Acta Cybernetica **20** (2011) 269–283.

Cooperating Distributed Grammar Systems with Random Context Grammars as Components

Zbyněk Křivka* and Tomáš Masopust†

Abstract

In this paper, we discuss cooperating distributed grammar systems where components are (variants of) random context grammars. We give an overview of known results and open problems, and prove some further results.

Keywords: Cooperating distributed grammar system, random context grammar, left-random context grammar.

1 Introduction

Rewriting systems based on a simple form of productions play an important role in formal language theory. Therefore, it is no surprise that context-free grammars and their variants are frequently studied models. However, many systems describing real-life applications, such as parsers of natural and programming languages, require some additional mechanisms that allow to check for context dependencies. From that viewpoint, context-free grammars are not fully convenient for those applications because they are too simple to handle such dependencies.

A natural method of handling more context dependencies with rewriting systems is to compose systems of several components, and to define a cooperation protocol for these components to generate a common sentential form. Such devices are known as *cooperating distributed (CD) grammar systems* [2, 3, 12]. Components are represented by grammars or other rewriting devices, and the protocol for mutual cooperation describes (roughly speaking) the number of steps one component has to perform before allowing another component to work. For instance, the most interesting protocol is the so-called *terminal derivation mode* (t-mode, for short) making the component work until it is not able to perform another derivation step. It is well-known that the cooperation has a significant effect on context-free grammars. Namely, working in non-trivial modes, context-free CD grammar systems are more powerful than ordinary context-free grammars [3].

*Faculty of Information Technology, Brno University of Technology, Božetěchova 2, 612 66 Brno, Czech Republic. E-mail: krivka@fit.vutbr.cz

†Institute of Mathematics of the Czech Academy of Sciences, Žižkova 22, 616 62 Brno, Czech Republic. E-mail: masopust@ipm.cz

DOI: 10.14232/actacyb.20.2.2011.4

Thus, rewriting systems that are simple and able to check for context dependencies are of interest as components of CD grammar systems [10]. One of such systems are *random context grammars* [14], which are a natural generalization of context-free grammars with respect to context dependency checking. Specifically, in random context grammars, two finite sets of non-terminals are attached to each context-free production—a *permitting* and a *forbidding* set—and such a production is applicable only if all permitting symbols appear in the current sentential form, while no forbidding symbol does. The family of random context languages contains properly the family of context-free languages and is properly included in the family of context-sensitive languages (coincides with the family of recursively enumerable languages, respectively, if erasing productions are allowed [1, 14]). In addition, random context grammars with all permitting (forbidding) sets empty result in the introduction of *forbidding* (*permitting*) *grammars*, which are less powerful than random context grammars (the reader is referred to [7, 15], respectively, for more details and for some pumping-like properties of those languages).

In this paper, we discuss the generative power of several variants of CD grammar systems with random context grammars as components, give an overview of known results and open problems, and prove some further results.

2 Preliminaries and Definitions

We assume that the reader is familiar with formal language theory [6, 12, 13]. An alphabet is a finite non-empty set. For an alphabet V, V^* represents the free monoid generated by V. The unit of V^*, the empty string, is denoted by λ, and the free semigroup generated by V is denoted by $V^+ = V^* - \{\lambda\}$. For a string $w \in V^*$, let $|w|$ denote the length of w and $alph(w)$ denote the set of all symbols occurring in w. Let **CF**, **CS**, and **RE** denote the families of context-free, context-sensitive, and recursively enumerable languages, respectively.

A *random context grammar* [14] is a quadruple $G = (N,T,P,S)$, where N is the alphabet of non-terminals, T is the alphabet of terminals such that $N \cap T = \emptyset$, $V = N \cup T$, $S \in N$ is the start symbol, and P is a finite set of productions of the form $(A \to x, Per, For)$, where $A \to x$ is a context-free production, $A \in N$ and $x \in V^*$, and $Per, For \subseteq N$. For $u, v \in V^*$ and a production $(A \to x, Per, For) \in P$, the relation $uAv \Rightarrow uxv$ holds provided that

$$Per \subseteq alph(uv) \quad \text{and} \quad alph(uv) \cap For = \emptyset. \tag{1}$$

The transitive closure and the reflexive and transitive closure of $\Rightarrow$ are denoted by $\Rightarrow^+$ and $\Rightarrow^*$, respectively. The language generated by G is defined as $L(G) = \{w \in T^* : S \Rightarrow^* w\}$. A *permitting (forbidding) grammar* is a random context grammar $G = (N,T,P,S)$, where for each production $(A \to x, Per, For) \in P$, it holds that $For = \emptyset$ ($Per = \emptyset$, respectively). The language families generated by random context grammars, permitting grammars, and forbidding grammars are denoted by $\mathbf{RC}_\lambda$, $\mathbf{PER}_\lambda$, and $\mathbf{FOR}_\lambda$, respectively, and by **RC**, **PER**, and **FOR**, respectively, if they are generated by corresponding grammars without erasing productions.

A *left-random context grammar* [4, 8] is a quadruple $G = (N,T,P,S)$, where N, T, P, and S are the same as in random context grammars. For $u, v \in V^*$ and a production $(A \to x, Per, For) \in P$, we define the relation $uAv \Rightarrow uxv$ provided that $Per \subseteq alph(u)$

and $alph(u) \cap For = \emptyset$. That is, only the symbols on the left side of the rewritten non-terminal are considered. The language generated by G is defined as $L(G) = \{w \in T^* : S \Rightarrow^* w\}$. A *left-permitting (left-forbidding) grammar* is a left-random context grammar $G = (N,T,P,S)$, where for each production $(A \to x, Per, For) \in P$, it holds that $For = \emptyset$ ($Per = \emptyset$, respectively). The language families generated by left-random context grammars, left-permitting grammars, and left-forbidding grammars are denoted by $\ell\mathbf{RC}_\lambda$, $\ell\mathbf{PER}_\lambda$, and $\ell\mathbf{FOR}_\lambda$, respectively, and by $\ell\mathbf{RC}$, $\ell\mathbf{PER}$, and $\ell\mathbf{FOR}$, respectively, if they are generated by grammars without erasing productions.

2.1 Cooperating Distributed Grammar Systems

A *cooperating distributed (CD) grammar system* is a construct $\Gamma = (N,T,P_1,P_2,\ldots,P_n,S)$, $n \geq 1$, where N is the alphabet of non-terminals, T is the alphabet of terminals, $N \cap T = \emptyset$, $S \in N$ is the start symbol, and for $1 \leq i \leq n$, each component P_i is a finite set of context-free productions. For $u,v \in V^*$, $V = N \cup T$, and $1 \leq k \leq n$, let $u \Rightarrow_k v$ denote a derivation step performed by the application of a production from P_k. As usual, extend the relation $\Rightarrow_k$ to $\Rightarrow_k^m$ (the m-step derivation), $m \geq 0$, $\Rightarrow_k^+$, and $\Rightarrow_k^*$. In addition, we define the relation $u \Rightarrow_k^t v$ so that $u \Rightarrow_k^+ v$ and there is no $w \in V^*$ such that $v \Rightarrow_k w$. The languages generated by Γ working in the f-mode, $f \in \{*,t\} \cup \{\leq m,\ =m,\ \geq m : m \geq 1\}$, denoted by $L_f(\Gamma)$, is defined as follows.

t-mode $L_t(\Gamma) = \{w \in T^* :$ there are $\ell \geq 1$ and sentential forms $\alpha_i, 1 \leq i \leq \ell$, such that $\alpha_i \Rightarrow_{k_i}^t \alpha_{i+1}$, $1 \leq k_i \leq n$, $\alpha_1 = S$, and $\alpha_\ell = w\}$.

$*$-mode $L_*(\Gamma) = \{w \in T^* :$ there are $\ell \geq 1$ and sentential forms $\alpha_i, 1 \leq i \leq \ell$, such that $\alpha_i \Rightarrow_{k_i}^* \alpha_{i+1}$, $1 \leq k_i \leq n$, $\alpha_1 = S$, and $\alpha_\ell = w\}$.

$=m$-mode $L_{=m}(\Gamma) = \{w \in T^* :$ there are $\ell \geq 1$ and sentential forms $\alpha_i, 1 \leq i \leq \ell$, such that $\alpha_i \Rightarrow_{k_i}^m \alpha_{i+1}$, $1 \leq k_i \leq n$, $\alpha_1 = S$, and $\alpha_\ell = w\}$, $m \geq 1$.

$\leq m$-mode $L_{\leq m}(\Gamma) = \{w \in T^* :$ there are $\ell \geq 1$ and sentential forms $\alpha_i, 1 \leq i \leq \ell$, such that $\alpha_i \Rightarrow_{k_i}^{j_i} \alpha_{i+1}$, $1 \leq k_i \leq n$, $1 \leq j_i \leq m$, $\alpha_1 = S$, and $\alpha_\ell = w\}$.

$\geq m$-mode $L_{\geq m}(\Gamma) = \{w \in T^* :$ there are $\ell \geq 1$ and sentential forms $\alpha_i, 1 \leq i \leq \ell$, such that $\alpha_i \Rightarrow_{k_i}^{j_i} \alpha_{i+1}$, $1 \leq k_i \leq n$, $1 \leq m \leq j_i$, $\alpha_1 = S$, and $\alpha_\ell = w\}$.

Language families generated by CD grammar systems with n context-free components working in the f-mode are denoted by $\mathbf{CD}_f(\mathbf{CF}_\lambda, n)$, or $\mathbf{CD}_f(\mathbf{CF}, n)$ if the components are non-erasing. The following results are well-known [3].

1. $\mathbf{CD}_f(\mathbf{CF}_\lambda, n) = \mathbf{CD}_f(\mathbf{CF}, n) = \mathbf{CF}$, for $n \geq 1$ and $f \in \{=1, \geq 1, *\} \cup \{\leq k : k \geq 1\}$,
2. $\mathbf{CF} \subset \mathbf{CD}_f(\mathbf{CF}, 2) \subseteq \mathbf{CD}_f(\mathbf{CF}, r) \subseteq \mathbf{MAT}$ and
   $\mathbf{CF} \subset \mathbf{CD}_f(\mathbf{CF}_\lambda, 2) \subseteq \mathbf{CD}_f(\mathbf{CF}_\lambda, r) \subseteq \mathbf{MAT}_\lambda$, for $f \in \{=k, \geq k : k \geq 2\}$ and $r \geq 3$,
3. $\mathbf{CD}_t(\mathbf{CF}_\lambda, 2) = \mathbf{CD}_t(\mathbf{CF}, 2) = \mathbf{CF}$ and $\mathbf{CD}_t(\mathbf{CF}_\lambda, n) = \mathbf{CD}_t(\mathbf{CF}, n) = \mathbf{ET0L}$, for $n \geq 3$,

where **ET0L** denotes the family of languages generated by extended tabled interactionless Lindenmayer systems [12], and **MAT** and $\mathbf{MAT}_\lambda$ denote the families of languages generated by matrix grammars without and with erasing productions [6], respectively.

Obviously, the definition of CD grammar systems can be generalized so that the components are sets of productions of any type. This leads to several new definitions of CD grammar systems with (permitting, forbidding, left-permitting, left-forbidding, left-) random context grammars as components. The family of languages generated by CD grammar systems with n components of type $\mathbf{X}$ working in the f-mode, $f \in \{*,t\} \cup \{\leq k, =k, \geq k : k \geq 1\}$, is denoted by $\mathbf{CD}_f(\mathbf{X}_\lambda, n)$, or $\mathbf{CD}_f(\mathbf{X}, n)$ if the components are non-erasing.

3 Results

First, let us recall that $\mathbf{CF} \subset \mathbf{PER}_\lambda = \mathbf{PER} \subset \mathbf{RC} \subset \mathbf{CS}$, $\mathbf{CF} \subset \mathbf{FOR}_\lambda \subset \mathbf{RC}_\lambda = \mathbf{RE}$, and $\mathbf{CF} \subset \mathbf{FOR} \subset \mathbf{RC}$ [7, 15, 16]. In addition, in the case of left-forbidding grammars $\ell\mathbf{FOR}_\lambda = \ell\mathbf{FOR} = \mathbf{CF}$ [8], i.e., left-forbidding languages coincide with context-free languages. However, it is of interest to compare this with results concerning context-free CD grammar systems and left-forbidding CD grammar systems (below), where it turns out that although the components are of the same power, left-forbidding CD grammar systems are more powerful than context-free CD grammar systems. Furthermore, in the case of left-permitting grammars $\mathbf{CF} \subset \ell\mathbf{PER}$ [4], which is surprising in comparison with the result concerning left-forbidding grammars. The inclusion is clear from the definition, and the strictness follows from the following example [4].

Example 1. Let $G = (\{S,A,C,A',C'\},\{a,b,c\},P,S)$ be a left-permitting grammar, where

$$P = \{(S \to AC,\emptyset),(A \to aA'b,\emptyset),(A \to ab,\emptyset),(A' \to A,\emptyset),\\ (C \to cC',\{A'\}),(C \to c,\emptyset),(C' \to C,\{A\})\}.$$

We can see that $L(G) = \{a^n b^n c^m : n \geq m \geq 1\}$, which is a non-context-free language.

Note that not all the relations among language families **PER**, **FOR**, ℓ**PER**, **RC**, ℓ**RC**, **CS** (and analogously among their erasing variants) are known. More specifically, only the relations depicted in Figure 1 are known. The reader is also referred to [1] for more details.

Open problem 1. *What are the relations among the above mentioned language families?*

3.1 Alternative Definition of the Direct Derivation Step

In the case of random context grammars, the direct derivation step is in the literature also defined so that the rewritten symbol is considered by the context-dependency checking mechanism (cf. [14] and [11]), i.e., for $u,v \in V^*$ and a production $(A \to x, Per, For) \in P$, the relation $uAv \Rightarrow uxv$ holds provided that

$$Per \subseteq alph(uAv) \quad \text{and} \quad alph(uAv) \cap For = \emptyset. \tag{2}$$

Lemma 1. *Definitions* (1) *and* (2) *are equivalent for random context (permitting, forbidding) grammars.*

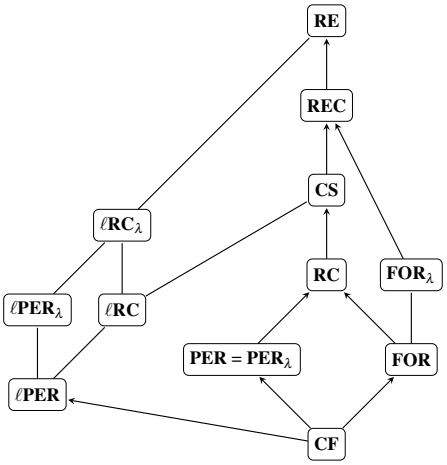

Figure 1: A hierarchy of language families. If two families are connected by a line (an arrow), the upper family includes (includes properly) the lower family. If two families are not connected, they are not necessarily incomparable.

Proof. $(1) \Rightarrow (2)$: Let $G = (N,T,P,S)$ be a random context (permitting, forbidding) grammar using definition (1). Construct the grammar $G' = (N \cup N',T,P',S)$ of the same type using definition (2) so that $N' = \{A' : A \in N\}$, $N \cap N' = \emptyset$, and P' is defined as follows.

(a) $P' = \{(A \to A',\emptyset,N'),(A' \to x,Per,For) : (A \to x,Per,For) \in P\}$ for forbidding and random context grammars.

(b) $P' = \{(A \to A',\emptyset,\emptyset),(A' \to x,Per,\emptyset) : (A \to x,Per,\emptyset) \in P\}$ for permitting grammars.

It is not hard to see that $L(G) = L(G')$.

$(2) \Rightarrow (1)$: Let G be a random context (permitting, forbidding) grammar using definition (2). Construct the grammar G' of the same type using definition (1) so that for each production $p = (A \to x,Per,For)$ of G with $A \notin For$, add $p' = (A \to x,Per - \{A\},For)$ to productions of G'. Clearly, p is applicable in G if and only if p' is applicable in G'. □

Similarly, we can modify definition (2) of the direct derivation step for left-random context grammars, i.e., for $u,v \in V^*$ and a production $(A \to x,Per,For) \in P$, the relation $uAv \Rightarrow uxv$ holds provided that $Per \subseteq alph(uA)$ and $alph(uA) \cap For = \emptyset$. As above, we refer to those two definitions as to definitions (1) and (2). The reader can see that the implication $(2) \Rightarrow (1)$ of the previous lemma holds for left-random context (left-permitting, left-forbidding) grammars. Thus, definition (2) is weaker than definition (1) in the sense that every left-random context (left-permitting, left-forbidding) grammar using definition (2) can be converted to an equivalent grammar of the same type using definition (1). As left-forbidding grammars generate only context-free languages, we have that these two definitions are equivalent for left-forbidding grammars. In addition, the implication $(1) \Rightarrow (2)$ holds for left-permitting grammars. Thus, we have the following result.

Lemma 2. *Definitions* (1) *and* (2) *are equivalent for left-permitting and left-forbidding grammars.*

However, the construction (1) $\Rightarrow$ (2) does not work for left-random context grammars.

Open problem 2. *Are these two definitions equivalent for left-random context grammars?*

As definition (2) is weaker than definition (1), in the sense mentioned above, we implicitly use definition (1) from now on. However, definition (2) is also discussed.

3.2 (Left-)Random Context Components

Let $\Gamma = (N, T, P_1, P_2, \ldots, P_n, S)$, $n \geq 1$, be a CD grammar system with (left-)random context components and consider an f-mode, $f \in \{*, =1, \geq 1\} \cup \{\leq k : k \geq 1\}$. The behavior of Γ is then characterized by choosing a component and applying any of its productions; the cycle is repeated. Thus, Γ behaves as the (left-)random context grammar $G = (N, T, P_1 \cup P_2 \cup \cdots \cup P_n, S)$. The following holds.

Lemma 3. *For $n \geq 1$ and $f \in \{*, =1, \geq 1\} \cup \{\leq k : k \geq 1\}$,*

1. $\boldsymbol{CD_f(RC_\lambda, n) = RC_\lambda}$ *and* $\boldsymbol{CD_f(RC, n) = RC}$,
2. $\boldsymbol{CD_f(FOR_\lambda, n) = FOR_\lambda}$ *and* $\boldsymbol{CD_f(FOR, n) = FOR}$,
3. $\boldsymbol{CD_f(PER_\lambda, n) = PER_\lambda = PER = CD_f(PER, n)}$ [16],
4. $\boldsymbol{CD_f(\ell PER_\lambda, n) = \ell PER_\lambda}$ *and* $\boldsymbol{CD_f(\ell PER, n) = \ell PER}$,
5. $\boldsymbol{CD_f(\ell FOR_\lambda, n) = CF = CD_f(\ell FOR, n)}$ [8].

By the well-known result $\mathbf{RC}_\lambda = \mathbf{RE}$ [6], we have the following result.

Lemma 4. *For $n \geq 1$ and $f \in \{t\} \cup \{=k, \geq k : k \geq 2\}$, $\boldsymbol{CD_f(RC_\lambda, n) = RC_\lambda}$.*

Given a random context grammar without erasing productions, the grammar can be considered as the only component (with productions $(A \to A, \emptyset, \emptyset)$ added, if needed), which gives the following.

Lemma 5. *For $n \geq 1$ and $f \in \{t\} \cup \{=k, \geq k : k \geq 2\}$, $\boldsymbol{RC \subseteq CD_f(RC, n)}$.*

We prove that the other inclusion holds true, too.

Lemma 6. *For $n \geq 1$, $\boldsymbol{CD_t(RC, n) \subseteq RC}$.*

Proof. Let $\Gamma = (N, T, P_1, P_2, \ldots, P_n, S)$ be a CD grammar system with n random context components. Construct the random context grammar $G = (\tilde{N} \cup \{\tilde{S}\}, T \cup \{c\}, P', \tilde{S})$, where c, $\tilde{S}$ are new symbols, $c, \tilde{S} \notin T \cup \tilde{N}$, $\tilde{N} = N \cup N' \cup \{[Q_i], \langle p, Q_i \rangle, [p, Q_i], [i] : Q_i \subseteq P_i, p \in Q_i, 1 \leq i \leq n\}$, $N' = \{X' : X \in N\}$, and P' is constructed as follows:

1. For each $(A \to x, Per, For) \in P_i$, add $(A \to x, Per \cup \{[i]\}, For)$ to P'.
2. For $1 \leq i, \ell \leq n$, add to P'

a) $(\tilde{S} \to S[i], \emptyset, \emptyset)$,
b) $([i] \to [P_i], \emptyset, \emptyset)$,
c) $([\emptyset] \to [\ell], \emptyset, \emptyset)$,
d) $([i] \to c, \emptyset, \emptyset)$.

3. For $Q_i \subseteq P_i$ and $p = (A \to x, Per, For) \in Q_i$, $1 \leq i \leq n$, add also to P'

e) $([Q_i] \to [Q_i - \{p\}], \emptyset, \{A\})$,
f) $([Q_i] \to \langle p, Q_i \rangle, \{A\}, \emptyset)$,
g) $(A \to A', \{\langle p, Q_i \rangle\}, \{A'\})$,
h) $(\langle p, Q_i \rangle \to [p, Q_i], \{A'\}, \{X\})$, for $X \in Per$,
i) $(\langle p, Q_i \rangle \to [p, Q_i], \{A', X\}, \emptyset)$, for $X \in For$,
j) $(A' \to A, \emptyset, \emptyset)$,
k) $([p, Q_i] \to [Q_i - \{p\}], \emptyset, \{A'\})$.

The main idea of the proof is to simulate the CD grammar system so that the non-terminal $[i]$ denotes the simulated component, the grammar uses productions of P_i, and in some moment it non-deterministically decides that no other productions of P_i are applicable, see production (2.b). This is then verified by productions in (3) so that a production p is removed from the non-terminal Q_i if it is not applicable. If no productions of P_i are applicable, production (2.c) can change the component.

We prove that $L_t(\Gamma)c = L(G)$. As non-erasing random context grammars are closed under restricted homomorphisms (Lemma 1.3.3 in [6]), we get that there is a random context grammar H such that $L_t(\Gamma) = L(H)$.

The simulation of Γ by G starts with a production constructed in (2.a) applied only once because $\tilde{S}$ does not occur on the right side of any production. Furthermore, every successful derivation is finished by the application of a production constructed in (2.d). If $uAv \Rightarrow_i uxv$ in Γ, for $1 \leq i \leq n$, $A \in N$, $u, v, x \in (N \cup T)^*$, then $uAv[i] \Rightarrow uxv[i]$ in G by $(A \to x, Per \cup \{[i]\}, For)$. If Γ changes components from P_i to P_ℓ, for some $1 \leq i, \ell \leq n$, and the sentential form is $w \in (N \cup T)^*$, then no production from P_i can be applied. This is verified in G by the following sequence of productions: $w[i] \Rightarrow_G w[P_i] \Rightarrow_G^* w[\emptyset] \Rightarrow_G w[\ell]$, by (2.b), (3)*, and (2.c). In more detail, for every $p = (A \to x, Per, For) \in P_i$, if A does not occur in w, then $w[Q_i] \Rightarrow_G w[Q_i - \{p\}]$, by (3.e); otherwise, if $w = uAv$, we need to check that it is not applicable because of permitting and forbidding sets:

$$\begin{aligned} uAv[Q_i] &\Rightarrow_G uAv\langle p, Q_i \rangle \Rightarrow_G uA'v\langle p, Q_i \rangle && \text{(by (3.f) and (3.g))} \\ &\Rightarrow_G uA'v[p, Q_i] && \text{(by (3.h) or (3.i))} \\ &\Rightarrow_G uAv[p, Q_i] \Rightarrow_G uAv[Q_i - \{p\}] && \text{(by (3.j) and (3.k))} \end{aligned}$$

where (3.h) is applied if there is $X \in Per$ missing in uv, whereas (3.i) is applied if there is $Y \in For$ occurring in uv. As p is not applicable to w in Γ, at least one of (3.h) and (3.i) must be applicable in G.

On the other hand, to prove $L(G) \subseteq L_t(\Gamma)c$, let h be a homomorphism defined as $h(X) = X$, $X \in N \cup T$, $h(A') = A$, $A' \in N'$, $h(B) = \lambda$ otherwise, and let $N_Q = \tilde{N} - (N \cup N')$. We prove that if $yY \Rightarrow zZ$ in G by (2.b), (2.c), (2.d), or (3), then $h(y) \Rightarrow^* h(z)$ in Γ, for $y, z \in (N \cup N' \cup T)^*$, $Y \in N_Q$, and $Z \in N_Q \cup \{c\}$. As yY contains exactly one symbol from N_Q and this is preserved by (2.b), (2.c), and (3), $yY \Rightarrow zZ$ by (2.b), (2.c), (2.d), or (3) implies that $|yY| = |zZ|$. Consider the form of Y: (I) If $Y = [i]$, productions (1), (2.b), and (2.d)

can be applied. (1) is applied if $yY = uAv[i] \Rightarrow uxv[i] = zZ$ in G, which corresponds to $y = uAv \Rightarrow_i uxv = z$ in Γ by $(A \to x, Per, For) \in P_i$. (2.b) implies that $y = z$ and $Z = [P_i]$. (2.d) is the last step of any successful derivation of G replacing the rightmost non-terminal with c; here, $y = z$ and $Z = c$. (II) If $Y = [Q_i]$, $p = (A \to x, Per, For) \in Q_i$ is tested for its non-applicability to y. If (3.e) is applied, A does not occur in y, which implies that p is not applicable; this is remembered by removing p from Q_i. Here, $y = z$ and $Z = [Q_i - \{p\}]$. If (3.f) is applied, then A appears in y, $y = z$, and $Z = \langle p, Q_i \rangle$ (see III below). If (2.c) is applied, then $Q_i = \emptyset$, $y = z$, and $Z = [\ell]$, for some $1 \leq \ell \leq n$. This means that there is no applicable production and the component can be changed to ℓ. (III) $Y = \langle p, Q_i \rangle$, $p = (A \to x, Per, For) \in Q_i$. If (3.g) is applied, then A' does not occur in y, A' occurs in z, $h(y) = h(z)$, and only (3.h), (3.i), and (3.j) can be applied. If (3.h) is applied, A occurs in $h(y)$ and at least one symbol $X \in Per$ is missing in uv, for $h(y) = h(z) = uAv$. Thus, p is not applicable in Γ, and $Z = [p, Q_i]$ (see IV below). Analogously, if (3.i) is applied, then there is $X \in For$ occurring in uv. Again, p is not applicable in Γ, and $Z = [p, Q_i]$. (IV) $Y = [p, Q_i]$, $p = (A \to x, Per, For) \in Q_i$. If (3.j) is applied, then A' occurs in y, $h(y) = h(z) = z$, and $Z = Y$. If (3.k) is applied, then A' does not occur in $y = z$ and $Z = [Q_i - \{p\}]$. Thus, we remember than p is not applicable. This cycle is repeated until the non-terminal $[\emptyset]$ is reached (see II above). As the successful derivation starts by (2.a), the proof proceeds by induction. Thus, for $u[i], v[\ell] \in (N \cup T)^* N_Q$, we have proved that if $u[i] \Rightarrow^* v[\ell]$ in G, then $u \Rightarrow_i^t v$ in Γ by productions from P_i, which completes the proof. □

The following lemma discusses the effect of the remaining two derivation modes.

Lemma 7. *For $n \geq 1$ and $f \in \{=k, \geq k : k \geq 2\}$, $\boldsymbol{CD_f(RC}, n) \subseteq \boldsymbol{RC}$.*

Proof. Let $\Gamma = (N, T, P_1, P_2, \ldots, P_n, S)$ be a CD grammar system with n random context components working in the $\geq k$-mode, for $k \geq 2$. Construct the random context grammar $G = (\tilde{N} \cup \{\tilde{S}\}, T \cup \{c\}, P', \tilde{S})$, where $c, \tilde{S} \notin T \cup \tilde{N}$, $\tilde{N} = N \cup N_{rhs} \cup \{[i,m], \langle i,m \rangle : 1 \leq i \leq n, 0 \leq m \leq k\}$, $N_{rhs} = \{\langle x \rangle : (A \to x, Per, For) \in P_i, 1 \leq i \leq n\}$, and for all P_i, $1 \leq i \leq n$, and all $(A \to x, Per, For) \in P_i$, add the following productions to P':

1. $(\tilde{S} \to S[i,k], \emptyset, \emptyset)$,
2. $(A \to \langle x \rangle, Per \cup \{[i,m]\}, For \cup N_{rhs})$, where $1 \leq m \leq k$
3. $(\langle x \rangle \to x, \{\langle i,m \rangle\}, \emptyset)$, where $0 \leq m \leq k$
4. $([i,k] \to \langle i,k \rangle, \{\langle x \rangle\}, \emptyset)$,
5. $([i,m] \to \langle i,m-1 \rangle, \{\langle x \rangle\}, \emptyset)$, where $1 \leq m \leq k$,
6. $(\langle i,m \rangle \to [i,m], \emptyset, \{\langle x \rangle\})$, where $0 \leq m \leq k$,
7. $([i,0] \to [j,k], \emptyset, \emptyset)$, where $1 \leq j \leq n$,
8. $([i,0] \to c, \emptyset, \emptyset)$.

Each non-terminal of the form $[i,m]$ or $\langle i,m \rangle$ consists of the index, i, of the simulated component of Γ and the counter, m, of the number of productions of P_i which need to be applied by Γ to allow another component to work. They are used to simulate productions of the ith component and to count the number of simulated components, respectively.

To prove that $L_{\geq k}(\Gamma)c \subseteq L(G)$, we demonstrate that if $uAv \Rightarrow_i uxv$ in Γ by a production $(A \to x, Per, For) \in P_i$, then

$$\begin{aligned} uAv[i,m] &\Rightarrow u\langle x\rangle v[i,m] && \text{(by (2))} \\ &\Rightarrow u\langle x\rangle v\langle i,o\rangle && \text{(by (4) or (5))} \\ &\Rightarrow uxv\langle i,o\rangle && \text{(by (3))} \\ &\Rightarrow uxv[i,o] && \text{(by (6))} \end{aligned}$$

where $o = m-1$, for $m < k$, or $o \in \{k, k-1\}$, for $m = k$. If Γ makes $k+\ell$ steps by productions from P_i, for some $\ell \geq 0$, then in $k+\ell$ repetitions of the derivation of G shown above, the first ℓ is made using production (4), while the last k is made using production (5). Furthermore, when Γ changes its component from P_i to P_j, for some $1 \leq i, j \leq n$, then G derives $x[i,0] \Rightarrow x[j,k]$ by production (7). As every derivation of G starts by a production constructed in (1), the proof proceeds by induction. Finally, the last non-terminal is rewritten to c by (8) as the last step of the simulation.

To complete the proof, we demonstrate that $L(G) \subseteq L_{\geq k}(\Gamma)c$. Let $N_Q = \tilde{N} - (N \cup N_{rhs})$, and let h be a homomorphism defined as $h(X) = X$, $X \in N \cup T$, $h(\langle x\rangle) = x$, $\langle x\rangle \in N_{rhs}$, and $h(B) = \lambda$ otherwise. As $\tilde{S} \Rightarrow S[i,k]$ starts the simulation, we prove that if $yY \Rightarrow zZ$ in G, then $h(y) \Rightarrow^* h(z)$ in Γ, for $y, z \in (N \cup N_{rhs} \cup T)^*$, $Y \in N_Q$, and $Z \in N_Q \cup \{c\}$. Clearly, $S[i,k]$ contains one symbol from N_Q and this is preserved by productions (4) and (2) through (7). The last non-terminal, Y, and the occurrence of a symbol from N_{rhs} control the derivation of G in the following way: (I) $Y = [i,m]$, $1 \leq m \leq k$. If y contains no symbol $\langle x\rangle$, (2) simulates the corresponding production of P_i including permitting and forbidding checks, i.e., $y = uAv \Rightarrow u\langle x\rangle v = z$ and $h(y) = uAv \Rightarrow uxv = h(z)$; $Z = [i,m]$. If it contains $\langle x\rangle$, (4) or (5) is applied to count the number of remaining steps of the current component. Thus, $h(y) = h(z)$, and $Z = \langle i,m\rangle$ or $Z = \langle i,m-1\rangle$. (II) $Y = \langle i,m\rangle$, $0 \leq m \leq k$. If $\langle x\rangle$ occurs in y, (3) finishes the simulation of a production from P_i, i.e., $y = u\langle x\rangle v \Rightarrow uxv = z$ and $h(y) = h(z)$; $Z = \langle i,m\rangle$. If there is no $\langle x\rangle$ in y, Y is rewritten by (6), $y = z$, and $Z = [i,m]$. (III) $Y = [i,0]$. Then, either (7) is applied to replace $[i,0]$ with $Z = [j,k]$, for some $1 \leq j \leq n$, or (8) finishes the simulation by replacing $[i,0]$ with $Z = c$. Thus, for $\alpha[i,m] \in (N \cup T)^* N_Q$, we have proved that only the sequence of productions (2), (4) or (5), (3), and (6) is applicable, resulting in a string over $(N \cup T)^* N_Q$. As the derivation starts by (1), generating a string over $(N \cup T)^* N_Q$, the proof proceeds by induction. As, in addition, productions (4) do not change the counter, (5) decrease the counter, and (2) (simulating productions from P_i) are applicable only if the counter is not zero, G simulates at least k applications of productions from P_i. Thus, we have proved that $L(G) = L_{\geq k}(\Gamma)c$ and, again, by [6, Lemma 1.3.3], there exists a random context grammar H such that $L_{\geq k}(\Gamma) = L(H)$.

Finally, by omitting productions constructed in (4), G simulates exactly k applications of productions of the given component. Thus, it proves the statement for $=k$-mode, where $k \geq 2$. Hence, the proof is complete. □

The constructions of proofs of Lemmas 6 and 7 can be modified so that the results also hold for random context components using definition (1). Productions constructed in (3.g) and (3.j) are removed from the construction of the proof of Lemma 6, and symbol A' is

replaced with A in the productions constructed in (3) of that proof. The proof of Lemma 7 holds for definition (1) as it is.

We summarize the results in the following theorem.

Theorem 1. *For $n \geq 1$ and $f \in \{t\} \cup \{=k, \geq k : k \geq 2\}$, $\boldsymbol{CD_f(RC}, n) = \boldsymbol{RC}$.*

3.3 (Left-)Forbidding Components

Considering terminal derivation mode, the following theorem is proved in [9]. Note that definition (2) of the direct derivation step is used there. However, by simple modifications, constructions are also valid for forbidding components using definition (1).

Theorem 2. *For $n \geq 2$, $\boldsymbol{CD_t(FOR_\lambda}, n) = \boldsymbol{RE}$ and $\boldsymbol{CD_t(FOR}, n) = \boldsymbol{RC} \subset \boldsymbol{CS}$.*

The following theorems discuss the remaining modes.

Theorem 3. *$\boldsymbol{RC} = \bigcup_{k,n\geq 1} \boldsymbol{CD_{=k}(FOR}, n)$ and $\boldsymbol{RC_\lambda} = \bigcup_{k,n\geq 1} \boldsymbol{CD_{=k}(FOR_\lambda}, n)$.*

Proof. The inclusions $\mathbf{CD}_{=k}(\mathbf{FOR}_\lambda, n) \subseteq \mathbf{RC}_\lambda$ and $\mathbf{CD}_{=k}(\mathbf{FOR}, n) \subseteq \mathbf{RC}$ follow by Lemmas 4 and 7, respectively. To prove the other inclusions, let $G = (N, T, P, S)$ be a random context grammar, and construct the CD grammar system Γ with $n = |P| + 1$ forbidding components. The main idea of the proof is to introduce a new component for each production of G in which the presence of all the permitting symbols is verified by replacing them one by one (see more details below). Thus, $\Gamma = (N \cup N' \cup N_{rhs} \cup N_{cnt}, T, P_1, P_2, \ldots, P_n, S)$, where

- $N' = \{A' : A \in N\}$, $N_{rhs} = \{\langle x \rangle : (A \to x, Per, For) \in P\}$,
- $m = \max\{|Per - \{A\}| : (A \to x, Per, For) \in P\}$,
- $N_{cnt} = \bigcup_{i=1}^{m} N^{(i)}$, for $N^{(i)} = \{A^{(i)} : A \in N\}$, $1 \leq i \leq m$,

and Γ works in $=k$-mode, where $k = m + 1$. Let β be a bijection from P to $\{1, 2, \ldots, |P|\}$. The components of Γ are constructed as follows:

1. For each production $(A \to x, Per, For) \in P$, add to $P_{\beta(A\to x, Per, For)}$:
   a) $(A \to A^{(m-|Per-\{A\}|)}, \emptyset, N_{cnt} \cup N_{rhs} \cup N')$, for $m - |Per - \{A\}| > 0$,
   b) $(A^{(i)} \to A^{(i-1)}, \emptyset, N_{rhs} \cup N')$, where $1 < i \leq m - |Per - \{A\}|$,
   c) $(B \to B', \emptyset, N_{rhs} \cup \{B'\})$, where $B \in Per - \{A\}$,
   d) $(A^{(1)} \to \langle x \rangle, \emptyset, For \cup N_{rhs})$,
   e) $(A \to \langle x \rangle, \emptyset, For \cup N_{rhs})$, for $m - |Per - \{A\}| = 0$.
2. To P_n add:
   a) $(B' \to B, \emptyset, N_{cnt})$, where $B \in N$,
   b) $(\langle x \rangle \to \langle x \rangle, \emptyset, N' \cup N_{cnt})$, where $\langle x \rangle \in N_{rhs}$.
   c) $(\langle x \rangle \to x, \emptyset, N' \cup N_{cnt})$, where $\langle x \rangle \in N_{rhs}$,

Note that the crucial point of the construction is that exactly $|Per - \{A\}|$ productions of type (1.c) have to be applied, where the forbidding context guarantees that only one occurrence of any nonterminal can be primed.

To prove $L(G) = L_{=k}(\Gamma)$, we show that each component P_i, $i = 1,2,\ldots,n-1$, simulates exactly one production of G. Let $p = (A \to x, Per, For)$ be a production of G. The simulation is done by a sequence of productions from $P_{\beta(p)}$ as follows. (A) For $m - |Per - \{A\}| > 0$: A production constructed in (1.a), then $m - |Per - \{A\}| - 1$ productions constructed in (1.b), then $|Per - \{A\}|$ productions constructed in (1.c), finished by a production constructed in (1.d). Summarized, $1 + (m - |Per - \{A\}| - 1) + (|Per - \{A\}|) + 1 = m + 1$ productions are applied, and the component is changed. (B) For $m - |Per - \{A\}| = 0$: $|Per - \{A\}|$ productions constructed in (1.c), finished by a production constructed in (1.e). Again, $|Per - \{A\}| + 1 = m - |Per - \{A\}| + |Per - \{A\}| + 1 = m + 1$ productions are applied, and the component is changed. After that, the current sentential form contains $|Per - \{A\}|$ primed non-terminals and one symbol $\langle x \rangle$, for some x. The following sequence of productions of P_n is applied to remove these symbols: $|Per - \{A\}|$ productions constructed in (2.a), $m - |Per - \{A\}|$ productions constructed in (2.b), and a production constructed in (2.c).

On the other hand, we show that this is the only possible behavior of Γ. (A) For $m - |Per - \{A\}| > 0$: Considering the work of P_i, a production constructed in (1.a) has to be applied first; otherwise, if (1.c) is applied first, then (1.a) (and also (1.b) and (1.d)) cannot be applied. However, as $|Per - \{A\}| < m$ and P_i has to perform $m + 1$ steps, the derivation is blocked. (B) For $m - |Per - \{A\}| = 0$: Considering the work of P_i, if $|Per - \{A\}| > 0$, then productions constructed in (1.c) have to be applied first; otherwise, if (1.e) is applied first, then (1.c) cannot be applied. However, as $|Per - \{A\}| = m > 0$, the derivation is blocked. Thus, each component P_i, $i = 1,2,\ldots,n-1$, generates no more than m primed non-terminals and one symbol $\langle x \rangle$, for some x. Considering the component P_n, productions constructed in (2.a) have to be applied first. Then, if (2.c) is not applied as the $(m+1)$st production, the derivation is blocked. On the other hand, if (2.c) is not applied at all, i.e., only (2.b) is applied, then only P_n contains applicable productions. Thus, until a production constructed in (2.c) is applied as the $(m+1)$st production, P_n is chosen repeatedly to work.

As we do not introduce any new erasing productions, the proof is complete. □

Corollary 1. $\boldsymbol{RC} = \bigcup_{k,n\geq 1} \boldsymbol{CD}_{\geq k}(\boldsymbol{FOR}, n)$ *and* $\boldsymbol{RC}_\lambda = \bigcup_{k,n\geq 1} \boldsymbol{CD}_{\geq k}(\boldsymbol{FOR}_\lambda, n)$.

Proof. In the proof of the previous theorem, each component P_i, $i = 1,2,\ldots,n-1$, performs exactly k steps, only P_n can perform more than k steps. □

Corollary 2.

1. $\boldsymbol{RC} = \bigcup_{n\geq 1} \boldsymbol{CD}_{=2}(\boldsymbol{FOR}, n) = \bigcup_{n\geq 1} \boldsymbol{CD}_{\geq 2}(\boldsymbol{FOR}, n)$, *and*
2. $\boldsymbol{RC}_\lambda = \bigcup_{n\geq 1} \boldsymbol{CD}_{=2}(\boldsymbol{FOR}_\lambda, n) = \bigcup_{n\geq 1} \boldsymbol{CD}_{\geq 2}(\boldsymbol{FOR}_\lambda, n)$.

Proof. It is shown in [5] that, for any random context grammar G', there is an equivalent random context grammar $G = (N, T, P, S)$ using definition (2) such that G is with erasing productions if and only if G' is, and for $(A \to w, Per, For) \in P$, $|Per \cup For| = 1$. Thus, in the proof of the previous theorem, we have $m \leq 1$, and if $m = 1$, then Γ works in $=$2-mode. □

Open problem 3. *Can the number of components be reduced?*

The following is proved for left-forbidding CD grammar systems in [8].

Theorem 4. *For $n \geq 2$ and $f \in \{t\} \cup \{=k, \geq k : k \geq 2\}$, $\boldsymbol{RE} = \boldsymbol{CD}_f(\ell\boldsymbol{FOR}_\lambda, n)$ and $\boldsymbol{CS} = \boldsymbol{CD}_f(\ell\boldsymbol{FOR}, n)$. In addition, any recursively enumerable language can be generated by a left-forbidding CD grammar system working in terminal derivation mode with two components and twelve non-terminals.*

3.4 (Left-)Permitting Components

Although the relation between the families **PER** and **FOR** is not known, permitting CD grammar systems and forbidding CD grammar systems working in terminal derivation mode are of the same power [4].

Theorem 5. *For $n \geq 2$, $\boldsymbol{CD}_t(\boldsymbol{PER}_\lambda, n) = \boldsymbol{CD}_t(\ell\boldsymbol{PER}_\lambda, n) = \boldsymbol{RE}$ and $\boldsymbol{CD}_t(\boldsymbol{PER}, n) = \boldsymbol{RC} \subset \boldsymbol{CS} = \boldsymbol{CD}_t(\ell\boldsymbol{PER}, n)$. In addition, any recursively enumerable language can be generated by a left-permitting CD grammar system working in terminal derivation mode with six components and nineteen non-terminals.*

Open problem 4. *What is the generative power of permitting (left-permitting) CD grammar systems working in the f-mode, for $f \in \{=k, \geq k : k \geq 2\}$?*

In this paper, the components use definition (1) of the direct derivation step. In comparison with forbidding CD grammar systems working in terminal derivation mode where these two definitions are equivalent, we do not know whether they are equivalent in the case of permitting CD grammar systems with at least two components, although they are equivalent for permitting grammars (see Lemma 1).

Open problem 5.

1. *What is the power of permitting CD grammar systems if the components (permitting grammars) use definition (2) of the direct derivation step?*
2. *What is the power of left-permitting CD grammar systems if the components use definition (2)?*

4 Conclusion and discussion

In this paper, we have discussed CD grammar systems where components are (variants of) random context grammars. Recall that CD grammar systems with only permitting and with only forbidding components have been studied in [4] and [9], respectively. Originally, the components of forbidding CD grammar systems used definition (2) of the direct derivation step. However, the constructions can be modified so that the results hold for forbidding CD grammar systems with components using definition (1) as well. In addition, all the results concerning CD grammar systems with random context components proved in this paper hold for both definitions. On the other hand, to achieve results concerning the generative

power of CD grammar systems with permitting components proved in [4], definition (1) of the direct derivation step is used. Unfortunately, in this case, we do not know whether the same results can also be achieved for CD grammar systems with permitting components using definition (2), although definitions (1) and (2) are equivalent for permitting grammars (see Lemma 1). Note that it is not hard to see that definition (1) allows us to check for at least two occurrences of a given non-terminal symbol (the rewritten one and the one occurring on the left or on the right of the rewritten symbol), while definition (2) seems to be too weak to check for that property. However, it might be possible to check for that property by using some mechanisms of CD grammar systems instead. The cases of CD grammar systems with left-permitting and left-forbidding grammars as components are studied in [4] and [8], respectively. In these cases, only definition (1) of the direct derivation step is considered. An overview of the results follows.

For any $n \geq 1$, derivation modes $f \in \{*, =1, \geq 1\} \cup \{\leq k : k \geq 1\}$, and language families $\mathbf{X} \in \{\mathbf{RC}_\lambda, \mathbf{RC}, \mathbf{FOR}_\lambda, \mathbf{FOR}, \mathbf{PER}_\lambda, \mathbf{PER}, \ell\mathbf{RC}_\lambda, \ell\mathbf{RC}, \ell\mathbf{PER}_\lambda, \ell\mathbf{PER}, \ell\mathbf{FOR}_\lambda, \ell\mathbf{FOR}\}$,

$$\mathbf{CD}_f(\mathbf{X}, n) = \mathbf{X},$$

and for any $n \geq 2$ and derivation modes $f \in \{t\} \cup \{=k, \geq k : k \geq 2\}$,

1. $\mathbf{CD}_f(\mathbf{RC}_\lambda, n) = \bigcup_{m \geq 1} \mathbf{CD}_f(\mathbf{FOR}_\lambda, m) = \mathbf{CD}_f(\ell\mathbf{RC}_\lambda, n) = \mathbf{CD}_f(\ell\mathbf{FOR}_\lambda, n) = \mathbf{RE}$,
2. $\mathbf{CD}_t(\mathbf{FOR}_\lambda, n) = \mathbf{CD}_t(\mathbf{PER}_\lambda, n) = \mathbf{CD}_t(\ell\mathbf{PER}_\lambda, n) = \mathbf{RE}$,
3. $\mathbf{CD}_f(\ell\mathbf{RC}, n) = \mathbf{CD}_f(\ell\mathbf{FOR}, n) = \mathbf{CD}_t(\ell\mathbf{PER}, n) = \mathbf{CS}$,
4. $\mathbf{CD}_f(\mathbf{RC}, n) = \bigcup_{m \geq 1} \mathbf{CD}_f(\mathbf{FOR}, m) = \mathbf{RC}$,
5. $\mathbf{CD}_t(\mathbf{FOR}, n) = \mathbf{CD}_t(\mathbf{PER}, n) = \mathbf{RC}$.

Recall that it has recently been shown in [16] that the generative power of permitting grammars coincides with the generative power of non-erasing permitting grammars. In other words, the erasing productions can be removed from permitting grammars. In addition, although left-permitting grammars are similar to permitting grammars, it is an open problem whether a similar relation holds for left-permitting grammars with and without erasing productions. Note that the proof from [16] cannot be directly applied to the case of left-permitting grammars because it uses the following property of permitting grammars that does not hold for left-permitting grammars: for any permitting production $(A \to w, U)$, if the production can be used to one of the occurrences of A, then it can be used to any occurrence of A in the sentential form.

Finally, given two families of grammars generating the same family of languages, an interesting question is whether or when it is also the case that the language families generated by CD grammar systems, using these two types of grammars as components, also coincide. From the results mentioned above, we can immediately see that, e.g., although $\mathbf{CF} = \ell\mathbf{FOR} = \ell\mathbf{FOR}_\lambda$, we have that for any $n \geq 2$, $\mathbf{CD}_t(\mathbf{CF}, n) \subset \mathbf{CD}_t(\ell\mathbf{FOR}, n) \subset \mathbf{CD}_t(\ell\mathbf{FOR}_\lambda, n)$. Similarly, although $\mathbf{PER} = \mathbf{PER}_\lambda$, the proper inclusion $\mathbf{CD}_t(\mathbf{PER}, n) \subset \mathbf{CD}_t(\mathbf{PER}_\lambda, n)$ holds for any $n \geq 2$. On the other hand, it is obvious that the equality of language families generated by CD grammar systems with different types of components does not imply that the language families generated by grammars of these types coincide. Thus, the question when the same power of components implies the same power of CD

grammar systems is open. Moreover, note that the discussion in Section 3.1 can also be considered in this way.

Acknowledgments

The authors gratefully acknowledge very useful suggestions and comments of the anonymous referees.

The research by the first author has been supported by the MŠMT Research Plan no. MSM0021630528, and by the MŠMT grants 2C06008 and MEB041003. The research by the second author has partially been supported by the Czech Academy of Sciences, Institutional Research Plan no. AV0Z10190503, and by the GAČR grant no. 202/11/P028.

References

[1] Bordihn, H. and Fernau, H. Accepting grammars and systems: An overview. In *Developments in Language Theory*, pages 199–208, 1995. Technical Report 9/94, Universität Karlsruhe, Fakultät für Informatik, 1994.

[2] Csuhaj-Varjú, E. and Dassow, J. On cooperating/distributed grammar systems. *Journal of Information Processing and Cybernetics (EIK)*, 26(1-2):49–63, 1990.

[3] Csuhaj-Varjú, E., Dassow, J., Kelemen, J., and Păun, Gh. *Grammar Systems: A Grammatical Approach to Distribution and Cooperation*. Gordon and Breach Science Publishers, Topics in Computer Mathematics 5, Yverdon, 1994.

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

[5] Dassow, J. and Masopust, T. On restricted context-free grammars. In *Developments in Language Theory*, pages 434–435, 2010.

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

[7] Ewert, S. and van der Walt, A. P. J. A pumping lemma for random permitting context languages. *Theoretical Computer Science*, 270(1–2):959–967, 2002.

[8] Goldefus, F., Masopust, T., and Meduna, A. Left-forbidding cooperating distributed grammar systems. *Theoretical Computer Science*, 411(40-42):3661–3667, 2010.

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

[10] Mihalache, V. Programmed grammar systems. In *Developments in Language Theory*, pages 430–437, 1993.

[11] Păun, Gh. A variant of random context grammars: Semi-conditional grammars. *Theoretical Computer Science*, 41:1–17, 1985.

[12] Rozenberg, G. and Salomaa, A., editors. *Handbook of Formal Languages*, volume 1–3. Springer, Berlin, 1997.

[13] Salomaa, A. *Formal languages*. Academic Press, New York, 1973.

[14] van der Walt, A. P. J. Random context grammars. In *Proceedings of the Symposium on Formal Languages*, pages 163–165, 1970.

[15] van der Walt, A. P. J. and Ewert, S. A shrinking lemma for random forbidding context languages. *Theoretical Computer Science*, 237(1-2):149–158, 2000.

[16] Zetzsche, G. On erasing productions in random context grammars. In *International Colloquium on Automata, Languages and Programming (2)*, pages 175–186, 2010.

Received 24th June 2010