Some hierarchies for the communication complexity measures of cooperating grammar systems

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

June 14, 2010

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 **C**ommunicating **G**rammar **S**ystems (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

[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

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.

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*.

If no communication symbol appears in any of the components, we perform a *rewriting step* which consists of a rewriting step performed synchronously in each of the grammars. If one of the components is a terminal string, it is left unchanged while the others are performing the rewriting step. If in one of the components the nonterminal cannot be rewritten any more, the derivation is blocked.

If in any of the components a communication symbol is present, a *communication step* is performed. It consists of replacing all the occurrences of communication symbols with the components they refer to, providing these components do not contain further communication symbols. If some communication symbols are not satisfied in this step, they may be satisfied in one of the next ones. Communication steps are performed until no more communication symbols are present. No rewriting is allowed if any communication symbol occurs in one of the components. Therefore, if circular queries emerge, the derivation is blocked.

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

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-PCGS, two-way array-PCGS, one-way array-PCGS, two-way ring-PCGS, 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

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

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 characterization 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 $f : \mathbf{N} \longrightarrow \mathbf{N}$. 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 1 $\mathcal{L}(\textit{tree-PCGS}) \subseteq \textit{NTIME}(n) \cap \textit{NSPACE}(\log_2 n)$.

Now let us consider the general simulation of PCGS by an off-line nondeterministic multitape Turing machines. A general PCGS$_m$ can be simulated by an m-tape nondeterministic Turing machine with the working tapes $T_1, \ldots, T_m$ in the following way. Each tape is used to simulate the rewriting work of one grammar. If a grammar G_i produces Q_j, then the content of the tape T_j is copied on the tape T_i and the content of the tape T_j is rewritten with S_j (the initial nonterminal of G_j). Obviously, we cannot assume that this simulation works in linear time. For instance, if one considers the systems of two grammars which communicate exactly in each second (even) step and the communication alternates (each odd communication flows from the second grammar to the first grammar and each even communication flows from the first grammar to the second grammar), then some produced terminals can be linear-time copied from one tape to the other one and back. Thus, the simulation can require $\Omega(n^2)$ steps. In what follows we shall show that there is a simulation working in linear time for dag-PCGS.

Theorem 2 $\mathcal{L}(\textit{dag-PCGS}) \subseteq \textit{NTIME}(n)$.

Proof. Let $\pi = (G_1, \ldots, G_m)$ be a dag-PCGS$_m$. We shall describe an off-line m-tape nondeterministic Turing machine M recognizing $L(M)$ in linear time.

During the whole simulation M stores the actual nonterminals of all G_is in its finite control. M uses its tapes $T_1, \ldots, T_m$ to store the current words generated by $G_1, \ldots, G_m$ respectively, or T_i may contain the blank symbol B if M has nondeterministically decided that the word curently produced by G_i will never be included in the final word generated by the dag-PCGS$_m$ π (This is to remove the unnecessary transfers from one tape to another one).

The simulation of π by M runs as follows. At the beginning M stores the initial nonterminals of all grammars in its finite control (state). For each $i = 1, \ldots, m$ M nondeterministically decides whether the next word generated by G_i will be a part of the final word generated by π or not. If M decides that the word generated by G_i will be a part of the final word, then M will simulate the rewriting procedure of G_i on the tape T_i. If M decides that the

word generated by G_i will never be a part of the final word generated by π, then M writes B on the tape T_i and does not simulate the next rewriting procedure of G_i on T_i. Thus, in this case M only simulates the work of G_i in its final control by storing the actual nonterminal. The simulation of π by M runs synchronously, i.e. one simulation step of M consists of the simulation of one rewriting step of all the grammars G_i on all tapes T_i. The simulation runs in this way until no nonterminal symbol is on the first tape T_1 or at least one symbol from $\{Q_1, \ldots, Q_m\}$ appears on a tape.

If no nonterminal symbol is on T_1, then the content of T_1 is compared with the content of the input tape. If these contents are identical, then M accepts the input word.

If the actual nonterminal of G_r, $r \in \{1, \ldots, m\}$, stored in the final control, is Q_i, $i \in \{1, \ldots, m\}$, and both T_r and T_i do not contain B, then the content of T_i is copied on the tape T_r (the copy is laid on T_r so that it starts on the position containing Q_i). After this M continues to simulate the work of G_r from the nonterminal (the last symbol of the copy) and the whole content of T_i is rewritten by one symbol S_i (the initial nonterminal of G_i). Now, M again nondeterministically decides whether the next word generated by G_i will be a subword of the final word or not. Depending on this decision M either simulates G_i on T_i or writes B on T_i.

If the actual nonterminal of G_r stored in the final control is Q_i for some $1 \in \{1, \ldots, m\}$ and exactly one of the tapes T_r and T_i contains B then M halts and does not accept the input word (some of the nondeterministic decisions of M was incorrect).

If the actual nonterminal of G_r stored in the final control is Q_i for some $i \in \{1, \ldots, m\}$, and both tapes T_r and T_i contain B, then M does not change the content of T_r and it nondeterministically decides whether the next word generated by G_i will be a subword of the final word or not. Again, depending on this decision M either simulates G_i on T_i or writes B on T_i. In the case that several communication symbols appear simultaneously, the transfer of the contents of the tape is made in a order such that a word ending with a symbol from $\{Q_1, \ldots, Q_m\}$ is never transferred. Note that this is possible because the communication structure does not contain any cycle.

Obviously, for each word $w \in L(\pi)$ there exists a right sequence of nondeterministic decisions of M which leads to the generation of w on the first working tape T_1. Since the communication structure of π does not contain any cycle, each symbol of the word w generated by π was copied at most $m-1$ times from one tape to another tape. Thus, the simulation of π by M works in linear time.

□

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 1 that $\mathcal{L}$(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 folowing 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 2 *$L \in \mathcal{L}$(c-$PCGS_2(n)$) $- \cup_{m\in\mathbf{N}}\mathcal{L}$(tree-$PCGS_m(f(n))$) for any $f(n) \notin \Omega(n)$.*

Proof. Let us first prove that $L \in \mathcal{L}(\text{c-PCGS}_2(n))$. In order to do it, it is sufficient to consider the folowing $c-\text{PCGS}_2(n)$ $\pi = (G_1, G_2)$, where

$$\begin{aligned} G_1 = \ & (\{S_1, S_2, Q_2\}, \{a, c\}, \{S_1\}, \\ & \{S_1 \longrightarrow aS_1, S_1 \longrightarrow aQ_2, S_2 \longrightarrow aS_1, S_2 \longrightarrow c\}) \\ G_2 = \ & (\{S_2\}, \{b\}, \{S_2\}, \{S_2 \longrightarrow bS_2\}) \end{aligned}$$

Now let us prove the fact that $L \notin \cup_{m\in\mathbf{N}}\mathcal{L}(\text{tree-PCGS}_m(f(n)))$ for $f(n) \notin \Omega(n)$ by contradiction.

Let $L \in \mathcal{L}(\text{tree-PCGS}_m(g(n)))$ for some $m \in \mathbf{N}$ and some $g(n) \notin \Omega(n)$. Following Lemma 1 we may assume that there is a real-time nondeterministic

$(m-1)$-counter machine M which recognizes L with 2 $g(n)$ reversals, and moreover M has the property (i).

Let M have k states. To realize our proof we need to define the following notion.

Let M read a group of k identical symbols in k steps. Clearly, there has to be a state q which will be entered twice or more in different configurations in this part of the computation consisting of $k-1$ configurations. If two occurrences of the state q are adjacent (no further state q and no two equal states different from q occur in between) we say that the part of the computation from q to q is a *cycle* with the *state characteristic* q, *reading head characteristic* e-the number of symbols over which the reading head moves to the right in the cycle- and *counter characteristics* $e_1, \ldots, e_m$, where $e_i \in [-k, k]$ is the difference between the counter contents at the beginning and at the end of the cycle. The vector $(q, e, e_1, \ldots, e_m)$ is the *characteristic vector* of the cycle. If all $e_1, \ldots, e_{m-1} \in \{0, h, -h\}$ for some $h \in \{1, \ldots, e\}$ we say that the cycle is (e, h)-regular. Note that every cycle of M laying between two reversals is a regular cycle because of the property (i).

The fact $g(n) \notin \Omega(n)$ implies that for all $n_0, d \in \mathbf{N}$, there is $n \geq n_0$ such that $g(n) \leq n/d$. Let us take $n_0 = 2(k+1)^3 \cdot 10$, $d = 8 \cdot (k+1)^3$, where k is the number of states of M. Thus, there exists an $n \in N$, $n \geq n_0$, such that for any $x \in L$, $|x| = n$, M has an accepting computation on x with at most $g(n) \leq n/2(k+1)^3$ reversals. This implies that there is a word

$$w = a^{i_1} b^{i_1} a^{i_2} b^{i_2} \ldots a^{i_n} b^{i_n} c \in L(\pi),$$

such that $i_v = d/8$ for each $v \in \{1, \ldots, n-1\}$, $d/4 \geq i_n \geq d/8$, and $|w| = n$ (note that $n \cdot d/4 \leq n \leq n \cdot d/4 + d/4$).

Now, let us consider the accepting computation C_w of M on w which contains at most n/d reversals. Since $n+1 \geq 4n/d$ there exists a $j \in \{1, \ldots, n\}$ such that C_w contains no reversal in the part of the computation in which $a^{i_j} b^{i_j}$ is read. Since $i_j \geq (k+1)^3$ there are at least $(k+1)^2$ disjoint regular cycles appearing by reading $a^{i_j}(b^{i_j})$. This implies that for some $r, z, r', z' \in \{1, \ldots, k\}$ there exists an (r, r')-regular cycle R of M appearing at least k times by reading a^{i_j} and a (z, z')-regular cycle X appearing at least k times by reading b^{i_j}.

Thus, the computation C_w on w may be written as

$$C_w = C_1 C_0 C_2, C_0 = P_0 R P_1 R P_2 \ldots P_{k-1} R Z_0 X Z_1 X \ldots Z_{k-1} X Z_k,$$

where C_1, C_0, C_2 are the parts of C_w on the words $a^{i_1} b^{i_1} \ldots a^{i_{j-1}} b^{i_{j-1}}$, $a^{i_j} b^{i_j}$, $a^{i_{j+1}} b^{i_{j+1}} \ldots a^{i_n} b^{i_n} c$ respectively, and $P_0, \ldots, P_k$, $Z_0, \ldots, Z_{k+1}$ are some parts (may be also empty) of the computation of M on $a^{i_j} b^{i_j}$.

Now, we show that the following word

$$w_1 = a^{i_1} b^{i_1} \ldots a^{i_{j-1}} b^{i_{j-1}} a^{i_j - rz'} b^{i_j + r'z} a^{i_{j+1}} b^{i_{j+1}} \ldots a^{i_n} b^{i_n} c$$

is accepted by M ($w_1 \in L(M)$) which is a contradiction with the fact that $w_1 \notin L$. To see this we write the accepting computation C_{w_1} of M on w.

$$C_{w_1} = C_1 P_0 P_1 \dots P_{z'} R P_{z'+1} \dots R P_{k-1} R Z_0(X)^{r'+1} Z_1 X Z_2 \dots Z_{k-1} X Z_k C_2.$$

C_{w_1} is an accepting computation on w_1 because of the following facts:

(1) No counter is emptied in the part C_0 of C_w and so no counter can be emptied in the part

$$P_0 P_1 \dots P_{z'} R P_{z'+1} \dots R P_{k-1} R Z_0(X)^{r'+1} Z_1 X Z_2 \dots X Z_k$$

of C_{w_1} (because of the property (i) of M).

(2) M after the computation

$$C_1 P_0 P_1 \dots P_{z'} R P_{z'+1} \dots R P_{k-1} R Z_0(X)^{r'+1}$$

on w_1 is in the same configuration (the same state, the same contents of counters, the same postfix of the input word on the input tape) as M after the computation part

$$C_1 P_0 R P_1 R \dots R P_z R P_{z+1} \dots R P_{k-1} R Z_0 X$$

of C_w on w.

Obviously, the fact (2) is a direct consequence of (1) and some trivial computation about the contents of counters. □

Following Lemma 2 we get the following hierarchies on the communication complexity.

Theorem 3 *For any function $f : \mathbf{N} \longrightarrow \mathbf{N}$, $f(n) \notin \Omega(n)$, and any $m \in \mathbf{N}$, $m \geq 2$:*

$$\mathcal{L}(\textit{one-way-array-PCGS}_m(f(n))) \subset \mathcal{L}(\textit{one-way-array-PCGS}_m(n)),$$

$$\mathcal{L}(\textit{c-PCGS}_m(f(n))) \subset \mathcal{L}(\textit{c-PCGS}_m(n)),$$

$$\mathcal{L}(\textit{tree-PCGS}_m(f(n))) \subset \mathcal{L}(\textit{tree-PCGS}_m(n)).$$

Besides Theorem 3, Lemma 3 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

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

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

Lemma 3 $L_k \in \mathcal{L}(c\text{-}PCGS_{k+1}(k)) - \cup_{m\in\mathbf{N}}\mathcal{L}(\ tree\text{-}PCGS_m(k-1))$.

Proof. To generate L_k the following c-PCGS$_{k+1}(k)$, $\pi = (G_0, G_1, \ldots, G_k)$ can be used.

$$\begin{aligned} G_0 = \quad & (\{S_1, \ldots, S_{k+1}, Q_1, \ldots, Q_k\}, \{a\}, \{S_1\}, \\ & \{S_i \longrightarrow aS_i, S_i \longrightarrow aQ_i, S_{k+1} \longrightarrow c |\ 1 \le i \le k\}), \\ G_j = \quad & (\{S_j, S_{j+1}\}, \{b\}, \{S_j\}, \{S_j \longrightarrow bS_{j+1}, S_{j+1} \longrightarrow bS_{j+1}\}), \\ & \text{for all } j = 1, \ldots, k. \end{aligned}$$

Now let us prove that $L_k \notin \mathcal{L}$(tree-PCGS$_m(k-1)$) for any $m \in \mathbf{N}$ by contradiction. Let there be a tree-PCGS$_m(k-1)$ generating L_k.

Then there exists (following Lemma 1) a linear-time $(m-1)$-counter automaton M recognizing L_k with at most $2(k-1)$ reversals and $k-1$ zerotests. Moreover, M has the property (i), and each zerotest of M coincides with a "lower" reversal of M.

Let M have s states. Choose $l = (s+1)^3$ and consider the word

$$w = a^{2l}b^{2l}a^{2l}b^{4l}a^{2l}b^{6l}\ldots a^{2l}b^{2kl}c,$$

that is, $i_1 = i_2 = \ldots = i_k = 2l$.

The word w can be decomposed as

$$w = a^l w_1 u_1 w_2 u_2 \ldots w_{k-1} u_{k-1} w_k b^{kl} c,$$

where $w_i = a^l b^{il}$ and $u_i = b^{il}a^l$. As we can have at most $2(k-1)$ reversals, there exists $i \in \{1, \ldots, k\}$ such that no reversal occurs when reading the subword w_i or u_i.

As in the proof of Lemma 2 we see that we can find constants $r, z, r', z' \in \{1, \ldots, s\}$ such that we can replace w_i (respectively u_i) by $w_i' = a^{l-rz'}b^{il+r'z}$ ($u_i' = b^{il-rz'}a^{l+r'z}$, respectively) obtaining

$$w' = a^l w_1 u_1 \ldots w_i' u_i \ldots w_k b^{kl} c,$$

$$w'' = a^l w_1 u_1 \ldots w_i u_i' \ldots w_k b^{kl} c, \text{ respectively,}$$

and M still recognizes the word obtained in this way. This further implies that w' (respectively w'') belongs to the language L_k - a contradiction with the definition of L_k. □

Theorem 4 *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 1 *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:*

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

Proof. The function defined by:
"$f(x)$ is the length of the longest path in G starting in node x"
satisfies the requested condition. □

The classical proof of the pumping lemma for regular languages is based on finding, along a sufficiently long derivation, of two "similar" sentential forms. "Similar" means that the two sentential forms contain the same nonterminal, 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 splitted 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 4 *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.*

Proof. Let $\pi = (G_1, G_2, \ldots, G_n)$ be a dag-PCGS, where $G_i = (V_{N,i}, \Sigma, S_i, P_i)$.

Denote by A the number of equivalence classes of the equivalence relation $\equiv$. Clearly,

$$A = \prod_{i=1}^{n} (|V_{N,i}| + 1).$$

Denote further by p the maximum number of productions that exists in any of the grammars.

Define recursively

$$\begin{array}{lcl} M_1 & = & A, \\ M_{j+1} & = & A \cdot \prod_{k=1}^{j} r_j^k \; + M_j, \end{array}$$

where, for each j, r_j^k, $0 < k < j$, are defined as:

$$\begin{array}{lcl} r_j^1 & = & (p+1)^{M_j}, \\ r_j^{k+1} & = & (p + 1 + \sum_{s=1}^{k} r_j^s)^{M_j}, \end{array}$$

Claim. For each j, $1 \leq j \leq n$, in any derivation D of length M_j, where less than j communication symbols are used, there are two D-similar configurations.

The claim will be proved by induction on j.

If $j = 1$ then no communication symbols are used in the derivation D. The length of D is $M_1 = A$ and therefore it contains $A + 1$ configurations. The number of equivalence classes of $\equiv$ is A, so the pigeon-hole principle says that

there are two equivalent configurations in D. Obviously they are D-similar as well, because no communication symbols appear during D.

$j \mapsto j+1$. Consider a derivation D of length M_{j+1} where at most j different communication symbols are used. If it contains a subderivation of length M_j, where less than j different communication symbols are used, we are through.

Otherwise, all the subderivations of length M_j from D contain all j different communication symbols used in D. Let us denote by D' the subderivation of D which starts after the first M_j steps of D.

The derivation D' contains $A \cdot \prod_{j=1}^{j} r_j^k \ + 1$ configurations. More than $\prod_{k=1}^{j} r_j^k$ of them must be in the same equivalence class of $\equiv$.

Let us order the j communication symbols used in D, according to the values of the function f (see Proposition 1) of the corresponding grammars. Thus, the communication symbols used during the derivation D are $Q_{i_j}, Q_{i_{j-1}}, \ldots, Q_{i_1}$ where $f(G_{i_j}) \geq f(G_{i_{j-1}}) \geq \ldots \geq f(G_{i_1})$. The nearest occurrence of Q_{i_1} preceding any configuration of D' must appear in one of the M_j predecessor configurations. As $f(G_{i_1})$ is the minimum value among $f(G_{i_j}), \ldots, f(G_{i_1})$, it results that G_{i_1} does not ask for strings from any other grammar during D. Therefore in a single derivation step performed during D', G_{i_1} can either use one of its p productions or remain unchanged. Consequently, there may exist at most $(p+1)^{M_j}$ different i_1-components in configurations appearing in D'. Define $r_j^1 = (p+1)^{M_j}$. It follows that there exist at most r_j^1 different i_1-components in configurations of D'.

Assume now that there are respectively $r_j^{k-1}, \ldots, r_j^1$ different possibilities for the components in positions $i_{k-1}, \ldots, i_1$ of any configuration of D'. Consider now the component in the position i_k. The nearest occurrence of Q_{i_k} preceding any configuration of D' appears in one of the M_j predecessor configurations. After the communication step demanded by Q_{i_k} the grammar G_{i_k} returns to the axiom. In a single derivation step, G_{i_k} may either use one of its p productions, or remain unchanged, or communicate with one of G_{i_j}, $j < k$. Indeed, recall that G_{i_k} can ask only for strings generated by grammars G_r with $f(G_{i_k}) > f(G_r)$. Consequently, for the i_k-component we have

$$r_j^k = (p + 1 + \sum_{l=1}^{k-1} r_j^l)^{M_j}$$

different possibilities in any configuration appearing in D'.

Counting the possibilities for all the components corresponding to $Q_{i_j}, \ldots, Q_{i_1}$, one gets $\prod_{k=1}^{j} r_j^k$ different possibilities. This means that we have at most $\prod_{k=1}^{j} r_j^k$ configurations along D' which differ by at least one component whose corresponding communication symbol has been used in the derivation D'. As along D' there are more than $\prod_{k=1}^{j} r_j^k$ equivalent configurations, an application of the pigeon-hole principle tells that we can find two D'-similar configurations.

Let us return to the proof of the lemma.

From the claim it follows that in any derivation D according to π of length at least M_n, there are two D-similar configurations. Indeed, the maximum number of communication symbols that can occur in any derivation is $n-1$: the communication symbols of grammars which have the in-degree zero do not occur. On the other hand such a node with in-degree zero always exists in a dag.

Taking now $q = M_n$, the lemma is proved.

□

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 5 (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 at least N can be decomposed as*

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

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

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

belongs to L.

Proof. Let $\pi = (G_1, \dots, G_n)$ be a dag-PCGS like in the preceding lemma. 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 4 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 4, during this derivation occur at least two D-similar configurations c_1 and c_2 as shown below:

$$\begin{array}{lll}(S_1, S_2, \ldots, S_n) & \Longrightarrow^* & c_1 = (x_1A_1, x_2A_2, \ldots, x_nA_n) \\ & \Longrightarrow^* & c_2 = (x_1z_1A_1, x_2z_2A_2, \ldots, x_nz_nA_n) \\ & \Longrightarrow^* & (\alpha, \ldots\ldots).\end{array}$$

If all the strings x_iz_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_iz_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 6 (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*

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

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

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

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

Proof. Similar to the one of the preceding pumping lemma. The only difference is that in a tree no vertex can have more than one father. Consequently, a word z_i cannot be read by more grammars at the same time, which implies that no word z_i^s can appear twice in α as a result of a communication. The word z_i^s can appear twice in α only if, by some coincidence, $z_i = z_j$ for some indices $i \neq j$, $i, j \leq n$. We conclude that in the case of trees we can pump on at most n positions. □

As a consequence of Lemma 5, 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 5 *For all $n > 1$, $\mathcal{L}(\textit{tree-PCGS}_n) - \mathcal{L}(\textit{dag-PCGS}_{n-1}) \neq \emptyset$.*

Proof. Consider the language

$$L_n = \{a_1^{k+1}a_2^{k+2} \dots a_n^{k+n} |\ k \geq 0\}.$$

The language L_n can be generated by the tree PCGS $\pi = (G_1, \dots, G_n)$, where

$$\begin{aligned}
G_i &= (V_{N,i}, \Sigma, S_i, P_i),\\
\Sigma &= \{a_1, \dots, a_n\},\\
V_{N,1} &= \{S_i |\ 1 \leq i \leq n\} \cup \{Q_i |\ 2 \leq i \leq n\},\\
V_{N,i} &= \{S_i\}, 2 \leq i \leq n,\\
P_1 &= \{S_1 \longrightarrow a_1S_1, S_n \longrightarrow a_n\} \cup \{S_i \longrightarrow a_iQ_{i+1} |\ 1 \leq i \leq n-1\},\\
P_i &= \{S_i \longrightarrow a_iS_i\}, 2 \leq i \leq n,
\end{aligned}$$

and therefore $L_n \in \mathcal{L}(\text{tree} - PCGS_n)$.

However, L_n cannot generated by any dag-PCGS of degree $n-1$ or smaller. Assume the contrary and let N be the number defined in Lemma 5. Consider the word

$$\alpha = a_1^{N+1}a_2^{N+2} \dots a_n^{N+n}.$$

Following Lemma 5, the words $\alpha^{(s)}$ obtained from α by pumping *at most* $n-1$ *different* subwords of it belong to L_n. First, note that the only words that

can be pumped are necessarily of the form a_i^k, $1 \leq i \leq n$. By pumping only $n-1$ of them, the exponent of the letter which is not pumped will remain bounded, whereas the exponents of the pumped ones will grow arbitrarily large. This contradicts the form of the words in L_n. Consequently, the language L_n belongs to $\mathcal{L}(\text{tree}-PCGS_n)-\mathcal{L}(\text{dag}-PCGS_{n-1})$ which is therefore nonempty. □

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

Corollary 1 *For all $n > 1$, $\mathcal{L}(\textit{dag-PCGS}_n) - \mathcal{L}(\textit{dag-PCGS}_{n-1}) \neq \emptyset$.*

Corollary 2 *The hierarchy $\{\mathcal{L}(\textit{dag-PCGS}_n)\}_{n\geq 1}$ is infinite.*

Corollary 3 *For all $n > 1$, $\mathcal{L}(\textit{tree-PCGS}_n) - \mathcal{L}(\textit{tree-PCGS}_{n-1}) \neq \emptyset$.*

Corollary 4 *The hierarchy $\{\mathcal{L}(\textit{tree-PCGS}_n)\}_{n\geq 1}$ 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 6 *For all $m \geq 1$,*

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

Proof. Let L be the language

$$L = \{a_1^n a_2^n \dots a_{2m}^n |\ n \geq 1\}.$$

We shall prove first that L can be generated by a two-way array-PCGS consisting of $m+1$ grammars. Indeed, let $\pi = (G_1, \dots, G_{m+1})$ where the communication

graph is a two-way linear array and $G_i = (V_{N,i}, \Sigma, S_i, P_i)$, $1 \leq i \leq m+1$,

$$\begin{aligned}
\Sigma &= \{a_1, a_2, \ldots, a_{2m}\},\\
V_{N,1} &= \{S_1, Q_2, Z, Z'\} \cup \{X_2^k |\ 1 \leq k \leq 2m\},\\
V_{N,j} &= \{S_j, Q_{j-1}, Q_{j+1}, Y_j\} \cup \{X_j^k |\ 1 \leq k \leq 2m\}\\
&\quad \cup \{X_{j+1}^k |\ j \leq k \leq 2m-j+1\},\ \text{for } 2 \leq j \leq m,\\
V_{N,m+1} &= \{S_{m+1}, Q_m, Z, Y_{m+1}\} \cup \{X_{m+1}^k |\ 1 \leq k \leq 2m\},\\
P_1 &= \{S_1 \longrightarrow a_1 Q_2, X_2^1 \longrightarrow a_2 X_2^2, S_1 \longrightarrow a_1 a_2 \ldots a_{2m} Z'\}\\
&\quad \cup \{Z \longrightarrow Z', Z' \longrightarrow \lambda\} \cup \{X_2^k \longrightarrow X_2^{k+1} |\ 2 \leq k < 2m\},\\
P_j &= \{S_j \longrightarrow X_j^1, S_j \longrightarrow a_{2j-1} Q_{j+1}, S_j \longrightarrow Q_{j-1}, S_j \longrightarrow Y_j, Y_j \longrightarrow Y_j\}\\
&\quad \cup \{X_j^k \longrightarrow X_j^{k+1} |\ 1 \leq k < j-1\} \cup \{X_{j+1}^j \longrightarrow a_{2j} X_{j+1}^{j+1}\}\\
&\quad \cup \{X_{j+1}^k \longrightarrow X_{j+1}^{k+1} |\ j+1 \leq k \leq 2m-j\}\\
&\quad \cup \{X_j^k \longrightarrow X_j^{k+1} |\ 2m-j+1 < k \leq 2m-1\}\\
&\quad \cup \{X_j^{2m} \longrightarrow X_j^1, X_j^{2m} \longrightarrow a_{2j-2} a_{2j-1} Q_{j+1}\},\\
&\quad \text{for all } j, 2 \leq j \leq m,\\
P_{m+1} &= \{S_{m+1} \longrightarrow X_{m+1}^1, S_{m+1} \longrightarrow Q_m, S_{m+1} \longrightarrow Y_{m+1}, Y_{m+1} \longrightarrow Y_{m+1},\\
&\quad X_{m+1}^{2m} \longrightarrow X_{m+1}^1, X_{m+1}^{2m} \longrightarrow a_{2m} Z\}\\
&\quad \cup \{X_{m+1}^k \longrightarrow X_{m+1}^{k+1} |\ 1 \leq k < 2m, k \neq m\}.
\end{aligned}$$

For proving that $L \subseteq L(\pi)$ we shall show that, for every n, the word $a_1^n a_2^n \ldots a_{2m}^n$ can be generated by π.

Claim. For all $n \in \mathbf{N}$ there exists a derivation

$$D:\ (S_1, S_2, S_3, \ldots, S_{m+1}) \Longrightarrow^* (a_1 Q_2, a_1^n a_2^n X_2^1, a_3^n a_4^n X_3^1, \ldots, a_{2m-1}^n a_{2m}^n X_{m+1}^1)$$

according to π.

The claim will be proved by induction on n. For $n = 0$, we can construct the derivation

$$\left(S_1, S_2, \ldots, S_{m+1}\right) \longrightarrow \left(a_1 Q_2, X_2^1, \ldots, X_{m+1}^1\right).$$

Let us suppose now that there exists a derivation D

$$(S_1, S_2, \ldots, S_{m+1}) \Longrightarrow^* \left(a_1 Q_2, a_1^n a_2^n X_2^1, \ldots, a_{2m-1}^n a_{2m}^n X_{m+1}^1\right).$$

We shall construct a valid derivation D' for the configuration

$$\left(a_1 Q_2, a_1^{n+1} a_2^{n+1} X_2^1, \ldots, a_{2m-1}^{n+1} a_{2m}^{n+1} X_{m+1}^1\right).$$

The idea of the construction is the following. We shall add a subderivation to the derivation D, such that every component, except the first one, will have in the end the exponent increased by one. The increasing of the exponent implies the catenation of one letter to the left side of the terminal word, and one to the right. This wouldn't be possible in an ordinary regular grammar, where

the letters are only added to one end. Using the communication, letters can be added here to both ends of the terminal word of some component. This is done first by communicating the word to the left component. Together with the communication symbol, a letter is produced, that means it is catenated to the left end of the word. Afterwards, working in this auxiliary component another letter is produced, that means it is catenated to the right. Finally, (after the change has been made in all components) the new word is communicated back to the original component where it belonged.

This procedure can be applied in a chain, from left to right, using the fact that we have one grammar for which we do not need to change the word, that is we have an auxiliary place. After all the needed letters are produced, the new strings are in components situated to the left of their original ones. Then, beginning with the m'th component, the strings are moved one position to the right, and the requested configuration is obtained. Special attention has to be paid to the components in the "waiting status", because the changing of the string is only done for one component at a time. Therefore, until the turn of a particular component to be communicated comes, only renamings are performed in it, the upper index of the nonterminals $X_j^k, 1 \leq j \leq m+1, 1 \leq k \leq 2m+1$ counting the "waiting" steps.

The derivation D' has therefore the following form:

$$\begin{gathered}
\left(a_1Q_2,\ldots,a_{2j-3}^n a_{2j-2}^n X_j^1, a_{2j-1}^n a_{2j}^n X_{j+1}^1,\ldots,a_{2m-1}^n a_{2m}^n X_{m+1}^1\right)\\
\Downarrow \begin{array}{l} j-1 \text{ rewriting steps and}\\ j-1 \text{ communication steps}\end{array}\\
\left(a_1^{n+1}a_2^{n+1}X_2^j,\ldots,a_{2j-1}Q_{j+1}, a_{2j-1}^n a_{2j}^n X_{j+1}^j,\ldots,a_{2m-1}^n a_{2m}^n X_{m+1}^j\right)\\
\Downarrow \text{ communication step}\\
\left(a_1^{n+1}a_2^{n+1}X_2^j,\ldots,a_{2j-1}^{n+1} a_{2j}^n X_{j+1}^j, S_{j+1},\ldots,a_{2m-1}^n a_{2m}^n X_{m+1}^j\right)\\
\Downarrow \text{ rewriting step}\\
\left(a_1^{n+1}a_2^{n+1}X_2^{j+1},\ldots,a_{2j-1}^{n+1} a_{2j}^{n+1} X_{j+1}^{j+1}, a_{2j+1}Q_{j+2},\ldots,a_{2m-1}^n a_{2m}^n X_{m+1}^{j+1}\right)\\
\Downarrow_* \begin{array}{l} m-j \text{ communication steps and}\\ m-j-1 \text{ rewriting steps}\end{array}\\
\left(a_1^{n+1}a_2^{n+1}X_2^m,\ldots,a_{2j-1}^{n+1} a_{2j}^{n+1} X_{j+1}^m, a_{2j+1}^{n+1} a_{2j+2}^{n+1} X_{j+2}^m,\ldots,S_{m+1}\right)\\
\Downarrow \text{ rewriting step}\\
\left(a_1^{n+1}a_2^{n+1}X_2^{m+1},\ldots,a_{2j-1}^{n+1} a_{2j}^{n+1} X_{j+1}^{m+1}, a_{2j+1}^{n+1} a_{2j+2}^{n+1} X_{j+2}^{m+1},\ldots,Q_m\right)\\
\Downarrow_* \begin{array}{l} m \text{ communication steps and}\\ m-1 \text{ rewriting steps}\end{array}\\
\left(S_1,\ldots,a_{2j-3}^{n+1} a_{2j-2}^{n+1} X_j^{2m}, a_{2j-1}^{n+1} a_{2j}^{n+1} X_{j+1}^{2m},\ldots,a_{2m-1}^{n+1} a_{2m}^{n+1} X_{m+1}^{2m}\right)
\end{gathered}$$

$$\Downarrow \text{ rewriting step}$$

$$(a_1Q_2, \ldots, a_{2j-3}^{n+1}a_{2j-2}^{n+1}X_j^1, a_{2j-1}^{n+1}a_{2j}^{n+1}X_{j+1}^1, \ldots, a_{2m-1}^{n+1}a_{2m}^{n+1}X_{m+1}^1).$$

We have found a derivation according to π for the configuration requested by the induction step, therefore the claim is proved.

The membership of the word $a_1^n a_2^n \ldots a_{2m}^n$ in $L(\pi)$ for every $n \geq 1$ follows now from the claim. Indeed, we replace the last step of the derivation D (in which a new round is started) with a subderivation which plays the role of collecting all the strings in the first component, in the correct order.

$$\begin{aligned}
(S_1, S_2, \ldots, S_{m+1}) &\Longrightarrow^* (S_1, a_1^n a_2^n X_2^{2m}, a_3^n a_4^n X_3^{2m}, \ldots, a_{2m-1}^n a_{2m}^n X_{m+1}^{2m}) \\
&\Longrightarrow (a_1Q_2, a_1^n a_2^{n+1} a_3 Q_3, a_3^n a_4^{n+1} a_5 Q_4, \ldots, a_{2m-1}^n a_{2m}^{n+1} Z) \\
&\Longrightarrow^* (a_1^{n+1} a_2^{n+1} \ldots a_{2m-1}^{n+1} a_{2m}^{n+1} Z, S_2, \ldots, S_{m+1}) \\
&\Longrightarrow (a_1^{n+1} a_2^{n+1} \ldots a_{2m-1}^{n+1} a_{2m}^{n+1} Z', Y_2, \ldots, Y_{m+1}) \\
&\Longrightarrow (a_1^{n+1} a_2^{n+1} \ldots a_{2m-1}^{n+1} a_{2m}^{n+1}, Y_2, \ldots, Y_{m+1})
\end{aligned}$$

The converse inclusion follows because, except the alternative of stoping the derivation, the use of other productions than the ones we have actually applied leads to the blocking of the derivation (either by introducing nonterminals that cannot be rewritten or by introducing circular communication requests). This implies that the only words that can be generated by the PCGS π are the ones of the form $a_1^n a_2^n \ldots a_{2m}^n$.

We have therefore proved that $L(\pi) = L$, which shows that L belongs to $\mathcal{L}(\text{two-way array} - PCGS_{m+1})$.

It has been proved in [14] that the language L cannot be generated by any PCGS with m grammars, regardless of the shape of its communication graph.

Consequently, we have showed a stronger result than the one stated in the theorem. For all $m > 1$ there exists a language 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 5 *The hierarchy* $\{\mathcal{L}(\textit{two-way array-PCGS}_n)\}_{n\geq 1}$ *is infinite.*

Corollary 6 *The hierarchy* $\{\mathcal{L}(\textit{two-way ring-PCGS}_n)\}_{n\geq 1}$ *is infinite.*

Proof. A two-way array is a two-way ring where one of the arcs is deleted. Consequently the preceding theorem holds also for two-way rings. □

The language used in the proof of Theorem 6 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 7 *For all $m \geq 1$,*

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

Proof. Consider the language L from the proof of Theorem 6. We shall show in the following that L can be generated by a one-way ring-PCGS π of degree $m+1$, where the sense of the arrows in the ring is clock-wise and

$$\begin{aligned}
\pi &= (G_1, G_2, \ldots, G_{m+1}),\\
G_i &= (V_{N,i}, \Sigma, S_i, P_i), 1 \leq i \leq m+1,\\
\Sigma &= \{a_1, a_2, \ldots, a_{2m}\},\\
V_{N,1} &= \{S_1, A_1, Q_2, Z, Z'\} \cup \{X_2^k \mid 1 \leq k \leq m+1\} \cup\\
&\quad \{B_2^k \mid 2 \leq k \leq m+1\} \cup \{Y_i^k \mid 2 \leq i \leq m+1, 1 \leq k \leq m^2\},\\
V_{N,j} &= \{S_j, A_j, Q_{j+1}\} \cup \{X_j^k \mid 1 \leq k \leq j\} \cup \{B_j^k \mid 1 \leq k \leq j\} \cup\\
&\quad \{X_{j+1}^k \mid j \leq k \leq m+1\} \cup \{B_{j+1}^k \mid j+1 \leq k \leq m\} \cup\\
&\quad \{Y_i^k \mid 2 \leq i \leq m+1, 1 \leq k \leq m^2\}, 2 \leq j \leq m\\
V_{N,m+1} &= \{S_{m+1}, C, C', Z, Q_1\} \cup \{X_{m+1}^k \mid 1 \leq k \leq m\} \cup\\
&\quad \{B_{m+1}^k \mid 1 \leq k \leq m\} \cup \{Y_i^k \mid 2 \leq i \leq m+1, 1 \leq k \leq m^2\},\\
P_1 &= \{S_1 \longrightarrow a_1 A_1, A_1 \longrightarrow Q_2, X_2^1 \longrightarrow a_2 B_2^2, B_2^2 \longrightarrow X_2^2\} \cup\\
&\quad \{X_2^k \longrightarrow B_2^{k+1}, B_2^{k+1} \longrightarrow X_2^{k+1} \mid 2 \leq k \leq m\} \cup\\
&\quad \{X_2^{m+1} \longrightarrow Y_2^1\} \cup \{Y_i^k \longrightarrow Y_i^{k+1} \mid 1 \leq k < m^2, 2 \leq i \leq m+1\} \cup\\
&\quad \{S_1 \longrightarrow Q_2, Z \longrightarrow Z', Z' \longrightarrow \lambda\},\\
P_j &= \{S_j \longrightarrow B_j^1, B_j^1 \longrightarrow X_j^1\} \cup \{S_j \longrightarrow a_{2j-1} A_j, A_j \longrightarrow Q_{j+1}\} \cup\\
&\quad \{X_j^k \longrightarrow B_j^{k+1}, B_j^{k+1} \longrightarrow X_j^{k+1} \mid 1 \leq k < j-1\} \cup\\
&\quad \{X_{j+1}^j \longrightarrow a_{2j} B_{j+1}^{j+1}, B_{j+1}^{j+1} \longrightarrow X_{j+1}^{j+1}\} \cup\\
&\quad \{X_{j+1}^k \longrightarrow B_{j+1}^k, B_{j+1}^k \longrightarrow X_{j+1}^{k+1} \mid j < k \leq m\} \cup\\
&\quad \{X_{j+1}^{m+1} \longrightarrow Y_{j+1}^1\} \cup \{Y_i^k \longrightarrow Y_i^{k+1} \mid 1 \leq k < m^2, 2 \leq i \leq m+1\} \cup\\
&\quad \{S_j \longrightarrow Q_{j+1}\} \cup \{Y_j^{m^2} \longrightarrow B_j^1\} \cup \{Y_j^{m^2} \longrightarrow Q_{j+1}\}, 2 \leq j \leq m,\\
P_{m+1} &= \{S_{m+1} \longrightarrow B_{m+1}^1, B_{m+1}^1 \longrightarrow X_{m+1}^1\} \cup\\
&\quad \{X_{m+1}^k \longrightarrow B_{m+1}^{k+1}, B_{m+1}^{k+1} \longrightarrow X_{m+1}^{k+1} \mid 1 \leq k < m\} \cup\\
&\quad \{S_{m+1} \longrightarrow C, C \longrightarrow C', C' \longrightarrow Q_1\} \cup \{S_{m+1} \longrightarrow Q_1\} \cup\\
&\quad \{Y_i^k \longrightarrow Y_i^{k+1} \mid 1 \leq k < m^2, 2 \leq i \leq m+1\} \cup\\
&\quad \{Y_{m+1}^{m^2} \longrightarrow B_{m+1}^1, Y_{m+1}^{m^2} \longrightarrow Z\}.
\end{aligned}$$

The proof of the fact that $L = L(\pi)$ is similar to that of Theorem 6. We shall omit the proof and explain instead in an informal way how the one-way ring-PCGS works.

The main idea is that each grammar G_{j+1}, $0 < j < m$, has to produce a sentential form $a_{2j-1}^n a_{2j}^n D$ for any n, where D is an arbitrary nonterminal. In order to increase the exponent of a_{2j}, a rule of P_{j+1} can be used. In order to increase the exponent of a_{2j-1}, the sentential form is communicated to the left grammar, i.e. G_j, and a rule of P_j produces the necessary a_{2j-1}. The communication to the left is always possible because the communication graph is a one-way ring where we considered the sense of the arrows to be clock-wise.

Two problems occur in performing the above described operation. The first one appeared already in the two-way array. It refers to the fact that the changing of the strings $a_{2j-1}a_{2j}$ is done successively, not simultaneously. This means that the components in the "waiting status" have to perform only renamings. In order to preserve the synchronization of the exponents, the upper indices of the nonterminals B_j^k and X_j^k will count the steps. This helps also to prevent communications to occur at undesirable moments: only components with certain upper indices can be rewritten in the neighbour grammars. The lower indices of the nonterminals refer to the indices of the corresponding grammar. Notice that in any grammar G_j, only nonterminals X and B with lower index j or $j+1$ can be rewritten.

The second problem refers to the fact that after an increase of the exponent n in $a_{2j-1}^n a_{2j}^n D$ has been accomplished for all j, the sentential forms are all in the wrong position. More explicitely, they are shifted one position to the left. In order to be able to repeat the process, the sentential forms have to return to their old positions. This cannot be accomplished by a shift to the right, because the ring is one-way. Therefore we have to rotate all components m positions to the left in order to obtain the changing of the position of *one* of the components to the right. In this moment the nonterminals Y_i^k enter the stage. The most important thing about them is the upper index which counts the number of steps. Their upper index can be updated in any grammar, as long as it is smaller than m^2. They can be changed into B_j^1 only after all the components have reached their correct places, that is, m rotations for each of the m grammars have been performed. The upper index takes care of the fact that no undesired rule is applied. Indeed, if this would happen, then Y would reach its grammar with wrong index, and the derivation would be blocked.

In the end of the derivation, we want to collect all the strings in the grammar G_1. This is done by simultaneously producing communication symbols in all the grammars. This will, in turn, trigger a chain-communication whose effect will be the catenation of the strings $a_{2j-1}^n a_{2j}^n$ in the correct order into G_1.

The above explanations show that $L = L(\pi)$. In [14] it has been proved that L cannot be generated by a PCGS with m components, regardless of the shape of its communication graph. Consequently, for any $m > 1$ we have found a language L that can be generated by a one-way ring-PCGS with $m+1$ components, but cannot be generated by any PCGS of smaller order. This result implies the relation requested by the theorem. □

Corollary 7 *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.