Schedae Informaticae Vol. 22 (2013): 9–18
doi: 10.4467/20838476SI.13.001.2086

On Normal Forms and Erasing Rules in Path Controlled Grammars

Jiří Koutný, Alexander Meduna
Formal Language Research Group
Department of Information Systems
Faculty of Information Technology Brno University of Technology
Božetěchova 2, 612 66 Brno, Czech Republic
e-mail: {*ikoutny, meduna*}*@fit.vutbr.cz*

Abstract. This paper discusses path controlled grammars – context-free grammars with a root-to-leaf path in their derivation trees restricted by a control language. First, it investigates the impact of erasing rules on the generative power of path controlled grammars. Then, it establishes two Chomsky-like normal forms for path controlled grammars - the first allows unit rules, the second allows just one erasing rule.

Keywords: context-free grammars, path controlled grammars, restricted derivation trees, paths, normal forms, erasing rules

1. Introduction

The investigation of context-free grammars with restricted derivation trees represents an important trend in today's formal language theory (see [3], [5], [6], [7], [8], [9], [10], [13], [15], and [16]). In essence, these grammars generate their languages just as ordinary context-free grammars do but their derivation trees have to satisfy some simple prescribed conditions. The present paper continues the investigation of these grammars.

In [8], path controlled grammars are introduced as an attempt to increase the generative power of context-free grammar without changing the basic formalism and without loosing some basic properties of the class of context-free languages. Basically, a derivation tree in a context-free grammar is accepted if it contains a *path* described by a control language. More precisely, consider a context-free grammar G and a context-free language R. A string w generated by G belongs to the language defined by G and R if there is a derivation tree t for w in G such that there exists a path p of t described by R. Further properties of path controlled grammars are studied in [6], [9], and [10], however, still many questions remain unanswered.

Since, in general, a restriction placed upon a path is a restriction placed upon a derivation tree, we use a slightly modified but equivalent formulation of the definitinions stated in [8], [9], and [10]. Consequently, aforementioned modifications allow us to study all derivation-tree-based restrictions using the same terminology – e.g. restriction on levels (see [3], [5], [13], [15], and [16]), paths (see [2], [6], [7], [8], [9], and [10]), or cuts (see [7]).

In the theory of regulated rewriting, the impact of erasing rules on the generative power of a rewriting system is usually studied. Typically, beyond the class of context-free grammars, erasing rules significantly affect the generative power of a studied model (see [4]). However, as it is demonstrated in the rest of the paper, erasing rules do not affect the generative power of path controlled grammars. Indeed, erasing rules outside the controlled path can be removed by a well-known algorithm (see Algorithm 5.1.3.2.3 in [12]) and since a path is defined as a sequence of non-terminal symbols followed by just one terminal symbol, erasing rules cannot be used along the controlled path by the definition.

As the main goal of this paper, we establish two normal forms of path controlled grammars. Normal form for any formal model is one of the fundamental model-characterizing properties that is important both from the theoretical as well as practical viewpoints. From the theoretical viewpoint, normal forms are often used to facilitate some kind of proofs - typically, the proofs based on the transformation of investigated formal model to a well-known one or vice versa. From the practical viewpoint, normal forms underlie some general parsing methods used in compiler construction. Since path controlled grammars generate several non-context-free languages used in linguistics (see [8]), parsing methods for path controlled are desirable to be established (see [2]).

Despite all our effort so far, we are not able to transform a path controlled grammar into an equivalent path controlled grammar with controlled grammar in Chomsky normal form. Indeed, we have concluded that we need either chain rules or erasing rules.

In Section 2., we introduce the needed terminology. Section 3. recalls the definition of path controlled grammars and introduces two normal forms for path controlled grammars. As the main result of this paper, Section 4. demonstrates that erasing rules do not affect the generative power of path controlled grammars and establishes the algorithms that transform any path controlled grammar into an equivalent path controlled grammar satisfying desired normal form. In the conclusion, we formulate some open problems in the investigation of grammars with restricted path.

2. Preliminaries

This paper assumes that the reader is familiar with graph theory (see [1]) and the theory of formal languages (see [12]) including the theory of regulated rewriting (see [4]). In this section, we introduce the terminology and the definitions needed in the sequel.

For an alphabet V, V^* denotes the free monoid (generated by V under the operation concatenation), ε is the unit of V^*, and $V^+ = V^* - \{\varepsilon\}$. Every subset $L \subseteq V^*$ is a *language* over V. As usual, when comparing two languages, the empty string (ε) is ignored – L_1 equals L_2 if $L_1 - \{\varepsilon\} = L_2 - \{\varepsilon\}$.

A *context-free grammar* is a quadruple $G = (V, T, P, S)$ where V is a total alphabet, $T \subseteq V$ is a terminal alphabet, P is a finite set of rules of the form $r : A \to x$ where r is unique label, $A \in V - T$, $x \in V^*$, and $S \in V - T$ is the starting symbol. A context-free grammar $G = (V, T, P, S)$ is referred to as *ε-free* if and only if for all $r : A \to x \in P$ it holds $x \neq \varepsilon$. A rule $r : A \to x$ is referred to as *chain rule* if and only if $x \in V - T$. A derivation step in G is defined for $u, v \in V^*$ and $r : A \to x \in P$ as $uAv \Rightarrow uxv\,[r]$. In the standard manner, we introduce the relations $\Rightarrow^i$, $\Rightarrow^+$, and $\Rightarrow^*$ (see [12]). A rule $r : A \to x \in P$ is referred to as *usable* if and only if there is a derivation $S \Rightarrow^* uAv \Rightarrow uxv\,[r]$ for some $u, v \in V^*$. The language of a context-free grammar G is called *context-free language* and defined as $L(G) = \{x \in T^* |\ S \Rightarrow^* x\}$. The family of context-free languages is denoted by **CF**. A context-free grammar $G = (V, T, P, S)$ is in *Chomsky normal form* if and only if for all $r : A \to x \in P$ it holds either $x \in (V - T)^2$ or $x \in T$.

Let $G = (V, T, P, S)$ be a context-free grammar and $x \in T^*$. Let ${}_G\triangle(x)$ denote a set of the derivation trees with frontier x in G. Let $t \in {}_G\triangle(x)$. A *path* of t is any sequence of the nodes where the first node is the root of t, last node is a leaf of t, and there is an edge in t between each two consecutive nodes of the sequence. Let s be any sequence of the nodes of t, then $word(s)$ denotes the string obtained by concatenation of all labels of the nodes of s in order from left to right.

A *rational transducer* (see [14]) is a 6-tuple $M = (Q, \Sigma, \Omega, \tau, s, F)$ such that Q is a finite set of states, Σ is an input alphabet, Ω is an output alphabet, τ is a finite subset of $S \times \Sigma^* \to S \times \Omega^*$ called transition function, $s \in Q$ is an initial state, $F \subseteq Q$ is a finite set of final states. A configuration of M is (p, u) with $p \in Q$, $u \in \Sigma^*$. A configuration (p, u) is initial, final if $p = s$, $p \in F$, respectively. Given a rational transducer M, for every input word $u \in \Sigma^*$, *rational transduction of u* is denoted as $RT_M(u)$ and defined as $RT_M(u) = \{v \in \Omega^* |\ (t, v) \in \tau(s, u)$ is a final configuration$\}$. A *rational transduction* of a language L is denoted as $RT_M(L)$ and defined as $RT_M(L) = \bigcup\{RT_M(u) |\ u \in L\}$. It is well known that **CF** is closed under rational transduction. Furhter properties of rational transducers can be found in [11].

3. Definitions

First, using the terminology of previous section, we recall the basic notions and definitions given in [6] and [8].

Definition 1 *A* tree-controlled *grammar, TC grammar for short, is a pair* (G, R) *where* $G = (V, T, P, S)$ *is a context-free grammar, and* $R \subseteq V^*$ *is a context-free control language. The* language that (G, R) generates under the path control by R *is denoted by* ${}_{path}L(G, R)$ *and defined by the following equivalence: For all* $z \in T^*$, $z \in {}_{path}L(G, R)$ *if and only if there exists a derivation tree* $t \in {}_G\triangle(z)$ *such that there is path* p *of* t *with* $word(p) \in R$. *Let* **path-TC** $= \{{}_{path}L(G, R) | \ (G, R)$ *is a TC grammar*$\}$ *and* **path-TC**$_{\varepsilon-free}$ $= \{{}_{path}L(G, R) | \ (G, R)$ *is a TC grammar where* G *is* ε*-free*$\}$.

Next, we define 1^{st} and 2^{nd} Chomsky-like normal forms of TC grammars that generates the language under path control. Roughly speaking, compared to Chomsky normal form (see Sect. 2.), 1^{st} normal form adds only unit rules, 2^{nd} normal form adds just one ε-rule.

Definition 2 *Let* (G, R) *be a TC grammar that generates the language under path control by* R*, where* $G = (V, T, P, S)$. (G, R) *is in* 1^{st} normal form *if every rule* $r : A \to x \in P$ *is of the form* $A \in V - T$ *and* $x \in T \cup (V - T) \cup (V - T)^2$.

Definition 3 *Let* (G, R) *be a TC grammar that generates the language under path control by* R*, where* $G = (V, T, P, S)$. (G, R) *is in* 2^{nd} normal form *if every rule* $r : A \to x \in P$ *is of the form* $A \in V - T$ *and* $x \in T \cup ((V \cup \{E\}) - T)^2$ *where* $E \cap V = \emptyset$ *and* $E \to \varepsilon \in P$. *The alphabet of* G *should now include* E*, with* $E \notin V$.

4. Results

Before transforming tree-controlled grammars that generate the language under path control to the normal forms, we establish the following lemma related to erasing rules.

Lemma 1 *For any TC grammar* (G, R) *there is a TC grammar* (G', R) *such that* ${}_{path}L(G, R) = {}_{path}L(G', R)$ *and* G' *is* ε*-free.*

Proof 1 *Let* (G, R) *where* $G = (V, T, P, S)$ *be a TC grammar generating* ${}_{path}L(G, R)$. *Without any loss of generality, assume* G *contains only usable rules. Basicaly, from all correct derivation trees for any* $z \in L(G)$*, we select just those of them containing a path* p *with* $word(p) \in R$.

Consider $t \in {}_{(G,R)}\triangle(z)$*, for any* $z \in {}_{path}L(G, R)$. *Clearly, there is a path* p *of* t *such that* $word(p) = A_1 \dots A_\ell a$ *with* $A_1, \dots, A_\ell \in V - T$*, for* $\ell \geq 1$, $A_1 = S$*, and*

$a \in T$. *Consider the rules* $A_i \to x_i A_{i+1} y_i$, *for* $1 \leq i \leq \ell - 1$, *used when passing from* A_i *to* A_{i+1} *on* p *and, corresponding to* p, *the rule* $A_\ell \to x_\ell a\ y_\ell$ *used in the last step of the derivation in* G. *Since* $word(p) \in \{S\}(V-T)^*T$, *no* $A_i \to x_i A_{i+1} y_i$ *is* ε*-rule, for* $1 \leq i \leq \ell - 1$.

Consider that any $x_i y_i$, $1 \leq i \leq \ell$, *contains* $B \in V - T$ *that does not belong to* p. *Consider a substring* z' *of* z *that is derived from* B. *Since* G *is context-free,* z' *can be generated from* B *without using* ε*-rules (see the well-known Algorithm 5.1.3.2.3 in [12]).*

Therefore, transformation G *into* G' *where* G' *is* ε*-free by aforementioned algorithm cannot affect the language describing a controlled path. Thus, such a transformation cannot restrict or extend* ${}_{path}L(G,R)$ *properly and therefore* ${}_{path}L(G,R) = {}_{path}L(G',R)$ *holds.*

Corollary 2 **path-TC** $=$ **path-TC**$_{\varepsilon-free}$.

Theorem 3 *Let* $L \in$ **path-TC**. *Then, there exists a TC grammar* (G,R) *in* 1^{st} *normal form such that* $L = {}_{path}L(G,R)$.

Proof 4 *Alghorithm 1 is based on well-known Algorithm 5.1.4.1.1 in [12] used for transformation of a context-free grammar to an equivalent context-free grammar in Chomsky normal form. Whenever a rule* $A \to X_1X_2\dots X_n$ *for* $n \geq 3$ *is transformed into the rules* $A \to X_1\langle X_2 \dots X_n\rangle$, $\dots$, $\langle X_{n-1}X_n\rangle \to X_{n-1}X_n$, *if* $word(p) = uAX_iv \in R$ *for a path* p, *for* $i = 1, 2 \dots n$, *rational transducer* M *nondeterministicaly replaces* AX_i *as a substring of* $word(p)$ *by a sequence* $A\langle X_2 \dots X_n\rangle$ $\langle X_3 \dots X_n\rangle \dots \langle X_{i-1} \dots X_n\rangle X_i$. *Whenever new nonterminal symbol* a' *for* $a \in V-T$ *is introduced, rational transducer* M *replaces* a *as a last symbol of* $word(p)$ *by a sequence* $a'a$. *Since* **CF** *is closed under rational transduction,* $R' \in$ **CF**.

Theorem 5 *Let* $L \in$ **path-TC**. *Then, there exists a TC grammar* (G,R) *in* 2^{nd} *normal form such that* $L = {}_{path}L(G,R)$.

Proof 6 *Alghorithm 2 is a straightforward modification of Algorithm 1. First, new symbol* $E \in V$ *and the rule* $E \to \varepsilon \in P'$ *are created, then each rule* $A \to x \in P$ *with* $x \in V - T$ *is replaced by* $A \to xE \in P'$. *Clearly, this transformation does not affect the language describing a controlled path and thus* $R = R' \in$ **CF**.

5. Examples

Next, we demonstrate two examples of typically non-context-free languages that belong to **path-TC** and corresponding TC grammars both in general as well as 1^{st} and 2^{nd} normal form. The following examples demonstrate the languages capturing multiple copy up to four parts and cross-referencing of two parts.

Algorithm 1: Conversion of a TC grammar (G, R) to a TC grammar (G', R') in 1^{st} normal form that generates the same language under the path restriction.

Input: A TC grammar (G, R) where $G = (V, T, P, S)$ and G is ε-free.
Output: A TC grammar (G', R') where $G' = (V', T, P', S)$ satisfying ${}_{path}L(G, R) = {}_{path}L(G', R')$ and (G', R') is in 1^{st} normal form.

begin
    $P' := \{r \mid r : A \to x \in P, x \in T \cup (V - T) \cup (V - T)^2\}$;
    $P_{aux} := \{r \mid r : A \to x \in P, |x| \leq 2, r \notin P'\}$;
    $V' := V$;
    assume rational transducer $M(Q, V', V', \tau, s, F)$ with $RT_M(R) = R$;
    foreach $r : A \to X_1 X_2 \dots X_n \in P$ *where*
    $X_i \in V,\ i = 1, 2, \dots, n$ *for some* $n \geq 3$ **do**
        begin
            $P_{aux} := P_{aux} \cup \{A \to X_1 \langle X_2 \dots X_n \rangle,$
                $\langle X_2 \dots X_n \rangle \to X_2 \langle X_3 \dots X_n \rangle,$
                $\dots$
                $\langle X_{n-2} \dots X_n \rangle \to X_{n-2} \langle X_{n-1} X_n \rangle,$
                $\langle X_{n-1} X_n \rangle \to X_{n-1} X_n\}$;
            $V' := V' \cup \{\langle X_i \dots X_n \rangle \mid i = 2, \dots, n - 1\}$;
            $\tau = \tau \cup \{(f, uA\langle X_2 \dots X_n \rangle \langle X_3 \dots X_n \rangle \dots \langle X_{i-1} \dots X_n \rangle X_i v) \mid$
                $(f, uAX_i v) \in \tau(s, uAX_i v),$
                $f \in F, u, v \in (V')^*, A, X_i \in V'$ for some $2 \leq i \leq n\}$;
        end
    end
    foreach $r : A \to x \in P_{aux}$ *with alph*$(x) \cap T \neq \emptyset$ **do**
        begin
            replace each terminal $a \in T$ with a new symbol $a' \in V'$ in x;
            $V' := V' \cup \{a'\}$;
            $P' := P' \cup \{a' \to a\}$;
            $\tau := \tau \cup \{(f, ua'a) \mid (f, ua) \in \tau(s, ua), f \in F, u \in (V')^*, a \in V - T\}$;
        end
    end
    $P' := P' \cup \{r : A \to x \mid p \in P_{aux}, x \in T \cup (V')^2\}$;
    produce $G' = (V', T, P', S)$;
    produce $R' = RT_M(R)$;
    produce (G', R');
end

Algorithm 2: Conversion of a TC grammar (G, R) to a TC grammar (G', R') in 2^{nd} normal form that generates the same language under the path restriction.

Input: A TC grammar (G, R) in 1^{st} normal form where $G = (V, T, P, S)$.
Output: A TC grammar (G', R') where $G' = (V', T, P', S)$ satisfying ${}_{path}L(G, R) = {}_{path}L(G', R')$ and (G', R') is in 2^{nd} normal form.

```
begin
    V' := V ∪ {E| E ∩ V = ∅};
    P' := P ∪ {E → ε};
    foreach r : A → x ∈ P with x ∈ V − T do
        P' := P' ∪ {A → xE};
    end
    produce G' = (V', T, P', S);
    produce R' = R;
    produce (G', R');
end
```

Example 1 *Consider the TC grammar that generates* ${}_{path}L(G, R)$ *where*

$$
\begin{aligned}
&G = (\{S, B, D, a, b, c, d\}, \{a, b, c, d\}, P, S),\\
&P = \{S \to aSd, \quad S \to aBd, \quad B \to bBc, \quad B \to D, \quad D \to bc\},\\
&R = \{S^n B^n Db \mid n \geq 1\}.
\end{aligned}
$$

Clearly, ${}_{path}L(G, R) = \{a^k b^k c^k d^k \mid k \geq 1\} \notin$ **CF**.

Next, let us transform (G, R) *to* 1^{st} *normal form by Algorithm 1 that outputs* (G', R') *where*

$$
\begin{aligned}
&G' = (\{S, B, D, \langle Sd\rangle, \langle Bd\rangle, \langle Bc\rangle, a', b', c', d', a, b, c, d, e, f\}, \{a, b, c, d, e, f\}, P', S),\\
&P' = \{S \to a'\langle Sd\rangle, \quad S \to a'\langle Bd\rangle, \quad B \to b'\langle Bc\rangle, \quad B \to D, \quad D \to b'c',\\
&\qquad \langle Sd\rangle \to Sd', \quad \langle Bd\rangle \to Bd', \quad \langle Bc\rangle \to Bc',\\
&\qquad a' \to a, \quad b' \to b, \quad c' \to c, \quad d' \to d\},\\
&R' = \{(S\langle Sd\rangle)^n S\langle Bd\rangle(B\langle Bc\rangle)^n Db'b \mid n \geq 1\}.
\end{aligned}
$$

Clearly, ${}_{path}L(G', R') = \{a^k b^k c^k d^k \mid k \geq 1\} \notin$ **CF** *and* (G', R') *is in* 1^{st} *normal form.*

Finally, let us transform (G, R) *to* 2^{st} *normal form by Algorithm 2 that outputs* $(G", R")$ *where*

$$
\begin{aligned}
&G' = (\{S, B, D, E, \langle Sd\rangle, \langle Bd\rangle, \langle Bc\rangle, a', b', c', d', a, b, c, d, e, f\}, \{a, b, c, d, e, f\}, P', S),\\
&P" = \{S \to a'\langle Sd\rangle, \quad S \to a'\langle Bd\rangle, \quad B \to b'\langle Bc\rangle, \quad B \to DE, \quad D \to b'c',\\
&\qquad \langle Sd\rangle \to Sd', \quad \langle Bd\rangle \to Bd', \quad \langle Bc\rangle \to Bc', \quad E \to \varepsilon,\\
&\qquad a' \to a, \quad b' \to b, \quad c' \to c, \quad d' \to d\},\\
&R" = \{(S\langle Sd\rangle)^n S\langle Bd\rangle(B\langle Bc\rangle)^n Db'b \mid n \geq 1\}.
\end{aligned}
$$

Clearly, ${}_{path}L(G", R") = \{a^k b^k c^k d^k \mid k \geq 1\} \notin$ **CF** *and* $(G", R")$ *is in* 2^{nd} *normal form.*

Example 2 *Consider the TC grammar that generates* ${}_{path}L(G, R)$ *where*

$$
\begin{aligned}
&G = (\{S, A, B, C, D, a, b\}, \{a, b\}, P, S),\\
&P = \{S \rightarrow aS, \quad S \rightarrow aB, \quad B \rightarrow bB, \quad B \rightarrow A, \quad A \rightarrow bA, \quad A \rightarrow C,\\
&\qquad C \rightarrow Ca, \;\; C \rightarrow D, \quad D \rightarrow a\},\\
&R = \{S^m B^n A^n C^m Da |\; m, n \geq 1\}.
\end{aligned}
$$

Clearly, ${}_{path}L(G, R) = \{a^k b^l a^k b^l |\; k, l \geq 1\} \notin \mathbf{CF}$.

Next, let us transform (G, R) *to* 1^{st} *normal form by Algorithm 1 that outputs* (G', R') *where*

$$
\begin{aligned}
&G' = (\{S, A, B, C, D, a', b', a, b\}, \{a, b\}, P', S),\\
&P' = \{S \rightarrow a'S, \quad S \rightarrow a'B, \quad B \rightarrow b'B, \quad B \rightarrow A, \quad A \rightarrow b'A, \quad A \rightarrow C,\\
&\qquad C \rightarrow Ca', \;\; C \rightarrow D, \quad D \rightarrow a', \quad a' \rightarrow a, \quad b' \rightarrow b\},\\
&R' = \{S^m B^n A^n C^m Da'a |\; m, n \geq 1\}.
\end{aligned}
$$

Clearly, ${}_{path}L(G, R) = \{a^k b^l a^k b^l |\; k, l \geq 1\} \notin \mathbf{CF}$ *and* (G', R') *is in* 1^{st} *normal form.*

Finally, let us transform (G, R) *to* 2^{st} *normal form by Algorithm 2 that outputs* $(G", R")$ *where*

$$
\begin{aligned}
&G" = (\{S, A, B, C, D, E, a', b', a, b\}, \{a, b\}, P", S),\\
&P" = \{S \rightarrow a'S, \quad S \rightarrow a'B, \quad B \rightarrow b'B, \quad B \rightarrow AE, \quad A \rightarrow b'A, \quad A \rightarrow CE,\\
&\qquad C \rightarrow Ca', \;\; C \rightarrow DE, \;\; D \rightarrow a', \quad a' \rightarrow a, \quad b' \rightarrow b, \quad E \rightarrow \varepsilon\},\\
&R" = \{S^m B^n A^n C^m Da'a |\; m, n \geq 1\}.
\end{aligned}
$$

Clearly, ${}_{path}L(G, R) = \{a^k b^l a^k b^l |\; k, l \geq 1\} \notin \mathbf{CF}$ *and* $(G", R")$ *is in* 2^{nd} *normal form.*

6. Conclusion

In this concluding section, we summarize the achieved results and point out some important open questions.

We have considered the impact of erasing rules in path controlled grammars on the generative power. As opposed to tree controlled grammars (see [3]) in which tree levels are restricted, erasing rules in path controlled grammars in which tree paths are controlled do not restrict nor extend the generative power. On the other hand, when controlling a path, the control language has to be at least linear to extend generative power beyond context-free languages whereas for controlling levels or cuts (see [7]), a regular language is enough. As a result, we have stated that erasing rules can be removed from a path controlled grammar without affecting its language. Note that by introducing path restrictions, the independence of context-free grammars on erasing rules has not been lost.

Then, we have studied two normal forms for path controlled grammars. Both of them are based on Chomsky normal form for context-free grammars. Although we were not able to establish Chomsky normal form for path controlled-grammars, we have introduced the normal form allowing chain rules (and no erasing rules) and the normal form allowing just one erasing rule (and no chain rules). Then we have formulated algorithms that transform a path controlled grammar to its normal form.

Let us point out that it is well-known that membership problem is dedicable in polynomial time for path controlled grammars (see [9] and [10]). Both of these newly established normal forms for path controlled grammars should be taken into consideration in the relation to the results of [2] in order to modify general parsing methods that are based on Chomsky normal form so as to parse path controlled grammars in polynomial time.

Since for context-free grammars there is a well-known algorithm that transforms any context-free grammar in Chomsky normal form into an equivalent context-free grammar in Greibach normal form (see [12]), future investigations concerning the subject of this paper should consider aforementioned algorithm and reformulate it so that it modifies not only controlled grammar but also its controlling language. In other words, such an algoritm should take path controlled grammar in general or in 1^{st} or 2^{nd} normal form and produce an equivalent path controlled grammar in a kind of Greibach-like normal form.

7. Acknowledgements

This work was supported by the research plan MSM0021630528, the BUT FIT grant FIT-S-11-2, GA ČR grant GD102/09/H042, and by the European Regional Development Fund in the IT4Innovations Centre of Excellence project (CZ.1.05/1.1.00-/02.0070).

8. References

[1] Bondy A.; *Graph Theory*, Springer, New York 2010.

[2] Čermák M., Koutný J., Meduna A.; *Parsing based on n-path tree-controlled grammars*, Theoretical and Applied Informatics 23, 2011, pp. 213–228.

[3] Čulik K., Maurer H.A.; *Tree controlled grammars*, Computing 19, 1977, pp. 129–139.

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

[5] Dassow J., Truthe B.; *Subregularly tree controlled grammars and languages*, Automata and Formal Languages – 12$^{\text{th}}$ International Conference AFL 2008,

Balatonfured, Computer and Automation Research Institute of the Hungarian Academy of Sciences, 2008, pp. 158–169.

[6] Koutný J., Křivka Z., Meduna A.; *Pumping properties of path-restricted tree-controlled languages*, 7^{th} Doctoral Workshop on Mathematical and Engineering Methods in Computer Science, 2011, pp. 61–69.

[7] Koutný J., Meduna A.; *Tree-controlled grammars with restrictions placed upon cuts and paths*, Kybernetika 48(1), 2012, pp. 165–175.

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

[9] Martín-Vide C., Mitrana. V.; *Further properties of path-controlled grammars.* In: Proceedings of FG-MoL 2005: The 10^{th} Conference on Formal Grammar and The 9^{th} Meeting on Mathematics of Language, University of Edinburgh, Edinburgh 2005, pp. 221–232.

[10] Martin-Vide C., Mitrana V.; *Decision problems on path-controlled grammars*, IJFCS: International Journal of Foundations of Computer Science 18, 2007.

[11] Mateescu A., Salomaa A.; *Handbook of formal languages*, vol. 1, chapter *Aspects of classical language theory*, Springer-Verlag New York, Inc., New York 1997, pp. 175–251.

[12] Meduna A.; *Automata and Languages: Theory and Applications*, Springer, New York 2005.

[13] Păun Gh.; *On the generative capacity of tree controlled grammars*, Computing 21(3), 1979, pp. 213–220.

[14] Salomaa A.; *Computation and Automata*, vol. 25 of *Encyclopedia of Mathematics and Its Applications*, Cambridge University Press, Cambridge 1985.

[15] Turaev S., Dassow J., Selamat M.H.; *Language classes generated by tree controlled grammars with bounded nonterminal complexity.* In: M. Holzer, M. Kutrib, G. Pighizzini (eds.); *DCFS*, vol. 6808 of *Lecture Notes in Computer Science*, Springer, New York 2011, pp. 289–300.

[16] Turaev S., Dassow J., Selamat M.H.; *Nonterminal complexity of tree controlled grammars*, Theoretical Computer Science 412(41), 2011, pp. 5789–5795.