Some Hierarchies for the Communication Complexity Measures of Cooperating Grammar Systems

Juraj Hromkovic[1], Jarkko Kari[2], Lila Kari[2]

Abstract. We investigate here the descriptional and the computational complexity of parallel communicating grammar systems (PCGS). A new descriptional complexity measure - the communication structure of the PCGS - is introduced and related to the communication complexity (the number of communications). Several hierarchies resulting from these complexity measures and some relations between the measures are established. The results are obtained due to the development of two lower-bound proof techniques for PCGS. The first one is a generalization of pumping lemmas from formal language theory and the second one reduces the lower bound problem for some PCGS to the proof of lower bounds on the number of reversals of certain sequential computing models.

1 Introduction

Parallel Communicating Grammar Systems (PCGS) represent one of the several attempts that have been made for finding a suitable model for parallel computing (see [4] for an algebraic and [6], [1] for an automata theoretical approach). PCGS have been introduced in [12] as a grammatical model in this aim, trying to involve as few as possible non-syntactic components.

A PCGS of degree n consists of n separate usual Chomsky grammars, working simultaneously, each of them starting from its own axiom; furthermore, each grammar i can ask from the grammar j the string generated so far. The result of this communication is that grammar i includes in its own string the string generated by grammar j, and that grammar j returns to its axiom and resumes working. One of the grammars is distinguished as a master grammar and the terminal strings generated by it constitute the language generated by the PCGS.

Many variants of PCGS can be defined, depending on the communication protocol (see [8]), on the type of the grammars involved (see [12], [9]), and so on. In [12], [9], [11], [10] and [8], [14] various properties of PCGS have been investigated, including the generative power, closure under basic operations, complexity, and efficiency. In this paper we restrict ourselves to the study of PCGS composed of *regular grammars*. As no confusion will arise, in the sequel we will use the more general term PCGS when referring to these particular PCGS consisting of regular grammars.

[1] Department of Mathematics and Computer Science, University of Paderborn, 4790 Paderborn, Germany

[2] Academy of Finland and Department of Mathematics, University of Turku, 20500 Turku, Finland, email- santean@sara.cc.utu.fi

The most investigated complexity measure for PCGS has been the number of grammars the PCGS consists of, which is clearly a descriptional complexity measure. Here we propose for investigation two further complexity measures. One is the communication structure of the PCGS (the shape of the graph consisting of the communication links between the grammars) which can be considered as an alternative descriptional complexity measure to the number of grammars. This measure may be essential for the computational power of the PCGS, as showed also by results established in this paper. Here we consider mostly the following graphs as communication structures: linear arrays, rings, trees and directed acyclic graphs. The second complexity measure proposed here is the number of communications between the grammars during the generation procedure. This measure is obviously a computational complexity measure which is considered as a function of the length of the generated word. Here we investigate these complexity measures and the relations between them.

First, in Section 3, we relate these complexity measures to some sequential complexity measures. It is shown that PCGS with tree communications structure and $f(n)$ communication complexity can be simulated in real-time by one-way nondeterministic multicounter machines with at most $2\ f(n)$ reversals. PCGS with acyclic communication structure can be simulated in linear time by nondeterministic off-line multitape Turing machines.

The first simulation result is used in Section 4 to prove some lower bounds on the communication complexity of tree-PCGS. The lower bounds are achieved due to the modification of the lower bound proof technique on the number of reversals of multicounter machines developed in [5], [2].

The consequences are not only some hierarchies of communication complexity but also the fact that for tree-PCGS the increase of descriptional complexity cannot compensate for some small decreases of communication complexity.

Section 5, devoted to descriptional complexity measures, involves pumping lemmas for PCGS with tree structure, ring structure and with acyclic structures. This enables to obtain several strong hierarchies on the number of grammars of such PCGS.

2 Definitions and Notations

We assume the reader familiar with basic definitions and notations in formal language theory (see [13]) and we specify only some notions related to PCGS.

For a vocabulary V, we denote by V^* the free monoid generated by V under the operation of concatenation, and by λ the null element. For $x \in V^*, |x|$ is the length of x and if K is a set, $|x|_K$ denotes the number of occurrences of letters of K in x.

All the grammars appearing in this paper are assumed to be regular, that is, with productions of the form $A \longrightarrow wB$, and $A \longrightarrow w$, where A, B are nonterminals and w is a terminal word or the empty word.

Definition 1 *A PCGS of degree n, $n \geq 1$, is an n-tuple*

$$\pi = (G_1, G_2, \ldots, G_n),$$

where

- $G_i = (V_{N,i}, \Sigma, S_i, P_i)$, $1 \leq i \leq n$, *are regular Chomsky grammars satisfying* $V_{N,i} \cap \Sigma = \emptyset$ *for all* $i \in \{1, 2, \ldots, n\}$;
- *there exists a set* $K \subseteq \{Q_1, Q_2, \ldots, Q_n\}$ *of special symbols, called communication symbols,* $K \subseteq \bigcup_{i=1}^{n} V_{N,i}$, *used in communications as will be shown below.*

The communication protocol in a PCGS π is determined by its *communication graph*. The vertices of this directed graph are labeled by $G_1, \ldots, G_n$. Moreover, for $i \neq j$ there exists an arc starting with G_i and ending with G_j in the communication graph iff the communication symbol Q_j belongs to the nonterminal vocabulary of G_i.

An n-tuple $(x_1, \ldots, x_n)$ where $x_i \in \Sigma^*(V_{N,i} \cup \lambda)$, $1 \leq i \leq n$, is called a *configuration*. The elements x_i, $1 \leq i \leq n$, will be called *components* of the configuration.

We say that the configuration $(x_1, \ldots, x_n)$ directly derives $(y_1, \ldots, y_n)$ and write $(x_1, \ldots, x_n) \Longrightarrow (y_1, \ldots, y_n)$, if one of the next two cases holds:

(i) $|x_i|_K = 0$ for all i, $1 \leq i \leq n$, and, for each i, either x_i contains a nonterminal and $x_i \Longrightarrow y_i$ in G_i or x_i is a terminal word and $x_i = y_i$.

(ii) $|x_i|_K > 0$ for some i, $1 \leq i \leq n$.
For each such i we write $x_i = z_i Q_j$, where $z_i \in \Sigma^*$.
(a) If $|x_j|_K = 0$ then $y_i = z_i x_j$ and $y_j = S_j$.
(b) If $|x_j|_K > 0$ then $y_i = x_i$.
For all the remaining indexes l, that is, for those indexes l, $1 \leq l \leq n$, for which x_l does not contain communication symbols and Q_l has not occurred in any of the x_i, $1 \leq i \leq n$, we put $y_l = x_l$.

Informally, an n-tuple $(x_1, x_2, \ldots, x_n)$ directly yields $(y_1, y_2, \ldots, y_n)$ if either no communication symbol appears in $x_1, \ldots, x_n$ and we have a componentwise derivation, $x_i \Longrightarrow y_i$ in G_i, for each i, $1 \leq i \leq n$, or communication symbols appear and we perform a *communication step*, as these symbols impose: each occurrence of Q_{i_j} in x_i is replaced by x_{i_j}, provided x_{i_j} does not contain further communication symbols.

A derivation consists of *rewriting steps* and *communication steps*.

The derivation relation, denoted $\Longrightarrow^*$, is the reflexive transitive closure of the relation $\Longrightarrow$. The language generated by the system consists of the terminal strings generated by the *master grammar*, G_1, regardless the other components (terminal or not).

Definition 2 $L(\pi) = \{\alpha \in \Sigma^* \mid (S_1, \ldots, S_n) \Longrightarrow^* (\alpha, \beta_2, \ldots, \beta_n)\}$.

Of special interest are the *centralized PCGS*, denoted by c-PCGS. In this case, only the master grammar can ask for the strings generated by the others. The communication graph is therefore a tree (star) consisting of a father and its sons.

Definition 3 *A dag-PCGS (tree-, two-way array-, one-way array-, two-way ring-, one-way ring- PCGS) is a PCGS whose communication graph is a directed acyclic graph (respectively tree, two-way linear array, one-way linear array, two-way ring, one-way ring).*

Denote by $x - PCGS_n$ the class of PCGS's of degree n whose communication graph is of type x, where $x \in \{$c, dag, tree, two-way array, one-way array, two-way ring, one-way ring$\}$. Moreover, denote by $\mathcal{L}(x - PCGS_n)$ the family of languages generated by $x - PCGS$'s of degree n whose communication graph is of type x, where x is as before.

If x denotes one of the above communication graphs, $x - PCGS_n(f(m))$ will denote the class of PCGS's with communication graph of shape x and using at most $f(m)$ communication steps to generate any word of length m. (Note that $0 \leq f(m) \leq m$.) As above, $\mathcal{L}(x - PCGS_n(f(m)))$ will denote the family of languages generated by PCGS of this type.

Let us give now a simple example that shows the generative power of PCGS.

Example 1. Let π be the PCGS $\pi = (G_1, G_2, G_3)$ where

$$\begin{aligned} G_1 = (\{S_1, S_1', S_2, S_3, Q_2, Q_3\}, \{a, b, c\}, S_1, \{S_1 &\longrightarrow abc, \\ S_1 &\longrightarrow a^2b^2c^2, S_1 \longrightarrow a^3b^3c^3, S_1 \longrightarrow aS_1', S_1' \longrightarrow aS_1', \\ S_1' &\longrightarrow a^3Q_2, S_2 \longrightarrow b^2Q_3, S_3 \longrightarrow c\}), \\ G_2 = (\{S_2\}, \{b\}, S_2, \{S_2 &\longrightarrow bS_2\}), \\ G_3 = (\{S_3\}, \{c\}, S_3, \{S_3 &\longrightarrow cS_3\}). \end{aligned}$$

This is a regular centralized PCGS of degree 3 and it is easy to see that we have

$$L(\pi) = \{a^n b^n c^n | n \geq 1\},$$

which is a non-context-free language. □

Let us now informally define one-way nondeterministic multicounter machines. The formal definition can be found in [3]. A multicounter machine consists of a finite state control, a one-way reading head which reads the input from the input tape, and a finite number of counters. We regard a counter as an arithmetic register containing an integer which may be positive or zero. In one step, a multicounter machine may increment or decrement a counter by 1. The action or the choice of actions of the machine is determined by the input symbol currently scanned, the state of the machine and the sign of each counter: positive or zero. A reversal is a change from increasing to decreasing contents of a counter or viceversa. The machine starts with all counters empty and accepts if it reaches a final state.

3 Characterization of PCGS by Sequential Complexity Measures

In this section we shall characterize the families of languages generated by PCGS by some sequential complexity classes. These characterizations will depend on the communication structure of PCGS and on the communication complexity of PCGS. This enables us to obtain some hierarchies for the communication complexity measures of PCGS as consequences of some hierarchies for sequential complexity measures.

Let us start first with the characterization of tree-PCGS by linear-time nondeterministic multicounter machines.

Lemma 1. *Let π be a tree-PCGS$_m(f(n))$ for some positive integer m and for some function f : N$\longrightarrow$N. Then there exists a linear-time nondeterministic $(m-1)$-counter automaton M recognizing $L(\pi)$, with 2 $f(n)$ reversals and $f(n)$ zerotests.*

Proof. Let $\pi = (G_1, \ldots, G_m)$ be a tree-PCGS$_m(f(n))$. The simulation of π by a real-time 1MC$(m-1)$ machine M is based on the following idea. The finite control of M is used to store the description of all regular grammars $G_1, \ldots, G_m$ and to simulate always the rewriting of one of the grammars which is responsible for the input part exactly scanned.

M uses its counters $C_2, C_3, \ldots, C_m$ in the following way which secures that none of the grammars $G_1, \ldots, G_m$ is used longer than possible in actual situations (configurations). In each configuration of M and for each $i \in \{2, \ldots, m\}$ the number $c(C_i)$ stored in C_i is the difference between the number of the rewriting steps of G_i already simulated by M and the number of simulated rewriting steps of the father of G_i in the communication tree (this means that if G_i is asked by its father to give its generated word then this word is generated by G_i in at most $c(C_i)$ steps).

Now let us describe the simulation. M nondeterministically simulates the work of π by using its finite control to alternatively simulate the work of $G_1, \ldots, G_m$ and checking in real-time whether the generated word is exactly the word laying on the input tape. The simulation starts by simulating the work of G_1 and with the simultaneous comparison of the generated terminals with the corresponding terminals on the input tape. During this procedure M increases after each simulated rewriting step of G_1 the content of all counters assigned to the sons of G_1 and does not change the content of any other counter. This simulation procedure ends when a communication nonterminal Q_i (for some i) is generated. Then M starts to simulate the generation procedure of G_i from the initial nonterminal of G_i. Now, in each simulation step of M the content of the counter C_i is decreased by 1 and the contents of all counters of the sons of G_i are increased by 1. If G_i rewrites its nonterminal in a terminal word, then M halts and it accepts the input word iff the whole input word has been read. If C_i is empty and G_i has produced a nonterminal A in the last step then the control is given to the father of G_i (G_1) which continues to rewrite from the nonterminal A (if A is not a nonterminal of G_1, then M rejects the input). If G_i has produced a communication symbol Q_j for some j, then the son G_j of G_i is required to continue to generate the input word. Now the simulation continues recursively as described above.

Obviously, the number of reversals is bounded by 2 $f(n)$ and the number of zerotests is bounded by $f(n)$ because the content of a counter C_i starts to be decreased iff the communication symbol Q_i was produced.

Clearly, if there are no rules $A \longrightarrow B$, where both A and B are nonterminals, then M works in real-time. If such rules may be used, then the simulation works in linear time because there exists a constant d such that for each word $w \in L(\pi)$ there exists a derivation of w which generates in each d steps at least one terminal symbol. □

Realizing the facts that each 1-multicounter-machine can be simulated in the same time by an off-line multitape Turing machine, and that the contents of counters of M from Lemma 1 is in $O(|w|)$ for any input w, we get the following result.

Theorem 2. $\mathcal{L}$(*tree-PCGS*) $\subseteq$ *NTIME*$(n) \cap$ *NSPACE*$(\log_2 n)$.

The following theorem shows that there is a simulation of dag-PCGS by an off-line nondeterministic multitape Turing machine, working in linear time.

Theorem 3. $\mathcal{L}(dag\text{-}PCGS) \subseteq NTIME(n)$.

Finally, we let open the problem whether the general PCGS can be simulated nondeterministically in linear time. Some effort in this direction has been made in [15], [7], where some PCGS with cycles in communication structures and with some additional restrictions are simulated nondeterministically in linear time.

Another interesting question is whether $\mathcal{L}(PCGS) \subseteq NLOG$. If YES, then each PCGS can be simulated deterministically in polynomial time because $NLOG \subseteq P$. We only know as a consequence of Theorem 2 that $\mathcal{L}(\text{tree-}PCGS)$ is included in P.

4 Communication Complexity Hierarchies

In this section we shall use the simulation result from Lemma 1 to get some strong hierarchies on the number of communication steps for tree-PCGS and its subclasses. Following Lemma 1 we have that $L \in \mathcal{L}(\text{tree} - PCGS_m(f(n)))$ implies $L = L(M)$ for a real-time nondeterministic $(m-1)$-counter automaton M with $2\,f(n)$ reversals. Following the proof of Lemma 1 we see that M has the following property.

(i) For any computation part D of M containing no reversal, the counters can be divided into three sets, $S_1 = \{$the counters whose contents is never changed in $D\}$, $S_2 = \{$the counters whose content is increased in $D\}$, and $S_3 = \{$the counters whose content is decreased in $D\}$, such that for each step of D one of the following conditions holds:

1. either no counter changes its content in the given step, or
2. the counters from S_1 do not change their contents, each counter in S_2 increases its content by 1, and each counter in S_3 decreases its content by 1.

So, the property (i) of D means that, for any subpart D' of D, there exists a constant d' such that the volume of the change of the content of any counter in D' is either $+d'$, or $-d'$, or 0.

Now we will use (i) to get the following result.

Let $L = \{a^{i_1}b^{i_1}a^{i_2}b^{i_2}\ldots a^{i_k}b^{i_k}c |\ k \geq 1, i_j \in \mathbf{N} \text{ for } j \in \{1,\ldots,k\}\}$.

Lemma 4. $L \in \mathcal{L}(c\text{-}PCGS_2(n)) - \cup_{m\in\mathbf{N}}\mathcal{L}(\ tree\text{-}PCGS_m(f(n)))$ *for any* $f(n) \notin \Omega(n)$.

Following Lemma 4 we get the following hierarchies on the communication complexity.

Theorem 5. *For any function* $f : \mathbf{N} \longrightarrow \mathbf{N}$, $f(n) \notin \Omega(n)$, *and any* $m \in \mathbf{N}$, $m \geq 2$:

$$\mathcal{L}(one\text{-}way\text{-}array\text{-}PCGS_m(f(n))) \subset \mathcal{L}(one\text{-}way\text{-}array\text{-}PCGS_m(n)),$$

$$\mathcal{L}(c\text{-}PCGS_m(f(n))) \subset \mathcal{L}(c\text{-}PCGS_m(n)),$$

$$\mathcal{L}(tree\text{-}PCGS_m(f(n))) \subset \mathcal{L}(tree\text{-}PCGS_m(n)).$$

Besides Theorem 5, Lemma 5 claims a more important result namely that no increase of the number of grammars and no increase of communication links in tree communication structure (i.e. no increase of the descriptional complexity under the tree communication structure) can compensate for the decrease of the number of communication steps (i.e. computational complexity).

Now we shall deal with PCGS whose communication complexity is bounded by a constant. Let

$$L_k = \{a^{i_1}b^{i_1}a^{i_2}b^{i_1+i_2}\dots a^{i_k}b^{i_1+i_2+\dots+i_k}c \mid i_j \in \mathbf{N} \text{ for } j = 1,\dots,k\},$$

for any $k \in \mathbf{N}$.

Lemma 6. $L_k \in \mathcal{L}(c\text{-}PCGS_{k+1}(k)) - \cup_{m\in\mathbf{N}}\mathcal{L}(\ tree\text{-}PCGS_m(k-1))$.

Theorem 7. *For any positive integer k and any $X \in \{c,$ tree, one-way array$\}$ we have*

$$\mathcal{L}(X\text{-}PCGS_{k+1}(k-1)) \subset \mathcal{L}(X\text{-}PCGS_{k+1}(k)) \text{ and}$$

$$\cup_{m\in\mathbf{N}}\mathcal{L}(X\text{-}PCGS_m(k-1)) \subset \cup_{m\in\mathbf{N}}\mathcal{L}(X\text{-}PCGS_m(k)).$$

An open problem is to prove hierarchies for more complicated communication structures. Some results in this direction have been recently established in [7].

5 Pumping Lemmas and Infinite Hierarchies

In this section descriptional complexity measures of PCGS are investigated. For PCGS with communication structures tree and dag, strong hierarchies on the number of grammars are proved. To obtain them, some pumping lemmas as lower bound proof techniques are established. In the case of PCGS with communication structures arrays and rings, no such pumping lemmas are known. However, the infinity of the hierarchies of such PCGS on the number of grammars is obtained as a consequence of the following stronger result. There exist languages that can be generated by two-way array-PCGS, two-way ring-PCGS and one-way ring-PCGS but cannot be generated by *any* PCGS of smaller degree, regardless of the complexity of its communication graph. This also shows that in some cases the increase in the descriptional complexity (the number of grammars the PCGS consists of) cannot be compensated by any increase in the complexity of the communication graph.

Before entering the proof of the pumping lemmas, an ordering of the vertices in a directed acyclic graph is needed.

Proposition 8. *Let $G = (X, \Gamma)$ be a dag, where X is the set of vertices and Γ the set of arcs. We can construct a function $f : X \longrightarrow \mathbf{N}$ such that for all $x, y \in X$ we have:*

$$f(x) \geq f(y) \text{ implies that there is no path from } y \text{ to } x \text{ in the graph } G.$$

The classical proof of the pumping lemma for regular languages is based on finding, along a sufficiently long derivation, two "similar" sentential forms. "Similar" means that the two sentential forms contain the same nonterminal, a fact that allows us to iterate the subderivation between them arbitrarily many times.

We will use an analogous procedure for dag-PCGS. The difference will be that, due to the communications we need a stronger notion of "similarity". The first request will obviously be that the correspondent components of the two "similar" configurations contain the same nonterminal. Moreover, we will require that, in case communications are involved, also the terminal strings are identical.

Definition 4 *Let $c_1 = (x_1A_1, \ldots, x_nA_n)$ and $c_2 = (y_1B_1, \ldots, y_nB_n)$ be two configurations where x_i, y_i are terminal strings and A_i, B_i are nonterminals or λ, for $1 \leq i \leq n$.*

The configurations are called equivalent, and we write $c_1 \equiv c_2$ if $A_i = B_i$ for each i, $1 \leq i \leq n$.

Clearly, $\equiv$ is an equivalence relation.

Let us consider a derivation according to π, $D : c \Longrightarrow^* c_1 \Longrightarrow^* c_2 \Longrightarrow^* c'$, where c_1 and c_2 are defined as in the previous definition.

Definition 5 *The configurations c_1 and c_2 are called D-similar iff*

(i) c_1 and c_2 are equivalent,

(ii) if a communication symbol Q_i, $1 \leq i \leq n$, is used in the derivation D between c_1 and c_2, then $x_i = y_i$.

We are now in position to prove the pumping lemma for dag-PCGS. For the sake of clarity, the proof is split in two parts. The first result claims that in any sufficiently long derivation according to a dag-PCGS we can find two "similar" configurations.

Lemma 9. *Let π be a dag-PCGS. There exists a constant $q \in \mathbf{N}$ such that in any derivation D according to π whose length is at least q, there are two D-similar configurations.*

The following pumping lemma shows that any sufficiently long word generated by a dag-PCGS can be decomposed such that, by simultaneously pumping a number of its subwords, we obtain words that still belong to the language. Due to the dag structure of the communication graph which allows a string to be read by more than one grammar (a vertex can have more fathers), the number of the pumped subwords can be arbitrarily large. However, the number of *distinct* pumped subwords is bounded by the degree of the dag-PCGS.

Lemma 10. (Pumping lemma for dag-PCGS) *Let L be a language generated by a dag-PCGS of degree $n > 1$. There exists a natural number N such that every word $\alpha \in L$ whose length is greater than N can be decomposed as*

$$\alpha = \alpha_1\beta_1 \ldots \alpha_m\beta_m\alpha_{m+1},$$

where $\beta_i \neq \lambda$ for every i, $1 \leq i \leq m$, and $1 \leq card\{\beta_1, \ldots, \beta_m\} \leq n$. Moreover, for all $s \geq 0$ the word

$$\alpha_1\beta_1^s \ldots \alpha_m\beta_m^s\alpha_{m+1}$$

belongs to L.

Proof. Let $\pi = (G_1, \ldots, G_n)$ be a dag-PCGS, where $G_i = (V_{N,i}, \Sigma, S_i, P_i)$. Denote by z the maximum length of the right sides of all productions.

Claim. The length of any component of a configuration produced by π starting from the axiom in k derivation steps is at most $z \cdot 2^{k-1}$.

The claim will be proved by induction on k.

If $k = 1$ then the claim obviously holds as π can produce in one step only words of length at most z.

$k \mapsto k + 1$. Let us consider a derivation according to π which starts from the axiom and has $k + 1$ steps. In the $(k + 1)$th step, the length of any component α is:

$$|\alpha| \leq |\alpha'| + \max\{z, |\alpha'|\} \leq 2 \cdot |\alpha'| = z \cdot 2^k.$$

where $|\alpha'|$ denotes the maximum length of any component of a configuration that can be obtained after k derivation steps, starting from the axiom. The proof of the claim is complete.

If we choose now $N = z \cdot 2^{q-1}$, where q is the number defined in Lemma 9 and a word α whose length is greater than N, then a minimal derivation D of α contains at least q steps.

According to the Lemma 9, during this derivation occur at least two D-similar configurations c_1 and c_2 as shown below:

$$\begin{aligned}(S_1, S_2, \ldots, S_n) &\Longrightarrow^* c_1 = (x_1 A_1, x_2 A_2, \ldots, x_n A_n) \\ &\Longrightarrow^* c_2 = (x_1 z_1 A_1, x_2 z_2 A_2, \ldots, x_n z_n A_n) \\ &\Longrightarrow^* (\alpha, \ldots\ldots).\end{aligned}$$

If all the strings $x_i z_i$ which occur in c_2 and become later subwords of α have the property $z_i = \lambda$ then D is not minimal. Indeed, if this would be the case, the subderivation between c_1 and c_2 could be eliminated – a contradiction with the minimality of D.

Consequently, there exist $i_1, \ldots, i_k \in \{1, \ldots, n\}$, such that

$$\alpha = \alpha_1 x_{i_1} z_{i_1} \alpha_2 x_{i_2} z_{i_2} \ldots \alpha_k x_{i_k} z_{i_k} \alpha_{k+1}$$

$z_{i_j} \neq \lambda$, $1 \leq j \leq k$, and $x_{i_j} z_{i_j}$, $1 \leq j \leq k$, are exactly the terminal strings that have appeared in the components with the corresponding index of c_2. Observe that we do not necessarily have $i_j \neq i_p$ for $j \neq p$, $1 \leq j, p \leq k$. Indeed, because of possible communications, the same string $x_{i_j} z_{i_j}$ originating from the i_j-component of c_2 can appear several times in α.

By iterating the subderivation between the two D-similar configurations c_1 and c_2 s times, for an arbitrary s, we obtain a valid derivation for

$$\alpha^{(s)} = \alpha_1 x_{i_1} z_{i_1}^s \alpha_2 x_{i_2} z_{i_2}^s \ldots \alpha_k x_{i_k} z_{i_k}^s \alpha_{k+1}.$$

The word $\alpha^{(s)}$ therefore belongs to L for all natural numbers $s > 0$. The derivation between c_1 and c_2 can also be omitted and therefore also $\alpha^{(0)}$ belongs to L.

Note that we do not give an upper bound for k. This follows from the fact that in a dag a vertex can have more fathers. Consequently, a component $x_i z_i$ can be read by more than one grammar and thus appear more than once in α. However, the

number of *different* words z_{i_j} is at most n. Indeed when iterating the subderivation $c_1 \Longrightarrow^* c_2$, we can only pump the z_i's already existing in some components of c_2, that is, at most n different ones. As explained before, because of the communications steps that occur after c_2, some of the words z_i^s can appear several times in $\alpha^{(s)}$. □

An analogous pumping lemma can be obtained for tree-PCGS, but in this case the number of pumped positions is bounded by the number of grammars of the tree-PCGS.

Lemma 11. (Pumping lemma for tree-PCGS) *Let L be a language generated by a tree-PCGS. There exists a natural number N such that every word $\alpha \in L$ whose length is greater than N can be decomposed as*

$$\alpha = \alpha_1\beta_1 \ldots \alpha_m\beta_m\alpha_{m+1},$$

where $1 \leq m \leq n$, $\beta_i \neq \lambda$ for every i, $1 \leq i \leq m$, and the word

$$\alpha_1\beta_1^s \ldots \alpha_m\beta_m^s\alpha_{m+1}$$

belongs to L for all $s \geq 0$.

As a consequence of Lemma 10, we can obtain a language that can be generated by a tree-PCGS but cannot be generated by any dag-PCGS of smaller degree.

Theorem 12. *For all $n > 1$, $\mathcal{L}(\text{tree-PCGS}_n) - \mathcal{L}(\text{dag-PCGS}_{n-1}) \neq \emptyset$.*

Proof. Consider the language $L_n = \{a_1^{k+1}a_2^{k+2} \ldots a_n^{k+n} \mid k \geq 0\}$. □

The following infinite hierarchies are obtained as consequences of the preceding result.

Corollary 13. *The hierarchy $\{\mathcal{L}(\text{dag-PCGS}_n)\}_{n\geq 1}$ is infinite.*

Corollary 14. *The hierarchy $\{\mathcal{L}(\text{tree-PCGS}_n)\}_{n\geq 1}$ is infinite.*

In the remaining part of this section we will consider some PCGS with communication structures for which no pumping lemmas are known, namely two-way array, two-way ring and one-way ring-PCGS. The following theorem provides a language that can be generated by a two-way array-PCGS but cannot be generated by *any* PCGS of smaller degree. This shows that in some cases the increase in descriptional complexity cannot be compensated by an increase in the complexity of the communication structure.

Theorem 15. *For all $m \geq 1$,*

$$\mathcal{L}(\text{two-way array-PCGS}_{m+1}) - \mathcal{L}(\text{two-way array-PCGS}_m) \neq \emptyset.$$

Proof. Consider the language

$$L_m = \{a_1^n a_2^n \ldots a_{2m}^n \mid n \geq 1\}.$$

We can show a stronger result than the one stated in the theorem. For all $m > 1$ there exists the language L_m that can be generated by a two-way array PCGS of degree $m+1$ but cannot be generated by *any* PCGS of smaller degree. □

Corollary 16. *The hierarchy* $\{\mathcal{L}(\textit{two-way array-PCGS}_n)\}_{n\geq 1}$ *is infinite.*

Corollary 17. *The hierarchy* $\{\mathcal{L}(\textit{two-way ring-PCGS}_n)\}_{n\geq 1}$ *is infinite.*

The language used in the proof of Theorem 15 can be used to show that the hierarchy of one-way ring-PCGS, relative to the number of the grammars in the PCGS, is infinite. When constructing the one-way ring-PCGS which generates the language, special care has to be payed to synchronization problems.

Theorem 18. *For all* $m \geq 1$,

$$\mathcal{L}(\textit{one-way ring-PCGS}_{m+1}) - \mathcal{L}(\textit{one-way ring-PCGS}_m) \neq \emptyset.$$

Corollary 19. *The hierarchy* $\{\mathcal{L}(\textit{one-way ring-PCGS}_n)\}_{n\geq 1}$ *is infinite.*

The study of hierarchies on the number of grammars for PCGS with other communication structures (planar graphs, hypercubes, etc) remains open.

References

1. K.Culik, J.Gruska, A.Salomaa. Systolic trellis automata. *International Journal of Computer Mathematics* **15** and **16**(1984).
2. P.Duris, J.Hromkovic: Zerotesting bounded multicounter machines. *Kybernetika* 23(1987), No.1, 13-18.
3. S.Ginsburg: *Algebraic and Automata-Theoretic Properties of Formal Languages.* North-Holland Publ.Comp., Amsterdam 1975.
4. C.A.R.Hoare. Communicating sequential processes. *Comm. ACM.* **21** vol. 8 (1978).
5. J.Hromkovic: Hierarchy of reversal bounded one-way multicounter machines. *Kybernetica* 22(1986), No.2, 200-206.
6. J.Kari. Decision problems concerning cellular automata. *University of Turku, PhD Thesis* (1990).
7. D.Pardubska. The communication hierarchies of parallel communicating systems. *Proceedings of IMYCS'92*, to appear.
8. Gh.Paun. On the power of synchronization in parallel communicating grammar systems. *Stud. Cerc. Matem.* **41** vol.3 (1989).
9. Gh.Paun. Parallel communicating grammar systems: the context-free case. *Found. Control Engineering* **14** vol.1 (1989).
10. Gh.Paun. On the syntactic complexity of parallel communicating grammar systems. *Kybernetika*, **28**(1992), 155-166.
11. Gh.Paun, L.Santean. Further remarks on parallel communicating grammar systems. *International Journal of Computer Mathematics* **35** (1990).
12. Gh.Paun, L.Santean. Parallel communicating grammar systems: the regular case. *Ann. Univ. Buc. Ser. Mat.-Inform.* **37** vol.2 (1989).
13. A.Salomaa. *Formal Languages.* Academic Press New York London (1973).
14. L.Santean, J.Kari:The impact of the number of cooperating grammars on the generative power, *Theoretical Computer Science*, **98**, 2(1992), 249-263.
15. D.Wierzchula: Systeme von parallellen Grammatiken (in German). Diploma thesis, Dept. of Mathematics and Computer Science, University of Paderborn, 1991.