Math. Appl. **1** (2012), 13–35

RULE-RESTRICTED AUTOMATON-GRAMMAR TRANSDUCERS: POWER AND LINGUISTIC APPLICATIONS

MARTIN ČERMÁK, PETR HORÁČEK and ALEXANDER MEDUNA

Abstract. This paper introduces the notion of a new transducer as a two-component system, which consists of a finite automaton and a context-free grammar. In essence, while the automaton reads its input string, the grammar produces its output string, and their cooperation is controlled by a set, which restricts the usage of their rules.

From a theoretical viewpoint, the present paper discusses the power of this system working in an ordinary way as well as in a leftmost way. In addition, the paper introduces an appearance checking, which allows us to check whether some symbols are present in the rewritten string, and studies its effect on the power. It achieves the following three main results. First, the system generates and accepts languages defined by matrix grammars and partially blind multi-counter automata, respectively. Second, if we place a leftmost restriction on derivation in the context-free grammar, both accepting and generating power of the system is equal to generative power of context-free grammars. Third, the system with appearance checking can accept and generate all recursively enumerable languages. From more pragmatical viewpoint, this paper describes several linguistic applications. A special attention is paid to the Japanese-Czech translation.

1. Introduction

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 [2], [18], [21], [14], or [22]). The second category is composed of language-translation systems or, more briefly, transducers. 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 (see [3], [10], [20], and [23]).

Although transducers represent language-translation devices, language theory often views them as language-defining devices and investigates the language family resulting from them. That is, 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

2010 *MSC:* primary 68Q45.

Keywords: n-languages, transducer, computational control.

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).

strings. The present paper contributes to this vivid investigation trend in formal language theory.

In this paper, we introduce a new type 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.

The present paper discusses the power of this system working in an ordinary way as well as in a leftmost way and investigates an effect of an appearance checking placed into the system. First, we show that the generative power is equal to the generative power of matrix grammars (see [1] or [7]). Second, the accepting power coincides with the power of partially blind multi-counter automata (see [8] and [9]). Third, under the context-free-grammar leftmost restriction, the accepting and generating power of these systems coincides with the power of context-free grammars. On the other hand, when an appearance checking is introduced into these systems, the accepting and generating power coincides with the power of Turing machines.

In the last part of the paper, we discuss application-related perspectives of the studied systems in linguistics. Particularly, we concentrate our attention on natural language translation. First, we demonstrate the basic idea in terms of simple English sentence, performing its analysis and passive transformation. Furthermore, we describe the translation of selected sentence structures between the Czech, English, and Japanese languages. We demonstrate that while English and Czech are structurally similar languages, some aspects of Japanese differ significantly. We also show that some linguistic Czech-language features complicate this translation very much. For example, compared to English, there is very rich inflection in Czech. Other difficult-to-handle features include non-projectivity (crossing dependencies between words in a sentence), which mainly results from the fact that Czech is a free-word-order language.

2. Preliminaries

In this paper, we assume the reader is familiar with the formal language theory (see [17]) and the basic aspects of computational linguistics (see [19]).

For a set, Q, $|Q|$ denotes the cardinality of Q. 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 $L(M) = \{w \mid w \in \Sigma^*, q_0 w \Rightarrow^* f \text{ is an acceptance of } w\}$.

A *partially blind k-counter automaton*, k-PBCA, is finite automaton $M = (Q, \Sigma, \delta, q_0, F)$ with k integers $v = (v_1, \dots, v_k)$ 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 $\chi_1 = paw(v_1, \dots, v_k)$ and $\chi_2 = qw(v'_1, \dots, v'_k)$ be two configurations of M and $r = pa \to q(t_1, \dots, t_k) \in \delta$, where $(v_1 + t_1, \dots, v_k + t_k) = (v'_1, \dots, v'_k)$. 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 $L(M) = \{w \mid w \in \Sigma^*, q_0 w(0, \dots, 0) \Rightarrow^* f(0, \dots, 0), f \in F\}$.

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

A *k-counter automaton*, k-CA, is finite automaton $M = (Q, \Sigma, \delta, q_0, F)$ with k integers $v = (v_1, \dots, v_k)$ in $\mathbb{N}_0^k$ as an additional storage. Transition rules in δ are of the form $pa \to q(t_1, \dots, t_n)$, 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 $\chi_1 = paw(v_1, \dots, v_k)$ and $\chi_2 = qw(v'_1, \dots, v'_k)$ be two configurations of M and $r = pa \to q(t_1, \dots, t_k) \in \delta$, 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 of M is defined as $L(M) = \{w \mid w \in \Sigma^*, q_0 w(0, \dots, 0) \Rightarrow^* f(0, \dots, 0), f \in F\}$.

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

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

A *matrix grammar with appearance checking*, MAT_{ac}, is a pair $H = (G, C)$, where $G = (N, T, P, S)$ is a context-free grammar and C is a finite set of strings, matrices, of pairs (p, t) with $p \in P$ and $t \in \{-, +\}$. 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 m, written as $u \Rightarrow v[m]$, or simply $u \Rightarrow v$, if $m = (p_1, t_1) \dots (p_m, t_m) \in C$ and there are $v_0, \dots, v_m$, where $v_0 = u$, $v_m = v$, and for all $i = 0, \dots, m-1$, either $v_i \Rightarrow v_{i+1}[p_{i+1}]$ in G, or $t_{i+1} \in \{-\}$, $v_i = v_{i+1}$, and p_{i+1} is not applicable on v_i in G. Let $\Rightarrow^*$ and $\Rightarrow^+$ denote transitive-reflexive and transitive closure of $\Rightarrow$. The language of H is defined as $L(H) = \{w| \ S \Rightarrow w_1[m_1] \Rightarrow \dots w_n[m_n], w_n = w, m_1, \dots, m_n \in C, w \in T^*, n \geq 0\}$. The class of languages generated by matrix grammars with appearance checking is denoted by $\mathscr{L}(MAT_{ac})$.

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

3. Rule-restricted transducer

In this section, we define and investigate a rule-restricted transducers consisting of a finite automaton, a context-free grammar, and a control set of pairs of rules. The finite automaton reads its own input string and, simultaneously, the context-free grammar generates an output string. During the computation, the control set determines which rules can be used at the same computation step performed by both components. The computation of the system is successful if and only if the finite automaton accepts the input string and the context-free grammar successfully generates a string of terminal symbols.

Definition 3.1. The *rule-restricted transducer*, RT for short, is a triplet $\Gamma = (M, G, \Psi)$, where $M = (Q, \Sigma, \delta, q_0, F)$ is a finite automaton, $G = (N, T, P, S)$ is a context-free grammar, and Ψ is a finite set of pairs of the form (r_1, r_2), where r_1 and r_2 are rules from δ and P, respectively.

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

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

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

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

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

It is well-known that finite automata and context-free grammars describe different classes of languages. Specifically, by the finite automata we can accept regular languages, whereas the context-free grammars define the class of context-free languages. However, in Example 3.2 is shown that by the combination of these two models, the system is able to accept and generate non-context-free languages.

Example 3.2. Consider RT $K = (M, G, \Psi)$ with
$M = (\{1, 2, 3', 3, 4, 5', 5, 6\}, \{a, b\}, \delta, 1, \{6\})$, where
– $\delta = \{$
$p_1 = 1a \to 2$, $p_2 = 2 \to 1$, $p_3 = 1b \to 3'$, $p_4 = 3' \to 3$,
$p_5 = 3b \to 4$, $p_6 = 4 \to 3$, $p_7 = 3a \to 5'$, $p_8 = 5' \to 5$,
$p_9 = 5a \to 5$, $p_{10} = 5b \to 6$, $p_{11} = 6b \to 6\}$
(see the graphical representation of M in Figure 1),
$G = (\{S, A, B, C, D, D'\}, \{a, b\}, P, S)$, where
– $P = \{$
$r_1 = S \to BbD'$, $r_2 = B \to Bb$, $r_3 = D' \to D'D$,
$r_4 = B \to aA$, $r_5 = D' \to C$, $r_6 = A \to aA$,
$r_7 = C \to CC$, $r_8 = D \to b$, $r_9 = A \to \varepsilon$,
$r_{10} = C \to a\}$,
$\Psi = \{(p_1, r_1), (p_1, r_2), (p_2, r_3), (p_3, r_4), (p_4, r_5), (p_5, r_6), (p_6, r_7), (p_7, r_8), (p_8, r_9), (p_9, r_8), (p_{10}, r_{10}), (p_{11}, r_{10})\}$.

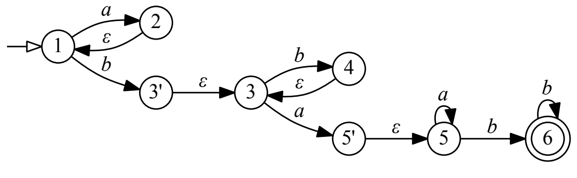


Figure 1. Definition of finite automaton M from Example 3.2.

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 matrix grammars.

Theorem 3.3. $\mathscr{L}(RT)_2 = \mathscr{L}(MAT)$.

Proof. 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 finite automaton ${}_\Gamma M = (Q, \Sigma, \delta, s, F)$ 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 finite automaton simulates matrices in I by moves. 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 $(r_1, p_1), \dots, (r_i, p_i) \in \Psi$. Therefore, $(s, x_1) \Rightarrow^i (s, x_2)$ in Γ. Similarly, if $(s, x_1) \Rightarrow^i (s, x_2)$ in Γ, for $i \in \mathbb{N}$, and there is no $j \in \mathbb{N}$ such that $0 < j < i$ and $(s, x_1) \Rightarrow^j (s, y) \Rightarrow^* (s, x_2)$, there has to be $p \in {}_I C$ and $x_1 \Rightarrow x_2[p]$ in I. Hence, if $(s, S) \Rightarrow^* (s, w)$ in Γ, where w is a string over the set of terminal symbols 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 ${}_I G$, then $(s, S) \Rightarrow^* (s, w)$ in Γ. The inclusion $\mathscr{L}(MAT) \subseteq \mathscr{L}(RT)_2$ has been proven.

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 $O = ({}_O G, {}_O C)$ such that $L(\Gamma)_2 = L(O)$ as follows: Set ${}_O G = ({}_\Gamma N \cup \{S'\}, {}_\Gamma T, {}_O P, S')$, ${}_O P = {}_\Gamma P \cup \{p_0 = S' \to \langle s \rangle {}_\Gamma S\}$, and ${}_O C = \{p_0\}$. 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 ${}_O P$ and $p_1 p_2$ into ${}_O C$. Furthermore, for all $q \in F$, add $p = \langle q \rangle \to \varepsilon$ into ${}_O P$ and p into ${}_O C$.

By the following claims, we prove that $L(\Gamma)_2 = L(O)$.

Claim 3.4. If $(sw, {}_\Gamma S) \Rightarrow^* (qw', \omega)$ in Γ, then $S' \Rightarrow^* \langle q \rangle \omega$ in O.

Proof. By induction on the number of computation steps.

- *Basis.* Let $(sw, {}_\Gamma S) \Rightarrow^0 (sw, {}_\Gamma S)$ in Γ. Then, $S' \Rightarrow \langle s \rangle {}_\Gamma S[p]$ in ${}_O G$ and $p \in {}_O C$. Hence, $S' \Rightarrow \langle s \rangle {}_\Gamma S[p]$ in O. Claim 3.4 holds for no steps in Γ.
- *Induction hypothesis.* Suppose that Claim 3.4 holds for j or fewer computation steps.
- *Induction step.* Let $(sw, {}_\Gamma S) \Rightarrow^j (qw', \omega) \Rightarrow (q'w'', \omega')$ in Γ. Then, by the induction hypothesis, $S' \Rightarrow^* \langle q \rangle \omega$ in O. Without any loss of generality suppose that $\omega = uAv$ for $u, v \in ({}_\Gamma N \cup {}_\Gamma T)^*$, $A \in {}_\Gamma N$, and $(qw', uAv) \Rightarrow (q'w'', uxv)$ with $x \in ({}_\Gamma T \cup {}_\Gamma N)^*$ and $\omega' = uxv$. From the construction of O we know that $p_1 = A \to x$ and $p_2 = \langle q \rangle \to \langle q' \rangle$ is in ${}_\Gamma P$ and $p_1 p_2 \in {}_O C$. Therefore, $S' \Rightarrow^* \langle q \rangle \omega \Rightarrow \langle q' \rangle uxv = \langle q' \rangle \omega'$ in O. Claim 3.4 holds. Furthermore, for all $f \in F$ there is a rule $p = \langle f \rangle \to \varepsilon \in {}_\Gamma P$ and $p \in {}_O C$. Hence, if $(sw, {}_\Gamma S) \Rightarrow^* (f, \omega)$, where $f \in F$ and $\omega \in {}_\Gamma T^*$ in Γ, $S' \Rightarrow^* \langle f \rangle \omega \Rightarrow \omega$ in O. That is, $L(\Gamma)_2 \subseteq L(O)$.

□

It remains to prove that $L(O) \subseteq L(\Gamma)_2$.

Claim 3.5. If $S' \Rightarrow^* \langle q \rangle \omega$ in O with $\omega \in {}_\Gamma T^*$, then $(sw, {}_\Gamma S) \Rightarrow^* (f, \omega)$ in Γ for some $w \in \Sigma^*$ and $f \in F$.

Proof. Consider any successful derivation of the form

$$S' \Rightarrow \langle q_0 \rangle \omega_0[p_0] \Rightarrow \langle q_1 \rangle \omega_1[p_1] \Rightarrow \langle q_2 \rangle \omega_2[p_2] \Rightarrow \dots \Rightarrow \langle q_k \rangle \omega_k[p_k]$$

in O, where $q_0 = s$, $q_k = q$, $\omega_0 = {}_\Gamma S$, and $\omega_k = \omega$. As it follows from the construction of O, for every $i = 1, \dots, k$, $p_i = p'_i p''_i$, where $p'_i = \langle q_{i-1}\rangle \to \langle q_i\rangle$, $\omega_{i-1} \Rightarrow \omega_i[p''_i]$ in ${}_\Gamma G$, and for $a \in \Sigma \cup \{\varepsilon\}$, $(q_{i-1}a \to q_i, p''_i) \in \Psi$. That is, $(q_{i-1}w_{i-1}, \omega_{i-1}) \Rightarrow (q_i w_i, \omega_i)$ for all $i = 1, \dots, k$, and hence, $(sw_0, {}_\Gamma S) \Rightarrow^* (q_k w_k, \omega_k)$ with $w = w_0$. Having $\omega_k \in {}_\Gamma T^*$ and using $p = p_1 p_2$, where $p_1 = \langle q\rangle \to \varepsilon \in {}_O P$, $\omega_{k-1} \Rightarrow \omega_k[p_2]$, and $p \in {}_O C$, implies $q \in F$, $w_k = \varepsilon$, and $w \in L({}_\Gamma M)$, and therefore, $\omega_k \in L(\Gamma)_2$. $L(O) \subseteq L(\Gamma)_2$. □

Theorem 3.3 holds by Claims 3.4 and 3.5. □

On the other hand, the context-free grammar in the RT can be exploited as an additional storage space of the finite automaton to remember some non-negative integers. If the automaton uses the context-free grammar 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 3.6. *For every k-PBCA I, there is an RT* $\Gamma = (M, G, \Psi)$ *such that* $L(I) = L(\Gamma)_1$.

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

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

- $q_1, \dots, q_n$ into ${}_M Q$;
- $r = S \to xS$, where $x \in (N - \{S\})^*$ and $occur(A_i, x) = \max(0, t_i)$, for $i = 1, \dots, k$, 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 $(r_{i+1}, \alpha_i \to \varepsilon)$, 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 finite automaton of the created system uses the context-free grammar 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 3.7 states that the context-free grammar is helpful for the finite automaton in RT at most with the preservation of the non-negative integers without possibility to check their values.

Lemma 3.7. *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 $N = \{A_1, \dots, A_n\}$, where $S = A_1$. The partially blind $|N|$-counter automaton $O = (Q \cup \{q'_0\}, \Sigma, {}_O\delta, q'_0, F)$ 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 $pa \to q(v_1, \dots, v_{|N|})$, where $v_i = occur(A_i, \beta) - occur(A_i, \alpha)$, for all $i = 1, \dots, |N|$, into ${}_O\delta$. Furthermore, add $q_0' \to q_0(1, 0, \dots, 0)$ into ${}_O\delta$.

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 a sentential form of grammar G is changed, O makes the same step and accordingly changes the number of occurrences of nonterminals in its counters. □

From Lemma 3.6 and Lemma 3.7, we can establish the following theorem.

Theorem 3.8. $\mathscr{L}(RT)_1 = \bigcup_{k=1}^{\infty} \mathscr{L}(k{-}PBCA)$.

Proof. It directly follows from Lemma 3.7 and Lemma 3.6. □

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 **RE** (see [1] or [7]), and the class of languages defined by partially blind k-counter automata, with respect to the number of counters, is superset of **CF** and properly included in **CS** (see [8] and [9]).

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

Definition 3.9 (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-$L_{lm}(\Gamma) = \{(w_1, w_2) |\ (q_0w_1, S) \Rightarrow_{lm}^* (f, w_2),\ w_1 \in \Sigma^*,\ w_2 \in T^*, \text{ and } f \in F\}$; we call Γ as *leftmost restricted RT*; and we define the languages given from 2-$L_{lm}(\Gamma)$ as $L_{lm}(\Gamma)_1 = \{w_1 |\ (w_1, w_2) \in 2\text{-}L_{lm}(\Gamma)\}$ and $L_{lm}(\Gamma)_2 = \{w_2 |\ (w_1, w_2) \in 2\text{-}L_{lm}(\Gamma)\}$. 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.

Theorem 3.10. $\mathscr{L}(RT_{lm})_2 = \mathbf{CF}$.

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 while 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 pushdown automata. It is sufficient to prove that every language $L_{lm}(\Gamma)_2$ of RT can be accepted by a pushdown automaton. Consider an RT $\Gamma = (M =$

$(Q, {}_{\Gamma}\Sigma, {}_{\Gamma}\delta, q_0, F), G = (N, T, P, S), \Psi)$ 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 ${}_{O}\delta$;

Now, we have to show that $L(O) = L_{lm}(\Gamma)_2$.

Claim 3.11. Let $(q_0w, S) \Rightarrow^* (pw', u\alpha v) \Rightarrow^* (f, \widehat{w})$ in RT Γ, where $u \in T^*$, $\alpha \in N$, and $v \in (N \cup T)^*$. Then, $Sq_0\widehat{w} \Rightarrow^* (v)^R \alpha p\widehat{w}'$ in PDA O, where $\widehat{w} = u\widehat{w}'$.

Proof. By the induction on the number of computation steps.

- *Basis.* Let $(q_0w, S) \Rightarrow^0 (q_0w, S) \Rightarrow^* (f, \widehat{w})$ in Γ. Trivially, $Sq_0\widehat{w} \Rightarrow^0 Sq_0\widehat{w}$ and Claim 3.11 holds.
- *Induction hypothesis.* Suppose that Claim 3.11 holds for j or fewer computation steps.
- *Induction step.* Let $(q_0w, S) \Rightarrow^j (paw', u\alpha v) \Rightarrow (qw', uxv) \Rightarrow^* (f, \widehat{w})$ in Γ, where $a \in {}_{\Gamma}\Sigma \cup \{\varepsilon\}$, $uxv = uu'\beta v'$ and β is the new leftmost nonterminal. Then, by the induction hypothesis, $Sq_0\widehat{w} \Rightarrow^* (v)^R \alpha pa\widehat{w}'$ in O. Since $(paw', u\alpha v) \Rightarrow (qw', uxv)$ in Γ, $paw' \Rightarrow qw'[r_1]$ in M, $u\alpha v \Rightarrow uxv[r_2]$ in G, and $(r_1, r_2) \in \Psi$. From the construction of ${}_{O}\delta$, O has rules $\alpha p \to (x)^R q$ and $bqb \to q$ for all $b \in T$. Hence, $(v)^R \alpha p\widehat{w}' \Rightarrow (xv)^R q\widehat{w}'$. Because $uxv = uu'\beta v'$, β is the leftmost nonterminal, and $(qw', uxv) \Rightarrow^* (f, \widehat{w})$, $(xv)^R q\widehat{w}' = (u'\beta v')^R qu'\widehat{w}''$, and obviously, $(u'\beta v')^R qu'\widehat{w}'' \Rightarrow^* (\beta v')^R q\widehat{w}''$ in O. Claim 3.11 holds.

□

The last step of every successful computation of Γ has to be of the form $(qa, u\alpha v) \Rightarrow (f, uxv)$, with $a \in T \cup \{\varepsilon\}$, $f \in F$, $uxv \in T^*$. By Claim 3.11, suppose that O is in configuration $(\alpha v)^R qw'$, where $uw' = uxv$. From the construction of ${}_{O}\delta$, $(\alpha v)^R qw' \Rightarrow (xv)^R fw' \Rightarrow^* f$ in O. Hence, $L_{lm}(\Gamma)_2 \subseteq L(O)$.

It remains to prove the opposite inclusion, that is, $L(O) \subseteq L_{lm}(\Gamma)_2$.

Claim 3.12. Let $Sq_0w \Rightarrow^* f$ in PDA O, where $f \in F$. Then, $(q_0\widehat{w}, S) \Rightarrow^* (f, w)$ in RT Γ.

Proof. Consider any successful acceptance:

$$Sq_0w \Rightarrow^* f \qquad (I)$$

in PDA O. Without any loss of generality, we can express (I) as $\alpha_0 q_0 w_0 \Rightarrow v_1\alpha_1u_1q_1w_0 \Rightarrow^* v_1\alpha_1q_1w_1 \Rightarrow v_2\alpha_2u_2q_2w_1 \Rightarrow^* v_2\alpha_2q_2w_2 \Rightarrow \dots \Rightarrow v_k\alpha_ku_kq_kw_{k-1} \Rightarrow^* v_k\alpha_kq_kw_k \Rightarrow v_ku_{k+1}fw_k \Rightarrow^* f$, where $\alpha_0 = S$ and for all $i = 1, \dots, k$ with $k \geq 0$, $\alpha_i \in N$, $u_i, u_{k+1}, v_k \in T^*$, $v_i \in (N \cup T)^*$, $w_{i-1} = (u_i)^R w_i$ and $w_k = (v_ku_{k+1})^R$. Openly, $(u_i)^R \alpha_i (v_i)^R \Rightarrow (u_{i+1}u_i)^R \alpha_{i+1} (v_{i+1})^R [r_i]$ in G, $q_{i-1}\widehat{w}_{i-1} \Rightarrow q_i\widehat{w}_i[r_i']$, and furthermore, $(r_i', r_i) \in \Psi$ for all $i = 0, \dots, k$. Hence, (I) can be simulated by $(q_0\widehat{w}_0, \alpha_0) \Rightarrow (q_1\widehat{w}_1, (u_1)^R\alpha_1(v_1)^R) \Rightarrow (q_2\widehat{w}_2, (u_2u_1)^R\alpha_2(v_2)^R) \Rightarrow \dots \Rightarrow (u_ku_{k-1}\dots u_1)^R\alpha_k(v_k)^R \Rightarrow (f, (u_{k+1}u_ku_{k-1}\dots u_1)^R(v_k)^R) = (f, w)$ in Γ. So, Claim 3.12 holds. □

As $L(O) \subseteq L_{lm}(\Gamma)_2$ and $L_{lm}(\Gamma)_2 \subseteq L(O)$, Theorem 3.10 holds. □

Lemma 3.13. *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 context-free grammar such that $L(I) = L$. For I, we can construct context-free grammar $H = ({}_HN, T, {}_HP, S)$, 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 finite automaton $M = (\{q_0\}, T, \delta, q_0, \{q_0\})$ with $\delta = \{q_0 \to q_0\} \cup \{q_0 a \to q_0 |\ a \in T\}$, and $\Psi = \{(q_0 \to q_0, A \to x) |\ A \to x \in {}_HP, A \in {}_IN\} \cup \{(q_0 a \to q_0, \langle a \rangle \to a) |\ a \in T\}$.

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 3.14. *Let Γ be 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 proof of Theorem 3.3, 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.

In the following, we demonstrate that $L(O) = L_{lm}(\Gamma)_1$.

Claim 3.15. Let $(q_0 w, S) \Rightarrow^* (pw', u\alpha v)$ in RT Γ, where $u \in T^*$, $\alpha \in N$, and $v \in (N \cup T)^*$. Then, $Sq_0 w \Rightarrow^* (\theta(v))^R \alpha p w'$ in PDA O.

Proof. By the induction on the number of computation steps.

- *Basis.* Let $(q_0 w, S) \Rightarrow^0 (q_0 w, S)$ in Γ. Then, surely, $Sq_0 w \Rightarrow^0 (\theta(S))^R q_0 w$. Claim 3.15 holds.
- *Induction hypothesis.* Suppose that Claim 3.15 holds for j or fewer computation steps.
- *Induction step.* Let $(q_0 w, S) \Rightarrow^j (paw', u\alpha v) \Rightarrow (qw', uxv)$ in Γ, where $a \in {}_\Gamma\Sigma \cup \{\varepsilon\}$, $uxv = uu'\beta v'$ and β is the leftmost nonterminal. By the induction hypothesis, $Sq_0 w \Rightarrow^* (\theta(v))^R \alpha paw'$ in O. Because $(paw', u\alpha v) \Rightarrow (qw', uxv)$ in Γ, $paw' \Rightarrow qw'[r_1]$ in M, $u\alpha v \Rightarrow uxv[r_2]$ in G, and $(r_1, r_2) \in \Psi$. From the construction of ${}_O\delta$, O has a rule $\alpha pa \to (\theta(x))^R q$, and $(\theta(v))^R \alpha paw' \Rightarrow (\theta(v'))^R \beta qw'$ in O. Claim 3.15 holds.

□

The last step of any successful computation in Γ is of the form $(qa, u\alpha v) \Rightarrow (f, uxv)$, where $f \in F$, $a \in {}_{\Gamma}\Sigma \cup \{\varepsilon\}$, $\alpha \in N$, and $uxv \in T^*$. Hence, $\alpha qa \to f \in {}_O\delta$ and $\alpha qa \Rightarrow f$ in O. So, $L_{lm}(\Gamma)_1 \subseteq L(O)$.

Claim 3.16. Let $Sq_0w \Rightarrow^* (\theta(v))^R\alpha pw'$ in PDA O. Then, $(q_0w, S) \Rightarrow^* (pw', u\alpha v)$ in RT Γ, where $u \in T^*$, $\alpha \in N$, and $v \in (N \cup T)^*$.

Proof. By the induction on the number of moves.

- *Basis.* Let $Sq_0w \Rightarrow^0 Sq_0w$. Then, $(q_0w, S) \Rightarrow^0 (q_0w, S)$ in Γ and Claim 3.16 holds.
- *Induction hypothesis.* Suppose that Claim 3.16 holds for j or fewer moves.
- *Induction step.* Let $Sq_0w \Rightarrow^j (\theta(v))^R\alpha paw' \Rightarrow (\theta(xv))^R qw'$ in O, where $a \in {}_{\Gamma}\Sigma \cup \{\varepsilon\}$. Then, by the induction hypothesis, $(q_0w, S) \Rightarrow^* (paw', u\alpha v)$ in Γ, where $u \in T^*$, $\alpha \in N$, and $v \in (N \cup T)^*$. Because there is a rule $\alpha pa \to (\theta(x))^R q$ in ${}_O\delta$, from the construction of ${}_O\delta$, there are rules $r_1 = pa \to q \in {}_{\Gamma}\delta$ and $r_2 = \alpha \to x \in P$, and $(r_1, r_2) \in \Psi$. Therefore, $(paw', u\alpha v) \Rightarrow (qw', uxv)$ in Γ. So, Claim 3.16 holds. Furthermore, if $\theta(xv)w' = \varepsilon$ and $q \in F$, then $(paw', u\alpha v) \Rightarrow (q, uxv)$ and $L(O) \subseteq L_{lm}(\Gamma)_1$.

□

Since $L(O) \subseteq L_{lm}(\Gamma)_1$ and $L_{lm}(\Gamma)_1 \subseteq L(O)$, $L(O) = L_{lm}(\Gamma)_1$. □

Theorem 3.17. $\mathscr{L}(RT_{lm})_1 = \mathbf{CF}$.

Proof. It directly follows from Lemma 3.13 and Lemma 3.14. □

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 the power of **CF**.

In the following, we extend RT with 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.

Definition 3.18. The *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 in Definition 3.1. 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.

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

Theorem 3.19. $\mathscr{L}(RT_{ac})_2 = \mathbf{RE}$.

Proof. Because $\mathscr{L}(MAT_{ac}) = \mathbf{RE}$ (see [7]), we only need to prove that $\mathscr{L}(MAT_{ac}) \subseteq \mathscr{L}(RT_{ac})_2$.

Consider a MAT 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 finite automaton ${}_\Gamma M = (Q, \Sigma, \delta, s, F)$ 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 finite automaton simulates matrices in I by moves. 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 3.20. $\mathscr{L}(RT_{ac})_1 = \mathbf{RE}$.

Proof. Let $I = ({}_IQ, \Sigma, {}_I\delta, q_0, F)$ be a k-CA for some $k \geq 1$ and construct RT $\Gamma = (M = ({}_MQ, \Sigma, {}_M\delta, q_0, F), G = (N, T, P, S), \Psi)$ as follows: Set $T = \{a\}, \Psi = \emptyset, N = \{S, \Diamond, A_1, \dots, A_k\}, P = \{A \to \varepsilon, A \to \Diamond \mid A \in N - \{\Diamond\}\} \cup \{S \to S\}$, and ${}_MQ = {}_IQ$, ${}_M\delta = \{f \to f \mid f \in F\}$. 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;
- $r_1 = q_0a \to q_1$, $r_2 = q_1 \to q_2$, $\dots$, $r_n = q_{n-1} \to q_n$, $r_{n+1} = q_n \to q$ 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 $q_0a \to q$, $S \to xS$ 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 3.6, the finite automaton of the created system uses context-free grammar 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 rule leads to an unsuccessful computation. The formal proof of the equivalence of languages is left to the reader. Since $\mathscr{L}(k{-}CA) = \mathbf{RE}$ for every $k \geq 2$ (see [13]), Theorem 3.20 holds. □

4. Applications in Natural Language Translation

In this section, we present several examples illustrating potential applications of the formal models discussed in this paper in natural language processing (NLP), particularly in translation.

Currently, mainly because of the availability of large corpora (both unannotated and annotated) for many languages, statistical approaches with the application of machine learning are dominant in machine translation and NLP in general. Many machine translation systems are based on n-gram models and use little or no syntactic information (see [5]). However, the recent trend consists in incorporating global information of this kind (global within the scope of the sentence) into translation systems in an effort to improve the results. This approach is usually called syntax-based or syntax augmented translation (see [16], [24]). An in-depth description of the formal background of one such system can be found in [4].

In this paper, we present a new formalism that can be applied within the context of such systems to describe syntactic structures and their transformations, and we study its properties. RT provides an alternative to formal models currently used in translation systems, such as synchronous grammars (see [6]).

One of the main advantages of the RT formalism lies in its straightforward and intuitive basic principle. Indeed, we simply read the input with an FA, while generating the corresponding output using a CFG. Another important advantage is the power of RT (see section 3). RT has the ability to describe features of natural languages that are difficult or even impossible to capture within a purely context-free framework, such as non-projectivity, demonstrated below.

RT can be easily extended and adapted for use in statistical NLP as well. For example, similarly to probabilistic CFG (see [15]), we can assign probabilities to rules, or to pairs of rules in the control set.

First, to demonstrate the basic principles, we perform the passive transformation of a simple English sentence

The cat caught the mouse.

The passive transformation means transforming a sentence in active voice into passive, and it is a well-known principle that is common to many languages. For the above sentence, the passive form is

The mouse was caught by the cat.

Figure 2 shows derivation trees for the above sentences. Throughout this section, we use the following notation to represent common linguistic constituents:

AUX	auxiliary verb	P	preposition
DET	determiner	PN	pronoun
N	noun	PP	prepositional phrase
NP	noun phrase	V	verb
NP-SBJ	noun phrase in the role of subject	VP	verb phrase

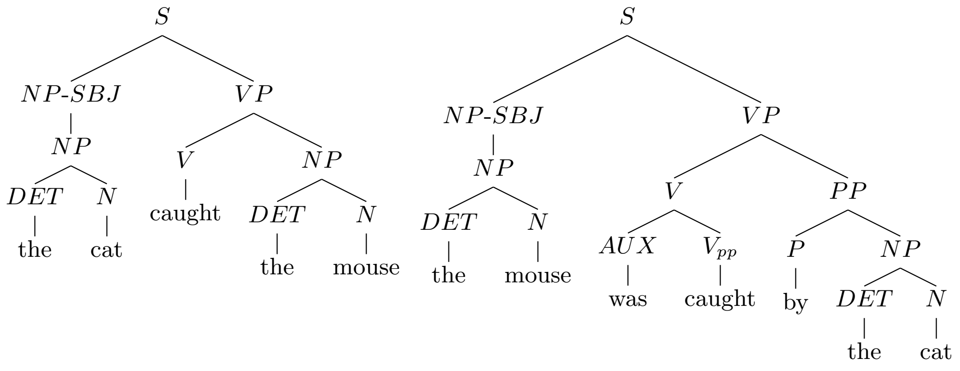


Figure 2. Example of the passive transformation.

Essentially, what we need to do is the following: swapping the subject and the object, adding the preposition *by* in the correct position, and changing the verb into passive, using the auxiliary verb *to be* in the appropriate form. The verb *to be* is irregular and has many different forms (paradigms) depending not only on tense, but also person and number. In most cases, we can see the tense directly from the main verb in the active form, but for the other two categories (person and number), we need to look at the subject (the object in the original sentence).

Example 4.1. Consider an RT $\Gamma = (M, G, \Psi)$, where $M = (Q, \Sigma, \delta, 0, F)$, $G = (N, T, P, S)$. Let $Q = \{0, 1, 2, 3, 4, 5, 6, 7, 8a, 8b, 8c, 8d, 8e, 8f, 9\}$, $\Sigma = \{\mathrm{N}_{1s}, \mathrm{N}_{2s}, \mathrm{N}_{3s}, \mathrm{N}_{1p}, \mathrm{N}_{2p}, \mathrm{N}_{3p}, \mathrm{V}_{pas}, \mathrm{V}_{ps}, \mathrm{V}_{pp}, \mathrm{DET}, \mathrm{P}, \mathrm{AUX}_{pas1s}, \mathrm{AUX}_{pas2s}, \mathrm{AUX}_{pas3s}, \mathrm{AUX}_{pas1p}, \mathrm{AUX}_{pas2p}, \mathrm{AUX}_{pas3p}, \mathrm{AUX}_{ps1s}, \mathrm{AUX}_{ps2s}, \mathrm{AUX}_{ps3s}, \mathrm{AUX}_{ps1p}, \mathrm{AUX}_{ps2p}, \mathrm{AUX}_{ps3p}\}$, $F = \{9\}$, $N = \{\mathrm{S}, \mathrm{NP\text{-}SBJ}, \mathrm{NP}, \mathrm{VP}, \mathrm{PP}, \mathrm{N}_?, \mathrm{V}_?, \mathrm{AUX}_{pas?}, \mathrm{AUX}_{ps?}\}$, $T = \Sigma$.

Let

$$
\delta = \{ \quad \begin{array}{rlrlrl}
r_1 = & 0 \to 1, & r_{6a} = & 3\mathrm{V}_{pas} \to 5, & r_{10a} = & 8a \to 9, \\
r_2 = & 1 \to 2, & r_{6b} = & 3\mathrm{V}_{ps} \to 5, & r_{10b} = & 8b \to 9, \\
r_3 = & 2 \to 3, & r_7 = & 5 \to 6, & & \vdots \\
r_4 = & 3\mathrm{DET} \to 4, & r_8 = & 6\mathrm{DET} \to 7, & & \\
r_{5a} = & 4\mathrm{N}_{1s} \to 3, & r_{9a} = & 7\mathrm{N}_{1s} \to 8a, & r_{10f} = & 8f \to 9, \\
r_{5b} = & 4\mathrm{N}_{2s} \to 3, & r_{9b} = & 7\mathrm{N}_{2s} \to 8b, & & \\
& \vdots & & \vdots & & \\
r_{5f} = & 4\mathrm{N}_{3p} \to 3, & r_{9f} = & 7\mathrm{N}_{3p} \to 8f\} & &
\end{array}
$$

and

$$
\begin{array}{lrlrl}
P=\{ & p_1 = & \text{S} \rightarrow \text{NP-SBJ VP}, & p_{8a} = & \text{AUX}_{pas?} \rightarrow \text{AUX}_{pas1s}, \\
 & p_2 = & \text{NP-SBJ} \rightarrow \text{NP}, & p_{8b} = & \text{AUX}_{pas?} \rightarrow \text{AUX}_{pas2s}, \\
 & p_3 = & \text{NP} \rightarrow \text{DET N}_?, & & \vdots \\
 & p_{4a} = & \text{N}_? \rightarrow \text{N}_{1s}, & & \\
 & p_{4b} = & \text{N}_? \rightarrow \text{N}_{2s}, & p_{8f} = & \text{AUX}_{pas?} \rightarrow \text{AUX}_{pas3p}, \\
 & \vdots & & p_{9a} = & \text{AUX}_{ps?} \rightarrow \text{AUX}_{ps1s}, \\
 & & & p_{9b} = & \text{AUX}_{ps?} \rightarrow \text{AUX}_{ps2s}, \\
 & p_{4f} = & \text{N}_? \rightarrow \text{N}_{3p}, & & \vdots \\
 & p_5 = & \text{VP} \rightarrow \text{V}_? \text{ PP}, & & \\
 & p_6 = & \text{PP} \rightarrow \text{P NP}, & p_{9f} = & \text{AUX}_{ps?} \rightarrow \text{AUX}_{ps3p}, \\
 & p_{7a} = & \text{V}_? \rightarrow \text{AUX}_{pas?} \text{ V}_{pp}, & & \\
 & p_{7b} = & \text{V}_? \rightarrow \text{AUX}_{ps?} \text{ V}_{pp}\}. & &
\end{array}
$$

Finally, let $\Psi = \{(r_1,p_1),(r_2,p_5),(r_3,p_6),(r_4,p_3),(r_{5a},p_{4a}),(r_{5b},p_{4b}),\dots,(r_{5f},p_{4f}),(r_{6a},p_{7a}),(r_{6b},p_{7b}),(r_7,p_2),(r_8,p_3),(r_{9a},p_{4a}),(r_{9b},p_{4b}),\dots,(r_{9f},p_{4f}),(r_{10a},p_{8a}),(r_{10b},p_{8b}),\dots,(r_{10f},p_{8f}),(r_{10a},p_{9a}),(r_{10b},p_{9b}),\dots,(r_{10f},p_{9f})\}$.

One may notice that the input alphabet of the automaton, as well as the terminal alphabet of the grammar, does not contain the actual words themselves, but rather symbols representing word categories and their properties (for example, N_{3s} represents a noun in third person singular). In the examples throughout this section, we consider syntax analysis and translation on the abstract level, transforming syntactic structures in languages. That is, we assume that we already have the input sentence split into words (or possibly some other units as appropriate), and these words are tagged as, for example, a noun, pronoun, or verb.

In the computation examples, the text in square brackets shows the words associated with the symbols for the given example sentence, but note that this is not a part of the formalism itself. This specifier is assigned to all terminals. Nonterminals are only specified by words when the relation can be established from the computation so far performed (for example, we cannot assign a word before we read the corresponding input token).

For the sentence *the cat caught the mouse* (for the purposes of this text, we disregard capitalization and punctuation) from the above example, the computation can proceed as follows:

(0 DET[*the*] N_{3s}[*cat*] V_{pas}[*caught*] DET[*the*] N_{3s}[*mouse*], $\underline{\text{S}}$)
$\Rightarrow$ (1 DET[*the*] N_{3s}[*cat*] V_{pas}[*caught*] DET[*the*] N_{3s}[*mouse*], NP-SBJ $\underline{\text{VP}}$) $[(r_1,p_1)]$
$\Rightarrow$ (2 DET[*the*] N_{3s}[*cat*] V_{pas}[*caught*] DET[*the*] N_{3s}[*mouse*], NP-SBJ $\text{V}_?$ $\underline{\text{PP}}$) $[(r_2,p_5)]$
$\Rightarrow$ (3 $\underline{\text{DET}[\textit{the}]}$ N_{3s}[*cat*] V_{pas}[*caught*] DET[*the*] N_{3s}[*mouse*], NP-SBJ $\text{V}_?$ P[*by*] $\underline{\text{NP}}$) $[(r_3,p_6)]$
$\Rightarrow$ (4 $\underline{\text{N}_{3s}[\textit{cat}]}$ V_{pas}[*caught*] DET[*the*] N_{3s}[*mouse*], NP-SBJ $\text{V}_?$ P[*by*] DET[*the*] $\underline{\text{N}_?}$) $[(r_4,p_3)]$
$\Rightarrow$ (3 $\underline{\text{V}_{pas}[\textit{caught}]}$ DET[*the*] N_{3s}[*mouse*], NP-SBJ $\underline{\text{V}_?}$ P[*by*] DET[*the*] N_{3s}[*cat*]) $[(r_{5c},p_{4c})]$
$\Rightarrow$ (5 DET[*the*] N_{3s}[*mouse*], $\underline{\text{NP-SBJ}}$ $\text{AUX}_{pas?}$[*be*] V_{pp}[*caught*] P[*by*] DET[*the*] N_{3s}[*cat*]) $[(r_{6a},p_{7a})]$

$\Rightarrow$ (6 $\underline{\mathrm{DET}[\mathit{the}]}$ $\mathrm{N}_{3s}[\mathit{mouse}]$, $\underline{\mathrm{NP}}$ $\mathrm{AUX}_{pas?}[\mathit{be}]$ $\mathrm{V}_{pp}[\mathit{caught}]$ $\mathrm{P}[\mathit{by}]$ $\mathrm{DET}[\mathit{the}]$ $\mathrm{N}_{3s}[\mathit{cat}]$) $[(r_7, p_2)]$

$\Rightarrow$ (7 $\underline{\mathrm{N}_{3s}[\mathit{mouse}]}$, $\mathrm{DET}[\mathit{the}]$ $\underline{\mathrm{N}_?}$ $\mathrm{AUX}_{pas?}[\mathit{be}]$ $\mathrm{V}_{pp}[\mathit{caught}]$ $\mathrm{P}[\mathit{by}]$ $\mathrm{DET}[\mathit{the}]$ $\mathrm{N}_{3s}[\mathit{cat}]$) $[(r_8, p_3)]$

$\Rightarrow$ (8*c*, $\mathrm{DET}[\mathit{the}]$ $\mathrm{N}_{3s}[\mathit{mouse}]$ $\underline{\mathrm{AUX}_{pas?}[\mathit{be}]}$ $\mathrm{V}_{pp}[\mathit{caught}]$ $\mathrm{P}[\mathit{by}]$ $\mathrm{DET}[\mathit{the}]$ $\mathrm{N}_{3s}[\mathit{cat}]$) $[(r_{9c}, p_{4c})]$

$\Rightarrow$ (9, $\mathrm{DET}[\mathit{the}]$ $\mathrm{N}_{3s}[\mathit{mouse}]$ $\mathrm{AUX}_{pas3s}[\mathit{was}]$ $\mathrm{V}_{pp}[\mathit{caught}]$ $\mathrm{P}[\mathit{by}]$ $\mathrm{DET}[\mathit{the}]$ $\mathrm{N}_{3s}[\mathit{cat}]$) $[(r_{10c}, p_{8c})]$

For clarity, in each computation step, the input symbol to be read (if any) and the nonterminal to be rewritten are underlined.

First (in states 0, 1, and 2), we generate the expected basic structure of the output sentence. Note that this is done before reading any input. In states 3 and 4, we read the subject of the original sentence, states 5 and 6 read the verb, and the rest of the states is used to process the object. When we read the verb, we generate its passive form, consisting of *to be* and the verb in past participle. However, at this point, we know the tense (in this case, past simple), but do not know the person or number yet. The missing information is represented by the question mark (?) symbol in the nonterminal $\mathrm{AUX}_{pas?}$. Later, when we read the object of the original sentence, we rewrite $\mathrm{AUX}_{pas?}$ to a terminal. In this case, the object is in third person singular, which gives us the terminal AUX_{pas3s} (meaning that the correct form to use here is *was*).

Next, we present examples of translation between different languages. We focus on Japanese, Czech, and English.

One problem when translating into Czech is that there is very rich inflection. The form of the words reflects many grammatical categories, such as case, gender, and number (see [11], where the author discusses this issue with regard to computational linguistics). To illustrate, compare the following sentences in Japanese, English, and Czech.

Zasshi o yondeitta onna no hito wa watashi no shiriai deshita.
Zasshi o yondeitta otoko no hito wa watashi no shiriai deshita.

The woman who was reading a magazine was an acquitance of mine.
The man who was reading a magazine was an acquitance of mine.

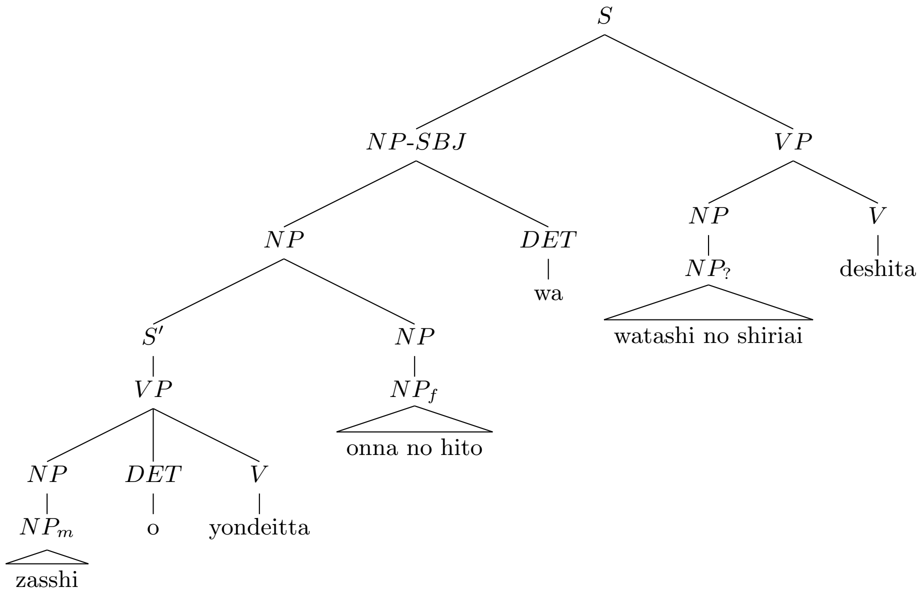
Žena, která četla časopis, byla moje známá.
Muž, který četl časopis, byl můj známý.

As we can see, in Czech, nearly every word is different, depending on the gender of the subject. In contrast, in both Japanese and English, the two sentences only differ in one word – *onna no hito* (*woman*) and *otoko no hito* (*man*).[1]

The above sentences also give us an example of some structural differences between Japanese and Czech. In Czech and English, the structure of the sentence is very similar, but in Japanese, there is no word that correspond directly to *který* (*which*, *who*, ...). Instead, this relation is represented by the form of the verb

[1]Technically, *onna no hito* literally translates to *woman's person* or *female person*, with *onna* itself meaning *woman*, *female*. However, referring to a person only by *onna* may have negative connotations in Japanese. Similarly for *otoko no hito*.

yondeitta (the dictionary form is *yomu*, meaning *to read*). Compare the derivation trees in Figure 3.



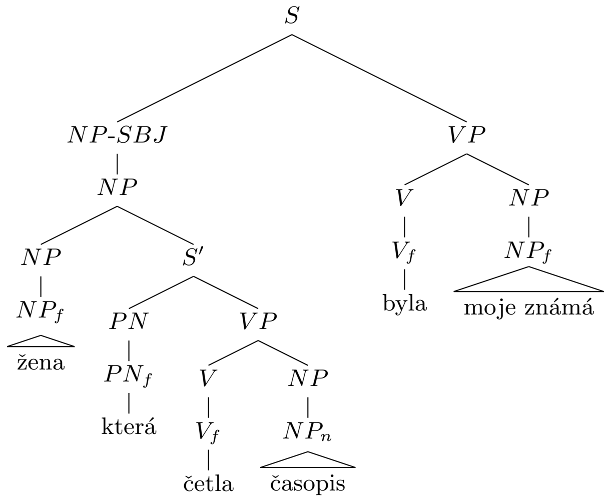


Figure 3. Derivation trees – Japanese (top) and Czech.

Example 4.2. Consider an RT $\Gamma = (M, G, \Psi)$, where $M = (Q, \Sigma, \delta, 0, F)$, $G = (N, T, P, S)$. Let $Q = \{0, m, m1, m2, f, f1, f2, n, n1, n2, 1m, 1f, 1n\}$, $\Sigma = \{\mathrm{NP}_m, \mathrm{NP}_f, \mathrm{NP}_n, \mathrm{NP}_?, \mathrm{V}, \mathrm{DET}, \#\}$, $F = \{1m, 1f, 1n\}$, $N = \{\mathrm{S}, \mathrm{NP\text{-}SBJ}, \mathrm{NP}_?, \mathrm{VP}, \mathrm{PN}_?, \mathrm{V}_?, \mathrm{X}\}$, $T = \{\mathrm{NP}_m, \mathrm{NP}_f, \mathrm{NP}_n, \mathrm{V}_m, \mathrm{V}_f, \mathrm{V}_n, \mathrm{PN}_m, \mathrm{PN}_f, \mathrm{PN}_n\}$.

Let

$$\delta = \{ \quad \begin{array}{rlrlrl}
r_1 = & 0\mathrm{V} \to 1, & r_{m1} = & m\mathrm{V} \to m1, & r_{f1} = & f\mathrm{V} \to f1, \\
r_2 = & 1 \to 0, & r_{m2} = & m1 \to m2, & r_{f2} = & f1 \to f2, \\
r_3 = & 0\mathrm{NP}_? \to 0, & r_{m3} = & m2 \to m, & & \vdots \\
r_4 = & 0\mathrm{DET} \to 0, & r_{m4} = & m\mathrm{DET} \to m, & & \\
r_{5m} = & 0\mathrm{NP}_m \to m, & r_{m5} = & m\mathrm{NP}_? \to m, & r_{f7} = & 1f \to 1f, \\
r_{5f} = & 0\mathrm{NP}_f \to f, & r_{m5m} = & m\mathrm{NP}_m \to m, & r_{n1} = & n\mathrm{V} \to n1, \\
r_{5n} = & 0\mathrm{NP}_n \to n, & r_{m5f} = & m\mathrm{NP}_f \to m, & r_{n2} = & n1 \to n2, \\
 & & r_{m5n} = & m\mathrm{NP}_n \to m, & & \vdots \\
 & & r_{m6} = & m\# \to 1m, & & \\
 & & r_{m7} = & 1m \to 1m, & r_{n7} = & 1n \to 1n\}
\end{array}$$

and

$$P = \{ \quad \begin{array}{rlrl}
p_1 = & \mathrm{S} \to \mathrm{NP\text{-}SBJ\ VP\ X}, & p_{7m} = & \mathrm{V}_? \to \mathrm{V}_m, \\
p_2 = & \mathrm{NP\text{-}SBJ} \to \mathrm{NP}_?, & p_{7f} = & \mathrm{V}_? \to \mathrm{V}_f, \\
p_{3m} = & \mathrm{NP}_? \to \mathrm{NP}_m, & p_{7n} = & \mathrm{V}_? \to \mathrm{V}_n, \\
p_{3f} = & \mathrm{NP}_? \to \mathrm{NP}_f, & p_8 = & \mathrm{S}' \to \mathrm{PN}_?\ \mathrm{VP}, \\
p_{3n} = & \mathrm{NP}_? \to \mathrm{NP}_n, & p_{9m} = & \mathrm{PN}_? \to \mathrm{PN}_m, \\
p_4 = & \mathrm{NP}_? \to \mathrm{NP}_?, & p_{9f} = & \mathrm{PN}_? \to \mathrm{PN}_f, \\
p_5 = & \mathrm{NP}_? \to \mathrm{NP}_?\ \mathrm{S}', & p_{9n} = & \mathrm{PN}_? \to \mathrm{PN}_n, \\
p_6 = & \mathrm{VP} \to \mathrm{V}_?\ \mathrm{NP}_?, & p_{10} = & \mathrm{X} \to \varepsilon\}.
\end{array}$$

Let $\Psi = \{(r_1, p_1), (r_2, p_6), (r_3, p_4), (r_4, p_2), (r_{5m}, p_4), (r_{5f}, p_4), (r_{5n}, p_4), (r_{m1}, p_5), (r_{m2}, p_8), (r_{m3}, p_6), (r_{m4}, p_4), (r_{m5}, p_4), (r_{m5m}, p_{3m}), (r_{m5f}, p_{3f}), (r_{m5n}, p_{3n}), (r_{m6}, p_{10}), (r_{m7}, p_{3m}), (r_{m7}, p_{7m}), (r_{m7}, p_{9m}), (r_{f1}, p_5), (r_{f2}, p_8), (r_{f3}, p_6), (r_{f4}, p_4), (r_{f5}, p_4), (r_{f5m}, p_{3m}), (r_{f5f}, p_{3f}), (r_{f5n}, p_{3n}), (r_{f6}, p_{10}), (r_{f7}, p_{3f}), (r_{f7}, p_{7f}), (r_{f7}, p_{9f}), (r_{n1}, p_5), (r_{n2}, p_8), (r_{n3}, p_6), (r_{n4}, p_4), (r_{n5}, p_4), (r_{n5m}, p_{3m}), (r_{n5f}, p_{3f}), (r_{n5n}, p_{3n}), (r_{n6}, p_{10}), (r_{n7}, p_{3n}), (r_{n7}, p_{7n}), (r_{n7}, p_{9n})\}$.

We have added two dummy symbols: the input symbol #, which acts as the endmarker, and the nonterminal X, which we generate at the beginning of the computation and then erase when all the input has been read (including #).

In this example, we read the input sentence in reverse order (right to left). Clearly, this makes no difference from a purely theoretical point of view, but it may be more suitable in practice due to the way how Japanese sentences are organized. The computation transforming the sentence *zasshi o yondeitta onna no hito wa watashi no shiriai deshita* into *žena, která četla časopis, byla moje známá* can proceed as follows:

(0 $\underline{\mathrm{V}[\textit{deshita}]}$ $\mathrm{NP}_?[\textit{watashi no shiriai}]$ $\mathrm{DET}[\textit{wa}]$ $\mathrm{NP}_f[\textit{onna no hito}]$
$\mathrm{V}[\textit{yondeitta}]$ $\mathrm{DET}[\textit{o}]$ $\mathrm{NP}_m[\textit{zasshi}]$ #, $\underline{\mathrm{S}}$)

$\Rightarrow$ (1 $\mathrm{NP}_?[\textit{watashi no shiriai}]$ $\mathrm{DET}[\textit{wa}]$ $\mathrm{NP}_f[\textit{onna no hito}]$ $\mathrm{V}[\textit{yondeitta}]$
$\mathrm{DET}[\textit{o}]$ $\mathrm{NP}_m[\textit{zasshi}]$ #, NP-SBJ $\underline{\mathrm{VP}}$ X) $[(r_1, p_1)]$

$\Rightarrow$ (0 $\underline{\mathrm{NP}_?[\textit{watashi no shiriai}]}$ $\mathrm{DET}[\textit{wa}]$ $\mathrm{NP}_f[\textit{onna no hito}]$ $\mathrm{V}[\textit{yondeitta}]$
$\mathrm{DET}[\textit{o}]$ $\mathrm{NP}_m[\textit{zasshi}]$ #, NP-SBJ $\mathrm{V}_?[\textit{byl}]$ $\underline{\mathrm{NP}_?}$ X) $[(r_2, p_6)]$

$\Rightarrow$ (0 $\underline{\mathrm{DET}[\textit{wa}]}$ $\mathrm{NP}_f[\textit{onna no hito}]$ $\mathrm{V}[\textit{yondeitta}]$ $\mathrm{DET}[\textit{o}]$ $\mathrm{NP}_m[\textit{zasshi}]$ #,
$\underline{\text{NP-SBJ}}$ $\mathrm{V}_?[\textit{byl}]$ $\mathrm{NP}_?[\textit{můj známý}]$ X) $[(r_3, p_4)]$

$\Rightarrow$ (0 $\underline{\mathrm{NP}_f[\textit{onna no hito}]}$ $\mathrm{V}[\textit{yondeitta}]$ $\mathrm{DET}[\textit{o}]$ $\mathrm{NP}_m[\textit{zasshi}]$ #, $\underline{\mathrm{NP}_?}$
$\mathrm{V}_?[\textit{byl}]$ $\mathrm{NP}_?[\textit{můj známý}]$ X) $[(r_4, p_2)]$

$\Rightarrow$ (f $\underline{\mathrm{V}[\textit{yondeitta}]}$ $\mathrm{DET}[\textit{o}]$ $\mathrm{NP}_m[\textit{zasshi}]$ #, $\underline{\mathrm{NP}_?[\textit{žena}]}$ $\mathrm{V}_?[\textit{byl}]$

NP$_?$[*můj známý*] X) [(r_{5f}, p_4)]
⇒ (f1 DET[*o*] NP$_m$[*zasshi*] #, NP$_?$[*žena*] $\underline{\text{S}'}$ V$_?$[*byl*] NP$_?$[*můj známý*] X) [(r_{f1}, p_5)]
⇒ (f2 DET[*o*] NP$_m$[*zasshi*] #, NP$_?$[*žena*] PN$_?$[*který*] $\underline{\text{VP}}$ V$_?$[*byl*] NP$_?$[*můj známý*] X) [(r_{f2}, p_8)]
⇒ (f $\underline{\text{DET}[\textit{o}]}$ NP$_m$[*zasshi*] #, NP$_?$[*žena*] PN$_?$[*který*] V$_?$[*četl*] $\underline{\text{NP}_?}$ V$_?$[*byl*] NP$_?$[*můj známý*] X) [(r_{f3}, p_6)]
⇒ (f $\underline{\text{NP}_m[\textit{zasshi}]}$ #, NP$_?$[*žena*] PN$_?$[*který*] V$_?$[*četl*] $\underline{\text{NP}_?}$ V$_?$[*byl*] NP$_?$[*můj známý*] X) [(r_{f4}, p_4)]
⇒ (f $\underline{\#}$, NP$_?$[*žena*] PN$_?$[*který*] V$_?$[*četl*] NP$_m$[*časopis*] V$_?$[*byl*] NP$_?$[*můj známý*] $\underline{\text{X}}$) [(r_{f5m}, p_{3m})]
⇒ (1f, $\underline{\text{NP}_?[\textit{žena}]}$ PN$_?$[*který*] V$_?$[*četl*] NP$_m$[*časopis*] V$_?$[*byl*] NP$_?$[*můj známý*]) [(r_{f6}, p_{10})]
⇒ (1f, NP$_f$[*žena*] $\underline{\text{PN}_?[\textit{který}]}$ V$_?$[*četl*] NP$_m$[*časopis*] V$_?$[*byl*] NP$_?$[*můj známý*]) [(r_{f7}, p_{3f})]
⇒ (1f, NP$_f$[*žena*] PN$_f$[*která*] $\underline{\text{V}_?[\textit{četl}]}$ NP$_m$[*časopis*] V$_?$[*byl*] NP$_?$[*můj známý*]) [(r_{f7}, p_{9f})]
⇒ (1f, NP$_f$[*žena*] PN$_f$[*která*] V$_f$[*četla*] NP$_m$[*časopis*] $\underline{\text{V}_?[\textit{byl}]}$ NP$_?$[*můj známý*]) [(r_{f7}, p_{7f})]
⇒ (1f, NP$_f$[*žena*] PN$_f$[*která*] V$_f$[*četla*] NP$_m$[*časopis*] V$_f$[*byla*] $\underline{\text{NP}_?[\textit{můj známý}]}$) [($r_{f7}, p_{7f}$)]
⇒ (1f, NP$_f$[*žena*] PN$_f$[*která*] V$_f$[*četla*] NP$_m$[*časopis*] V$_f$[*byla*] NP$_f$[*moje známá*]) [(r_{f7}, p_{3f})]

When we first read the word that determines the gender, we move to the state that represents this gender (state m, n, or f). Note that these states are functionally identical in the sense that we can read the same input symbols, while performing the same computation steps in the grammar generating the output. After we have reached the end of input, we rewrite the nonterminal symbols representing words with as of yet unknown gender to the corresponding terminal symbols, depending on the state.

Czech is considered a free-word-order language, that is, it allows for a wide range of permutations of words in a sentence without changing its syntactic structure (the meaning of the sentence may be affected). This is perhaps the main source of the relatively high amount of non-projectivity in Czech sentences.

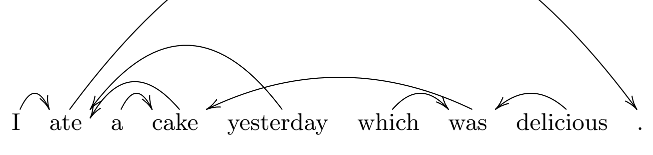


Figure 4. Non-projective dependency tree (English).

Non-projectivity means that there are cross-dependencies. For example, consider the English sentence

I ate a cake yesterday which was delicious.

As shown in the dependency tree in Figure 4, there is a crossing of dependencies represented by arrows (*yesterday*, *ate*) and (*was*, *cake*). Arrows are drawn from child (modifier) to parent (head).

Arguably, the English example is somewhat artificial – even though the sentence is well-formed, in most cases it might be more natural to say simply

I ate a delicious cake yesterday.

In contrast, in Czech, a sentence such as

Nevím, jaký je mezi nimi rozdíl.

(I don't know what the difference between them is.)

is not at all unusual. The dependency tree for this sentence (see Figure 5) is also non-projective.

For further information about projectivity, and the issue of non-projectivity in the Czech language in particular, see [12].

The following example illustrates how our formalism can account for the non-projectivity.

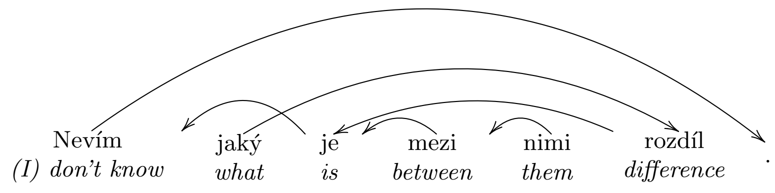


Figure 5. Non-projective dependency tree (Czech).

Example 4.3. Consider an RT $\Gamma = (M, G, \Psi)$, where $M = (Q, \Sigma, \delta, 0, F)$, $G = (N, T, P, S)$. Let $Q = \{0, 1, 2, 3, 4, 5, 6m, 6f, 6n\}$, $\Sigma = \{\mathrm{N}_m, \mathrm{N}_f, \mathrm{N}_n, \mathrm{V}, \mathrm{PN}, \mathrm{PN}_i, \mathrm{DET}, \mathrm{P}\}$, $F = \{0\}$, $N = \{\mathrm{S}, \text{NP-SBJ}, \mathrm{NP}, \mathrm{VP}, \mathrm{PP}, \mathrm{V}_?\}$, $T = \{\mathrm{N}_m, \mathrm{N}_f, \mathrm{N}_n, \mathrm{V}, \mathrm{PN}_m, \mathrm{PN}_f, \mathrm{PN}_n, \mathrm{P}\}$.

Let

$$\begin{array}{lrlrlrl}
\delta = \{ & r_{1m} = & 0\mathrm{N}_m \to 6m, & r_5 = & 0\mathrm{DET} \to 0, & r_{11} = & 5 \to 0, \\
& r_{1f} = & 0\mathrm{N}_f \to 6f, & r_6 = & 0\mathrm{P} \to 0, & r_{12m} = & 6m \to 0, \\
& r_{1n} = & 0\mathrm{N}_n \to 6n, & r_7 = & 1 \to 2, & r_{12f} = & 6f \to 0, \\
& r_2 = & 0\mathrm{V} \to 0, & r_8 = & 2 \to 0, & r_{12n} = & 6n \to 0, \\
& r_3 = & 0\mathrm{PN} \to 1, & r_9 = & 3 \to 4, & r_{13} = & 0\mathrm{PN} \to 0, \\
& r_4 = & 0\mathrm{PN}_i \to 3, & r_{10} = & 4 \to 5\} & &
\end{array}$$

and

$$\begin{array}{lrlrl}
P = \{ & p_1 = & \mathrm{S} \to \text{NP-SBJ VP}, & p_{6m} = & \mathrm{NP} \to \mathrm{PN}_m, \\
& p_2 = & \text{NP-SBJ} \to \mathrm{NP}, & p_{6f} = & \mathrm{NP} \to \mathrm{PN}_f, \\
& p_3 = & \text{NP-SBJ} \to \varepsilon, & p_{6n} = & \mathrm{NP} \to \mathrm{PN}_n, \\
& p_{4m} = & \mathrm{NP} \to \mathrm{N}_m, & p_7 = & \mathrm{VP} \to \mathrm{V}_? \ \mathrm{PP}\ \mathrm{NP}, \\
& p_{4f} = & \mathrm{NP} \to \mathrm{N}_f, & p_8 = & \mathrm{V}_? \to \mathrm{V}, \\
& p_{4n} = & \mathrm{NP} \to \mathrm{N}_n, & p_9 = & \mathrm{PP} \to \mathrm{P}\ \mathrm{NP}, \\
& p_5 = & \mathrm{NP} \to \mathrm{S}, & p_{10} = & \mathrm{PP} \to \varepsilon\}.
\end{array}$$

Let $\Psi = \{(r_{1m}, p_{4m}), (r_{1f}, p_{4f}), (r_{1n}, p_{4n}), (r_2, p_8), (r_3, p_1), (r_4, p_{10}), (r_5, p_7), (r_6, p_9), (r_7, p_3), (r_8, p_7), (r_9, p_5), (r_{10}, p_1), (r_{11}, p_2), (r_{12m}, p_{6m}), (r_{12f}, p_{6f}), (r_{12n}, p_{6n}), (r_{13}, p_{6m})\}$.

The computation transforming the English sentence *I don't know what the difference between them is* into the (non-projective) Czech sentence *nevím, jaký je mezi nimi rozdíl* proceeds as follows:

(0 $\underline{\text{PN}[\textit{I}]}$ V[*don't know*] PN$_i$[*what*] DET[*the*] N$_m$[*difference*] P[*between*] PN[*them*] V[*is*], $\underline{\text{S}}$)

$\Rightarrow$ (1 V[*don't know*] PN$_i$[*what*] DET[*the*] N$_m$[*difference*] P[*between*] PN[*them*] V[*is*], $\underline{\text{NP-SBJ}}$ VP) $[(r_3, p_1)]$

$\Rightarrow$ (2 V[*don't know*] PN$_i$[*what*] DET[*the*] N$_m$[*difference*] P[*between*] PN[*them*] V[*is*], $\underline{\text{VP}}$) $[(r_7, p_3)]$

$\Rightarrow$ (0 $\underline{\text{V}[\textit{don't know}]}$ PN$_i$[*what*] DET[*the*] N$_m$[*difference*] P[*between*] PN[*them*] V[*is*], $\underline{\text{V}_?}$ PP NP) $[(r_8, p_7)]$

$\Rightarrow$ (0 $\underline{\text{PN}_i[\textit{what}]}$ DET[*the*] N$_m$[*difference*] P[*between*] PN[*them*] V[*is*], V[*nevím*] $\underline{\text{PP}}$ NP) $[(r_2, p_8)]$

$\Rightarrow$ (3 DET[*the*] N$_m$[*difference*] P[*between*] PN[*them*] V[*is*], V[*nevím*] $\underline{\text{NP}}$) $[(r_4, p_{10})]$

$\Rightarrow$ (4 DET[*the*] N$_m$[*difference*] P[*between*] PN[*them*] V[*is*], V[*nevím*] $\underline{\text{S}}$) $[(r_9, p_5)]$

$\Rightarrow$ (5 DET[*the*] N$_m$[*difference*] P[*between*] PN[*them*] V[*is*], V[*nevím*] $\underline{\text{NP-SBJ}}$ VP) $[(r_{10}, p_1)]$

$\Rightarrow$ (0 $\underline{\text{DET}[\textit{the}]}$ N$_m$[*difference*] P[*between*] PN[*them*] V[*is*], V[*nevím*] NP[*jaký*] $\underline{\text{VP}}$) $[(r_{11}, p_2)]$

$\Rightarrow$ (0 $\underline{\text{N}_m[\textit{difference}]}$ P[*between*] PN[*them*] V[*is*], V[*nevím*] NP[*jaký*] V$_?$ PP $\underline{\text{NP}}$) $[(r_5, p_7)]$

$\Rightarrow$ (6*m* P[*between*] PN[*them*] V[*is*], V[*nevím*] $\underline{\text{NP}[\textit{jaký}]}$ V$_?$ PP N$_m$[*rozdíl*]) $[(r_{1m},\ p_{4m})]$

$\Rightarrow$ (0 $\underline{\text{P}[\textit{between}]}$ PN[*them*] V[*is*], V[*nevím*] PN$_m$[*jaký*] V$_?$ $\underline{\text{PP}}$ N$_m$[*rozdíl*]) $[(r_{12m},\ p_{6m})]$

$\Rightarrow$ (0 $\underline{\text{PN}[\textit{them}]}$ V[*is*], V[*nevím*] PN$_m$[*jaký*] V$_?$ P[*mezi*] $\underline{\text{NP}}$ N$_m$[*rozdíl*]) $[(r_6, p_9)]$

$\Rightarrow$ (0 $\underline{\text{V}[\textit{is}]}$, V[*nevím*] PN$_m$[*jaký*] $\underline{\text{V}_?}$ P[*mezi*] PN$_m$[*nimi*] N$_m$[*rozdíl*]) $[(r_{13}, p_{6m})]$

$\Rightarrow$ (0, V[*nevím*] PN$_m$[*jaký*] V[*je*] P[*mezi*] PN$_m$[*nimi*] N$_m$[*rozdíl*]) $[(r_2, p_8)]$

The corresponding derivation tree of G is shown in Figure 6.

In the examples presented in this paper, we have made two important assumptions. First, we already have the input sentence analysed on a low level – we know where every word starts and ends (which may be a non-trivial problem in itself in some languages, such as Japanese) and have some basic grammatical information about it. Furthermore, we know the translation of the individual words. For practical applications in natural language translation, we need a more complex system, with at least two other components: a part-of-speech tagger and a dictionary to translate the actual meanings of the words. Then, the component based on the discussed formal model can be used to transform the syntactic structure and also ensure that the words in the translated sentence are in the correct form.

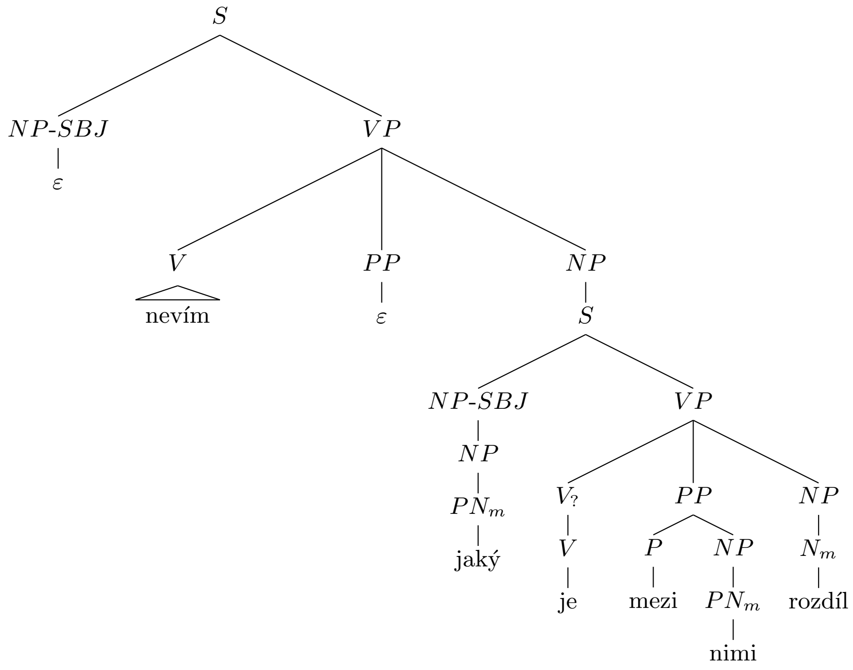


Figure 6. The derivation tree of G.

References

[1] S. Abraham, *Some questions of language theory*, in: Proceedings of the 1965 conference on Computational linguistics (COLING '65, Bonn, Germany), Association for Computational Linguistics, Stroudsburg, 1965, 1–11.

[2] A. V. Aho, *Compilers: Principles, Techniques, & Tools*, Pearson/Addison Wesley, 2007.

[3] M. Bál, O. Carton, C. Prieur and J. Sakarovitch, *Squaring transducers: an effcient procedure for deciding functionality and sequentiality*, Theor. Comput. Sci. **292** (2003), 45–63.

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

[5] P. F. Brown, J. Cocke, S. A. D. Pietra, V. J. D. Pietra, F. Jelinek, J. D. Lafferty, R. L. Mercer and P. S. Roossin, *A statistical approach to machine translation*, Comput. Linguist. **16** (1990), 79–85.

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

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

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

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

[10] E. M. Gurari and O. H. Ibarra, *A note on finite-valued and finitely ambiguous transducers*, Theor. Comput. Syst. **16** (1983), 61–66.

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

[12] E. Hajičová, P. Sgall and D. Zeman, *Issues of projectivity in the Prague dependency treebank*, in: Prague Bulletin of Mathematical Linguistics, 2004.

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

[14] O. Jirák and Z. Křivka, *Design and implementation of back-end for picoblaze C-compiler*, in: Proceedings of the IADIS International Conference Applied Computing (Rome, Italy), International Association for Development of the Information Society, 2009, 135–138.

[15] M. Johnson, *PCFG models of linguistic tree representations*, Comput. Linguist. **24** (1998), 613–632.

[16] M. Khalilov and J. A. R. Fonollosa, *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, 2009, 424–432.

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

[18] 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, Vol. 2, Morgan Kaufmann Publ. Inc., 1991, 1012–1017.

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

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

[21] S. S. Muchnick, *Advanced Compiler Design and Implementation*, Morgan Kaufmann Publ. Inc., San Francisco, 1997.

[22] P. Šaloun, *Parallel LR parsing*, in: Proceedings of the 5th International Scientific Conference Electronic Computers and Informatics 2002, The University of Technology Košice, 2002.

[23] A. Weber, *On the valuedness of finite transducers*, Acta Informatica **27** (1990), 749–780.

[24] A. Zollmann and A. Venugopal, *Syntax augmented machine translation via chart parsing*, in: Proceedings of the Workshop on Statistical Machine Translation (StatMT '06), Association for Computational Linguistics, Stroudsburg, 2006, 138–141.

Martin Čermák, Faculty of Information Technology, Brno University of Technology, Božetěchova 2, 612 66 Brno, Czech Republic,
e-mail: `icermak@fit.vutbr.cz`

Petr Horáček, Faculty of Information Technology, Brno University of Technology, Božetěchova 2, 612 66 Brno, Czech Republic,
e-mail: `ihoracekp@fit.vutbr.cz`

Alexander Meduna, Faculty of Information Technology, Brno University of Technology, Božetěchova 2, 612 66 Brno, Czech Republic,
e-mail: `meduna@fit.vutbr.cz`