Parallel Communicating Grammar Systems

Lila Santean
Academy of Finland and Mathematics Department
University of Turku
20500 Turku, Finland

Several attempts have been made to find a suitable model for parallel processing. Cellular automata [9], Lindenmayer systems [18], systolic trellis automata [3], Russian parallel [4] and Indian parallel [4] grammars are some examples of such models based on formal language and automata theories. In these devices the parallelism is local. Symbols are rewritten independently of each other. No major cooperation between the parallel processes occurs, although, for instance, in L-systems with interactions some minor cooperation appears.

However, the development of massively parallel processing systems increased the importance of interprocessor communication in the new generation computer design. Communication plays a major role in parallel processing architectures, where inappropriate interconnection topologies could lengthen the paths of messages, reduce the system reliability and introduce most embarassing performance limitations.

Cooperating /distributed grammar systems are an attempt of modelling the process of communication [2]. They consist of a system of grammars working together to produce words of a language. Each of the grammars rewrites the sentential form until a certain condition is met. Then it passes it to the next grammar, and so on, until a terminal string is obtained. The model captures the main features of a communication process, but the individual grammars work sequentially in the sense that, at each moment, only one grammar is allowed to rewrite the sentential form.

P*arallel* **C***ommunicating* **G***rammar* **S***ystems* are aimed to combine the notions of *parallelism* and *communication* into a suitable model for theoretical investigation of the properties of parallel processing systems.

Parallel communicating grammar systems have evolved from the following considerations:

- Knowledge processing systems are characterized by an intimate cooperation between logic and functional programming, which require an adequate communication discipline;

- Processing requirements for knowledge based problem solving are of a different nature, making heterogeneous parallel systems more appropriate;
- Although interprocess communication could be decided at process level, overall supervision is necessary for efficient task distribution, resource allocation and management.

Parallel communicating grammar systems (PCGS, for short) were introduced in [16] and their properties such as generative capacity, syntactic complexity, closure with respect to various operations, decision problems, have been studied in [1], [5], [11]—[17], [20].

A PCGS of degree n consists of n separated rewriting systems, say Chomsky grammars. One of the grammars is distinguished: the language it generates, by cooperating with the other grammars, is the language of the system. Each grammar has its own vocabularies, axioms and rewriting rules. As sentential forms can be transmitted between grammars, no terminal symbol from one grammar may be nonterminal for other.

Grammars in a PCGS work in parallel, each of them starting from its own axiom and, in well defined circumstances, communicate with each other. The moments of communication depend on the *query symbols* appearing in the current sentential forms generated by the grammars. Query symbols are special nonterminals, indexed from *1* to n, to refer to the grammars. Such a symbol may belong to the nonterminal vocabulary of any grammar, except the one whose index it bears (a grammar cannot transmit strings to itself — communication is not reflexive).

The appearance of a query symbol in any of the sentential forms imposes a communication, as the query symbols are nonterminals that cannot be rewritten. A communication consists of the replacing of all the query symbols with the current strings of the grammars they refer to. However, a restriction is imposed: no replacing takes place for the sentential forms containing query symbols referring to strings containing further query symbols. Circular communications are not admitted. The status of the grammar which has sent its current string depends on the type of the PCGS considered. The grammar may continue working (non-returning PCGS) or may erase its string and resume working from the axiom (returning PCGS). The language generated by a PCGS consists of all the terminal strings generated by the distinguished grammar, regardless the status of the others (whether their current strings are terminal or not).

We give the definition of a PCGS where:

- the components are Chomsky grammars;
- the sending grammars resume working from the axiom after each communication;
- the grammars work synchronously.

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

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

where each G_i is a Chomsky grammar, $G_i = (V_{N,i}, V_{T,i}, S_i, P_i), 1 \leq i \leq n$, such that $V_{N,i} \bigcap V_{T,j} = \emptyset$ for all $i, j \in \{1, 2, \ldots, n\}$, and there is a set $K \subseteq \{Q_1, Q_2, \ldots, Q_n\}$, of query symbols, $K \subseteq \bigcup_{i=1}^{n} V_{N,i}$.

Definition 2 *A configuration in a PCGS of degree n is an n-tuple $(x_1, \ldots, x_n)$ where $x_i \in V_{G,i}^*$ $(V_{G,i} = V_{T,i} \bigcup V_{N,i})$, for all $1 \leq i \leq n$.*

Depending on the context, we name "component i" either the grammar G_i or its string $x_i \in V_{G,i}^*$ in the current configuration. If x is a string over an alphabet V and $V' \subseteq V$, $|x|_{V'}$ denotes the number of occurrences of letters of V' in x.

Definition 3 *For two configurations $(x_1, x_2, \ldots, x_n)$, $(y_1, y_2, \ldots, y_n)$ in a PCGS $\pi = (G_1, \ldots, G_n)$, we write*

$$(x_1, x_2, \ldots, x_n) \Longrightarrow (y_1, y_2, \ldots, y_n)$$

if one of the next two cases holds:

(i) *$|x_i|_K = 0$ for all $i, 1 \leq i \leq n$, and for each $i, 1 \leq i \leq n$, we have either $x_i \Longrightarrow y_i$ in the grammar G_i, or $x_i \in V_{T,i}^*$ and $x_i = y_i$;*

(ii) *there is an $i, 1 \leq i \leq n$, such that $|x_i|_K > 0$; then for each such i we write $x_i = z_1 Q_{i_1} z_2 Q_{i_2} \ldots z_t Q_{i_t} z_{t+1}, t \geq 1$, with $z_j \in V_{G,i}^*, |z_j|_K = 0$, $1 \leq j \leq t+1$; if $|x_{i_j}|_K = 0, 1 \leq j \leq t$, then $y_i = z_1 x_{i_1} z_2 x_{i_2} \ldots z_t x_{i_t} z_{t+1}$ and $y_{i_j} = S_{i_j}, 1 \leq j \leq t$; when, for some $j, 1 \leq j \leq t, |x_{i_j}|_K \neq 0$, then $y_i = x_i$; for all the remaining indexes r, we put $y_r = x_r$.*

A derivation consists of *rewriting steps* **(i)** and *communication steps* **(ii)**. If no query 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 none of the nonterminals can be rewritten any more, the derivation is blocked.

If in any of the components a query symbol is present, a *communication step* is performed. It consists of the replacing of all the occurrences of query symbols with the components they refer to, providing these components do not contain further query symbols. More exactly, a component is modified only when all its occurrences of query symbols refer to strings without query symbols. In a communication operation, the communicated strings replace the corresponding query symbols (we say that the query symbols are *satisfied* in this way). After the communication, the sending grammars resume working from the axiom. If some query symbols are not satisfied in this step, they may be satisfied in one

of the next ones. Communication steps are performed until no more query symbols are present. No rewriting is allowed if a query symbol occurs in one of the components of the configuration. Therefore, if circular queries emerge, the derivation is blocked.

Definition 4 *The language generated by the PCGS* $\pi = (G_1, \ldots, G_n)$ *is :*

$$L(\pi) = \{\alpha \in V_{T,1}^* | (S_1, S_2, \ldots, S_n) \Longrightarrow^* (\alpha, \beta_2, \beta_3, \ldots, \beta_n), \beta_i \in V_{G,i}^*, 2 \leq i \leq n\}.$$

If we impose the restriction that only the first grammar may ask for strings generated by the others, that is $K \bigcap (\bigcup_{i=2}^{n} V_{N,i}) = \emptyset$, we say that π is a *centralized* PCGS; in contrast, the unrestricted case is called *non-centralized*.

Moreover, the above definitions refer to *returning* PCGS's (after communicating, each component whose string has been sent to another component *returns to axiom*). A PCGS is *non-returning* if in the point **(ii)** of the definition we remove the following words: "and $y_{i_j} = S_{i_j}, 1 \leq i \leq t$". That means, after communicating, the grammar G_{i_j} does not return to S_{i_j}, but continues to process the current string.

A PCGS is said to be regular, context-free, context-sensitive, λ-free etc. when all the component grammars $G_1, \ldots, G_n$ are of these types. Depending on the context, REG, LIN, CF, CS, RE denote the classes of λ-free regular, λ-free linear, λ-free context-free, context-sensitive, recursively enumerable grammars, or the family of languages generated by them. If the subscript λ is added, we are referring to such grammars which contain λ-rules. Let X be one of the above mentioned classes of grammars. We denote by $PC_n(X)$ the family of languages generated by non-centralized returning PCGS's of type X, of degree at most $n, n \geq 1$; when centralized PCGS's are used, the corresponding families are denoted $CPC_n(X)$. When non-returning PCGS's are considered, we denote the families $NPC_n(X), NCPC_n(X)$. Denote also

$$PC(X) = \bigcup_{n \geq 1} PC_n(X)$$

and similarly for $CPC, NPC, NCPC$ (the families of languages generated by PCGS's of given types, of arbitrary degree).

Example 1 Let $\pi = (G_1, G_2, G_3)$ where

$$\begin{aligned}
G_1 &= (\{S_1, B, B_1, Q_2\}, \{a\}, S_1, \{S_1 \longrightarrow aB, \\
&\quad S_1 \longrightarrow Q_2, B_1 \longrightarrow B, B_1 \longrightarrow \lambda\}) \\
G_2 &= (\{S_2, B, Q_1, Q_3\}, \{a\}, S_2, \{S_2 \longrightarrow Q_1, B \longrightarrow Q_3\}) \\
G_3 &= (\{S_3, Q_1, B, B_1\}, \{a\}, S_3, \{S_3 \longrightarrow Q_1, B \longrightarrow B_1\}).
\end{aligned}$$

A derivation according to π will have the following form:

$$(S_1, S_2, S_3) \quad \Longrightarrow \quad (aB, Q_1, Q_1) \Longrightarrow (S_1, aB, aB) \Longrightarrow (Q_2, aQ_3, aB_1)$$

$$
\begin{aligned}
\Longrightarrow \quad & (Q_2, a^2B_1, S_3) \Longrightarrow (a^2B_1, S_2, S_3) \\
\Longrightarrow \quad & (a^2B, Q_1, Q_1) \Longrightarrow (S_1, a^2B, a^2B) \\
\Longrightarrow^* \quad & (a^{2^{n-1}}B, Q_1, Q_1) \Longrightarrow (S_1, a^{2^{n-1}}B, a^{2^{n-1}}B) \\
\Longrightarrow \quad & (Q_2, a^{2^{n-1}}Q_3, a^{2^{n-1}}B_1) \Longrightarrow (Q_2, a^{2^n}B_1, S_3) \\
\Longrightarrow \quad & (a^{2^n}B_1, S_2, S_3) \Longrightarrow (a^{2^n}, Q_1, Q_1), \text{for any } n \geq 1.
\end{aligned}
$$

We notice that if $S_1 \longrightarrow aB$ is applied to a configuration (S_1, a^iB, a^iB) then the derivation is blocked:

$$(S_1, a^iB, a^iB) \Longrightarrow (aB, a^iQ_3, a^iB_1) \Longrightarrow (aB, a^{2i}B_1, S_3),$$

and B_1 cannot be rewritten in G_2. The same thing happens if we apply the rule $S_1 \longrightarrow Q_2$ to the configuration (S_1, S_2, S_3). We conclude that

$$L(\pi) = \{a^{2^n} | n \geq 1\},$$

where π is a non-centralized returning regular PCGS of degree 3.

Example 2 Consider following centralized non-returning regular PCGS $\pi = (G_1, G_2)$ where,

$$
\begin{aligned}
G_1 \quad = \quad & (\{S_1, S_2, Q_2\}, \{a, b, c\}, S_1, \{S_1 \longrightarrow aS_1, \\
& S_1 \longrightarrow aQ_2, S_2 \longrightarrow cQ_2, S_2 \longrightarrow c\}) \\
G_2 \quad = \quad & (\{S_2\}, \{b\}, S_2, \{S_2 \longrightarrow bS_2\})
\end{aligned}
$$

The derivations in π are of the following form:

$$
\begin{aligned}
(S_1, S_2) \quad \Longrightarrow^* \quad & (a^kS_1, b^kS_2) \Longrightarrow (a^{k+1}Q_2, b^{k+1}S_2) \\
\Longrightarrow \quad & (a^{k+1}b^{k+1}S_2, b^{k+1}S_2) \Longrightarrow (a^{k+1}b^{k+1}cQ_2, b^{k+2}S_2) \\
\Longrightarrow \quad & (a^{k+1}b^{k+1}cb^{k+2}S_2, b^{k+2}S_2) \\
\Longrightarrow^* \quad & (a^{k+1}b^{k+1}cb^{k+2}cb^{k+3} \ldots cb^{k+j}c, b^{k+j}S_2),
\end{aligned}
$$

that is

$$L(\pi) = \{a^nb^ncb^{n+1}c \ldots cb^{n+j}c | n \geq 1, j \geq 0\}$$

which is a non-context-free language.

As the above examples show, the generative capacity of PCGS's is much larger than that of the corresponding rewriting components: a PCGS with two or three regular grammar components can generate non-context-free languages. This indicates that an increase of the number of components adds generative power, that is, parallelism and communication are indeed useful. It has been proved in [20] that the hierarchy of classes of languages generated by regular returning PCGS's is infinite, where one class is determined by the number of components. For the centralized case, the proof uses the following auxiliary result:

Lemma 1 *(Pumping lemma) Let L be a language in $CPC_n(REG)$. There exists a natural number N such that every word α in L satisfying $|\alpha| > N$ can be decomposed as*

$$\alpha = \alpha_1\beta_1\alpha_2\beta_2 \ldots \alpha_m\beta_m\alpha_{m+1}$$

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

$$\alpha_1\beta_1^k\alpha_2\beta_2^k \ldots \alpha_m\beta_m^k\alpha_{m+1}$$

is in L for all $k \geq 0$.

However, for the noncentralized case a direct example has to be used as, due to example **1**, a similar lemma cannot be proved. For the context-sensitive case we have no hierarchy as it has been proved in [1] that

$$CS = CPC_n(CS) = CPC(CS) = NCPC_n(CS) = NCPC(CS), n \geq 1.$$

Relations between classes of languages generated by PCGS's and other language families:

- $CPC_n(REG), NCPC_n(REG)$ are incomparable with LIN for $n \geq 2$,
- $CPC_n(REG), NCPC_n(REG)$ are incomparable with CF for $n \geq 3$,
- $CPC_2(REG) \subset CF$, strict inclusion,
- each language in $CPC(REG), CPC(LIN)$ is semi-linear,
- the families $NPC_2(REG), NCPC_2(REG), PC_3(REG), CPC_3(CF)$ contain non-semi-linear languages,
- $CPC_2(REG)$ contains languages which are not linear simple matrix,
- $CPC_2(CF)$ contains languages which are not simple matrix (see [19] for definitions).

We have discussed so far only the increase in the generative power obtained by means of parallelism, without paying attention to the degree of communication. The parameter *com* has been introduced and studied in [14], [17], as a measure of *communication*.

Definition 5 *Consider a PCGS π and a derivation in π:*

$$\begin{aligned} D: \quad & (S_1, S_2, \ldots, S_n) \Longrightarrow (w_{1,1}, w_{1,2}, \ldots, w_{1,n}) \Longrightarrow \\ & (w_{2,1}, w_{2,2}, \ldots, w_{2,n}) \Longrightarrow^* (w_{k,1}, w_{k,2}, \ldots, w_{k,n}) \end{aligned}$$

Denote:

$$\begin{aligned} & com((w_{i,1}, w_{i,2}, \ldots, w_{i,n})) = \textstyle\sum_{j=1}^{n} |w_{i,j}|_K \\ & com(D) = \textstyle\sum_{i=1}^{k} com((w_{i,1}, w_{i,2}, \ldots, w_{i,n})) \end{aligned}$$

For $x \in L(\pi)$ define

$$com(x,\pi) = min\{com(D) | D : (S_1, S_2, \ldots, S_n) \Longrightarrow^* (x, \alpha_2, \ldots, \alpha_n)\}.$$

Then,

$$com(\pi) = sup\{com(x,\pi) | x \in L(\pi)\},$$

and, for a language L and a class X, $X \in \{PC, CPC, NPC, NCPC\}$,

$$com_X(L) = inf\{com(\pi) | L = L(\pi), \pi \in X\}.$$

The parameter *com* evaluates the total number of query symbols appearing in a derivation. We consider only centralized returning PCGS's, hence we do not specify the class X of PCGS's and write *com* for com_{CPC}. The following theorem states that the increase of communication influences the generative power:

Theorem 1 *$CPC_n(REG) \subset PC_n(REG), n \geq 1$, strict inclusion.*

A more general result, which shows the impact of the parameter *com* is:

Theorem 2 *If π is a regular (centralized or noncentralized) returning PCGS such that $com(\pi) = 1$, then $L(\pi)$ is context-free.*

As example **1** indicates, regular PCGS's can generate also non-context-free languages, therefore in this case only the communication has caused the increase in generative capacity.

Another interesting characteristic which, as parallelism and communication, can modify the power of a PCGS is the *synchronization*. Until now, we have only considered synchronized derivations in a PCGS, that is, each grammar uses exactly one rule in a derivation step, the only components which may "wait" being the terminal ones. What about the case when the grammars may wait without restrictions? The problem has been raised by J.Hromkovic (Bratislava) during *IMYCS'88, Smolenice*, and studied in [11]. Formally, for defining an *unsynchronized derivation* in a PCGS π, we replace the condition **(i)** in the definition **3** by:

(i') $|x_i|_K = 0, 1 \leq i \leq n$, and for each $i, 1 \leq i \leq n$ we have either $x_i \Longrightarrow y_i$ in the grammar G_i or $x_i = y_i$ (in a non-communication step, each grammar may either use a rule or wait).

We denote by $L_u(\pi)$ the languages generated in this way.

Example 3 Consider the centralized, non-returning unsynchronized context-free PCGS $\pi = (G_1, G_2)$ with

$$\begin{aligned} G_1 &= (\{S_1, Q_2\}, \{a, b\}, \{S_1 \longrightarrow Q_2Q_2Q_2\})\} \\ G_2 &= (\{S_2\}, \{a, b\}, S_2, \{S_2 \longrightarrow aS_2, S_2 \longrightarrow b\}) \end{aligned}$$

Each terminal derivation in π must contain a step where the rule $S_1 \longrightarrow Q_2Q_2Q_2$ is used in G_1. This means a communication step must follow, which will communicate to G_1 a terminal string in G_2 (G_1 cannot rewrite the symbol S_2). Therefore,

$$L_u(\pi) = \{a^nba^nba^nb | n \geq 0\}$$

which is not a context-free language.

As to be expected, the synchronization is useful, the unsynchronized PCGS's are strictly weaker than the synchronized ones of the same type. An extreme situation is that unsynchronized regular and linear centralized returning PCGS's can be simulated by usual regular and linear grammars, respectively.

We end the considerations about the generative capacity of PCGS's considering such systems which have L systems as components. They combine the local parallelism at string-rewriting level with the parallelism at component level. The definition of an L system PCGS is obvious (similar to the definitions of grammar PCGS's, with the derivation in the L-sense [18]).

As in the case of grammar PCGS's, the generative power of PCGS's with L-components is larger than that of the corresponding type of components. This has been proved in [15] for OL, DOL, EOL, EDOL, TOL, DTOL, EDTOL, ETOL systems. For instance, a PCGS with two DTOL components can generate languages which are not ETOL, the largest family of interactionless L languages.

Example 4 Consider the returning PCGS $\pi = (G_1, G_2)$ where

$$\begin{aligned} G_1 &= (\{a, b, c, d, e, Q_2\}, ec, \{\{e \longrightarrow ec, c \longrightarrow c\} \\ &\quad \{e \longrightarrow d, c \longrightarrow Q_2\}\}) \\ G_2 &= (\{a, b\}, a, \{\{a \longrightarrow ab, b \longrightarrow b\}\}) \end{aligned}$$

The derivations in π are of the following form:

$$\begin{aligned} (ec, a) \Longrightarrow^* &\ (ec^{k+1}, ab^k) \Longrightarrow (dQ_2^{k+1}, ab^{k+1}) \Longrightarrow \\ &\ (d(ab^{k+1})^{k+1}, a), k \geq 0. \end{aligned}$$

Therefore,

$$L(\pi) = \{ec^n | n \geq 1\} \bigcup \{d(ab^n)^n | n \geq 1\},$$

which is not an ETOL language.

However, the EDOL PCGS's cannot generate languages not in EDTOL. This result says that, in the deterministic case, the parallel work of EOL systems can be simulated by tables, also deterministic.

In general it seems to be hard to say something about closure properties of languages generated by PCGS's because, on the one hand, it is not easy to prove positive results and, on the other hand, no languages not in $CPC(CF)$, $PC(CF)$ and other such families are known. In [1], [17] it has been proved that the family

$PC(CF)$ is closed under union, concatenation, Kleene closure, substitution by λ-free context-free languages, and intersection by regular sets. The families $PC(LIN), NPC(LIN)$ are closed under union. The family $PC(CF_\lambda)$ is a full AFL.

As concerning the decidability results, it is shown in [1] that:

- it is undecidable if an arbitrarily given context-free PCGS generates a context-free (or right-linear simple matrix or simple matrix) language;
- the emptiness and the finiteness problems are decidable for linear centralized returning PCGS's.

The *syntactic complexity* measures *var* (number of nonterminals), *prod* (number of productions), *symb* (the sum of the lengths of the right-sides of the productions) defined in [6], [7] for context-free grammars were generalized for context-free PCGS's in [14]. Concerning the measure *com*, it has been proved that it is a connected measure (for each $n \geq n_0, n_0$ a given constant, there exists an L_n such that $com(L_n) = n$) over *CPC(CF)*. Obviously, the parameters $var, prod, symb$ can be computed for an arbitrary PCGS by a simple counting. The situation is different for the measure *com* due to its *dynamical* character (it is evaluated on an infinite set, namely, that of all possible derivations). Therefore, it has been shown that:

Theorem 3 *For an arbitrarily given context-free centralized returning PCGS π, $com(\pi)$ and $com(L(\pi))$ cannot be algorithmicaly computed.*

Theorem 4 *It is not decidable whether $com(\pi) = com(L(\pi))$, for an arbitrarily given context-free, centralized, returning PCGS π.*

Theorem 5 *Let us assume that π is a regular (linear) returning, non-centralized PCGS. If $com(\pi) < \infty$ then $L_u(\pi) \in REG$ (LIN, respectively).*

The measure *com* is incompatible with each of the measures $var, prod, symb$ (they cannot be simultaneously minimized).

Another complexity parameter is *time*:

Definition 6 *Given a grammar $G = (V_N, V_T, S, P)$ and a derivation $D : S \Longrightarrow w_1 \Longrightarrow \ldots \Longrightarrow w_n$ we put $time(D) = n$ and, for $x \in L(G)$,*

$$time_G(x) = min\{time(D) | D : S \Longrightarrow^* x \text{ in } G\}.$$

A mapping $time_G : L(G) \longrightarrow \mathbf{N}$ is obtained in this way. A similar definition holds for any type of generative devices, including PCGS's (both the rewriting and the communication steps are counted).

The following result [17] shows that in generating a linear language using a PCGS instead of a grammar, any linear speed-up can be obtained. Moreover the syntactic complexity of the obtained PCGS is not too big.

Theorem 6 *Let L be an infinite linear language and G a linear grammar such that $L = L(G), var(G) = p$. For each given natural number t there is a centralized (returning or non-returning) linear PCGS such that $L = L(\pi)$ and*

$$\begin{aligned} var(\pi) &= var(G) + pt \\ prod(\pi) &= prod(G) + p(t+1) \\ symb(\pi) &= symb(G) + 3p(t+1) \end{aligned}$$

and, for each $x \in L(G)$ we have

$$time_\pi(x) < (1/t)time_G(x) + 3t.$$

Recently, in [21], a new type of PCGS's has been investigated. The components of such PCGS's are placed in the vertices of a given *communication graph*, and only the communications on the edges of this graph are possible.

Denote by $x-PCGS_n$ the class of PCGS's of degree n whose communication graph is of type x, where $x \in \{$C(entralized), directed acyclic graph (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's of this type.

In [21], the descriptional complexity (communication structure) and the computational complexity (number of communications) of such PCGS's are investigated. Several hierarchies resulting from these complexity measures and some relations between the measures are established. Namely, the following hierarchies are proved to be infinite:

$$\begin{aligned} &\{\mathcal{L}(\text{one-way ring} - PCGS_n)\}_{n\geq 1}, \\ &\{\mathcal{L}(\text{two-way ring} - PCGS_n)\}_{n\geq 1}, \\ &\{\mathcal{L}(\text{two-way array} - PCGS_n)\}_{n\geq 1}, \\ &\{\mathcal{L}(\text{dag} - PCGS_n)\}_{n\geq 1}, \\ &\{\mathcal{L}(\text{tree} - PCGS_n)\}_{n\geq 1}, \end{aligned}$$

Moreover, for any function $f : \mathbf{N} \longrightarrow \mathbf{N}$, $f(n) \notin \Omega(n)$, and any $m \in \mathbf{N}$, $m \geq 2$, we have:

$$\begin{aligned} &\mathcal{L}(\text{one-way array} - PCGS_m(f(n)) \subset \mathcal{L}(\text{one-way array-}PCGS_m(n)), \\ &\mathcal{L}(\text{C} - PCGS_m(f(n)) \subset \mathcal{L}(\text{C-}PCGS_m(n)), \\ &\mathcal{L}(\text{tree} - PCGS_m(f(n)) \subset \mathcal{L}(\text{tree-}PCGS_m(n)), \end{aligned}$$

and for any positive integer k and any $x \in \{$C, tree, one-way array$\}$ we have:

$$\begin{aligned} &\mathcal{L}(x - PCGS_{k+1}(k-1)) \subset \mathcal{L}(x - PCGS_{k+1}(k)), \\ &\cup_{m\in N}\mathcal{L}(x - PCGS_m(k-1)) \subset \cup_{m\in N}\mathcal{L}(x - PCGS_m(k)). \end{aligned}$$

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 classical formal language theory and the second one reduces the lower bound problem for some PCGS's to the proof of lower bounds on the number of reversals of certain sequential computing models.

As the study of PCGS's is just starting, a wealth of questions remain to be investigated. Many specific problems have remained open as regards the generative capacity, the closure properties, complexity, etc. We list here some of them:

- Are the hierarchies induced by the degree of context-free PCGS's infinite?
- What is the relation between CS and $CPC(CF_\lambda)$?
- Are the inclusions $CPC_n(CF) \subseteq CPC_n(CF_\lambda), n \geq 1$, proper?
- Are the inclusions $CPC_n(X) \subseteq PC_n(X)$ proper for $X \in \{CF, CF_\lambda\}$?
- Is it decidable whether, for an arbitrary regular PCGS π we have $com(\pi) = com(L(\pi))$?

The following extentions of PCGS's suggest themselves rather naturally:

- the communication is not only by *request* but also by *command*. That is, under certain circumstances, a grammar could impose the sending of its string to another grammar;
- further restrictions are imposed on the strings that may be communicated or on the strings of a final configuration (for example, we can impose that all are terminal strings);
- the parallelism is *partial*. According to a time dependent control structure, only some of the components work in parallel, the others being in a waiting status;
- the PCGS has any type of rewriting systems as components. In fact, in [1] the study of pure grammar ([19]) and contextual grammar ([10]) PCGS's has already been initiated;
- the components of the PCGS are *dynamic*. That is, they change their set of rules in a dynamic way during a derivation, commanded by a control structure;
- the PCGS is not homogeneous, but consists of rewriting systems of different types, clustered upon the functions they have to perform.

References

[1] E.Csuhaj-Varju,J.Dassow,J.Kelemen,Gh.Păun. *Grammar Systems.* (to appear).

[2] E.Csuhaj-Varju, J.Kelemen. Cooperating grammar systems: a syntactical framework for blackboard model of problem solving. *AI Control Syst. of Robots* (Ed.I.Plander), Elsevier Sci.Publ. (1989).

[3] K.Culik, J.Gruska, A.Salomaa. Systolic trellis automata. *International Journal of Computer Mathematics* **15**, **16** (1984).

[4] J.Dassow,Gh.Păun. *Regulated Rewriting in Formal Language Theory.* Akademie-Verlag, Berlin (1989), Springer-Verlag, Berlin (1990).

[5] I.Georgescu, Gh.Păun, L.Santean. Parallel communicating grammar systems—a grammatical approach to parallel processing. *Res. Rep. 14-89,Institute for Informatics*, Bucharest (1989).

[6] J.Gruska. On a classification of context-free languages, *Kybernetica* **1,3** (1967).

[7] J.Gruska. Descriptional complexity of context-free languages, *Proc. MFCS'73*, High Tatras (1973).

[8] C.A.R.Hoare. Communicating sequential processes. *Comm.ACM* **1,8**(1978).

[9] J.Kari. Games played on the plane: solitaire & cellular automata (The formal language theory column). *EATCS Bulletin* **40** (1990).

[10] S.Marcus. Contextual grammars. *Rev.Roumaine Math.Pures Appl.* **14**, 10(1969).

[11] Gh.Păun. On the power of synchronization in parallel communicating grammar systems. *Stud.Cerc.Matem.***41**, 3(1989).

[12] Gh.Păun. Parallel communicating grammar systems: the context-free case. *Found. Control Engineering* **14** vol.1 (1989).

[13] Gh.Păun. Non-centralized parallel communicating grammar systems. *EATCS Bulletin* **40**(1990).

[14] Gh.Păun. On the syntactic complexity of parallel communicating grammar systems. *RAIRO/Th. Informatics* (to appear).

[15] Gh.Păun. Parallel communicating systems of L-systems. (to appear).

[16] Gh.Păun, L.Santean. Parallel communicating grammar systems: the regular case. *Ann. Univ. Buc. Ser. Mat.-Inform.* **37**, 2(1989).

[17] Gh.Păun, L.Santean. Further remarks on parallel communicating grammar systems. *International Journal of Computer Mathematics* **35** (1990).

[18] G.Rozenberg, A.Salomaa. *The Mathematical Theory of L Systems*, Academic Press, New York (1980).

[19] A.Salomaa. *Formal Languages.* Academic Press, New York, London (1973).

[20] L.Santean, J.Kari. The impact of the number of cooperating grammars on the generative power. *Theoretical Computer Science* **98**, 2(1992), 249-263.

[21] J.Hromkovic, J.Kari, L.Kari. Some hierarchies for the communication complexity measures of cooperating grammar systems, *Theoretical Computer Science*, (to appear).