Acta Cybernetica, Vol. 11, No. 3, Szeged, 1994

Structuring grammar systems by priorities and hierarchies*

Victor Mitrana † Gheorghe Păun‡ Grzegorz Rozenberg §

Abstract

A grammar system is a finite set of grammars that cooperate to generate a language. We consider two generalizations of grammar systems: (1) adding a priority relation between single grammar components, and (2) considering hierarchical components which by themselves are grammar systems. The generative power of these generalized grammar systems is investigated, and compared with the generative power of ordinary grammar systems and of some well-known types of grammars with regulated rewriting (such as matrix grammars). We prove that for many cooperating strategies the use of priority relation increases the generative capacity, however this is not the case for the maximal mode of derivation (an important case, because it gives a characterization of the ETOL languages). We also demonstrate that in many cases the use of hierarchical components does not increase the generative power.

1 Introduction

A cooperating grammar system (introduced in [7], and motivated by considerations related to two level grammars), is a set of usual Chomsky grammars which cooperate in rewriting sentential forms. In [7] a component that is currently rewriting a sentential form cannot quit until it introduces a symbol which it cannot rewrite (the current sentential form is not a sentential form of this component). Only one component at a time rewrites a sentential form. The set of terminal strings obtained in

*Research supported by project 11281 of the Academy of Finland, the Basic Research ASMICS II Working Group, and, in the case of the second author, also by the Alexander von Humboldt Foundation.

†University of Bucharest, Department of Mathematics Str. Academiei 14, 70109 Bucureşti, Romania

‡Institute of Mathematics of the Romanian Academy of Sciences P.O.Box 1 – 764, 70700 Bucureşti, Romania

§University of Leiden, Department of Computer Science Niels Bohrweg 1, 2333 CA Leiden, The Netherlands and Department of Computer Science, University of Colorado at Boulder Boulder, CO 80309, USA

this way is the language generated by the system. It is shown in [7] that this type of cooperating grammar systems (equiped with a control over the sequencing of the individual components) generates the family of programmed languages (which is equal to the family of languages generated by matrix grammars).

The cooperating grammar systems were rediscovered in [1], under the name of *modular grammars* (a term related to the time varying grammars). A rather intensive study of cooperating grammar systems has been initiated in [2], where the grammar systems were related to the notions from artificial intelligence, such as the blackboard model in problem solving [9]. (See also Chapter 1 of [3] for further links between grammar systems and topics in artificial intelligence, computer science, and cognitive psychology.) Within this framework, more conditions on enabling and disabling of individual components were considered. Two, quite basic, examples of this type are: the step limitations (a component must work exactly, or at least, or at most a given prescribed number of steps), and the maximal competence strategy (a component must work as long as it can) – this is similar in some extent to the stoping condition from [7]. The latter strategy is particularly interesting, because it yields a characterization of the family of ETOL languages.

A number of novel cooperating strategies has been considered recently – forming the *teams* of components, as in [6] and [9], is one of such strategies.

In this paper we consider two quite natural modifications of the basic model. The first of these is adding a *priority relation* between the components of a system. A component can become active only when no other component with a greater priority can rewrite the current string. The other modification consists of allowing components which by themselves are grammar systems, or systems of grammar systems, etc.

We demonstrate that neither of the two modifications increases the generative capacity when maximal competence strategy is used. For the other strategies, adding the priority relation strictly increases the generative power.

We end this section by pointing out that both modifications of grammar systems we consider in this paper, viz. priorities and hierarchies, are very natural. Adding priorities in rewriting systems in order to ensure the deterministic applicability of rules is a rather standard mechanism – e.g. it is used in regulated rewriting in context-free grammars and in term rewriting systems. Also, the way that a computation in a grammar system is defined on the base of computations of basic units (grammars) may be seen as just a specific cooperation mechanism. In order to understand its power, it is natural to consider the bootstrapping of this mechanism

- take grammar systems as basic units and obtain "grammar systems of depth 2" by organizing their work together by a given cooperation mechanism,

and proceeding inductively

- take grammar systems of depth $i \geq 2$ and organize their work together by a given cooperation mechanism obtaining "grammar systems of depth $i+1$".

Then a way to understand a given cooperation mechanism as defined in grammar systems is to investigate the relationship between the generative power of grammar systems of different depth. This leads one then to hierarchical grammar systems.

2 Basic definitions

For an alphabet V, V^* denotes the free monoid generated by V; the empty string is denoted by λ, and $|x|$ denotes the length of $x \in V^*$. The families of context-free, context-sensitive and recursively enumerable languages are denoted by CF, CS, and RE, respectively; $ET0L$ denotes the family of ET0L languages.

A *matrix grammar* is a construct $G = (N, T, S, M, F)$, where N, T are disjoint alphabets, $S \in N, M$ is a finite set of sequences, called *matrices*, $(A_1 \to x_1, \ldots, A_n \to x_n), n \geq 1$, of context-free rules over $N \cup T$, and F is a set of occurrences of rules in matrices of M.

For $m = (A_1 \to x_1, \ldots, A_n \to x_n) \in M$, and $w, w' \in (N \cup T)^*$, we define $w \Longrightarrow_m w'$ iff there are $w_1, w_2, \ldots, w_{n+1}$ in $(N \cup T)^*$ such that $w = w_1, w' = w_{n+1}$, and for each $i, 1 \leq i \leq n$, either $w_i = w_i' A_i w_i'', w_{i+1} = w_i' x_i w_i''$, or A_i does not occur in $w_i, w_{i+1} = w_i$ and $A_i \to x_i$ appears in F.

If $F = \emptyset$, then the grammar is said to be without *appearance checking* (and the component F is omitted from the specification of G).

We denote by MAT_{ac} (respectively, MAT_{ac}^{λ}) the family of languages generated by λ-free (arbitrary) matrix grammars; when the appearance checking feature is not present we remove the subscript ac.

A (context-free) *ordered grammar* is a construct $G = (N, T, S, P, \succ)$, where N, T, S, P are as in a context-free grammar, and $\succ$ is a partial order relation over P. A rule $A \to x$ in P can be used for rewriting a string w only if no rule $B \to y$ in P with $B \to y \succ A \to x$ can rewrite the string w. The family of languages generated by λ-free ordered grammars is denoted by ORD, and ORD^{λ} is used for the case when λ-rules are allowed.

It is known that

$$CF \subset MAT \subset MAT_{ac} \subset CS,$$
$$MAT \subset MAT^{\lambda} \subset MAT_{ac}^{\lambda} = RE,$$
$$CF \subset ET0L \subset ORD \subset MAT_{ac}.$$

For the basic elements of formal language theory the reader is referred to [11]; for Lindenmayer systems we refer to [10] and for regulated rewriting to [4].

Definition 1 *A cooperating distributed (cd, for short) grammar system is a construct*

$$\Gamma = (N, T, S, P_1, P_2, \ldots, P_n),$$

where N, T are disjoint alphabets, $S \in N$, and $P_i, 1 \leq i \leq n$, are finite sets of context-free rules over $N \cup T$.

The sets P_i are called the *components* of Γ; we also say that Γ is a cd grammar system *of degree n*.

For a component P_i from a grammar system Γ as above, $dom(P_i) = \{A \in N \mid A \to x \in P_i\}$, and we define the derivation relation $\Longrightarrow_{P_i}$ in the usual way. Then we can consider derivations in P_i of exactly k successive steps, of at least k steps, at most k steps, and of an arbitrary number of steps; they are denoted by $\Longrightarrow_{P_i}^{=k}, \Longrightarrow_{P_i}^{\geq k}, \Longrightarrow_{P_i}^{\leq k}$, and $\Longrightarrow_{P_i}^{*}$, respectively. Another important relation is

$$x \Longrightarrow_{P_i}^{t} y \text{ iff } x \Longrightarrow_{P_i}^{*} y \text{ and there is no } z \in (N \cup T)^* \text{ such that } y \Longrightarrow_{P_i} z$$

(the derivation is maximal in the component P_i).

In this way we have specified stop conditions for the components, i.e. conditions under which an active component must/can become inactive.

For $f \in \{*, t\} \cup \{\leq k, = k, \geq k \mid k \geq 1\}$ the language generated by Γ in the f mode is defined by

$$L_f(\Gamma) = \{x \in T^* \mid S \Longrightarrow^f_{P_{i_1}} x_1 \Longrightarrow^f_{P_{i_2}} x_2 \Longrightarrow^f_{P_{i_3}} \dots \Longrightarrow^f_{P_{i_r}} x_r = x, \\ r \geq 1, 1 \leq i_j \leq n, 1 \leq j \leq r\}.$$

The family of such languages, generated by systems with at most n components (all of them without λ-rules) is denoted by $CD_n(f)$ (if λ-rules are allowed, then we write $CD_n^\lambda(f)$). The union of the families $CD_n(f)$ for all n is denoted by $CD_\infty(f)$.

In [2] and [3] it is proved that:

$$CF = CD_\infty(= 1) = CD_\infty(\geq 1) = CD_\infty(*) = CD_\infty(\leq k),\ k \geq 1,$$
$$CF \subset CD_n(= k) \cap CD_n(\geq k),\ n \geq 2, k \geq 2,$$
$$CD_\infty(= k) \subseteq MAT,\ CD_\infty(\geq k) \subseteq MAT,\ k \geq 1,$$
$$CF = CD_1(t) = CD_2(t) \subset CD_n(t) = ET0L$$
(hence also $CD_n^\lambda(t) = ET0L$), $n \geq 3$.

3 Introducing orderings and hierarchies into grammar systems

We introduce now new classes of grammar systems which will be investigated in this paper.

Definition 2 *A grammar system with priorities (pcd grammar system) is a construct* $\Gamma = (N, T, S, P_1, \dots, P_n, \succ)$, *where* $N, T, S, P_1, \dots, P_n$ *are as in a cd grammar system, and* $\succ$ *is a partial order relation over the set of components. For a derivation mode* f, *two strings* $x, y \in (N \cup T)^*$, *and a component* P_i *of* Γ *we write* $x \Longrightarrow^{(f,\succ)}_{P_i} y$ *if and only if* $x \Longrightarrow^f_{P_i} y$ *and for no component* P_j *with* $P_j \succ P_i$ *and no string* $z \in (N \cup T)^*$, $x \Longrightarrow^f_{P_j} z$ *holds.*

Note that if $x \Longrightarrow^f_{P_i} y$, then no P_j with $P_j \succ P_i$ can rewrite x in the f mode – but there may be P_j with $P_j \succ P_i$ that can rewrite x in some way (e.g. P_j can make only one rewriting step on x while $f = \text{"} \geq 2\text{"}$).

We denote by $PCD_n(f)$ the family of languages generated by (λ-free) pcd grammar systems of degree at most n in the derivation mode f. Again, we add the superscript λ when also λ-rules may be used, and we replace n with ∞ when the degree is not bounded.

Here is an **example** of a pcd gramar system. Let

$$\begin{aligned}\Gamma = (&\{S, A, B, A', B', A'', B''\}, \{a, b, c\}, S, P_1, P_2, P_3, P_4, P_5, \succ),\\ &P_1 = \{A \to aA'b,\ B \to cB'\},\\ &P_2 = \{A \to A'',\ B \to B''\},\\ &P_3 = \{A' \to A,\ B' \to B,\ A'' \to ab,\ B'' \to c\},\\ &P_4 = \{A' \to A',\ B' \to B',\ A'' \to A'',\ B'' \to B''\},\\ &P_5 = \{A \to A,\ B \to B,\ S \to AB\},\\ \text{and}\quad &P_4 \succ P_1,\ P_4 \succ P_2,\ P_5 \succ P_3.\end{aligned}$$

Then

$$L_f(\Gamma) = \{a^n b^n c^n \mid n \geq 1\},$$

for all $f \in \{*, \geq 1\} \cup \{\leq k \mid k \geq 2\}$ (and also for $f \in \{= 2, \geq 2\}$).

Indeed, take a string $a^n A b^n c^n B, n \geq 0$; after using P_5, the component in which we must start any derivation, we have $n = 0$. We can apply either P_1 or P_2, using only one or both rules from each of these components. If we use only one rule, then we obtain either $a^{n+1}A'b^{n+1}c^nB$ or $a^nAb^nc^{n+1}B'$ when using P_1, and we obtain either $a^nA''b^nc^nB$ or $a^nAb^nc^nB''$ when using P_2. In all cases, both P_4 and P_5 can be used afterwards (and one of them has to be used, because they have the priority over P_1, P_2, P_3). However, nothing changes then in the current string, and so the derivation is blocked. Consequently, when using P_1, P_2 we must use both rules from each of them, thus obtaining either $a^{n+1}A'b^{n+1}c^{n+1}B'$ or $a^nA''b^nc^nB''$. Now P_4 is applicable and it changes nothing, but it does not forbid the use of P_3 (P_5 is not applicable). If, using P_3, only one of A', B' in $a^{n+1}A'b^{n+1}c^{n+1}B'$ is replaced by A, B, respectively, then again the derivation is blocked in the components P_4, P_5, hence we must produce $a^{n+1}Ab^{n+1}c^{n+1}B$ – this is a string of the form that we have started with, hence the derivation can be iterated. If from $a^nA''b^nc^nB''$ we produce either $a^{n+1}b^{n+1}c^nB''$ or $a^nA''b^nc^{n+1}$, then the only applicable components are P_3 and P_4; P_4 changes nothing, hence we eventually will use P_3 again, and get in this way a terminal string $a^{n+1}b^{n+1}c^{n+1}$.

Definition 3 *A hierarchical grammar system (hcd grammar system) of depth* $h, h \geq 0$, *is*

1. *a context-free grammar* $\Gamma = (N, T, S, P)$ *if* $h = 0$,
2. *a construct* $\Gamma = (N, T, S, \gamma_1, \gamma_2, \ldots, \gamma_m)$, $m \geq 1$, *if* $h \geq 1$, *where* $\Gamma_i = (N, T, S, \gamma_i), 1 \leq i \leq m$, *are grammar systems of depth* $h - 1$.

Thus, at the bottom level of a hcd grammar system we have sets of context-free rules, on the next level it contains sets of such sets, then sets of sets of sets and so on. The systems $\gamma_1, \ldots, \gamma_m$ from the specification of Γ in point 2 of the above definition are called *components* or *subsystems of* Γ *of depth* $h - 1$.

Here is an **example** of a hcd grammar system of depth 2:

$$\begin{aligned}&\text{level two}: \Gamma = (\{S, A, B, A', B'\}, \{a, b, c\}, S, \gamma_1, \gamma_2),\\ &\text{level one}: \gamma_1 = \{\gamma_{1,1}, \gamma_{1,2}\},\\ &\qquad\qquad \gamma_2 = \{\gamma_{2,1}\},\end{aligned}$$

$$\text{level zero} : \gamma_{1,1} = \{A \to aA'b,\ B \to cB'\},$$
$$\gamma_{1,2} = \{A' \to A,\ B' \to B\},$$
$$\gamma_{2,1} = \{S \to AB,\ A \to A,\ A \to ab,\ B \to c\}.$$

We known how to define a derivation step in a system of depth 0 (this is a usual derivation step in a context-free grammar), and we know how to define the derivation modes $\Longrightarrow_P^f$, for $f \in \{*, t\} \cup \{\leq k, = k, \geq k \mid k \geq 1\}$ in a set P of rules. Then, for a system of depth $h \geq 2$, $\Gamma = (N, T, S, \gamma_1, \ldots, \gamma_m)$ we define, for the component γ_j, $1 \leq j \leq m$,

$$x \Longrightarrow_{\gamma_j}^{=k} y \text{ iff } x \Longrightarrow_{\gamma_{j,i_1}}^{=k} x_1 \Longrightarrow_{\gamma_{j,i_2}}^{=k} \ldots \Longrightarrow_{\gamma_{j,i_k}}^{=k} x_k = y,$$
$$\gamma_{j,i_r}, 1 \leq r \leq k, \text{ are components of } \gamma_j;$$
$$x \Longrightarrow_{\gamma_j}^{\leq k} y \text{ iff } x \Longrightarrow_{\gamma_{j,i_1}}^{\leq k} \Longrightarrow_{\gamma_{j,i_2}}^{\leq k} \ldots \Longrightarrow_{\gamma_{j,i_s}}^{\leq k} x_s = y,$$
$$\gamma_{j,i_r}, 1 \leq r \leq s, \text{ are components of } \gamma_j, r \leq k,$$
$$x \Longrightarrow_{\gamma_j}^{\geq k} y \text{ iff } x \Longrightarrow_{\gamma_{j,i_1}}^{\geq k} \Longrightarrow_{\gamma_{j,i_2}}^{\geq k} \ldots \Longrightarrow_{\gamma_{j,i_s}}^{\geq k} x_s = y,$$
$$\gamma_{j,i_r}, 1 \leq r \leq s, \text{ are components of } \gamma_j, r \geq k,$$
$$x \Longrightarrow_{\gamma_j}^{*} y \text{ iff } x \Longrightarrow_{\gamma_{j,i_1}}^{*} \Longrightarrow_{\gamma_{j,i_2}}^{*} \ldots \Longrightarrow_{\gamma_{j,i_s}}^{*} x_s = y,$$
$$\gamma_{j,i_r}, 1 \leq r \leq s, \text{ are components of } \gamma_j, r \geq 0,$$
$$x \Longrightarrow_{\gamma_j}^{t} y \text{ iff } x \Longrightarrow_{\gamma_j}^{*} y \text{ and there is no } z \in (N \cup T)^*$$
$$\text{such that } y \Longrightarrow_{\gamma_j}^{=1} z.$$

Continuing the previous **example**, let us consider the $= 2$ derivation mode. Starting from S, we must use γ_2, which contains only one subsystem, hence

$$S \Longrightarrow_{\gamma_2}^{=2} x \text{ means } S \Longrightarrow_{\gamma_{2,1}}^{=2} x_1 \Longrightarrow_{\gamma_{2,1}}^{=2} x.$$

Hence after using $S \to AB$ and $A \to A$ (three times) we obtain $x = AB$. Now γ_1 must be applied, that is we must find a derivation

$$AB \Longrightarrow_{\gamma_{1,i}}^{=2} y_1 \Longrightarrow_{\gamma_{1,j}}^{=2} y_2,$$

for $i, j \in \{1, 2\}$. The only possibility is $i = 1, j = 2$, hence we get

$$AB \Longrightarrow_{\gamma_1}^{=2} aAbcB, \text{ because } AB \Longrightarrow_{\gamma_{1,1}}^{=2} aA'bcB' \Longrightarrow_{\gamma_{1,2}}^{=2} aAbcB.$$

This step can be iterated, obtaining $a^n Ab^n c^n B, n \geq 0$, and then γ_2 can be used for replacing A, B with ab, c, respectively. If the current string contains only one nonterminal, then γ_1 cannot be applied, hence after using γ_2 either a nonterminal string as above is produced or a terminal string must be obtained. It is easy to see that the generated language is

$$L_{=2}(\Gamma) = \{a^n b^n c^n \mid n \geq 1\}.$$

We denote by $H_h CD(f)$ the family of languages generated by grammar systems of depth at most $h, h \geq 1$, in the derivation mode f; we also set $H_0 CD(f) = CF$, for all f.

4 The generative power of grammar systems with priorities

In this section we will consider the effect of adding a priority relation on the generative power of grammar systems.

The next results follow directly from the definitions.

Lemma 1 $CD_n(f) \subseteq PCD_n(f), PCD_n(f) \subseteq PCD_{n+1}(f), n \geq 1$, *for all* $f \in \{*, t\} \cup \{\leq k, = k, \geq k \mid k \geq 1\}$.

The example from the end of the previous section implies that $PCD_n(f) - CF \neq \emptyset$, for $n \geq 5, f \in \{*, \geq 1\} \cup \{\leq k \mid k \geq 2\}$. Since $CD_n(f) = CF$, for all $n \geq 1$, and f as above (see the end of Section 2), this demonstrate that adding priorities strictly increases the generative power. This result can be extended also to modes of derivation other than t.

Theorem 1 $PCD_n(f) - CD_\infty(f) \neq \emptyset$, $n \geq 10, f \in \{*\} \cup \{\leq k, = k, \geq k \mid k \geq 1\}$.

Proof. Consider the system

$$\Gamma = (\{S, A, A', B, B', B''\}, \{a\}, S, P_1, P_2, \ldots, P_{10}, \succ),$$

with the components and the priority relations given in the following figure, where the components P_i, P_j are in relation $P_i \succ P_j$ iff P_i is placed above P_j in one of the "composite boxes" below:

$P_1 : \begin{matrix} B' \to B' \\ B'' \to B'' \end{matrix}$	$P_3 : B \to B$	$P_5 : \begin{matrix} S \to AB \\ A \to A \end{matrix}$
$P_2 : \begin{matrix} A \to A \\ A \to A'A' \end{matrix}$	$P_4 : \begin{matrix} A' \to A' \\ A' \to A \end{matrix}$	$P_6 : \begin{matrix} B \to B \\ B \to B' \\ B'' \to B'' \\ B'' \to a \end{matrix}$
$P_7 : A' \to A'$	$P_9 : \begin{matrix} B \to B \\ B' \to B' \end{matrix}$	
$P_8 : \begin{matrix} B' \to B' \\ B' \to B \\ B' \to B'' \end{matrix}$	$P_{10} : \begin{matrix} A \to A \\ A \to a \end{matrix}$	

Notice first that the components P_1, P_3, P_7, P_9 consist of rules of the form $X \to X$ only, hence their application does not change the current string. The same is true for P_5, except for the first step of a derivation, because S never appears later in a sentential form. Therefore, all components P_1, P_3, P_7, P_9 (as well as P_5 after the first step) check the appearance of the corresponding nonterminals and block the components P_2, P_4, P_8, P_{10} (and P_6), respectively. For this reason we will call P_1, P_3, P_5, P_7, P_9 the *control components* and $P_2, P_4, P_6, P_8, P_{10}$ the *rewriting components*.

The derivation starts in P_5 by producing the string AB (if we have a derivation mode $= k$ or $\geq k$ for $k \geq 2$, then we can use $k-1$ times the rule $A \to A$; this is true for all rewriting components, because they contain rules of the form $X \to X$, which do not modify the current string). Assume then that we have already generated a string $A^n B, n \geq 1$. The presence of the rules $A \to A$ and $B \to B$ in P_5 and P_9 forbides the use of components P_6 and P_{10}; P_4 and P_8 are not applicable to A and B. Thus P_2 is the only component which changes the current string. The obtained string will contain occurrences of both A and A' (and of B). Due to the presence of A we cannot use P_6, and due to the presence of B we cannot use P_4 and P_{10}; P_8 is not applicable. Therefore we must again use P_2 until all occurrences of A are replaced by A'. The so obtained string is of the form $A'^{2^n}B$. Now the only applicable component which changes the string is P_6, and its use leads to a string of the form $A'^{2^n}B'$, which allows the use of P_4 (and only of P_4, with the exception of control components like P_7 and P_9 which do not change the string under rewriting) which replaces occurrences of A' by A. As long as A, A' and B' are present, the only possibility is to continue to apply P_4 until each A' is replaced by A, obtaining in this way $A^{2^n}B'$. Now one can apply P_8 (and only P_8 with the exception of P_1, P_5, P_9 which do not change the string under rewriting). If $A^{2^n}B$, is obtained, then the above process can be iterated. If $A^{2^n}B''$ is obtained, then the only applicable component (which changes the string) is P_{10}; it must be then used until each A is replaced by a. When A is not present anymore, one can use P_6, finishing the derivation by replacing B'' with a.

Consequently,

$$L_f(\Gamma) = \{a^{2^n+1} \mid n \geq 1\}.$$

Since $L_f(\Gamma)$ is not context-free, it is not in $CD_\infty(f)$, for $f \in \{*, = 1, \geq 1\} \cup \{\leq k \mid k \geq 1\}$. Moreover, it is proved in [5] that the length set of every infinite language in $CD_\infty(f)$, for $f \in \{= k, \geq k \mid k \geq 1\}$, contains an infinite arithmetical progression. This implies that $L_f(\Gamma)$ is not in $CD_\infty(f)$, for $f \in \{= k, \geq k \mid k \geq 1\}$, which concludes the proof.

□

Our proof of the above theorem holds for $n \geq 10$. The question: "what is the smallest n for which Theorem 1 holds ?" remains *open*. Of course, the equalities $PCD_1(f) = CD_1(f) = CF$ are true for all f. Moreover, $PCD_2(= 1) \subseteq CF$. Indeed, for $\Gamma = (N, T, S, P_1, P_2, \succ)$ with $P_1 \succ P_2$ (the same argument holds for $P_2 \succ P_1$) we may assume that $dom(P_1) \cap dom(P_2) = \emptyset$ (the rules $A \to x \in P_2$ with $A \in dom(P_1)$ can never be used, hence they can be eliminated). Thus $L_{=1}(\Gamma) = L(G)$ for $G = (N, T, S, P_1 \cup P_2)$ (the derivations in G and in Γ are the same up to a change of the order of using the rules).

The above language $\{a^{2^n+1} \mid n \geq 1\}$ is probably not in the family MAT (it is conjectured already in [11] that the one-letter matrix languages are regular). Since $CD_\infty(f) \subseteq MAT$ for all f as in Theorem 1 (and in some cases, $CD_\infty(f) = CF$), the increase in generative power by adding priorities is quite considerable for those derivation modes. Hence it is somewhat surprising that for the t mode of derivation adding a priority relation does not increase the generative power.

Theorem 2 $PCD_\infty(t) = CD_\infty(t)$.

Proof. We have to prove only the inclusion $\subseteq$.

For a pcd grammar system $\Gamma = (N, T, S, P_1, \ldots, P_n, \succ)$, we construct the cd grammar system Γ' as follows.

$$\Gamma' = (N', T, S', P_0, P_1', P_1'', P_2', P_2'', \ldots, P_n', P_n'', P_{n+1}),$$

$$N' = N \cup \{S, X, \#\} \cup \{X_i \mid 1 \leq i \leq n\},$$
$$P_0 = \{S' \to SX\} \cup \{X_i \to X \mid 1 \leq i \leq n\},$$
$$P_i' = P_i \cup \{X \to \#\} \cup \{X_j \to \# \mid 1 \leq j \leq n, j \neq i\}, 1 \leq i \leq n,$$
$$P_i'' = \{X \to X_i\} \cup \{A \to \# \mid B \in dom(P_j), P_j \succ P_i, 1 \leq j \leq n\}, 1 \leq i \leq n,$$
$$P_{n+1} = \{X \to \lambda\} \cup \{A \to \# \mid A \in N\}.$$

Once introduced in a sentential form, the symbol # cannot be removed (it is a "trap-symbol"). The symbols $X_1, \ldots, X_n$ identify the components $P_1, \ldots, P_n$ of Γ. In the presence of X_i the component P_i will be simulated by P_i' and X_i can appear (introduced by P_i'') only when no component P_j with $P_j \succ P_i$ is applicable to the current string.

Let us see how these principles work in Γ' by examining in some detail a derivation. Consider a sentential form wX (initially we have $w = S$, obtained after using P_0, which is the only component which can be applied to S'). The component P_{n+1} can be used only if $w \in T^*$ – hence only as the final step of the derivation. A component P_i' introduces the trap-symbol #. If P_i is maximal with respect to the relation $\succ$ among the components which can be applied to w, then P_i'' can be used without blocking the derivation; it changes X into X_i, thus leading to wX_i. Now to a string wX_i we can apply either P_0, replacing again X_i with X (hence not achieving anything) or the component P_i', which will simulate the application of P_i to w. The string $w'X_i$ obtained in this way can be rewritten only by P_0, which leads to $w'X$, and so the process can be iterated. In the presence of X_i, every component $P_j', j \neq i$, will introduce the trap-symbol. Consequently, $L_t(\Gamma) = L_t(\Gamma')$. (Note that the λ-rule in P_{n+1} causes no problem, because $CD_\infty(t) = CD_\infty^\lambda(t) = ET0L$.) □

Let us return to families $PCD_\infty(f)$ for $f \neq t$. It is quite natural to compare these families with ORD, the family of languages generated by ordered grammars. Given an ordered grammar $G = (N, T, S, P, \succ)$, it is obvious that we have $L(G) = L_{=1}(\Gamma) = L_{\leq 1}(\Gamma)$ where Γ is a pcd grammar system obtained by considering each rule of P as a separate component and the relation $\succ$ defined as in G. Therefore $ORD \subseteq PCD_\infty(= 1) = PCD_\infty(\leq 1)$. This implies that the families $PCD_\infty(f)$, $f \in \{= 1, \leq 1\}$, strictly include $ET0L$ (and hence $CD_\infty(t)$).

A similar result is obtained for the $= k$ and $\leq k$ modes of derivation for all $k \geq 1$.

Theorem 3 $ORD \subseteq PCD_\infty(f), f \in \{\leq k, = k \mid k \geq 1\}$.

Proof. For $k = 1$ the statement follows by the argument as above. Consider $k \geq 2$. Let $G = (N, T, S, P, \succ)$ be an ordered grammar with

$$P = \{r_1, \ldots r_n\},\ r_i : A_i \to x_i,\ 1 \leq i \leq n,\ n \geq 1.$$

We construct the pcd grammar system

$$\Gamma = (N', T, S, P_0, P_1, P_2, \ldots, P_n, \succ),$$

where

$$N' = N \cup \{A_{i,j} \mid 1 \leq i \leq n, 1 \leq j \leq k-1\},$$
$$P_0 = \{A_{i,j} \to A_{i,j} \mid 1 \leq i \leq n, 1 \leq j \leq k-1\},$$
$$P_i = \{A_i \to A_{i,1}, A_{i,1} \to A_{i,2}, \ldots, A_{i,k-2} \to A_{i,k-1}, A_{i,k-1} \to x_i\},\ 1 \leq i \leq n,$$

and

$$P_0 \succ P_i \quad \text{for all } 1 \leq i \leq n,$$
$$P_i \succ P_j \quad \text{iff } r_i \succ r_j \text{ in } G.$$

Then $L(G) = L_{=k}(\Gamma) = L_{\leq k}(\Gamma)$.

Indeed, if we have a sentential form w to which P_0 can be applied, then on the one hand no other component of Γ can be used for rewriting w, while on the other hand the use of P_0 does not change the string w. Consequently, the derivation is blocked, and P_0 is a trap-component. As P_0 can be applied whenever any of the symbols in $N' - N$ is present, it follows that the components $P_i, 1 \leq i \leq n$, upon completing their derivations cannot produce strings containing symbols in $N' - N$. This implies that using a component $P_i, 1 \leq i \leq n$, in $\leq k$ or in $= k$ mode of derivation, means to use all the rules from P_i exactly once, hence to replace an occurrence of A_i first by $A_{i,1}$, then by $A_{i,2}, \ldots$, then by $A_{i,k-1}$, and finally by x_i. This is exactly the effect of using the rule $A_i \to x_i$. As the priority relation among the components P_i of Γ corresponds to the order relation among the rules of G, the equalities $L(G) = L_{=k}(\Gamma) = L_{\leq k}(\Gamma)$ follow.

□

It is an *open* question whether or not Theorem 3 holds also for the $\geq k$ mode of derivation.

We will now demonstrate that all the families $PCD_\infty(f)$ with $f \neq t$, are included in MAT_{ac}. In view of the strong generative power of matrix grammars with appearance checking this inclusion is somewhat expected, however is really cumbersome to write the detailed proof of this result. This is due to the fact that we have to check whether or not all the components greater than a given component (in the sense of the $\succ$ relation) are applicable to a given string in a specified mode of derivation. This is easy for modes $*, = 1, \geq 1, \leq k$, for all k, but much more difficult for the cases $= k, \geq k$, for $k \geq 2$, when all combinations of k rules in a component must be checked. For this reason the proof of the following theorem will be rather sketchy, but certainly containing enough information so that the interested reader may complete it to a detalied proof.

Theorem 4 $PCD_\infty(f) \subseteq MAT_{ac}$, $f \in \{*\} \cup \{\leq k, = k, \geq k \mid k \geq 1\}$.

Proof. (1) For $f \in \{*\} \cup \{\leq k \mid k \geq 1\}$, consider a system $\Gamma = (N, T, S, P_1, \ldots, P_n, \succ)$, and construct the matrix grammar

$$\begin{aligned}
G = {} & (N', T \cup \{c\}, S', M, F), \\
& N' = N \cup \{X, S', \#\} \cup \{[i,j] \mid 1 \leq i \leq n, 0 \leq j \leq k\}, \\
& M = \{(S' \to SX)\} \cup \\
& \quad \cup \{(X \to [i,0], A_1 \to \#, \ldots, A_s \to \#) \mid \{A_1, \ldots, A_s\} = \\
& \qquad \{A \in dom(P_j) \mid P_j \succ P_i, 1 \leq j \leq n\}, 1 \leq i \leq n\} \cup \\
& \quad \cup \{([i,j] \to [i,j+1], A \to x) \mid A \to x \in P_i, 1 \leq i \leq n, \\
& \qquad 0 \leq j \leq k-1\} \cup \\
& \quad \cup \{([i,j] \to X) \mid 1 \leq i \leq n, 0 \leq j \leq k\} \cup \\
& \quad \cup \{(X \to c)\}, \\
& F \text{ contains all rules } A \to \# \ (\# \text{ is a trap} - \text{symbol}).
\end{aligned}$$

We have $L(G) = L_{\leq k}(\Gamma)\{c\}$. The first component of nonterminals $[i,j]$ specifies the simulated component, while the second one counts the used rules. The symbol X is replaced by $[i,0]$, starting the simulation of P_i, only when no component P_j with $P_j \succ P_i$ can use at least one of its rules for rewriting the current sentential form. After using j rules of P_i, for some $0 \leq j \leq k$, the symbol $[i,j]$ can be replaced by X and another component of Γ can be simulated.

If all symbols $[i,j]$ are replaced by $[i]$, and no reference is made to the number of used rules, then we obtain $L(G) = L_*(\Gamma)\{c\}$. As MAT_{ac} is closed under restricted morphisms, the new symbol c can be erased, and so $L_f(\Gamma) \in MAT_{ac}$, for $f \in \{*\} \cup \{\leq k \mid k \geq 1\}$.

(2) In the case of the derivation mode $= k$, starting from $\Gamma = (N, T, S, P_1, \ldots, P_n, \succ)$ with $N = \{A_1, \ldots, A_s\}$, we shall use again the closure of the family MAT_{ac} under restricted morphisms. We construct a matrix grammar G with appearance checking working as follows. The new axiom S' introduces a string SX, where S is the axiom of Γ and X a control symbol; X or its variants will be present during all derivation steps. Moreover, for each symbol $A \in N$ we have its *copy* A_c. In order to be able to check whether a component P_j can be applied to the current string w, we introduce a copy of each nonterminal appearing in w, obtaining in this way a string w_c scattered among the symbols of w; we try to use the rules of P_j on w_c so that the original string w is not destroyed.

Here is a "sub-routine" for such a copying, called for by the control symbol X_c (here and in the matrices below, # is a trap-symbol):

$$(X_c \to X_c, A \to A'A_c), \text{ for each } A \in N,$$
$$(X_c \to X', A_1 \to \#, \ldots, A_s \to \#),$$
$$(X' \to X', A' \to A), \text{ for each } A \in N,$$
$$(X' \to X'', A'_1 \to \#, \ldots, A'_s \to \#).$$

(In the presence of X_c, each symbol $A \in N$ is replaced by $A'A_c$; when all symbols $A \in N$ have been so replaced, X_c can be replaced by X', and then in the presence of X' each A' is rewritten back to A; when this has been completed, X' is removed and the symbol X'' is introduced.)

Then, the control symbol X'' will guess a component, say P_i, to be used, by changing to X_i. Now all the components $P_j \succ P_i$ must be tested and if any of them can be used, then the derivation is blocked. This can be done as follows.

Having an ordered list $GR(P_i) = (P_{j_1}, \ldots, P_{j_{t_i}})$, of components that are "greater" than P_i, we inspect them in this order $P_{j_1}, \ldots, P_{j_{t_i}}$. If some P_{j_r} is applicable, then the derivation is blocked; if P_{j_r} is not applicable, then we pass to $P_{j_{r+1}}$. Finally when also $P_{j_{t_i}}$ is not applicable, the control symbol is changed to some Y_i, which leads to the simulation of P_i. This is done as in the $\leq k$ mode, introducing a counter which terminates the simulation of P_i when exactly k rules were used; then again the "general controller" X is introduced in order to start the simulation of another component. The derivation terminates (the control symbol, the copy symbols and their variants are replaced by the new terminal c) when no nonterminal from N is present in the current string.

Hence to complete the proof of the theorem for the $= k$ mode we have to show how to test whether or not a given component P_j is applicable in the $= k$ mode to the current string w (hence to the corresponding nonterminal string w_c containing copies of the nonterminals in w).

Consider the set P_j^k of all sequences of k rules in P_j,

$$P_j^k = \{m_1, m_2, \ldots, m_q\}, q = (card(P_j))^k,$$
$$m_l = (A_{l_1} \to x_{l_1}, \ldots, A_{l_k} \to x_{l_k}),\ A_{l_r} \to x_{l_r} \in P_j, 1 \leq r \leq k.$$

We have to check all these sequences – if at least one of them is applicable, then P_j is applicable. By appropriately modifying the control symbol, we check, one by one all the sequences $m_1, m_2, \ldots, m_q$. If some m_l is applicable, then the derivation is blocked by introducing the trap-symbol #; if m_l is not applicable, then we pass to m_{l+1}. Finally, when m_q is not applicable, then we conclude that P_j is not applicable.

We explain now the basic idea behind the checking whether or not some $m_l = (A_{l_1} \to x_{l_1}, \ldots, A_{l_k} \to x_{l_k})$ is applicable. Assume that the current string contains the control symbol $[i, j, l]$ (meaning: "for using P_i, we must be sure that $P_j \succ P_i$ is not applicable, and we will try the sequence m_l in P_j^k"). Consider the set of all sequences $C(m_l)$ associated with m_l as follows

$$C(m_l) = \{(A_{l_1} \to \alpha_{l_1}, \ldots, A_{l_k} \to \alpha_{l_k}) \mid \alpha_{l_r} \in \{x_{l_r}, \#\},$$
$$1 \leq r \leq k, \text{ and at least for one } r \text{ we have } \alpha_{l_r} = \#\}.$$

If m_l is applicable, then each sequence in $C(m_l)$, considered as a matrix with the rules $A \to \#$ used in the appearance checking manner, will introduce at least one occurrence of #. Conversely, if m_l is not applicable, then there is exactly one sequence in $C(m_l)$ which can be used without introducing the trap-symbol.

Indeed, take a rule $A_{l_r} \to x_{l_r}$. If it is applicable in m_l to the current string w, then it is also applicable in all sequences of $C(m_l)$, whether or not it is replaced by $A_{l_r} \to \#$. If it is applicable in m_l to a symbol not in w, but introduced by a previous rule $A_{l_p} \to x_{l_p}$, with x_{l_p} containing A_{l_r}, then we examine this rule, $A_{l_p} \to x_{l_p}$. If it remains unchanged in a sequence of $C(m_l)$, then it introduces A_{l_r}, hence also $A_{l_r} \to \alpha_{l_r}$ is applicable, introducing # when $\alpha_{l_r} = \#$. If it is replaced by $A_{l_p} \to \#$, then the above argument can be iterated again, considering two possible cases for A_{l_p}: either it appears in w or it is introduced by a previous rule. Since each sequence in $C(m_l)$ contains at least one rule $A_{l_g} \to \#$ whenever m_l is applicable, at least one # is introduced. When m_l is not applicable, at least one of its rules is not applicable. If we replace all not applicable rules by $A_{l_g} \to \#$, then we obtain a sequence in $C(m_l)$ which can be applied in the appearance checking mode without introducing the trap-symbol.

Consequently, for checking whether or not m_l is applicable it suffices to guess which sequence in $C(m_l)$ is applicable in the appearance checking mode (if the guessing is incorrect, then the derivation is blocked).

To this aim, the current control symbol $[i, j, l]$ is non-deterministicaly replaced by $[i, j, l; h]$, where h is the label of a sequence $(A_{l_1} \to \alpha_{l_1}, \ldots, A_{l_k} \to \alpha_{l_k})$ in $C(m_l)$. Here is the "sub-routine" for this step:

$$([i, j, l; h] \to [i, j, l; OK], (A_{l_1})_c \to (\alpha_{l_1})_c, \ldots, (A_{l_k})_c \to (\alpha_{l_k})_c),$$

where $(\alpha_{l_r})_c = \#$ if $\alpha_{l_r} = \#$ and it is obtained by replacing in x_{l_r} (whenever $\alpha_{l_r} = x_{l_r}$) all nonterminals $B \in N$ by their copies B_c and removing all the terminals; the terminal rules $B \to x$ are replaced by $B_c \to D$, where D is a special nonterminal (we do not introduce λ-rules).

Then, because the copy symbols have been altered, we replace all of them by D and for checking the next sequence in P_j^k (namely m_{l+1}) we produce a new series of copy symbols, using the following matrices:

$$
\begin{aligned}
&([i,j,l;OK] \to [i,j,l;OK], A_c \to D),\ A \in N,\\
&([i,j,l;OK] \to [i,j,l;copy], (A_1)_c \to \#, \ldots, (A_s)_c \to \#),\\
&([i,j,l;copy] \to [i,j,l;copy], A \to A', D \to A_c),\ A \in N,\\
&([i,j,l;copy] \to [i,j,l;copy], A \to A'A_c),\ A \in N,\\
&([i,j,l;copy] \to [i,j,l;copy'], A_1 \to \#, \ldots, A_s \to \#),\\
&([i,j,l;copy'] \to [i,j,l;copy'], A' \to A),\ A \in N,\\
&([i,j,l;copy'] \to [i,j,l+1], A_1' \to \#, \ldots, A_s' \to \#).
\end{aligned}
$$

In this way the new copies of nonterminals in the current string of Γ as simulated in G use "the places" of the old copies (the order is not relevant if matrices are used only for testing their applicability); new places for copies of nonterminals are introduced only when we do not have enough occurrences of the "place holder" symbol D (this is important when we pass from the simulation of P_i, which can introduce new nonterminals, to the simulation of another component). Therefore the length of the string is not increased more than by a factor of three (more exactly, for a string $x \in L_{=k}(\Gamma)$ we can obtain in $L(G)$ a string with the length less than $2|x|+1$).

We believe that the description of G given above allows one to give a formal (quite tedious) construction of a matrix grammar G with appearance checking such that $L_{=k}(\Gamma) = h(L(G))$, where $h : (T \cup \{c\})^* \longrightarrow T^*$ is a 3-bounded morphism defined by $h(a) = a$ for $a \in T$, and $h(c) = \lambda$. Consequently, $L_{=k}(\Gamma) \in MAT_{ac}$.

The modifications for the $\geq k$ mode of derivation concern only the counting of rules used in P_i whenever the use of P_i is permitted. (A component $P_j \succ P_i$ is applicable in the $\geq k$ mode if and only if it is applicable in the $= k$ mode, hence the "checking part" of the construction from the above proof remains unchanged.) □

5 The power of hierarchical grammar systems

We begin by pointing out the relations which follow directly from definitions:

Lemma 2 $CF = H_0CD(f) \subseteq H_1CD(f) = CD_\infty(f) \subseteq H_2CD(f) \subseteq H_3CD(f) \subseteq \ldots, f \in \{*,t\} \cup \{\leq k, = k, \geq k \mid k \geq 1\}$.

For many derivation modes, this hierarchy is finite.

Theorem 5 $H_hCD(t) = H_1CD(t)$, *for each* $h \geq 1$.

Proof. We only have to prove the inclusion $H_hCD(t) \subseteq H_1CD(t)$, and to this aim it suffices to show that $H_2CD(t) \subseteq H_1CD(t)$ (by induction: having a system of arbitrary depth $h \geq 2$, if its subsystems of depth $h-1$ can be reduced to systems of depth 1, then we replace them by such systems and obtain in this way a system of depth 2 equivalent with the initial one; then again using the reduction from depth 2 to depth 1, we prove the theorem).

Hence consider a system $\Gamma = (N, T, S, \gamma_1, \ldots, \gamma_m)$ of depth 2, with $\gamma_i = \{\gamma_{i,1}, \ldots, \gamma_{i,r_i}\}, r_i \geq 1$, $1 \leq i \leq m$, where $\gamma_{i,j}$ are sets of context-free rules over $N \cup T$. We construct the system Γ', of depth 1, with the nonterminal alphabet

$$N' = \{S, \#\} \cup \{[A,i] \mid A \in N, 1 \leq i \leq m\},$$

the terminal alphabet T, the axiom S', and the following components:

$$\begin{aligned}
P_1 &= \{S' \to [S,1],\ S' \to [S,2], \ldots, S' \to [S,m]\},\\
P_{i,j} &= \{[A,i] \to h_i(x) \mid A \to x \in \gamma_{i,j}\},\ 1 \leq i \leq m, 1 \leq j \leq r_i,\\
P'_{i,j} &= \{[A,i] \to [A,j] \mid A \in N - (\cup_{s=1}^{r_i} dom(\gamma_{i,s})) \cup\\
&\qquad \cup \{[A,i] \to \# \mid A \in \cup_{s=1}^{r_i} dom(\gamma_{i,s})\},\ 1 \leq i,j \leq m, i \neq j,
\end{aligned}$$

where, for each $1 \leq i \leq m$, $h_i : (N \cup T)^* \longrightarrow (N' \cup T)^*$ is the morphism defined by $h_i(A) = [A,i]$ for all $A \in N$, and $h_i(a) = a$ for all $a \in T$.

Each derivation in Γ' begins by a rule $S' \to [S,i]$, which selects a component γ_i of Γ which is simulated first. Assume we use now a component $P_{i,j}, 1 \leq j \leq r_i$. All the introduced nonterminals will be of the form $[A,i], A \in N$. The derivation will be maximal in $P_{i,j}$, hence it corresponds to a maximal derivation in $\gamma_{i,j}$. After finishing the derivation in $P_{i,j}$, another component $P_{i,s}$ for $1 \leq s \leq r_i$, can be used, and so on. At each moment, all the nonterminals present in the current string are of the form $[A,i]$, for the chosen i. When no component $P_{i,j}, 1 \leq j \leq r_i$, can be used (this corresponds to a maximal derivation in γ_i), a component $P'_{i,j}, j \neq i$, of Γ' can be used. It changes all nonterminals in the sentential form from $[A,i]$ to $[A,j]$. A component $P'_{i,j}, i \neq j$, can be used without blocking the derivation only when no derivation step in $P_{i,s}, 1 \leq s \leq r_i$, can be done, that is the corresponding derivation in γ_i is maximal (otherwise a rule $[A,i] \to \#, A \in dom(\gamma_{i,s})$, for some $1 \leq s \leq r_i$, can be used, which introduces the trap-symbol $\#$). Consequently, the terminal derivations in Γ' simulate derivations in Γ.

Conversely, it is obvious that each derivation in Γ can be simulated in Γ'.

Consequently, $L_t(\Gamma) = L_t(\Gamma')$, that is $H_2CD(t) \subseteq H_1CD(t) = CD_\infty(t)$, which concludes the proof.

□

Theorem 6 *$H_hCD(f) = H_1CD(f) = CF$, for $f \in \{*, = 1, \geq 1\} \cup \{\leq k \mid k \geq 1\}$, and $h \geq 1$.*

Proof. We proceed again as in the previous proof, reducing the problem to the inclusion $H_2CD(f) \subseteq H_1CD(f)$; because we know that $H_1CD(f) = CF$, for f as in the statement of the theorem, we shall prove the relation $H_2CD(f) \subseteq CF$.

Consider a system of depth 2, $\Gamma = (N, T, S, \gamma_1, \ldots, \gamma_m)$, with $\gamma_i = \{\gamma_{i,1}, \ldots, \gamma_{i,s_i}\}$, for each $1 \leq i \leq m$, where $\gamma_{i,j}$ is a set of context-free rules, $1 \leq j \leq s_i$. Let G be the context-free grammar $(N, T, S, \{A \to x \mid A \to x \in \gamma_{i,j}, 1 \leq i \leq m, 1 \leq j \leq s_i\})$.

Every derivation in Γ ammounts to the use of rules from sets $\gamma_{i,j}$, hence the inclusion $L_f(\Gamma) \subseteq L(G)$ is obvious (and actually holds for all modes of derivation, and not only for the modes f as in the statement of the theorem). Conversely, every derivation in G is correct with respect to the f mode in Γ, because we can reproduce all derivations in G as $= 1$ derivations in Γ. Consequently, $L(G) = L_f(\Gamma)$, that is $L_f(\Gamma) \in CF$. □

It is an *open* problem whether Theorem 6 can also be extended to the derivation modes $= k$ and $\geq k$, for $k \geq 2$. This question seems to be related to the unsolved problems about usual grammar systems concerning (1) the relations between families $CD_\infty(= k)$ and $CD_\infty(= j)$ for $k \neq j$, and (2) the strictness of the inclusions $CD_\infty(\geq k) \subseteq CD_\infty(\geq k+1)$ for $k \geq 2$ (weak inclusions are proved in [3]). In the example from Section 3 we have seen that a derivation in the $= 2$ mode at the level of the system corresponds, in some sense, to a derivation in the $= 4$ mode at the level of components: two rules from the first sub-component and two rules from the second sub-component are used.

We will demonstrate now that the result analogous to Theorem 3 holds for hcd grammar systems.

Theorem 7 *$H_hCD(f) \subseteq MAT$, for all $h \geq 0$ and for $f \in \{= k, \geq k \mid k \geq 2\}$.*

Proof. First of all notice that for each f as in the statement of the theorem, $H_0CD(f) = CF$, and $H_1CD(f) = CD(f)$ – thus (see also the end of Section 2) $H_0CD(f) \subseteq MAT$ and $H_1CD(f) \subseteq MAT$. Hence we may assume that $h \geq 2$.

Let Γ be a hcd grammar system of depth h, $\Gamma = (N, T, S, \gamma_1, \ldots, \gamma_m)$. Using a component γ_i in the $= k$ mode for $k \geq 2$, means to use k of its subsystems. This in turn means that k sub-subsystems are used, and so on until one reaches the level 0 (of sets of rules) where we use k rules from each set chosen by the previous steps. This means that from the sets P_j on the level 0 we use sequences in the sets P_j^k; then "concatenating" such sequences, we obtain sequences corresponding to the next level and so on. The so obtained sequences are matrices of rules, and so the work of Γ in the $= k$ mode can be simulated in a matrix grammar which is defined as follows.

For a sequence of matrices of context-free rules, $m_i = (r_{i,1}, \ldots, r_{i,s_i}), 1 \leq i \leq p$, we define $(m_1, \ldots, m_p) = (r_{i,1}, \ldots, r_{i,s_1}, r_{2,1}, \ldots, r_{2,s_2}, r_{3,1}, \ldots, r_{p,s_p})$, which is again a matrix of rules.

For a set P of context-free rules let $mat(P, k) = P^k$ (all matrices, in all orders and combinations, of k rules in P), and then, for a system $\delta = (N, T, S, \delta_1, \ldots, \delta_q)$ of depth $h \geq 1$, we define recursively

$$mat(\delta, k) = \{mat(\delta_1, k), mat(\delta_2, k), \ldots, mat(\delta_s, k)\}^k.$$

The matrix grammar $G = (N, T, S, mat(\Gamma, k))$ has the property $L(G) = L_{=k}(\Gamma)$, which proves the inclusion $H_hCD(= k) \subseteq MAT$.

The inclusion $H_hCD(\geq k) \subseteq MAT$ can be obtained in the same way, using the observation that every derivation in a system Γ in the mode $\geq k$ can be decomposed into one or more derivations in the mode $= j$, for $k \leq j \leq 2k-1$. Therefore, if we define now $mat'(P, k) = \cup_{j=k}^{2k-1} mat(P, j)$ and we modify in the same way the definition of $mat(\delta, k)$, then we obtain a matrix grammar G' generating the language $L_{\geq k}(\Gamma)$.

□

Note that in the above theorem we have dealt with matrix grammars without appearance checking.

References

[1] A. Atanasiu, V. Mitrana, Modular grammars, *Intern. J. Computer Math.*, 30 (1989), 101 - 122.

[2] E. Csuhaj-Varju, J. Dassow, On cooperating distributed grammar systems, *J. Inform. Process. Cybern., EIK*, 26 (1990), 49 – 63.

[3] E. Csuhaj-Varju, J. Dassow, J. Kelemen, Gh. Păun, *Grammar Systems*, Gordon and Breach, 1994.

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

[5] J. Dassow, Gh. Păun, S. Vicolov, On the generative capacity of certain classes of cooperating grammar systems, *Fundamenta Informaticae*, to appear.

[6] L. Kari, Al. Mateescu, Gh. Păun, A. Salomaa, Teams in cooperating grammar systems, *J. Experimental and Theoretical AI*, to appear.

[7] R. Meersman, G. Rozenberg, Cooperating grammar systems, *Proc. MFCS '78 Symp., LNCS 64*, Springer-Verlag, 1978, 364 – 374.

[8] P. H. Nii, Blackboard systems, in *The Handbook of AI*, vol. 4 (A. Barr, P. R. Cohen, E. A. Feigenbaum, eds.), Addison-Wesley, 1989.

[9] Gh. Păun, G. Rozenberg, Prescribed teams of grammars, *Acta Informatica*, to appear.

[10] G. Rozenberg, A. Salomaa, *The Mathematical Theory of L Systems*, Academic Press, 1980.

[11] A. Salomaa, *Formal Languages*, Academic Press, 1973.

Received October 6, 1993