{L}(\mathrm{RT_{ac}})_2$.

Consider a MAT_{ac} with appearance checking $I = ({}_IG, {}_IC)$ and construct RT $\Gamma = ({}_\Gamma M, {}_\Gamma G, \Psi)$, such that $L(I) = L(\Gamma)_2$, as follows: Set ${}_\Gamma G = {}_IG$; add a new initial nonterminal S', nonterminal Δ, and rules $\Delta \to \Delta$, $\Delta \to \varepsilon$, $S' \to S\Delta$ into grammar ${}_\Gamma G$; and construct FA ${}_\Gamma M = (Q, \Sigma, \delta, s, F)$ and Ψ in the following way: Set

$F = Q = \{s\}$, $\delta = \{s \to s\}$, and $\Psi = \{(s \to s, \Delta \to \varepsilon, \Delta \to \varepsilon), (s \to s, S' \to S\Delta, S' \to S\Delta)\}$; for every $m = (p_1, t_1) \dots (p_k, t_k) \in {}_IC$, add $q_1, q_2, \dots, q_{k-1}$ into Q, $s \to q_1, q_1 \to q_2, \dots, q_{k-2} \to q_{k-1}, q_{k-1} \to s$ into δ, and $(s \to q_1, p_1, c_1), (q_1 \to q_2, p_2, c_2), \dots, (q_{k-2} \to q_{k-1}, p_{k-1}, c_{k-1})$, $(q_{k-1} \to q_s, p_k, c_k)$ into Ψ, where, for $1 \leq i \leq k$, if $t_i = -$, then $c_i = p_i$; otherwise, $c_i = \Delta \to \Delta$.

Since S' is the initial symbol, the first computation step in Γ is $(s, S') \Rightarrow (s, S\Delta)$. After this step, the FA simulates matrices in I by computation step. That is, if $x_1 \Rightarrow x_2[p]$ in I, where $p = p_1, \dots, p_i$ for some $i \in \mathbb{N}$, then there is $q_1, \dots, q_{i-1} \in Q$ such that $r_1 = s \to q_1, r_2 = q_1 \to q_2, \dots, r_{i-1} = q_{i-2} \to q_{i-1}, r_i = q_{i-1} \to s \in \delta$ and $(r_1, p_1, c_1), \dots, (r_i, p_i, c_i) \in \Psi$. Therefore, $(s, x_1) \Rightarrow^i (s, x_2)$ in Γ. Notice that if I can overleap some grammar rule in $m \in {}_IC$, Γ represents the fact by using $\Delta \to \Delta$ with the move in ${}_\Gamma M$. Similarly, if, for some $i \in \mathbb{N}$, $(s, x_1) \Rightarrow^i (s, x_2)$ in Γ and there is no $j < i$ such that $(s, x_1) \Rightarrow^j (s, y) \Rightarrow^* (s, x_2)$, there exists $p \in {}_IC$ such that $x_1 \Rightarrow x_2[p]$ in I. Hence, if $(s, S) \Rightarrow^* (s, w)$ in Γ, where w is a string over the set of terminals in ${}_\Gamma G$, then $S \Rightarrow^* w$ in I; and, on the other hand, if $S \Rightarrow^* w$ in I for a string over the set of terminals in ${}_IG$, then $(s, S') \Rightarrow (s, S\Delta) \Rightarrow^* (s, w\Delta) \Rightarrow (s, w)$ in Γ.

Theorem 13

$$\mathscr{L}(RT_{ac})_1 = \mathbf{RE}$$

Proof. Let $I = ({}_IQ, \Sigma, {}_I\delta, q_0, F)$ be a k-CA for some $k \geq 1$ and construct RT $\Gamma = (M, G, \Psi)$, where $M = ({}_MQ, \Sigma, {}_M\delta, q_0, F)$, $G = (N, T, P, S)$, as follows: Set $T = \{a\}, \Psi = \emptyset, P = \{A \to \varepsilon, A \to \Diamond |\ A \in N - \{\Diamond\}\} \cup \{S \to S\}$, ${}_MQ = {}_IQ$, ${}_M\delta = \{f \to f |\ f \in F\}$, and $N = \{S, \Diamond, A_1, \dots, A_k\}$. For each $pa \to q(t_1, \dots, t_k)$ in ${}_I\delta$, $n = \Sigma_{i=1}^k \theta(t_i)$, and $m = \Sigma_{i=1}^k \widehat{\theta}(t_i)$, where if $t_i \in \mathbb{Z}$, $\theta(t_i) = max(0, -t_i)$ and $\widehat{\theta}(t_i) = max(0, t_i)$; otherwise $\theta(t_i) = 1$ and $\widehat{\theta}(t_i) = 0$, add:

- $q_1, \dots, q_n$ into ${}_MQ$;
- $r = S \to xS$, where $x \in (N - \{S, \Diamond\})^*$ and $occur(A_i, x) = \widehat{\theta}(t_i)$, for each $i = 1, \dots, k$, into P;
- $r_1 = q_0 a \to q_1$, $r_2 = q_1 \to q_2$, …, $r_n = q_{n-1} \to q_n$, $r_{n+1} = q_n \to q$ into ${}_M\delta$ with $q_0 = p$; and for each $i = 1, \dots, n$, add $(r_{i+1}, \tau_i, \tau_i')$, where for each $j = 1, \dots, k$, if $t_j \in \mathbb{N}$, for $\theta(t_j)$ is, $\tau_i = \tau_i' = A_j \to \varepsilon$; otherwise, if $t_j = -$, $\tau_i = A_j \to \Diamond$ and $\tau_i' = S \to S$, into Ψ. Notice that $n = 0$ means that only $q_0 a \to q$, $S \to xS$ are considered. Furthermore, add (r_1, r, r) into Ψ;
- $(f \to f, S \to \varepsilon, S \to \varepsilon)$ into Ψ for all $f \in F$.

Similarly as in the proof of Theorem 5, the FA of the created system uses CFG as an external storage, and each counter of the I is represented by a nonterminal. If I modifies some counters during a move from state p to state q, M moves from p to q in several steps during which it changes the numbers of occurrences of nonterminals correspondingly. Rules applicable only if some counters are equal to zero are simulated by using an appearance checking, where Γ tries to replace all nonterminals representing counters which have to be 0 by $\Diamond$. If it is not possible, Γ applies the rule $S \to S$ and continue with computation. Otherwise, since $\Diamond$ cannot be rewritten

during the rest of computation, use of such rules leads to an unsuccessful computation. The formal proof of the equivalence of languages is left to the reader. Since, $\mathscr{L}(k\text{-CA}) = \mathbf{RE}$ for every $k \geq 2$ [11], Theorem 13 holds.

5. Linguistic application perspectives

In this section, we discuss advantages of the new formal models regarding their potential applications in natural language processing.

One of the main advantages of both proposed systems is their power, which, as demonstrated above, goes beyond the class of context-free languages. This allows us to describe even features of natural languages that are difficult to capture within a purely context-free framework.

Another advantage of our synchronous grammars is their high flexibility, especially if we synchronize models that have higher generative power themselves, such as regulated grammars. In particular, let us consider the case of SMAT. As shown above, if we synchronize MATs in the proposed fashion, we do not obtain any further increase in power of the whole system compared to RSCFG (equal to the power of MAT). However, more powerful individual components allow for easier description of each individual language.

Unlike synchronous grammars, which are symmetric and therefore can be used for bidirectional translation, RT can only describe translation in one direction. However, an important advantage of RT, especially with regards to practical implementation, lies in its straightforward basic principle: read input with FA, generate output with CFG. Even with these relatively simple compononents, the system as a whole can describe even some non-context-free features.

To illustrate some of the main advantages, let us now consider translation between Czech and English. Czech is a relatively challenging language in terms of NLP. It is a free-word-order language with rich inflection [10].

For example, consider the Czech sentence *dva růžoví sloni přišli na přednášku* (*two pink elephants came to the lecture*). All of the following permutations of words also make for a valid sentence:

dva růžoví sloni přišli na přednášku	*dva růžoví sloni na přednášku přišli*
růžoví sloni přišli na přednášku dva	*růžoví sloni na přednášku přišli dva*
dva sloni přišli na přednášku růžoví	*dva sloni na přednášku přišli růžoví*
sloni přišli na přednášku dva růžoví	*sloni na přednášku přišli dva růžoví*

There may be differences in meaning or emphasis, but the syntactic structure remains the same. Why is this problematic? Compare the syntax trees in Figure 2. Because of the crossing branches (non-projectivity), the second tree cannot be produced by any CFG. Of course, it is still possible to construct a CFG that generates the sentence *růžoví sloni přišli na přednášku dva* if we consider a different syntax tree, for example such as in Figure 3. However, this tree no longer captures the relation between the noun *sloni* and its modifying numeral *dva* (represented

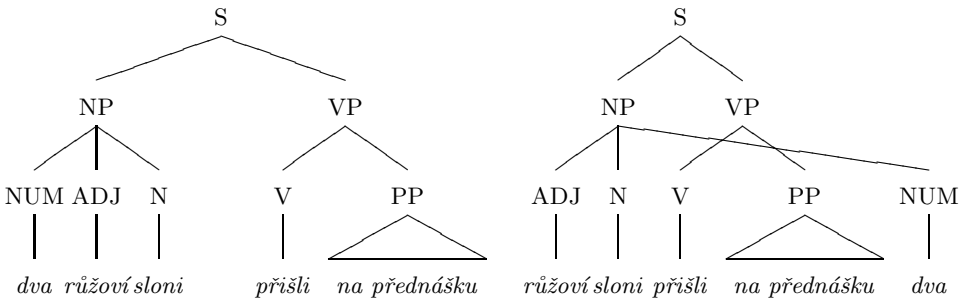

Fig. 2. Syntax trees for example sentences

by the dotted line). We need to know this relation for instance to ensure agreement between the words (person, number, gender...), so that we can choose their appropriate forms.

In a purely context-free framework, this is complicated. The necessary information has to be propagated through the tree, even if the structure is not actually affected. This can result in a high number of rules. With MAT, we can instead represent the relations using matrices.

Example 3 Here, we present an example of SMAT $H = (G_{cz}, M_{cz}, G_{en}, M_{en}, \Psi, \varphi_{cz}, \varphi_{en})$ that describes the translations between the English sentence *two pink elephants came to the lecture* and any of the above Czech sentences, correctly distinguishing between male and female gender in Czech (to demonstrate female gender, we also include *opice* in Czech, *monkeys* in English). Note that H is actually more general (for example allowing multiple adjectives). It is designed for easy extension to include other grammatical categories (person...) as well as different syntactic structures.

Context-free rules in G_{cz} (Czech), nonterminals are in capitals, S is the start symbol:

s :	$\mathrm{S} \to \mathrm{NP\ VP\ NUM\ ADJS}$,	np :	$\mathrm{NP} \to \mathrm{NUM\ ADJS\ N}$,
vp :	$\mathrm{VP} \to \mathrm{ADVS\ V\ ADVS}$,	num_ε :	$\mathrm{NUM} \to \varepsilon$,
adjs :	$\mathrm{ADJS} \to \mathrm{ADJ\ ADJS}$,	$\mathrm{adjs}_\varepsilon$:	$\mathrm{ADJS} \to \varepsilon$,
advs :	$\mathrm{ADVS} \to \mathrm{ADV\ ADVS}$,	$\mathrm{advs}_\varepsilon$:	$\mathrm{ADVS} \to \varepsilon$,
n_m :	$\mathrm{N} \to \mathrm{N}_m$,	n_f :	$\mathrm{N} \to \mathrm{N}_f$,
n_{mm} :	$\mathrm{N}_m \to \mathrm{N}_m$,	n_{ff} :	$\mathrm{N}_f \to \mathrm{N}_f$,
v_m :	$\mathrm{V} \to \mathrm{V}_m$,	v_f :	$\mathrm{V} \to \mathrm{V}_f$,
adj_m :	$\mathrm{ADJ} \to \mathrm{ADJ}_m$,	adj_f :	$\mathrm{ADJ} \to \mathrm{ADJ}_f$,
adv :	$\mathrm{ADV} \to \mathrm{PP}$,	num_m :	$\mathrm{NUM} \to \mathrm{NUM}_m$,
num_f :	$\mathrm{NUM} \to \mathrm{NUM}_f$,	dict_1 :	$\mathrm{N}_m \to$ *sloni*,
dict_2 :	$\mathrm{N}_f \to$ *opice*,	dict_{3m} :	$\mathrm{V}_m \to$ *přišli*,
dict_{3f} :	$\mathrm{V}_f \to$ *přišly*,	dict_{4m} :	$\mathrm{ADJ}_m \to$ *růžoví*,
dict_{4f} :	$\mathrm{ADJ}_f \to$ *růžové*,	dict_{5m} :	$\mathrm{NUM}_m \to$ *dva*,
dict_{5f} :	$\mathrm{NUM}_f \to$ *dvě*,	dict_6 :	$\mathrm{PP} \to$ *na přednášku.*

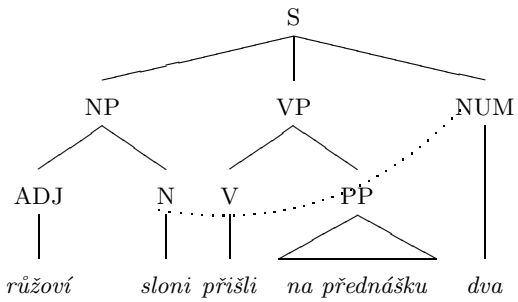

Fig. 3. Modified syntax tree

Context-free rules in G_{en} (English), nonterminals are in capitals, S is the start symbol:

s :	S → NP VP,	np :	NP → NUM ADJS N,
vp :	VP → V ADVS,	num_ε :	NUM → ε,
adjs :	ADJS → ADJ ADJS,	adjs_ε :	ADJS → ε,
advs :	ADVS → ADV ADVS,	advs_ε :	ADVS → ε,
adv :	ADV → PP,	dict_1 :	N → *elephants*,
dict_2 :	N → *monkeys*,	dict_3 :	V → *came*,
dict_4 :	ADJ → *pink*,	dict_5 :	NUM → *two*,
dict_6 :	PP → *to the lecture*.		

Matrices:

	M_{cz}	M_{en}		M_{cz}	M_{en}
s :	s	s	np :	np	np
vp :	vp	vp	num :	num_ε	ε
$num_{\varepsilon\varepsilon}$:	num_ε num_ε	num_ε	$adjs$:	adjs	adjs
$adjs_\varepsilon$:	adjs_ε adjs_ε	adjs_ε	$advs$:	adjs	adjs
$advs_\varepsilon$:	advs_ε advs_ε	advs_ε	n_m :	n_m	ε
n_f :	n_f	ε	v_m :	v_m n_{mm}	ε
v_f :	v_f n_{ff}	ε	adj_m :	adj_m n_{mm}	ε
adj_f :	adj_f n_{ff}	ε	adv :	adv	adv
num_m :	num_m n_{mm}	ε	num_f :	num_f n_{ff}	ε
$dict_1$:	dict_1	dict_1	$dict_2$:	dict_2	dict_2
$dict_{3m}$:	dict_{3m}	dict_3	$dict_{3f}$:	dict_{3f}	dict_3
$dict_{4m}$:	dict_{4m}	dict_4	$dict_{4f}$:	dict_{4f}	dict_4
$dict_{5m}$:	dict_{5m}	dict_5	$dict_{5f}$:	dict_{5f}	dict_5
$dict_6$:	dict_6	dict_6			

Note for example the matrix adj_f in M_{cz}, which ensures agreement between noun and adjective (both must be in female gender). Another interesting matrix is $adjs_\varepsilon$, which terminates generation of adjectives. In the Czech sentence in this example, we have two positions where adjectives can be placed (directly within the noun phrase or at the end of the sentence). In English, there is only one (within the noun phrase). This is why the rule ADJS → ε is used twice in Czech, but only once in English.

Also note that the linked matrices (sharing the same label) in M_{cz} and M_{en} may contain completely different rules and in some cases one can even be empty (ε). The definitions of MAT and SMAT allow for this kind of flexibility when describing both individual languages and their translations.

Example of a derivation in Czech follows:

S $\Rightarrow$ NP VP NUM ADJS [*s*] $\Rightarrow$ NUM ADJS N VP NUM ADJS [*np*] $\Rightarrow$ NUM ADJS N ADVS V ADVS NUM ADJS [*vp*] $\Rightarrow$ ADJS N ADVS V ADVS NUM ADJS [*num*] $\Rightarrow$ ADJ ADJS N ADVS V ADVS NUM ADJS [*adjs*] $\Rightarrow$ ADJ N ADVS V ADVS NUM [$adjs_\varepsilon$] $\Rightarrow$ ADJ N ADVS V ADV ADVS NUM [*advs*] $\Rightarrow$ ADJ N V ADV NUM [$advs_\varepsilon$] $\Rightarrow$ ADJ N_m V ADV NUM [n_m] $\Rightarrow$ ADJ N_m V_m ADV NUM [v_m] $\Rightarrow$ ADJ_m N_m V_m ADV NUM [adj_m] $\Rightarrow$ ADJ_m N_m V_m PP NUM [*adv*] $\Rightarrow$ ADJ_m N_m V_m PP NUM_m [num_m] $\Rightarrow$ ADJ_m *sloni* V_m PP NUM_m [$dict_1$] $\Rightarrow$ ADJ_m *sloni přišli* PP NUM_m [$dict_{3m}$] $\Rightarrow$ *růžoví sloni přišli* PP NUM_m [$dict_{4m}$] $\Rightarrow$ *růžoví sloni přišli na přednášku* NUM_m [$dict_5$] $\Rightarrow$ *růžoví sloni přišli na přednášku dva* [$dict_{6m}$]

The corresponding derivation in English may look like this:

S $\Rightarrow$ NP VP [*s*] $\Rightarrow$ NUM ADJS N VP [*np*] $\Rightarrow$ NUM ADJS N V ADVS [*vp*] $\Rightarrow$ NUM ADJS N V ADVS [*num*] $\Rightarrow$ NUM ADJ ADJS N V ADVS [*adjs*] $\Rightarrow$ NUM ADJ N V ADVS [$adjs_\varepsilon$] $\Rightarrow$ NUM ADJ N V ADV ADVS [*advs*] $\Rightarrow$ NUM ADJ N V ADV [$advs_\varepsilon$] $\Rightarrow$ NUM ADJ N V ADV [n_m] $\Rightarrow$ NUM ADJ N V ADV [v_m] $\Rightarrow$ NUM ADJ N V ADV [adj_m] $\Rightarrow$ NUM ADJ N V PP [*adv*] $\Rightarrow$ NUM ADJ N V PP [num_m] $\Rightarrow$ NUM ADJ *elephants* V_m PP [$dict_1$] $\Rightarrow$ NUM ADJ *elephants came* PP [$dict_{3m}$] $\Rightarrow$ NUM *pink elephants came* PP [$dict_{4m}$] $\Rightarrow$ NUM *pink elephants came to the lecture* [$dict_5$] $\Rightarrow$ *two pink elephants came to the lecture* [$dict_{6m}$]

The entire derivation tree for the Czech sentence is shown in Figure 4. The dotted lines represent relations described by matrices. The triangle from N_m to N_m is an abstraction which in this particular case essentially means that this step is repeated until all agreement issues are resolved.

We can achieve similar results using SSCGs. For example the matrix adj_f in M_{cz} can be represented by two scattered-context rules (ADJ, N_f) $\rightarrow$ (ADJ_f, N_f) and (N_f, ADJ) $\rightarrow$ (N_f, ADJ_f). Note that we need two rules, because the nonterminal order is important in SCG. This is one of the key differences between SMAT and SSCG. In this case, we need an additional rule in SSCG. However, this can also be an advantage, because it allows us to easily distinguish between left and right modifiers. For example, if we only have the first rule (ADJ, N_f) $\rightarrow$ (ADJ_f, N_f), it means that the adjective always has to occur on the left of the noun.

With RT, we can also represent the relations discussed in the above example. For instance, we can use the states of FA to store the information about gender (as well as other grammatical categories).

Further examples of application perspectives of our synchronous grammars and RT can be found in [13] and [17], respectively.

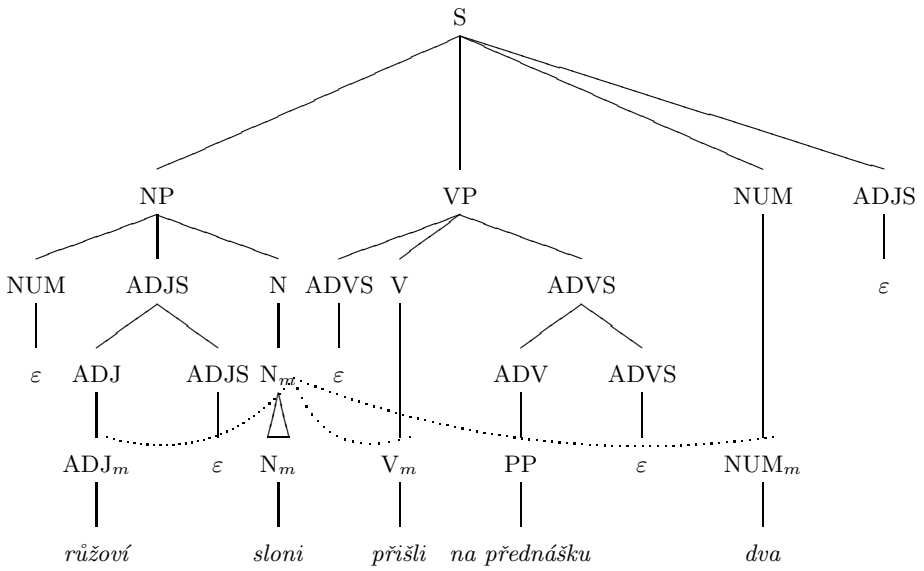

Fig. 4. Derivation tree

6. Conclusion

We have presented new grammar systems that can formally describe translations (or, more specifically, transformations of syntactic structures). We have discussed some of the theoretical properties of the new models, in particular their generative and accepting power. We have also briefly discussed application perspectives in translation of natural languages. Besides natural language processing, the models can be useful in other translation and transformation tasks, such as compilers.

Further research prospects include the study of other theoretical properties of the proposed models, such as descriptional complexity. We can also consider other variants and restrictions, systems consisting of other well-known grammars and automata, and even systems with more than two components. Finally, note that although our synchronous grammars and RT represent different approaches and are defined differently, there is an important similarity in their basic principles. In essence, they are all systems in which the cooperation of components is achieved by synchronization of their rules. It might be useful to introduce a more general formalism encompassing all such rule-synchronized systems.

From a more practical viewpoint, efficient parsing with matrix grammars and scattered context grammars still represents an open problem.

7. References

[1] Abraham S.; *Some questions of language theory.* In: *Proceedings of the 1965 conference on Computational linguistics*, COLING '65, Stroudsburg, PA, USA, 1965, Association for Computational Linguistics, pp. 1–11.

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

[3] Allen J.; *Natural language understanding (2nd edition)*, Benjamin/Cummings series in computer science. Benjamin/Cummings Pub. Co., 1995.

[4] Bojar O., Čmejrek M.; *Mathematical model of tree transformations.* In: *Project Euromatrix Deliverable 3.2*, Charles University, Prague 2007.

[5] Chiang. D.; *An introduction to synchronous grammars.* In: *44th Annual Meeting of the Association for Computational Linguistics*, 2006.

[6] Dassow J., Păun Gh.; *Regulated Rewriting in Formal Language Theory*, Springer-Verlag, Berlin 1989.

[7] Ginsburg S.; *Algebraic and Automata-Theoretic Properties of Formal Languages*, Elsevier Science Inc., New York, NY, USA, 1975.

[8] Greibach S.A.; *Remarks on blind and partially blind one-way multicounter machines*, Theor. Comput. Sci. 7, 1978, pp. 311–324.

[9] Gurari E.M., Ibarra O.H.; *A note on finite-valued and finitely ambiguous transducers*, Theory of Computing Systems 16, 1983, pp. 61–66.

[10] Hajič J.; *Disambiguation of Rich Inflection: Computational Morphology of Czech*, Wisconsin Center for Pushkin Studies, Karolinum 2004.

[11] Hopcroft J.E., Motwani R., Ullman J.D.; *Introduction to Automata Theory, Languages, and Computation*, Addison Wesley, 2000.

[12] Horáček P., Meduna A.; *Regulated rewriting in natural language translation.* In: *7th Doctoral Workshop on Mathematical and Engineering Methods in Computer Science*, Brno University of Technology, Brno, CZ, 2011, pp. 35–42.

[13] Horáček P., Meduna A.; *Synchronous versions of regulated grammars: Generative power and linguistic applications*, Theoretical and Applied Informatics 24(3), 2012, pp. 175–190.

[14] Khalilov M., Fonollosa J.A.R.; *N-gram-based statistical machine translation versus syntax augmented machine translation: comparison and system combination.* In: *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '09, Association for Computational Linguistics, Stroudsburg, PA, USA, 2009, pp. 424–432.

[15] Meduna A.; *Automata and Languages: Theory and Applications*, Springer, London 2000.

[16] Meduna A., Techet J.; *Scattered Context Grammars and their Applications*, WIT Press, United Kingdom, 2010.

[17] Meduna A., Čermák M., Horáček P.; *Rule-restricted automaton-grammar tranduscers: Power and linguistic applications*, Mathematics for Applications 1(1), 2012, pp. 13–35.

[18] Mitkov P. (ed.); *The Oxford Handbook of Computational Linguistics*, Oxford University Press, Oxford 2003.

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

[20] Rozenberg G., Salomaa A. (eds.); *Handbook of Formal Languages*, Springer-Verlag, Berlin 1997.

[21] Zollmann A., Venugopal A.; *Syntax augmented machine translation via chart parsing*. In: *Proceedings of the Workshop on Statistical Machine Translation*, StatMT '06, Association for Computational Linguistics, Stroudsburg, PA, USA, 2006, pp. 138–141.