FURTHER REMARKS ON PARALLEL COMMUNICATING GRAMMAR SYSTEMS

GHEORGHE PĂUN and LILA SĂNTEAN

Institute of Mathematics, Str. Academiei 14, București, Romania

(Received 19 December 1989; in final form 20 January 1990)

We continue the study of parallel communicating grammar systems introduced in Păun and Săntean [7] as a grammatical model of parallel computing. The investigated topics are: closure properties, the efficiency of generating a (linear) language by such a system compared with usual grammars, hierarchies.

KEY WORDS: Context-free grammars, parallel computing, grammar systems

C.R. CATEGORIES: F.4.3, G.2.1

1. INTRODUCTION

The parallel communicating grammar systems (PCGS, for short) have been introduced in Păun and Săntean [7] and investigated, from various points of view, in Păun [4, 5, 6]. They prove to be a grammatical model of parallel computing with quite interesting properties and raising many appealing mathematical problems. As architecture, they are similar to the cooperating/distributed grammar systems in Csjaj-Varju and Dassow [1] and Dassow and Păun [2], but there the grammars work sequentially, not in parallel as in a PCGS.

The theory of PCGS (of grammar systems, in general) is very young, therefore many questions are still unsettled in this area. We consider here three of them: closure properties, hierarchies, efficiency.

2. DEFINITIONS

We assume the reader familiar with basic facts in formal language theory (from Salomaa [8], for instance), and we specify only some notations and the definitions related to PCGS.

For a vocabulary \( V \), we denote by \( V^* \) the free monoid generated by \( V \) under the operation of concatenation and the null element \( \lambda \). For \( x \in V^* \), \(|x|\) is the length of \( x \). If \( U \subseteq V \), \( x \in V^* \), then \(|x|_U\) is the length of the string obtained by erasing from \( x \) all symbols not in \( U \).

We denote by \( \text{REG}, \text{LIN, CF, CS} \) the classes of regular, linear, context-free, context sensitive grammars, respectively, without using \( \zeta \)-rules. For a class \( X \) of grammars as above, \( X_\zeta \) is the class of arbitrary grammars of type \( X \) (that is using \( \zeta \)-rules) and \( \mathcal{L}(X) \) is the family of languages generated by grammars in \( X \).
A PCGS (of degree \(n, n \geq 1\)) is a system

\[ \gamma = (G_1, G_2, \ldots, G_n) \]

where \(G_i = (V_{T,i}, V_T, S_i, P_i), 1 \leq i \leq n\), are Chomsky grammars such that \(V_{T,i} \subseteq V_{T,1}, 2 \leq i \leq n\), and there is a set \(K \subseteq \{Q_1, Q_2, \ldots, Q_n\}\), of special symbols (query symbols), \(K \subseteq \bigcup_{i=1}^{n} V_{X,i}\), used in derivations as follows.

For \((x_1, \ldots, x_n), (y_1, \ldots, y_n)\), \(x_i, y_i \in V_{G_i}^*, 1 \leq i \leq n\) \((V_{G_i} = V_{X,i} \cup V_{T,i})\), we write \((x_1, \ldots, x_n) \Rightarrow (y_1, \ldots, y_n)\) if one of the next two cases holds:

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

ii) if \(|x_i|_K > 0\) for some \(i, 1 \leq i \leq n\), then for each such \(i\) we write \(x_i = z_1Q_{i1}z_2Q_{i2} \cdots z_jQ_{ij}z_{j+1}, t \geq 1\), \(|x_i|_K = 0\), \(1 \leq j \leq t+1\); if \(|x_i|_K = 0\), \(1 \leq j \leq t\), then \(y_i = z_1x_i z_2x_i \cdots z_jx_i z_{j+1}\) and \(y_j = S_{ij}\), \(1 \leq j \leq t\); when, for some \(j, 1 \leq j \leq t\), \(|x_i|_K > 0\), then \(y_i = x_i\). For all \(i, 1 \leq i \leq n\), for which \(y_i\) was not defined above, we put \(y_i = x_i\).

In words, an \(n\)-tuple \((x_1, \ldots, x_n)\) directly yields \((y_1, \ldots, y_n)\) if either no query symbol appears in \(x_1, \ldots, x_n\), and then we have a componentwise derivation, \(x_i \Rightarrow y_i\) in \(G_i\), for each \(i, 1 \leq i \leq n\), or, in the case of query symbols appearing, we perform a communication step, \(x_i\) replaced by \(x_{ij}\), providing \(x_{ij}\) does not contain query symbols. More exactly, a component \(x_i\) is modified only when all its occurrences of query symbols refer to strings without query symbols occurrences. After a communication operation, the communicated string \(x_{ij}\) replaces the query symbol \(Q_{ij}\), whereas the grammar \(G_{ij}\) resumes working from its axiom. The communication has priority over rewriting. If some query symbols are not satisfied at a given communication step, then they will be satisfied at the next one (providing they ask for strings without query symbols in that moment) and so on. No rewriting is possible when at least a query symbol is present. This implies that when a circular query appears, the work of the system is blocked. Similarly, the derivation is blocked when no query symbol appears, but some nonterminal component \(x_i\) cannot be further rewritten in \(G_i\).

The language generated by \(\gamma\) is

\[ L(\gamma) = \{x \in V_{T,1}^* | (S_1, \ldots, S_n) \Rightarrow (x, x_2, \ldots, x_n), x_i \in V_{G_i}^*, 2 \leq i \leq n\} \]

A derivation consists of repeated rewriting and communication steps, starting from \((S_1, \ldots, S_n)\); we retain in \(L(\gamma)\) the string generated in this way on the first component, terminal with respect to \(G_1\), without care about the strings generated by \(G_2, \ldots, G_n\) \((G_1\) is the master grammar of the system).

A PCGS as above is called non-centralized. When \(K \cap V_{X,i} = \emptyset, 2 \leq i \leq n\), then \(\gamma\) is called centralized (only \(G_1\) may ask for the strings generated by other grammars in the system).

A further classification can be considered, according to the following criterion: the PCGS as above are called returning to the axiom; when in point (ii) of the
previous definition we erase the words “and \( y_i = S_{ij}, \ 1 \leq i \leq t \)”, then we obtain a non-returning PCGS (after communicating a string \( x_{ij} \) to some \( x_i \), the grammar \( G_{ij} \) does not return to \( S_{ij} \), but continues to process the current string \( x_{ij} \)).

In this paper we deal with returning PCGS. We shall denote by \( PC \) their class and by \( CPC \) the class of centralized ones. When only systems of degree at most \( n \) are considered, we add the subscript \( n \) (\( PC_n, CPC_n \)). According to the type of grammars \( G_1, \ldots, G_n \), a PCGS can be regular, linear, context-free, \( \tilde{\alpha} \)-free etc. We can write \( PC(REG), CPC(CF) \) and so on, for distinguishing such classes.

Here is a simple example: \( \gamma = (G_1, G_2, G_3) \), with

\[
G_1 = (\{S_1, S'_1, S_2, S_3, Q_2, Q_3\}, \{a, b, c\}, S_1,
\]
\[
\{S_1 \rightarrow abc, S_1 \rightarrow a^2b^2c^2, S_1 \rightarrow a^3b^3c^3, S_1 \rightarrow aS'_1, S'_1 \rightarrow aS'_1, S'_1 \rightarrow a^3Q_2, S_2 \rightarrow bQ_3, S_3 \rightarrow c\}
\]
\[
G_2 = (\{S_2\}, \{b\}, S_2, \{S_2 \rightarrow bS_2\})
\]
\[
G_3 = (\{S_3\}, \{c\}, S_3, \{S_3 \rightarrow cS_3\}).
\]

This is a regular centralized PCGS of degree 3 and it is easy to see that (both in the returning and the non-returning mode of derivation) we have

\[
L(\gamma) = \{a^nb^nc^n| \ n \geq 1\}
\]

which is a non-context-free language.

3. ON HIERARCHIES

In Pâun [5] and Pâun and Sântean [7] only the power of centralized PCGS is investigated. Clearly \( CPC_n(X) \subseteq PC_n(X) \), hence \( \mathcal{L}(CPC_n(X)) \subseteq \mathcal{L}(PC_n(X)) \), for all \( n \geq 1 \). \( X \) as above, and it is expected that these inclusions are proper. We shall prove here that this is the case for regular PCGS. For other classes of grammars the problem remains open.

**Theorem 1** \( \mathcal{L}(PC_3(REG)) - \mathcal{L}(CPC(REG)) \neq \emptyset \) (hence all inclusions \( \mathcal{L}(CPC_n(REG)) \subseteq \mathcal{L}(PC_n(REG)) \), \( \mathcal{L}(CPC(REG)) \subseteq \mathcal{L}(PC(REG)) \) are proper, \( n \geq 3 \)).

**Proof** Consider the language

\[
L = \{xcmi(x)|x \in \{a,b\}^*\}
\]

(\( mi(x) \) is the mirror image of \( x \)). In Pâun and Sântean [7] it is proved that \( \{xcmi(x)|x \in \{a,b\}^*\} \) is not in \( \mathcal{L}(CPC(REG)) \); the same proof shows that \( L \), the previous language, is not in \( \mathcal{L}(CPC(REG)) \).
On the other hand, this language can be generated by the non-centralized PCGS $\gamma=(G_1,G_2,G_3)$ with

$$G_1=(\{S_1,S_2,S_3,B_2,B_3,C,Z,Q_2,Q_3\},\{a,b,c,d\},S_1,\{S_1\to cC,S_1\to Q_2,S_1\to Q_3,C\to aB_2,C\to bB_3,\}$$

$$S_2\to Z,S_3\to Z,C\to d,B_2\to Z,B_3\to Z))$$

$$G_2=(\{S_2,S_3,B_2,B_3,C,Z,Q_1\},\{a,b,c\},S_2,\{S_2\to aQ_1,B_2\to C,B_3\to Z,S_3\to Z,C\to Z\})$$

$$G_3=(\{S_2,S_3,B_2,B_3,C,Z,Q_1\},\{a,b,c\},S_3,\{S_3\to bQ_1,B_3\to Z,S_2\to Z,C\to Z\}).$$

As one can see, $Z$ is a trap-symbol; after introducing it, at most communication steps can be performed and then the derivation is blocked.

The first step of a derivation is of the form $(S_1,S_2,S_3)^*(a_1,a_2,a_3),a_1\in\{cC,Q_2,Q_3\},a_2\in\{S_2,aQ_1\},a_3\in\{S_3,bQ_1\}$. If $a_1=Q_2$ or $a_1=Q_3$, then either the derivation is blocked by circularity when $a_2=aQ_1$, or $a_3=bQ_1$, or we communicate $S_2,S_3$ to $G_1$ and, at the next step, we have to use one of the rules $S_2\to Z$,

$S_3\to Z$. Consequently, we must have $a_1=cC$. As both $G_2$ and $G_3$ contain the rule $C\to Z$, the derivation is blocked also when one of $a_2,a_3$ contains $Q_1$. Thus, the beginning of a derivation must be of the form $(S_1,S_2,S_3)^*(cC,S_1,S_2)^*(cBl,P_2,P_3),(l,j_1\in\{d,b,b\},P_2\in\{S_2,aQ_1\},P_3\in\{S_3,bQ_1\}).$ If $\beta_1=d$, then the derivation ends by producing the string $cd$. If $\beta_1=aB_2$ (or $\beta_1=bB_3$) and $\beta_2=S_2$ ($\beta_3=S_3$, respectively), then at the next step $G_1$ introduces the symbol $Z$. If both $\beta_2=aQ_1,\beta_3=bQ_1$ (and, for example, $\beta_1=aB_2$), then after communicating $cB_2$ to $G_3$ we have to introduce $Z$ in $G_3$. Consequently, we must have $(cC,S_2,S_3)^*(c\beta_1,\beta_2,\beta_3),\beta_1\in\{d,aB_2,bB_3\}$, $\beta_2\in\{S_2,aQ_1\}$, $\beta_3\in\{S_3,bQ_1\}$. If $\beta_1=d$, then the derivation ends by producing the string $cd$. If $\beta_1=aB_2$ (or $\beta_2=bB_3$) and $\beta_2=S_2$ ($\beta_3=S_3$, respectively), then at the next step $G_1$ introduces the symbol $Z$. If both $\beta_2=aQ_1,\beta_3=bQ_1$ (and, for example, $\beta_1=aB_2$), then after communicating $cB_2$ to $G_3$ we have to introduce $Z$ in $G_3$. Consequently, we must have $(cC,S_2,S_3)^*(c\beta_1,\beta_2,\beta_3),\beta_1\in\{d,aB_2,bB_3\}$, $\beta_2\in\{S_2,aQ_1\}$, $\beta_3\in\{S_3,bQ_1\}$. If $\beta_1=d$, then the derivation ends by producing the string $cd$. If $\beta_1=aB_2$ (or $\beta_2=bB_3$) and $\beta_2=S_2$ ($\beta_3=S_3$, respectively), then at the next step $G_1$ introduces the symbol $Z$. Therefore $\delta_1=Q_2$ and we have $(Q_2,acaC,\delta_3)=(acaC,\delta_3)$. If $\delta_3=bQ_1$, then $C$ will be replaced in $G_3$ by $Z$. We started from $(cC,S_2,S_3)$ and we obtained $(acaC,S_2,S_3)$. The process can be iterated and, after each such four steps, we add to the terminal string in $G_1$ a prefix $\sigma$ and a suffix $\sigma_1,\sigma\in\{a,b\}$. In this way we can obtain each string of the form $xc\text{mi}(x),\ x\in\{a,b\}^*,$ therefore $L(\gamma)=L$ and the proof is over.

As we have said, it is highly probably that also $L(CPC(CF)) \subsetneq L(PC(CF))$ is a strict inclusion. For instance, we believe that the language $LL$, with $L=\{a^n b^n c^n | n \geq 1\}$, is not in $L(CPC(CF))$ (for generating $a^n b^n c^n a^m b^m c^m$ with unrelated arbitrarily large $n, m$, we have to generate "independently" $a^n b^n c^n$ and $a^m b^m c^m$, which implies we need derivations of the form $A \Rightarrow A$ in at least two grammars,
and this can lead to parasitic derivations producing strings $a^n b^m c^p$, with different $n, m, p$). If this conjecture would be proved, then it would follow that $L(CPC(CF))$ is not closed under concatenation. Note, in contrast, that the family $L(PC(CF))$ is closed under concatenation (Theorem 5).

On the other hand, the problem whether $L(CPC_n(X))$, $L(PC_n(X))$, $n \geq 1$, are infinite hierarchies, for $X \in \{REG, LIN, CF, CF_2\}$, is open. (It is known that $\lambda$-rules can be eliminated from regular and linear PCGS (Păun [5]).) The answer of this important open question is conjectured to be affirmative. Possible languages for proving this would be those of the forms

$$\{a_1^n a_2^n \ldots a_k^n | n \geq 1\} \in L(CPC_k(REG)) - L(PC_{k-1}(REG))$$

$$\{a_1^n a_2^n \ldots a_k^n | n \geq 1\} \in L(CPC_k(LIN)) - L(PC_{k-1}(CF))$$

Anyway, languages of the next form are not useful:

$$L_k = \bigcup_{i=1}^{k} \{a_i^n b_i^n | n \geq 1\}.$$  

Although apparently a different component is necessary for generating each substring $b_i^n$, the language $L_k$ can be generated by a regular PCGS of degree 2. We shall obtain this as a consequence of a more general result.

Given a PCGS $\gamma = (G_1, \ldots, G_n)$ as above and a derivation $D: (S_1, \ldots, S_n) \Rightarrow (w_1, \ldots, w_{1,n}) \Rightarrow (w_2, \ldots, w_{2,n}) \Rightarrow \cdots \Rightarrow (w_{r}, \ldots, w_{r,n})$ in $\gamma$, denote

$$Com(w_{i,1}, \ldots, w_{i,n}) = \sum_{j=1}^{n} |w_{i,j}|_{k}$$

$$Com(D) = \sum_{i=1}^{r} Com(w_{i,1}, \ldots, w_{i,n}).$$

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

$$Com(x, \gamma) = \min \{|Com(D)| : D: (S_1, \ldots, S_n) \Rightarrow (x, z_2, \ldots, z_n)\}$$

Then

$$Com(\gamma) = \sup \{|Com(x, \gamma)| : x \in L(\gamma)\}$$

The parameter $Com$ has been introduced in Păun [6], as a sort of cost of producing a string in $\gamma$ (the total number of query symbols appearing in the derivation).

**Theorem 2** If $\gamma$ is a linear (regular) PCGS of degree $n$ such that $Com(\gamma) = 1$, then there is a linear (regular) PCGS $\gamma'$ of degree 2 such that $L(\gamma) = L(\gamma')$. 

Proof Clearly, if \( \text{Com}(\gamma) = 1 \), then \( \gamma \) is a centralized PCGS. Moreover, in Păun [5, Theorem 4] it is proved that \( \mathcal{L}(\text{CPC}_\delta(\text{REG})) = \mathcal{L}(\text{CPC}_\delta(\text{REG}_\lambda)) \). Therefore, we can assume \( \gamma \) to be \( \lambda \)-free.

We shall consider here only linear PCGS; the regular case is a particular one.

Assume \( \gamma = (G_1, \ldots, G_n) \). \( G_i = (V_{i,n}, V_{T,i}, S_i, P_i) \) \( 1 \leq i \leq n \), and construct \( \gamma' = (G'_1, G'_2) \) as follows.

\[
G'_1 = (V'_{N,1}, V'_{T,1}, S'_1, P'_1)
\]

where

\[
V'_{N,1} = V_{N,1} \cup \{S'_1\} \cup \{[A, i, 0] \mid A \in (V_{N,1} \cap V_{N,i}) \cup V_{T,i}, 2 \leq i \leq n\}
\]

\[
\cup \{(x, i, j) \mid x \in V_{N,1} \cup \{*\}, 2 \leq i \leq n, 0 \leq j \leq n - 1\}
\]

and \( P'_1 \) contains the next groups of rules:

1) \( S'_1 \rightarrow (\ast, i, 1), \quad 2 \leq i \leq n \),

\( (\ast, i, j) \rightarrow (\ast, i, j + 1), \quad 2 \leq i \leq n, 1 \leq j \leq i - 2 \),

\( (\ast, i, i - 1) \rightarrow (S_i, i, 0), \quad 2 \leq i \leq n \).

(Each derivation in \( G'_1 \) starts by \( i \) steps when no rule in \( G_1 \) is involved, \( 2 \leq i \leq n \).)

2) \( (A, i, j) \rightarrow (A, i, j + 1), \quad A \in V_{N,1}, \quad 2 \leq i \leq n, \quad 0 \leq j \leq n - 2 \),

\( (A, i, n - 1) \rightarrow x(B, i, 0)y, \quad \text{for } A \rightarrow xBy \in P_i, \quad 2 \leq i \leq n \).

(At each step \( i + rn, r \geq 1 \), of a derivation in \( G'_1 \), for given \( i, 2 \leq i \leq n \), a rule of \( P_i \) is simulated; the second component of the current nonterminal specifies the chosen \( i \).)

3) \( (A, i, n - 1) \rightarrow xQ_2y, \quad \text{for } A \rightarrow xQ_iy \in P_i \).

(Also the queries are simulated at the moments \( i + rn, r \geq 1, \ 2 \leq i \leq n \).)

4) \( [A, i, 0] \rightarrow A, \quad A \in (V_{N,1} \cap V_{N,i}), \quad 2 \leq i \leq n \),

\( A \rightarrow x, \quad \text{for } A \rightarrow x \in P_i, \quad |x|_k = 0 \).

(After communicating a nonterminal string, the derivation can continue in \( G'_1 \) as in \( G_1 \), without further communications. As the rules \( A \rightarrow x \) of \( P_i \) without queries are introduced in \( P'_1 \), each no communication terminal derivation in \( G_1 \) can be reproduced in \( G'_1 \) too.)

5) \( [a, i, 0] \rightarrow a, \quad a \in V_{T,i}, \quad 2 \leq i \leq n \).
(When a terminal string is communicated, the derivation ends by a rule as above.)

\[ G'_2 = \left( V'_{N,2}, \bigcup_{i=2}^{n} V_{T,i}, S'_2, P'_2 \right) \]

where

\[ V'_{N,2} = \{ S'_2 \} \cup \{ [x,i,j] \mid x \in \{ \ast \} \cup V_{N,i} \cup V_{T,i} \}, \]

\[ 2 \leq i \leq n, \quad 0 \leq j \leq n-1 \}

and \( P'_2 \) contains the following groups of rules:

1) \[ S'_2 \rightarrow [\ast, i, 1], \quad 2 \leq i \leq n, \]

\[ [\ast, i, j] \rightarrow [\ast, i, j+1], \quad 2 \leq i \leq n, \quad 1 \leq j \leq i-2, \]

\[ [\ast, i, i-1] \rightarrow [S, i, 0], \quad 2 \leq i \leq n. \]

(The derivations in grammars \( G_i \), \( 2 \leq i \leq n \), will be simulated in \( G_2 \), involving nonterminals having \( i \) on the second component.)

2) \[ [A, i, j] \rightarrow [A, i, j+1], \quad A \in V_{N,i}, \quad 2 \leq i \leq n, \quad 0 \leq j \leq n-2, \]

\[ [A, i, n-1] \rightarrow x[B, i, 0]y, \quad \text{for} \ A \rightarrow xBy \in P_i, \quad 2 \leq i \leq n. \]

(At each moment \( i + rn, r \geq 1, 2 \leq i \leq n \), in the presence of the component \( i \), a rule of \( G_i \) is simulated in \( G'_2 \).)

3) \[ [A, i, n-1] \rightarrow x'[A, i, 0], \quad \text{for} \ A \rightarrow x'a \in P_i, \quad x' \in V^*_T,i, \]

\[ a \in V_{T,i}, \quad 2 \leq i \leq n. \]

(For each terminal rule \( A \rightarrow x'a \) in \( P_i \)—all of them are \( \lambda \)-free—we consider rules as above, in order to make possibly to finish only the derivations in which the queries from \( G'_1 \) at moments \( i + rn, r \geq 1, 2 \leq i \leq n \), are satisfied by derivations in \( G'_2 \) simulating derivations in \( G_i \), that is introducing at moments \( i + rn \) nonterminals of the form \([x, i, 0]\).)

4) \[ [a, i, j] \rightarrow [a, i, j+1], \quad a \in V_{T,i}, \quad 2 \leq i \leq n, \quad 0 \leq j \leq n-2, \]

\[ [a, i, n-1] \rightarrow [a, i, 0], \quad a \in V_{T,i}, \quad 2 \leq i \leq n. \]

(Symbols \([a, i, 0]\) are introduced only at moments \( i + rn, r \geq 1, 2 \leq i \leq n \).)

From the above explanations, it is easy to see that the symbols \([x, i, j], j \neq 0\), cannot be rewritten in \( G'_1 \), hence the only successful derivations are those which
simulate in \( G' \), a derivation in \( G_i \), \( 2 \leq i \leq n \), and \( G_i \) asks exactly at moments \( i + rn \), \( r \geq 1 \), for the string generated in \( G' \), thus receiving a nonterminal of the form \( [x, i, 0] \), \( x \in (V_{x,1} \cap V_{x,i}) \cup V_{r,i} \). In conclusion, \( L(\gamma) = L(\gamma') \) and the proof is finished.

**Corollary** If \( \gamma \) is a regular PCGS such that \( \text{Com}(\gamma) = 1 \), then \( L(\gamma) \in \mathcal{L}(CF) \).

**Proof** Theorem 3 in Păun and Sântean [7] proves that \( \mathcal{L}(\text{CPC}_2(\text{REG})) \subseteq \mathcal{L}(CF) \). Combining with the result of the previous theorem, we obtain the assertion in corollary.

**Remark** The example in Section 2 proves that the \( \text{Com}(\gamma) = 1 \) consideration in Theorem 1 is essential: for \( \gamma \) as in that example (it is centralized) we have \( \text{Com}(\gamma) = 2 \) and \( L(\gamma) \) is not context-free.

**Open problem** Is the above theorem valid also for PCGS with context-free components? (Conjecture: no.)

4. THE EFFICIENCY OF PCGS

As in Păun [5] and Păun and Sântean [7] it is proved, the generative capacity of PCGS is much larger than that of corresponding grammars (PCGS with regular components can generate non-context-free languages, PCGS with at least three context-free components can generate non-semilinear languages etc.). Moreover, the syntactic complexity, in the sense of Gruska [3], of context-free languages can be considerably decreased (see Păun [6] for a precise meaning of this assertion). But what about dynamical complexity, about time parameter, for instance? Although in formal language theory the time parameter is not investigated for grammars, we can define such a measure in the following way.

Given a grammar \( G = (V_N, V_T, S, P) \) and a derivation \( D: S \Rightarrow w_1 \Rightarrow \cdots \Rightarrow w_n \), we put

\[
\text{Time}(D) = n
\]

and define, for \( x \in L(G) \),

\[
\text{Time}_G(x) = \min \{ \text{Time}(D) \mid D: S \Rightarrow x \}
\]

A mapping \( \text{Time}_G: L(G) \rightarrow \mathbb{N} \) is obtained in this way. A similar definition holds for any other generative device, including a PCGS (both the rewriting and the communication steps are counted).

In this frame, a natural question is: given a grammar \( G \) and a mapping \( f \), can we construct (algorithmically) a PCGS \( \gamma \) such that \( L(G) = L(\gamma) \) and \( \text{Time}_G(x) < f(\text{Time}_G(x)) \), for (almost) all strings \( x \in L(G) \)? Of course, such an improvement in the parameter \( \text{Time} \) must not increase "too much" other measures of (syntactic) complexity. Here are some such measures. For a PCGS \( \gamma \) as above, define

\[
\text{Var}(\gamma) = \text{card} \bigcup_{i=1}^{n} V_{x,i}
\]
CONTEXT-FREE GRAMMARS

\[ Prod(\gamma) = \text{card} \bigcup_{i=1}^{n} P_i \]
\[ Symb(\gamma) = \sum_{r} Symb(r), \quad \text{for all } r \in \bigcup_{i=1}^{n} P_i, \quad \text{where} \]
\[ Symb(A \rightarrow x) = |x| + 2, \]
\[ Length(\gamma) = \max \left\{ |x| \mid A \rightarrow x \in \bigcup_{i=1}^{n} P_i \right\}. \]

These parameters can be defined in the same way for grammars (particular case).

**Theorem 3** Given an infinite linear language \( L \) and a linear grammar \( G \) such that \( L = L(G) \), \( \text{Var}(G) = p \), for each given natural number \( t \) there is a centralized PCGS \( \gamma \) such that

\[ L = L(\gamma) \]
\[ \text{Var}(\gamma) = \text{Var}(G) + pt \]
\[ Prod(\gamma) = Prod(G) + p(t + 1) \]
\[ Symb(\gamma) = Symb(G) + 3p(t + 1) \]
\[ Length(\gamma) = Length(G) \]

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

\[ Time_{\gamma}(x) < \frac{1}{t} Time_{\gamma}(x) + 3t. \]

**Proof** For \( G = (\{A_1, \ldots, A_p\}, V_T, A_1, P) \), we construct the PCGS \( \gamma = (G_1, \ldots, G_{pt + 1}) \), where

\[ G_1 = (V_{N,1}, V_T, A_1, P_1) \]
\[ V_{N,1} = \{A_1, \ldots, A_p\} \cup \{Q_{ij} \mid 1 \leq i \leq p, 1 \leq j \leq t\} \]
\[ P_1 = P \cup \{A_1 \rightarrow A_1\} \cup \{A_1 \rightarrow Q_{ij} \mid 1 \leq i \leq p, 1 \leq j \leq t\} \]

and, for each \( 1 \leq i \leq p, 1 \leq j \leq t \).
The inclusion \( L(G) \subseteq L(G) \) is obvious (each grammar \( G_{ij+1} \) has \( A_i \) as axiom, \( 1 \leq i \leq p \), \( 1 \leq j \leq t \); when \( G_1 \) introduces a query symbol \( Q_{ij+1} \), this means it used a rule \( A_i \rightarrow Q_{ij+1} \) and the string which will replace \( Q_{ij+1} \) is obtained by a derivation of the form \( A_i \Rightarrow w \), in the grammar \( G_{ij+1} \); this implies the derivations on various components of \( \gamma \) which will be communicated to \( G_1 \) will complete a derivation in \( G \); note that the rules \( A_i \rightarrow A_i \) introduced in each \( G_{ij+1} \) in order to synchronize the derivations do not lead to derivations not in \( G \).)

Conversely, take a string \( x \in L(G) \) with \( \text{Time}_G(x) = m \) and take a derivation \( D \) of \( x \) with exactly \( m \) steps. If \( m \leq 3t \), then \( D \) can be considered a derivation in \( \gamma \) as it can be reproduced in \( G_1 \). If \( m > 3t \), then we write \( D \) in the form

\[
D: A_1 \Rightarrow u_1 A_i v_1 \Rightarrow u_1 u_2 A_i v_2 v_1 \Rightarrow \ldots
\]

\[
\ldots \Rightarrow u_1 \ldots u_{t-1} A_i v_{t-1} \ldots v_1 \Rightarrow u_1 \ldots u_{t-1} w v_{t-1} \ldots v_1
\]

such that

\[
\text{Time}_G(A_1 \Rightarrow u_1 A_i v_1) = \left[ \frac{m}{t} \right]
\]

\[
\text{Time}_G(A_i \Rightarrow u_i A_{i+1} v_{i+1}) = \left[ \frac{m}{j} \right], \quad 1 \leq j \leq t - 2
\]

([q] is the greatest integer smaller than \( q \)). If \( m \) is a multiple of \( t \), then

\[
\text{Time}_G(A_{i-1} \Rightarrow w) = \left[ \frac{m}{t} \right]
\]

If \( m \) is not a multiple of \( t \), then we further write \( A_{i-1} \Rightarrow w \) as

\[
A_{i-1} \Rightarrow u_i A_i v_i \Rightarrow u_i w' v_i
\]

with

\[
\text{Time}_G(A_{i-1} \Rightarrow u_i A_i v_i) = \left[ \frac{m}{t} \right]
\]

\[
\text{Time}_G(A_i \Rightarrow w') = m - \left[ \frac{m}{t} \right] t.
\]

Clearly, \( (m - [m/t]t) < t \). We construct the following derivation in \( \gamma \):
When \( m \) is a multiple of \( t \), then on the last component one generates \( u_iw'v_e = w \) from \( A_{i-1} \) in \([m/t]\) steps.

Besides the \([m/t]\) initial rewriting steps, in the previous derivation we have \( t-1 \) steps of using rules of the form \( A_j \rightarrow Q_j \) on the first component and of the form \( A_i \rightarrow A_i \) on the others, \( t-1 \) steps of the communication and, possibly, further at most \( t-1 \) steps in the final stage (for \( A_i \rightarrow w' \)). In conclusion, we have less than \([m/t]+3t\) steps, therefore \( \text{Time}_e(x) < (1/t)\text{Time}_g(x) + 3t \).

It is easy to see that the relations between \( \text{Var}(G) \), \( \text{Prod}(G) \), \( \text{Symb}(G) \), \( \text{Length}(G) \) and \( \text{Var}(\gamma) \), \( \text{Prod}(\gamma) \), \( \text{Symb}(\gamma) \), \( \text{Length}(\gamma) \) are as specified in theorem.

5. CLOSURE PROPERTIES

In general, it seems to be hard to say something about closure properties of families of languages generated by \( \text{PCGS} \), because, on the one hand, it is not easy to prove positive results and, on the other hand, there are no known languages not in \( \mathcal{L}(\text{PC}(\text{CF})) \), \( \mathcal{L}(\text{PC}(\text{CF})) \) and other such families. Here we shall investigate only the families \( \mathcal{L}(\text{PC}(\text{CF})) \), \( \mathcal{L}(\text{PC}(\text{CF})) \) (more closure properties can be proved to hold for them).

First, let us note that by standard proofs, all families involving arbitrary rules are closed under arbitrary homomorphisms and all families involving \( \lambda \)-free rules are closed under \( \lambda \)-free homomorphisms.

**Theorem 4** The families \( \mathcal{L}(\text{PC}(\text{CF})) \), \( \mathcal{L}(\text{PC}(\text{CF})) \) are closed under union.

**Proof** Consider two \( \text{PCGS} \), \( \gamma_1 = (G_1', \ldots, G_n') \), \( \gamma_2 = (G_1'', \ldots, G_m'') \), \( G_i = (V_{N,i}, V_{T,i}, S'_i, P_i), 1 \leq i \leq n, G_i'' = (V''_{N,i}, V''_{T,i}, S''_i, P_i'), 1 \leq i \leq m \). Without loss of generality, we may assume that \( V''_{T,1} \cap (\bigcup_{i=1}^n V''_{N,i}) = \emptyset \) and \( V''_{T,1} \cap (\bigcup_{i=1}^n V''_{N,i}) = \emptyset \).

If both of \( L(\gamma_1) \), \( L(\gamma_2) \) are finite, then the assertion is trivial. If one of \( L(\gamma_1) \), \( L(\gamma_2) \) is finite—say \( L(\gamma_2) \)—then we construct the \( \text{PCGS} \) \( \gamma = (G_1, G_2', \ldots, G_m') \) with

\[
G_1 = (V_{N,1} \cup \{S_1\}, V_{T,1} \cup V''_{T,1}, S_1, P_1)
\]

\[
P_1 = P_1' \cup \{S_1 \rightarrow x | x \in P_i' \}
\]

\[
\cup \{S_1 \rightarrow w | w \in L(\gamma_2) \}.
\]
The equality \( L(\gamma) = L(\gamma_1) \cup L(\gamma_2) \) is obvious.
Assume now that both \( L(\gamma_1) \) and \( L(\gamma_2) \) are infinite. We construct the PCGS

\[
\gamma = (G_1, G_2, \ldots, G_{n+1}, G_{n+2}, \ldots, G_{n+m+1})
\]

where

\[
G_1 = (\{S_1, Q_2, Q_{n+2} \}, V_{T.1} \cup V_{T.2}', S_1, \\
\{S_1 \rightarrow S_1, S_1 \rightarrow Q_2, S_1 \rightarrow Q_{n+2} \})
\]

\( G_2, \ldots, G_{n+1} \) are exactly \( G'_1, \ldots, G'_n \) with each occurrence of

\[
Q_i, \quad 1 \leq i \leq n, \text{ replaced by } Q_{i+1},
\]

\( G_{n+2}, \ldots, G_{n+m+1} \) are exactly \( G''_1, \ldots, G''_m \) with each occurrence of

\[
Q_i, \quad 1 \leq i \leq m, \text{ replaced by } Q_{i+n+1}.
\]

Each derivation in \( G_1 \) is of the form \( S_1 \xrightarrow{*} S \Rightarrow Q_2 \) or of the form \( S_1 \xrightarrow{*} S \Rightarrow Q_{n+2} \). The components \( G_2, \ldots, G_{n+1} \) work exactly as \( \gamma_1 \), the components \( G_{n+2}, \ldots, G_{n+m+1} \) work as \( \gamma_2 \). The string in \( G_2 \) \( (G_{n+2}, \text{ respectively}) \) communicated to \( G_1 \) must be in \( V_{T.1}' \) \( (\text{in } V_{T.1}'' \text{, respectively}) \). When a string in \( L(\gamma_1) \) is to be prepared on the components \( G_2, \ldots, G_{n+1} \) of \( \gamma \), the components \( G_{n+2}, \ldots, G_{n+m+1} \) work on a long enough string, in order to not block the derivation, and similarly \( G_2, \ldots, G_{n+1} \) during preparing a string in \( L(\gamma_2) \) \( (\text{the languages } L(\gamma_1), L(\gamma_2) \text{ are infinite}) \). In conclusion, \( L(\gamma) = L(\gamma_1) \cup L(\gamma_2) \).

**Theorem 5** The families \( \mathcal{L}(PC(CF)), \mathcal{L}(PC(CF_\wedge)) \) are closed under concatenation.

**Proof** Consider two PCGS, \( \gamma_1 = (G'_1, \ldots, G'_n) \), \( \gamma_2 = (G''_1, \ldots, G''_m) \) as in the previous proof. Again, if both \( L(\gamma_1) \), \( L(\gamma_2) \) are finite, then the question is trivial, and if exactly one of them is finite, the question can be easily settled. If both \( L(\gamma_1) \), \( L(\gamma_2) \) are infinite, then we construct the PCGS

\[
\gamma = (G_1, G_2, \ldots, G_{n+1}, G_{n+2}, \ldots, G_{n+m+1}, G_{n+m+2})
\]

where

\[
G_1 = (\{S_1, R, Q_2, Q_{n+2} \}, V_{T.1} \cup V_{T.1}'', S_1, \\
\{S_1 \rightarrow S_1, S_1 \rightarrow Q_2 R, R \rightarrow R, R \rightarrow Q_{n+2} \}),
\]

\( G_2, \ldots, G_{n+1} \) are exactly \( G'_1, \ldots, G'_n \) with each occurrence of

\[
Q_i, \quad 1 \leq i \leq n, \text{ replaced by } Q_{i+1},
\]

\( G_{n+2}, \ldots, G_{n+m+1} \) are exactly \( G''_1, \ldots, G''_m \) with each occurrence of
CONTEXT-FREE GRAMMARS

Each derivation in $G_1$ must use the rules $S \rightarrow Q_2 R$, $R \rightarrow Q_{n+2}$. As no symbol in $V_{n,1}$, $V'_n,1$ can be rewritten in $G_1$, the strings communicated to $G_1$ after introducing $Q_2$, $Q_{n+2}$ must be terminal, hence in $L(\gamma_1)$, $L(\gamma_2)$. This implies $L(\gamma) \subseteq L(\gamma_1) L(\gamma_2)$.

Conversely, each pair of derivations $(S'_1, \ldots, S'_n) \Rightarrow (x, x_2, \ldots, x_n)$, $(S''_1, \ldots, S''_m) \Rightarrow (y, \beta_2, \ldots, \beta_m)$ can be simulated in $\gamma$ as follows:

\[
(S'_1, \ldots, S'_n) \Rightarrow (Q_2 R, x, x_2, \ldots, x_n, \delta''_1, \ldots, \delta''_m) \Rightarrow (Q_2 \cdots Q_{n+1} Q_{n+2} \cdots Q_{n+m+1} S_{n+m+2})
\]

\[
= (xR, S'_1, \ldots, S'_{n'}, S''_1, \ldots, S''_{m'}, x_2, \ldots, x_n, \delta''_1, \ldots, \delta''_m S_{n+m+2})
\]

\[
\Rightarrow (x, \delta'_1, \ldots, \delta'_n, S'_1, \beta_2, \ldots, \beta_m, x_2, \ldots, x_n, \delta'_1, \ldots, \delta'_m S_{n+m+2}).
\]

(Remember that $L(\gamma_1)$, $L(\gamma_2)$ are infinite, hence we can find long enough derivations in $G_2, \ldots, G_n$ and in $G_{n+1}, \ldots, G_{n+m+1}$, respectively.) In conclusion, $L(\gamma_1) L(\gamma_2) \subseteq L(\gamma)$, hence $L(\gamma) = L(\gamma_1) L(\gamma_2)$.

**Theorem 6** The families $\mathcal{L}(PC(CF))$, $\mathcal{L}'(PC(CF))$ are closed under Kleene $+$ and $\ast$.

**Proof** Consider a PCGS $\gamma_0 = (G_1, \ldots, G_n)$, with $G_i = (V_{n,1}^i, V_{T,i}^i, S_i^i, P_i^i)$, $1 \leq i \leq n$, and construct the PCGS

\[
\gamma = (G_1, G_2, \ldots, G_n, G_{n+1}, G_{n+2}, G_{n+3})
\]

where

\[
G_1 = \{ S_1, S_{n+2}, Q_2, Q_{n+2}, V_{T,1}, S_1, 1 \leq i \leq n \},
\]

\[
G_2 = (V_{n,1}^i \cup \{ S_1^i, Q_2, Q_{n+2}, S_{n+2}, \rightarrow S_1, S_1 \rightarrow Q_2 \}),
\]

\[
G_3 = (V_{T,1}^i \cup \{ S_1^i, Q_2, Q_{n+2}, S_{n+2}, \rightarrow S_1, S_1 \rightarrow Q_2 \}),
\]

\[
\{ S_1^i \rightarrow h(x) \mid S_1 \rightarrow x \in P_1 \} \cup \{ A \rightarrow h(x) \mid A \rightarrow x \in P_1 \},
\]

\[
\{ S_1^i \rightarrow x \in P_1 \} \cup \{ A \rightarrow x \in P_1 \}.
\]
where

\[ h: \left( \bigcup_{i=1}^{n} (V_{N,i} \cup V_{T,i}) \right) \rightarrow \left( \bigcup_{i=1}^{n} (V_{N,i} \cup V_{T,i}) \cup \{Q_{n+1}\} \right) \]

is the homomorphism defined by \( h(x) = x \) for \( x \notin K \), and \( h(Q_j) = Q_{j+1}, 1 \leq j \leq n \);

\[ G_{n+2} = (\{S_{n+2}, S'_{n+2}, T\}, \{a\}, S_{n+2}, \{S_{n+2} \rightarrow S'_{n+2}, S'_{n+2} \rightarrow T, T \rightarrow T\}) \]

\[ G_{n+3} = (\{S_{n+3}, S'_{n+3}, Q_{2}, \ldots, Q_{n+2}\}, \{a\}, S_{n+3}, \{S_{n+3} \rightarrow Q_{2} S'_{n+3}, S'_{n+3} \rightarrow Q_{3} \ldots Q_{n+2} S_{n+3}, S'_{n+3} \rightarrow S'_{n+3}\}) \]

Let us examine a derivation in \( \gamma \).

\( (S_1, S_2', S_3', \ldots, S_n', S_{n+1}, S_{n+2}, S_{n+3}) \)

\[ \Rightarrow (x_1, x_2, S_3', \ldots, S'_{n+1}, S_{n+2}, Q_2 S'_{n+3}) \]

\[ \Rightarrow (x_1, S_2', S_3', \ldots, S'_{n+1}, x_{n+2}, x_2 S'_{n+3}) \]

If \( x_1 = S_1 \), then \( x'_1 = S_1, x_{n+2} = S'_{n+2} \). If \( x_1 = Q_2 \), then \( x'_1 = x_2 \) and the derivation ends (correctly when \( x_2 \in V_{T,1}' \)). If \( x_1 = Q_2 Q_{n+2} \), then \( x'_1 = x_2 S'_{n+2} \). \( x_{n+2} = S_{n+2} \) and we continue by

\[ (x_2 S'_{n+2}, S''_2, S_3', \ldots, S_{n+1}, S_{n+2}, x_2 S'_{n+3}) \]

\[ \Rightarrow (x_2 S_1, \beta_2, \beta_3, \ldots, \beta_{n+1}, S'_{n+2}, x_2 S'_{n+3}) \]

Therefore either the derivation ends, or we have a current \( n \)-tuple of the form \( (x S_1, \beta_2, \beta_3, \ldots, \beta_{n+1}, S'_{n+2}, S_{n+2}) \), with \( (\beta_2, \ldots, \beta_{n+1}) \) correctly generated in \( \gamma_0 \). Now, \( G_{n+2} \) introduces the symbol \( T \), which will be repeated; it cannot be rewritten in \( G_1 \). If the next query in \( G_1 \) will be \( Q_2 \), then the string generated by \( G_2 \) (that is by \( G_1 \) in \( \gamma_0 \)) must be terminal and the derivation ends. If the next query in \( G_1 \) will be \( Q_2 Q_{n+2} \), then the string in \( G_2 \) must be terminal and that in \( G_{n+2} \) must be \( S'_{n+2} \) (the only nonterminal of \( G_{n+2} \) which can be rewritten in \( G_1 \)). This means, we have the next steps of derivation:

\[ (x S'_1, \delta_2, \delta_3, \ldots, \delta_{n+1}, T, x' S'_{n+3}) \]
If $S_1$ is not replaced by $Q_2$ or by $Q_2Q_{n+1}$ in the previous step of derivation, then the string $\delta_2'$ is "lost" as being requested by $G_{n+3}$. Now we obtain

$$(x'\delta_2'S_1, \pi_2, \ldots, \pi_{n+3}, S_{n+2}', \gamma' \delta_3 \ldots \delta'_{n+1} T \delta_2' S_{n+3})$$

and the process can be iterated. When the rule $S_1 \rightarrow Q_2$ is used in $G_1$, the derivation ends by producing a string in $L(\gamma_0)^*$. In order to obtain the closure under $*$, a rule $S_1 \rightarrow \lambda$ must be added to $G_1$.

**Theorem 7** The families $\mathcal{L}(PC(CF))$, $\mathcal{L}(PC(CF_{\lambda}))$ are closed under substitution by $\lambda$-free languages.

**Proof** Consider a PCGS $\gamma_0 = (G_1, \ldots, G_r)$ with $G_i = (V_N, V_T, I,S_i, P_i)$, $1 \leq i \leq n$, $V_T = \{a_1, \ldots, a_r\}$, and take the context-free grammars $G_i'' = (V_N, V_T, S_i', P_i')$, $1 \leq i \leq r$. If $L(\gamma_0)$ is finite, then each context-free substitution maps it to a context-free language. Thus, we have to discuss only the case when $L(\gamma_0)$ is infinite. We construct the PCGS

$$\gamma = (G_1, G_2, \ldots, G_{n+3}, \ldots, G_{n+r+1}, G_{n+r+2})$$

where

$$G_1 = \{S_1, Q_2\} \cup \{Q_{n+i+1} | 1 \leq i \leq r\} \cup \{a_i | 1 \leq i \leq r\} \cup \bigcup_{i=1}^r V_T,i,$$

$$S_1 \rightarrow S_1, S_1 \rightarrow Q_2 \cup \{a_i \rightarrow a_i', a_i \rightarrow Q_{n+i+1} | 1 \leq i \leq r\},$$

$$G_{i+1} = (V_N, i \cup \{a | a \in V_T,i\}, \{a_i\}, S_i',$$

$$\{A \rightarrow h(x) | A \rightarrow x \in P_i\}, \ 1 \leq i \leq n,$$

with

$$h: \left(\bigcup_{i=1}^n (V_N,i \cup V_T,i)\right)^* \rightarrow \left(\bigcup_{i=1}^n V_N,i \cup \{Q_{n+1}\} \cup \{a | a \in \bigcup_{i=1}^n V_T,i\}\right)^*$$

the homomorphism defined by
G. PĂUN AND L. SĂNTAN

\[ h(A) = A, A \in \bigcup_{i=1}^{n} V'_{N,i} - K, \]

\[ h(a) = a', a \in \bigcup_{i=1}^{n} V'_{T,i}, h(Q_j) = Q_{j+1}, \quad 1 \leq j \leq n, \]

\[ G_{n+i+1} = (V''_{N,i} \cup \{S_{n+i+1}\}, V''_{T,i}, S_{n+i+1}, \]

\[ P''_i \cup \{S_{n+i+1} \rightarrow S''_i, S_{n+i+1} \rightarrow S_{n+i+1}\}, \quad 1 \leq i \leq r, \]

\[ G_{n+r+2} = (\{S_{n+r+2}, Q_{n+2}, \ldots, Q_{n+r+1}\}, \{a\}, S_{n+r+2}, \]

\[ \{S_{n+r+2} \rightarrow S_{n+r+2}, S_{n+r+2} \rightarrow Q_{n+2} \ldots Q_{n+r+1} S_{n+r+2}\}). \]

It is easy to see that (1) each derivation in \( G_1 \) must use a rule \( S_1 \rightarrow Q_2 \), (2) the components \( G_2, \ldots, G_{n+1} \) work exactly as \( \gamma_0 \), but the produced string has each symbol \( a \in V'_{T,1} \) replaced by \( a' \), (3) the components \( G_{n+i+1}, 1 \leq i \leq r \) produce strings in \( L(G''_i) \) and (4) the last component, \( G_{n+r+2} \), only "cleans" the strings generated by \( G_{n+i+1}, 1 \leq i \leq r \), returning to axioms these components. After introducing \( Q_2 \) in \( G_1 \) and communicating the string \( x \) generated in that moment by \( G_2, \ldots, G_{n+1} \) (hence by \( \gamma_0 \), modulo the homomorphism \( h \)), the only rewrites in \( G_1 \) are done by rules \( a_i' \rightarrow a_i' \) and \( a_i' \rightarrow Q_{n+i+1}, 1 \leq i \leq r \). In this way, each \( a_i' \) is replaced by a string generated by \( G''_i \). If this string is not terminal, the derivation is blocked. When generating the string \( x \) in \( G_2, \ldots, G_{n+1} \), the components \( G_{n+2}, \ldots, G_{n+r+1} \) can be returned to axioms as many times as necessary (by queries using the rule \( S_{n+r+2} \rightarrow Q_{n+2} \ldots Q_{n+r+1} S_{n+r+2} \) in \( G_{n+r+2} \)), thus not limiting the length of the derivation in \( G_2, \ldots, G_{n+1} \). Similarly, when generating strings in \( G_{n+2}, \ldots, G_{n+r+1} \), the components \( G_2, \ldots, G_{n+1} \) work to a long enough string in \( L(\gamma_0) \) (this language is infinite). Moreover, after using a rule \( a_i' \rightarrow Q_{n+i+1} \) in \( G_1 \) and substituting \( Q_{n+i+1} \) by the corresponding string in \( G_{n+i+1} \), the components \( G_{n+2}, \ldots, G_{n+r+1} \) can be again returned to axioms, in order to obtain a new string in some \( G_{n+j+1} \), maybe also in \( G_{n+i+1} \), which will be communicated to \( G_1 \) for substituting a symbol \( a_i' \).

In conclusion, \( L(\gamma) = s(L(\gamma_0)) \), for \( s \) the substitution defined by \( s(a_i) = L(G''_i) \), \( 1 \leq i \leq r \); and the proof is ended.

**Remark** Note that the proof of Theorem 4 holds also for regular and for linear PCGS; the proofs of Theorems 5, 6, 7 do not hold for these cases.

**Open problem** Are the families \( \mathcal{L}(PC(CF)) \), \( \mathcal{L}(PC(CF_j)) \) closed under intersection by regular sets? If the answer would be affirmative, this will imply that \( \mathcal{L}(PC(CF_j)) \) is a full AFL, and, as probably \( \mathcal{L}(PC(CF)) \) is closed under restricted homomorphisms, this family will be an AFL.

**Acknowledgement**

Useful discussions with Dr. Marian Gheorghe are gratefully acknowledged.
References