On the Correspondence between Parallel Communicating Grammar Systems and Restarting Automata

Dana Pardubská[⋆1], Martin Plátek[⋆⋆2], and Friedrich Otto[3]

[1] Dept. of Computer Science, Comenius University, Bratislava
e-mail: pardubska@dcs.fmph.uniba.sk
[2] Dept. of Computer Science, Charles University, Prague
e-mail: Martin.Platek@mff.cuni.cz
[3] Fachbereich Elektrotechnik/Informatik, Universität Kassel, Kassel
e-mail: otto@theory.informatik.uni-kassel.de

Abstract This technical report integrates the results of [8] and [9] in which the study of the relation between Parallel Communicating Grammar Systems (PCGS) and Freely Rewriting Restarting Automata (FRR) has been initiated. Besides communication complexity we introduce and study so-called distribution and generation complexity. It is shown that analysis by reduction for a PCGS with distribution complexity bounded by a constant t and generation complexity bounded by a constant j can be implemented by a strongly linearized deterministic FRR-automaton with t rewrites per cycle. We show that, from a communication complexity point of view, this implementation is almost optimal. As consequences we obtain a pumping lemma for the class of languages generated by PCGS with communication complexity bounded by a constant, and the fact that this class of languages is semi-linear. That makes this class of languages interesting from the point of view of formal linguistics. Further, we present infinite hierarchies of classes of languages based on the parameters t, j and on the new notion of *skeleton*. Here we also consider non-deterministic linearized FRR-automata.

1 Introduction

The main goal of this paper is to study the relation between Freely Rewriting Restarting Automata (FRR, [5]) and Parallel Communicating Grammar Systems (PCGS, [1,10]).Namely, the so-called linearized FRR-automaton is introduced for this purpose. The motivation for our study is the usefulness of both models in computational linguistics.

Freely rewriting restarting automata form a suitable tool for modelling the so-called *analysis by reduction*. In general, analysis by reduction facilitates the development and testing of categories for syntactic and semantic disambiguation of sentences of natural languages. The Functional Generative Description for the Czech language developed in Prague (see, e.g., [2]) is based on this method.

FRR-automata work on so-called *characteristic languages*, that is, on languages with auxiliary symbols (categories) included in addition to the input symbols. The *proper language* is obtained from a characteristic language by removing all auxiliary symbols from its sentences. By requiring that the automata considered are *linearized* we restrict the number of auxiliary symbols allowed

⋆ Partially supported by the Slovak Grant Agency for Science (VEGA) under contract "Theory of Models, Complexity and Algorithms".
⋆⋆ Partially supported by the Grant Agency of the Czech Republic under Grant-No. 405/08/0681 and by the program Information Society under project 1ET100300517.

on the tape by a function linear in the number of terminals on the tape. We mainly focus on deterministic restarting automata in order to ensure the *correctness preserving property* for the analysis, i.e., after any restart within an accepting computation the content of the tape is a word from the characteristic language, and after any restart within a non-accepting computation the content of the tape is a word from the complement of the characteristic language. In fact, we mainly consider *strongly linearized* restarting automata. This additional restriction requires that all rewrite operations must be deletions.

Parallel Communicating Grammar Systems handle the creation of copies of generated strings and their regular mappings in a natural way. This ability has a strong similarity to the generation of coordinations in the Czech language (and some other natural languages). However, the synonymy of coordinations has not yet been appropriately modelled.

In this paper the notions of distribution and generation complexity for PCGS are introduced and studied. It is shown that analysis by reduction for a PCGS with distribution complexity bounded by a constant t and generation complexity bounded by a constant j can be implemented by a strongly linearized deterministic FRR-automaton with t rewrites per cycle. We show that this implementation is almost optimal due to communication complexity. As consequences we obtain a pumping lemma for the class of languages generated by PCGS with communication complexity bounded by a constant, and the fact that this class of languages is semi-linear. Further, we present infinite hierarchies of classes of languages based on the parameters t and j, and on the notion of *skeleton*. Here we consider also the non-deterministic versions of linearized FRR-automata.

The notion of skeleton is introduced in order to model the borders of so-called segments in (Czech) sentences (or in text). The elements of skeletons are called *islands*. They are used to model the so-called separators of segments (see [3]).

2 Basic notions

A *freely rewriting restarting automaton*, abbreviated as FRR-automaton, is described by an 8-tuple $M = (Q, \Sigma, \Gamma, ¢, \$, q_0, k, \delta)$. It consists of a finite-state control, a flexible tape, and a read/write window of a fixed size $k \geq 1$. Here Q denotes a finite set of (internal) states that contains the initial state q_0, Σ is a finite input alphabet, and Γ is a finite tape alphabet that contains Σ. The elements of $\Gamma \setminus \Sigma$ are called *auxiliary symbols*. The additional symbols $¢, \$ \notin \Gamma$ are used as markers for the left and right end of the workspace, respectively. They cannot be removed from the tape. The behavior of M is described by a transition function δ that associates transition steps to certain pairs of the form (q, u) consisting of a state q and a possible content u of the read/write window. There are four types of transition steps: *move-right steps*, *rewrite steps*, *restart steps*, and *accept steps*. A *move-right step* simply shifts the read/write window one position to the right and changes the internal state. A *rewrite step* causes M to replace a non-empty prefix u of the content of the read/write window by a shorter word v, thereby shortening the length of the tape, and to change the state. Further, the read/write window is placed immediately to the right of the string v. A *restart step* causes M to place its read/write window over the left end of the tape, so that the first symbol it sees is the left sentinel ¢, and to reenter the initial state q_0. Finally, an *accept step* simply causes M to halt and accept.

A *configuration* of M is described by a string $\alpha q \beta$, where $q \in Q$, and either $\alpha = \varepsilon$ (the empty word) and $\beta \in \{¢\} \cdot \Gamma^* \cdot \{\$\}$ or $\alpha \in \{¢\} \cdot \Gamma^*$ and $\beta \in \Gamma^* \cdot \{\$\}$; here q represents the current state, $\alpha\beta$ is the current content of the tape, and it is understood that the window contains the first k symbols of β or all of β when $|\beta| \leq k$. A *restarting configuration* is of the form $q_0 ¢ w \$$, where $w \in \Gamma^*$.

Any computation of M consists of certain phases. A phase, called a *cycle*, starts in a restarting configuration. The window is shifted along the tape by move-right and rewrite operations until a restart operation is performed and thus a new restarting configuration is reached. If no further restart operation is performed, the computation necessarily finishes in a halting configuration – such a phase is called a *tail*. It is required that in each cycle M performs at least one rewrite step. As each rewrite step shortens the tape, we see that each cycle reduces the length of the tape. We use the notation $u \vdash_M^c v$ to denote a cycle of M that begins with the restarting configuration $q_0 ¢ u \$$ and ends with the restarting configuration $q_0 ¢ v \$$; the relation $\vdash_M^{c^*}$ is the reflexive and transitive closure of $\vdash_M^c$.

A word $w \in \Gamma^*$ is *accepted* by M, if there is a computation which starts from the restarting configuration $q_0 ¢ w \$$, and ends with an application of an accept step. By $L_{\mathrm{C}}(M)$ we denote the language consisting of all words accepted by M. It is the *characteristic language* of M.

By Pr^{Σ} we denote the projection from Γ^* onto Σ^*, that is, Pr^{Σ} is the morphism defined by $a \mapsto a$ $(a \in \Sigma)$ and $A \mapsto \varepsilon$ $(A \in \Gamma \setminus \Sigma)$. If $v := \mathsf{Pr}^{\Sigma}(w)$, then v is the *Σ-projection* of w, and w is an *expanded version* of v. For a language $L \subseteq \Gamma^*$, $\mathsf{Pr}^{\Sigma}(L) := \{\, \mathsf{Pr}^{\Sigma}(w) \mid w \in L \,\}$. Further, for $K \subseteq \Gamma$, $|x|_K$ denotes the number of occurrences of symbols from K in x.

In recent papers (see, e.g., [4]) restarting automata were mainly used as acceptors. The (*input*) *language* accepted by a restarting automaton M is the set $L(M) := L_{\mathrm{C}}(M) \cap \Sigma^*$. Here, motivated by linguistic considerations to model the analysis by reduction with parallel processing, we are rather interested in the so-called *proper language of* M, which is the set of words $L_{\mathrm{P}}(M) := \mathsf{Pr}^{\Sigma}(L_{\mathrm{C}}(M))$. Hence, a word $v \in \Sigma^*$ belongs to $L_{\mathrm{P}}(M)$ if and only if there exists an expanded version u of v such that $u \in L_{\mathrm{C}}(M)$.

For each type X of restarting automata, we use $\mathcal{L}_{\mathrm{C}}(\mathsf{X})$ and $\mathcal{L}_{\mathrm{P}}(\mathsf{X})$ to denote the class of all characteristic languages and the class of all proper languages of automata of this type.

The following basic properties of FRR-automata are often used in proofs.

(Error Preserving Property) Each FRR-automaton M is *error preserving*, i.e., if $u \notin L_{\mathrm{C}}(M)$ and $u \vdash_M^{c^*} v$, then $v \notin L_{\mathrm{C}}(M)$, either.

(Weak Correctness Preserving Property) Each FRR-automaton M is *weakly correctness preserving*, i.e., if $u \vdash_M^{c^*} v$ during an accepting computation of M, then $u, v \in L_{\mathrm{C}}(M)$.

(Correctness Preserving Property) Each deterministic FRR-automaton M is *correctness preserving*, i.e., if $u \in L_{\mathrm{C}}(M)$ and $u \vdash_M^{c^*} v$, then $v \in L_{\mathrm{C}}(M)$, too.

(Cycle Pumping Lemma) For any FRR-automaton M, there exists a constant p such that the following property holds. Assume that $uxvyz \vdash_M^c ux'vy'z$ is a cycle of M, where $u = u_1 u_2 \cdots u_n$ for some non-empty words $u_1, \ldots, u_n$ and an integer $n > p$. Then there exist $r, s \in \mathbb{N}_+$, $1 \leq r < s \leq n$, such that

$u_1 \cdots u_{r-1}(u_r \cdots u_{s-1})^i u_s \cdots u_n xvyz \vdash_M^c u_1 \cdots u_{r-1}(u_r \cdots u_{s-1})^i u_s \cdots u_n x'vy'z$

holds for all $i \geq 0$, that is, $u_r \cdots u_{s-1}$ is a "pumping factor" in the above cycle. Similarly, such a pumping factor can be found in any factorization of length greater than p of v or z as well as in any factorization of length greater than p of a word accepted in a tail computation.

We focus our attention on FRR-automata, for which the use of auxiliary symbols is less restricted than in [5].

Definition 1. *Let $M = (Q, \Sigma, \Gamma, ¢, \$, q_0, k, \delta)$ be an* FRR*-automaton.*

(a) *M is called* linearized *if there exists a constant $j \in \mathbb{N}_+$ such that $|w|_{\Gamma - \Sigma} \leq j \cdot |w|_{\Sigma} + j$ for each $w \in L_C(M)$.*

(b) *M is called* strongly linearized *if it is linearized, and if each of its rewrite operations just deletes some symbols.*

Since a linearized FRR-automaton only uses linear space, we have the following:

Corollary 1. *If M is a linearized* FRR*-automaton, then the proper language $L_{\mathrm{P}}(M)$ is context-sensitive.*

In what follows we are mainly interested in strongly linearized FRR-automata and their proper languages. We denote by (S)LnRR the class of (strongly) linearized deterministic FRR-automata, by N(S)LnRR the class of non-deterministic (strongly) linearized FRR-automata, and, for $t \in \mathbb{N}_+$, we use the prefix t- to denote types of FRR-automata that execute at most t rewrite steps in any cycle.

2.1 Parallel Communicating Grammar Systems

A PCGS *of degree m, $m \geq 1$,* is an $(m+1)$-tuple $\Pi = (G_1, \ldots, G_m, K)$, where, for all $i \in \{1, \ldots, m\}$, $G_i = (N_i, T, S_i, P_i)$ are regular grammars, called *component grammars*, satisfying $N_i \cap T = \emptyset$, and $K \subseteq \{Q_1, \ldots, Q_m\} \bigcap \bigcup_{i=1}^{m} N_i$ is a set of special symbols, called *communication symbols*.
A *configuration* of Π is an m-tuple $C = (x_1, \ldots, x_m),\ x_i = \alpha_i A_i,\ \alpha_i \in T^*, A_i \in (N_i \cup \varepsilon)$; we call x_i the *i-th component of the configuration* (resp. component). The *nonterminal cut* of configuration C is the $m-$tuple $N(C) = (A_1, A_2, \ldots, A_m)$. If $N(C)$ contains at least one communication symbol, it is denoted $NC(C)$ and called an *NC-cut*.

We say that a *configuration $X = (x_1, \ldots, x_m)$ directly derives a configuration $Y = (y_1, \ldots, y_m)$*, written as $X \Rightarrow Y$, if Y is derived from X by one *generative* or *communication* step (see below). Informally, in a communication step any occurrence of a communication symbol Q_i in X is substituted by the i-th component of X (assuming that this component does not contain any communication symbol).

1. (Generative step) If $|x_i|_K = 0$ for all i, $1 \leq i \leq m$, then
   $x_i \overset{G_i}{\Rightarrow} y_i$ for $x_i \in T^* N_i$, and
   $y_i = x_i$ for $x_i \in T^+$;
2. (Communication step) If $|x_i|_K > 0$ for some i, $1 \leq i \leq m$, then for each k such that $x_k = z_k Q_{j_k}$, $z_k \in T^*, Q_{j_k} \in K$, the following holds:
   (a) If $|x_{j_k}|_K = 0$, then $y_k = z_k x_{j_k}$ and $y_{j_k} = S_{j_k}$.
   (b) If $|x_{j_k}|_K = 1$, then $y_k = x_k$.

For all remaining indices t, for which x_t does not contain a communication symbol and Q_t does not occur in any of x_i's, we put $y_t = x_t$.

Now we describe the derivations in a PCGS. A *derivation* of a PCGS Π is a sequence of configurations $D = C_1, C_2, \ldots, C_t$, where $C_i \Rightarrow C_{i+1}$ in Π. If the first component of C_t is a terminal word w, then we usually write $D(w)$ instead of D. Analogously, we denote by $W(D)$ the terminal word generated within the derivation D. Every derivation can be viewed as a sequence of *generative steps* and *communication steps*.

If no communication symbol appears in any of the components, then we perform a *generative step* consisting of rewriting steps synchronously performed in each of the component grammars $G_i, 1 \leq i \leq m$. If any of the components is a terminal string, it is left unchanged. If any of the components contains a nonterminal that cannot be rewritten, the derivation is blocked. If the first

component is a terminal word w, the derivation finishes and w is the word generated by Π in this derivation.

If a communication symbol is present in any of the components, then a *communication step* is performed. It consists of replacing those communication symbols with the phrases they refer to for which the phrases do not contain communication symbols. Such an individual replacement is called a *communication*. Obviously, in one communication step at most $m-1$ communications can be performed. If some communication symbol was not replaced in this communication step, it may be replaced in one of the next communication steps. Communication steps are performed until no more communication symbols are present or the derivation is blocked, because no communication symbol can be replaced in the last communication step.

The *(terminal) language* $L(\Pi)$ generated by a PCGS Π is the set of terminal words generated by G_1 (in cooperation with the other grammars):

$$L(\Pi) = \{\, \alpha \in T^* \mid (S_1, \ldots, S_m) \Rightarrow^+ (\alpha, \beta_2, \ldots, \beta_m) \,\}.$$

Let $D = D(w) = C_0, C_1, \ldots, C_t$ be a derivation of w by Π; $D(w)$, Π, and w are fixed in what follows. With derivation $D(w)$, several notions can be associated which help to analyze the derivation of Π and to unambiguously determine w.

The *trace* of a (sub)derivation D is the sequence $T(D) = N(C_0)N(C_1)\ldots N(C_t)$ of the nonterminal cuts of the individual configurations of D.

The *NC-sequence* is defined analogously; $NCS(D)$ is the sequence of all NC-cuts in the (sub)derivation D. Let us recall that any NC-cut contains at least one communication symbol.

A *cycle in a derivation* is a subsequence $N(C), N(C_1), \ldots, N(C_j), N(C)$ of nonterminal cuts of the derivation[1] in which the first and last nonterminal cuts ($N(C)$) are identical. If $N(C)$ is an NC-cut, and none of the intermediate cuts $N(C_i)$ is an NC-cut, then the cycle is called a *communication cycle*. A *generative cycle* is defined analogously, we only require that *none* of its cuts is an NC-cut.

Note that, if there is a cycle in the derivation $D(w)$, then manifold repetition of the cycle is possible and the resulting derivation is again a derivation of some terminal word. We call a derivation $D(w)$ *reduced*, if every repetition of each of its cycles leads to a *longer* terminal word ω; $|w| < |\omega|$. Obviously, to every derivation $D(w)$ there is an equivalent reduced derivation $D'(w)$.

A *generative section* is a non-empty sequence of generative steps between two consecutive communication steps in $D(w)$[2], resp. before the first and/or after the last communication steps in $D(w)$.

The *degree of generation* $DG(D(w))$ is the number of generative sections of $D(w)$.

In what follows we only consider PCGS **without communication cycles**, i.e., $DG(D(w))$ is bounded by a constant depending only on Π, since $DG(D(w)) = 1+$ *number of communication steps of D(w).*

g(i,j) **(g(i,j,D(w)))** denotes the terminal part generated by G_i within the j-th generative section of $D(w)$, we call it the (i,j)-(generative) factor (of $D(w)$);

n(i,j) **(n(i,j,D(w)))** denotes the number of occurrences of $g(i,j)$ in w;

g(i,j,l) **(g(i,j,l,D(w)))** denotes the l-th occurrence of $g(i,j)$ in w, we call it the (i,j,l)-(generative) factor.

[1] More precisely it is a subsequence of the trace of the derivation.

[2] Note that if some communication cut contains more than one communication symbol, then there might be no generative step between two communication steps.

The *communication structure* of $D(w)$ $CS(D(w))$ captures the connection between the terminal word w and its particular derivation $D(w)$:

$CS(D(w)) = (i_1, j_1, l_1), (i_2, j_2, l_2), \ldots, (i_r, j_r, l_r)$, where $w = g(i_1, j_1, l_1), g(i_2, j_2, l_2) \ldots g(i_r, j_r, l_r)$. The *set* of these indices is denoted $I(D(w))$.

$\mathbf{N(j, D(w))} = \Sigma_i\ n(i, j, D(w))$, where the sum is taken over such i for which $\exists s$: $i{=}i_s$ & $(i_s, j_s, l_s){\in}$ $I(D(w))$.

The *degree of distribution* $DD(D(w))$ of $D(w)$ is the maximum over all (defined) $N(j, D(w))$.

Now we are ready to introduce the notions of *distribution complexity* and *generation complexity*. First, the distribution complexity of a derivation D (denoted $DD(D)$) is the degree of distribution introduced above.

Then the distribution complexity of a language and the associated complexity class are defined in the usual way (always considering the corresponding maximum): distribution complexity of a derivation $\rightsquigarrow$ distribution complexity of a language L (denoted $DD(L)$) as a function of the length of the word $\rightsquigarrow$ $f(n) - DD$ as the class of languages whose distribution complexity is bounded by $f(n)$. Analogously for the generation complexity.

Definition 2. *Let $f : \mathbb{N} \to \mathbb{N}^+$ be a function, Π be a* PCGS*, and $w \in L(\Pi)$.*

We say that Π is of distribution (generation) complexity *f if, for any derivation $D(w)$ by Π, $DD(D(w)) \leq f(|w|)$ ($DG(D(w)) \leq f(|w|)$). We say that a* language L has distribution (generation) complexity *f, if there exists a* PCGS *Π of distribution (generation) complexity f such that $L = L(\Pi)$. In particular, if $f(n) = k$ for each $n \geq 0$, then we say that L has distribution (generation) complexity k (resp. constant distribution (generation) complexity).*

Here, we are mainly interested in the classes of languages with constant communication and/or generation complexity. For natural numbers j, t we denote the corresponding complexity classes by j-DG, t-DD, and j-t-DDG, respectively.

We also utilize the notion of *communication complexity* from [1,6]. Informally, the communication complexity of a derivation $D(w)$ (denoted $com(D(w))$) is defined as the number of communications performed within the derivation $D(w)$. For example, the communication complexity of the derivation $D_1(w)$ in Example 1 is 2 (see Section 3 below).

Definition 3. *Let $f : \mathbb{N} \to \mathbb{N}^+$ be a function, Π be a* PCGS*, and $w \in L(\Pi)$. We say that Π* is of communication complexity *f if, for any derivation $D(w)$ by Π, $com(D(w)) \leq f(|w|)$. We say that a* language L has communication complexity *f, if there exists a* PCGS *Π of communication complexity f such $L = L(\Pi)$.*

We use $com(\Pi)$ and $com(L)$ to denote the communication complexity of a PCGS Π and a language L, respectively. The corresponding complexity class is defined in the usual way:

$$\mathsf{COM(f(n))} = \{\, L \mid com(L) \leq f(n) \,\}.$$

If $f(n) = k$ for each $n \in \mathbb{N}$, then we write COM(k) and speak about the class of languages with constant communication complexity.

Relevant observations about the derivations of PCGS (see [6] for more information) are summarized in the following fact.

Fact 1 *Let $com(\Pi) = k$. Then every derivation by Π is without communication cycle, and there are constants $d(\Pi), \ell(\Pi), s(\Pi), e(\Pi)$ such that*

1. *the number $n(i,j)$ of occurrences of individual $g(i,j)'s$ in a reduced derivation $D(w)$ is bounded by $d(\Pi)$; $n(i,j) \leq d(\Pi)$;*
2. *the length of the communication structure for every reduced derivation $D(w)$ is bounded by $\ell(\Pi)$;*
3. *the cardinality of the set of possible communication structures corresponding to reduced derivations by Π is bounded by $s(\Pi)$;*
4. *if more than $e(\Pi)$ generative steps of one generative section are performed in a reduced derivation $D(w)$, then at least one $g(i,j,D(w))$ is changed (see Example 1 in the next section).*

The following observation is a direct consequence of the corresponding definitions.

Observation 4 *Let Π be a* PCGS *of degree $m > 1$. Then $DD(\Pi) \leq com(\Pi) + (m-1)^{com(\Pi)}$.*

Proof. Let us recall that, for any derivation $D(w)$ of Π, $N(j, D(w)) = \Sigma_i\, n(i,j,D(w))$, where the sum is taken over such i for which $\exists s : i = i_s$ & $(i_s, j_s, l_s) \in I(D(w))$, and that the degree of distribution $DD(D(w))$ is the maximum over all (defined) $N(j, D(w))$. Obviously, $0 \leq j \leq com(\Pi)$. Since the number of realized communications in one communication step is bounded by $m-1$, the result follows. The augmenter $com(\Pi)$ is added because of the case $m = 2$. □

3 Bounded Degree of Distribution and Communication

We start this section by showing that a language generated by a PCGS Π with constant communication complexity can be analyzed (by reduction) by a t-SLnRR-automaton M.

The high-level idea is to merge the terminal word w with the information describing its reduced derivation $D(w)$ in a way allowing simultaneously the "simulation/reduction" of the derivation $D(w)$ and the correctness checking. Analysis by reduction is based on the deletion of the parts of a (characteristic) word which correspond to parts generated within one generative cycle. We call such a merged (characteristic) word a Π-description of w.

Let $(\alpha_1 A_1, \ldots, \alpha_m A_m)$ be the configuration at the beginning of the j-th generative section, and let

$(A_1, \ldots, A_m), (\alpha_{1,1}A_{1,1}, \ldots, \alpha_{1,m}A_{1,m}), \ldots (\alpha_{1,1}\alpha_{2,1}\ldots\alpha_{s,1}A_{s,1}, \ldots, \alpha_{1,m}\alpha_{2,m}\ldots\alpha_{s,m}A_{s,m})$
be the sub-derivation corresponding to this generative section. Merging the description of this sub-derivation into $g(i,j,l)$ we obtain the extended version of $g(i,j,l)$:

$$[b,i,j,l]\begin{pmatrix} A_1 \\ A_2 \\ \cdots \\ A_m \end{pmatrix} \alpha_{1,i} \begin{pmatrix} A_{1,1} \\ A_{1,2} \\ \cdots \\ A_{1,m} \end{pmatrix} \alpha_{2,i} \begin{pmatrix} A_{2,1} \\ A_{2,2} \\ \cdots \\ A_{2,m} \end{pmatrix} \ldots\ldots \alpha_{s,i} \begin{pmatrix} A_{s,1} \\ A_{s,2} \\ \cdots \\ A_{s,m} \end{pmatrix} [e,i,j,l].$$

This description of $g(i,j,l)$ is denoted *ex-*$g(i,j,l)$. We use *ex-*$g(i,j,l)$ to merge the (topological) information about derivation $D(w)$ into w. Obviously, we can speak about *traces* and *factor cycles* in *ex-*$g(i,j,l)$ similarly as we speak about traces and generative cycles in derivations.

Let $w, D(w), ex\text{-}g(i,j,l)$, be as above. Replace any $g(i,j,l)$ in w by *ex-*$g(i,j,l)$; the result is denoted *ex-*w. Then, concatenating the NC-sequence of $D(w)$, the communication structure given by $D(w)$, and *ex-*w we obtain the *Π-description of w*:

$$\Pi d(D(w)) = NCS(D(w))CS(D(w))ex\text{-}w.$$

Example 1. For an illustration we give a PCGS $\Pi_{ab}(2)$ generating the language

$$L_{ab}(2) = \{\, a^{i_1} b^{i_1} a^{i_2} b^{i_1+i_2} \mid i_j \in \mathbb{N} \,\} :$$

$G_1 : S_1 \to aS_1 | aQ_2, \quad G_2 : S_2 \to bZ_2, \quad G_3 : S_3 \to Z_3,$
$Z_2 \to aZ_2 | aQ_3, \qquad Z_2, \to bZ_2, \qquad Z_3 \to bZ_3.$
$Z_3 \to b,$
Let us consider the following derivation $D_1(w)$:

$$\left|\begin{matrix} S_1 \\ S_2 \\ S_3 \end{matrix}\right| \left|\begin{matrix} aS_1 \\ bZ_2 \\ Z_3 \end{matrix}\right| \left|\begin{matrix} a^2S_1 \\ b^2Z_2 \\ bZ_3 \end{matrix}\right| \left|\begin{matrix} a^3Q_2 \\ b^3Z_2 \\ b^2Z_3 \end{matrix}\right| \left\|\begin{matrix} a^3b^3Z_2 \\ S_2 \\ b^2Z_3 \end{matrix}\right| \left|\begin{matrix} a^3b^3aZ_2 \\ bZ_2 \\ b^2bZ_3 \end{matrix}\right| \left|\begin{matrix} a^3b^3a^2Q_3 \\ b^2Z_2 \\ b^2b^2Z_3 \end{matrix}\right| \left\|\begin{matrix} a^3b^3a^2b^2b^2Z_3 \\ b^2Z_2 \\ S_3 \end{matrix}\right| \left|\begin{matrix} a^3b^3a^2b^2b^2b \\ b^2bZ_2 \\ Z_3 \end{matrix}\right|$$

Then $g(1,1) = a^3, \quad g(1,2) = a^2, \quad g(1,3) = b,$
$g(2,1) = b^3, \quad g(2,2) = b^2, \quad g(2,3) = b,$
$g(3,1) = b^2, \quad g(3,2) = b^2, \quad g(3,3) = \varepsilon,$

and $w = a^3b^3a^2b^5 = g(1,1,1)g(2,1,1)g(1,2,1)g(3,1,1)g(3,2,1)g(1,3,1)$.

We see that the communication structure of $D(w)$ has the following form:

$CS(D_1(w)) = (1,1,1)(2,1,1)(1,2,1)(3,1,1)(3,2,1)(1,3,1).$

Hence, ex-$g(1,1,1)$, ex-$g(2,1,1)$, and ex-$g(3,1,1)$ are defined as follows:

$$[b,1,1,1] \begin{pmatrix} S_1 \\ S_2 \\ S_3 \end{pmatrix} a \begin{pmatrix} S_1 \\ Z_2 \\ Z_3 \end{pmatrix} a \begin{pmatrix} S_1 \\ Z_2 \\ Z_3 \end{pmatrix} a \begin{pmatrix} Q_2 \\ Z_2 \\ Z_3 \end{pmatrix} [e,1,1,1],$$

$$[b,2,1,1] \begin{pmatrix} S_1 \\ S_2 \\ S_3 \end{pmatrix} b \begin{pmatrix} S_1 \\ Z_2 \\ Z_3 \end{pmatrix} b \begin{pmatrix} S_1 \\ Z_2 \\ Z_3 \end{pmatrix} b \begin{pmatrix} Q_2 \\ Z_2 \\ Z_3 \end{pmatrix} [e,2,1,1],$$

$$[b,3,1,1] \begin{pmatrix} S_1 \\ S_2 \\ S_3 \end{pmatrix} \quad \begin{pmatrix} S_1 \\ Z_2 \\ Z_3 \end{pmatrix} b \begin{pmatrix} S_1 \\ Z_2 \\ Z_3 \end{pmatrix} b \begin{pmatrix} Q_2 \\ Z_2 \\ Z_3 \end{pmatrix} [e,3,1,1].$$

The NC-sequence given by $D_1(w)$ has the following form:

$$NCS(D_1(w)) = \begin{pmatrix} Q_2 \\ Z_2 \\ Z_3 \end{pmatrix} \begin{pmatrix} Q_3 \\ Z_2 \\ Z_3 \end{pmatrix}.$$

The following derivation $D_2(w)$ is obtained by deleting the first generative cycle from $D_1(w)$:

$$\left|\begin{matrix} S_1 \\ S_2 \\ S_3 \end{matrix}\right| \left|\begin{matrix} aS_1 \\ bZ_2 \\ Z_3 \end{matrix}\right| \left|\begin{matrix} a^2Q_2 \\ b^2Z_2 \\ bZ_3 \end{matrix}\right| \left\|\begin{matrix} a^2b^2Z_2 \\ S_2 \\ bZ_3 \end{matrix}\right| \left|\begin{matrix} a^2b^2aZ_2 \\ bZ_2 \\ bbZ_3 \end{matrix}\right| \left|\begin{matrix} a^2b^2a^2Q_3 \\ b^2Z_2 \\ bb^2Z_3 \end{matrix}\right| \left\|\begin{matrix} a^2b^2a^2bb^2Z_3 \\ b^2Z_2 \\ S_3 \end{matrix}\right| \left|\begin{matrix} a^2b^2a^2bb^2b \\ b^2bZ_2 \\ Z_3 \end{matrix}\right|$$

As the structure of $\Pi d(D(w))$ illustrates, the relevant properties summarized in the next observation should be obvious.

Observation 5 *Let $\Pi d(D(w))$ be a Π-description of w.*

(a) When a reduced derivation $D(w)$ is taken, then the length of $\Pi d(D(w))$ is bounded from above by $c_\Pi \cdot |w| + c_\Pi$, where c_Π is a constant depending on Π only.

(b) The terminal word w is easily obtained from $\Pi d(D(w))$ by deleting all symbols which are not terminal symbols of Π.

(c) Let $T(D(w))$ be the trace of $D(w)$, and $T(\Pi) := \{\, T(D(w)) \mid w \in L(\Pi) \,\}$. Then $T(\Pi)$ is a regular language, and the sets of NC-cuts and communication sequences of Π are finite. Note that a finite automaton is also able to check whether a given string x is a correct ex-$g(i,j,l)$, $NCS(D(w))$, or $CS(D(w))$ given by Π.

The construction of a k-SLnRR-automaton M accepting the characteristic language $L_C(M) = \{\Pi d(D(w)) \mid w \in L(\Pi)\}$ is outlined in the proof of the next theorem.

Theorem 1. *Let $k \in \mathbb{N}$ be a constant. Then to every* PCGS *Π of degree $m > 1$ with communication complexity k, there is a* t-SLnRR*-automaton M such that $L(\Pi) = L_P(M)$, and $t \leq k + (m-1)^k$.*

Proof. We outline the construction of a k-SLnRR-automaton M accepting the characteristic language $L_C(M) = \{\Pi d(D(w)) \mid w \in L(\Pi)\}$. Let ω be an input word to M. M starts to "simulate" the derivation $D(w)$ supposing $\omega = \Pi d(D(w))$. The main idea is to reduce ω in a cycle of M to $\omega_1 \in L_C(M)$ accordingly to a reduction (deleting) of a deterministically chosen occurrence of a generative cycle of $D(w)$; namely, the chosen generative cycle corresponds to the leftmost factor cycle of ω. Let us denote the newly reduced derivation $D(w_1)$ (using the corresponding shortened word w_1 for that). This process can be repeated until a word with no factor cycle is obtained. Since the length of a terminal word generated without a (generative or communication) cycle is bounded by a constant depending on Π only, the set of these words is finite, and so is the set of their Π-descriptions. It is therefore easy to check whether the Π-description of a "cycle free" word belongs to $L_C(M)$ or not.

Simultaneously, we need to ensure that every $\omega' \notin L_C(M)$ is either directly rejected or reduced to some $\omega'' \notin L_C(M)$. To check the mutual (in)consistency of $NCS(D(w))$, $SC(D(w))$, the syntax of the factor part *ex-w*, and the individual factors of *ex-w*, M uses $NCS(D(w))$ and $SC(D(w))$. In this way the correctness preserving property of M is ensured by the finite control of M.

The more detailed description of a computation of M follows. First, we give the description of one cycle of M, and then the rejecting and accepting tail computations are described.

A cycle by M:

Consider a cycle C of M on $\omega = NCS(D(w))\ CS(D(w))$ *ex-w*. In order to check the consistency of ω, M stores $NCS(D(w))$ and $SC(D(w))$ in its finite control, and then it deterministically finds the leftmost factor cycle in *ex-w*. Realize that, for a correct input to M, this leftmost factor cycle is located within some extended factor *ex-*$g(i,j,1)$. Let us denote this (occurrence of the) generative cycle $c(i,j)$. Since $c(i,j)$ is derived in the j-th generative section of $D(w)$ by the i-th grammar of Π, within the cycle C

– machine M has to delete the first occurrence of $c(i,j)$ in all extended factors *ex-*$g(i,j,1)$, *ex-*$g(i,j,2), \cdots,$ *ex-*$g(i,j,l(i,j))$;
– because of the correctness preserving property M also has to delete the first occurrence of the factor cycles with the same trace as $c(i,j)$ placed in an (extended) (i_I,j,l_L)-factor (*ex-*$g(i_I,j,l_L)$'s) such that (i_I,j,l_L) is an item from $CS(D(w))$ and $i \neq i_I$.

It is not hard to see that the described construction guarantees that the number of rewrite operations within one cycle of M to be bounded from above by k, the communication complexity of the PCGS Π.

More precisely:

1. Let $ex\text{-}g(i,j,1) = [b,i,j,1]\beta_{\ell,i}\underbrace{NC_1\alpha_{1,i}NC_2\alpha_{2,i}\dots NC_{s-1}\alpha_{s,i}NC_1}_{c(i,j)}\beta_{R,i}[e,i,j,1]$, where $\beta_{\ell,i}$ does not contain any factor cycle, and $\alpha_i := \alpha_{1,i}\dots\alpha_{s,i}$ is the (possibly empty) terminal part generated within $c(i,j)$. M stores $[j, NC_1\dots NC_1]$, $[(i,j),\beta_{\ell,i},\alpha_i)]$ in its finite control and replaces $c(i,j)$ by NC_1 in $ex\text{-}g(i,j,1)$. The stored information enables M to check the requested correctness of the other extended sub-words starting with $[b,i_I,j,l_L]$. Namely, for $i_I = i$ the prefix has to be exactly of the form $[b,i,j,1]\beta_{\ell,i}c(i,j)$, while for $i_I \neq i$, it should only fit the trace of the prefix.
2. M thereafter moves its head to the right until the next occurrence of $[b,i_I,j,l_L]$ or the right sentinel is found.
   – If the visited symbol is of the form $[b,i_w,j,1]$, where $i_I \neq i$, then M proceeds in the same way as in the case for $[b,i,j,1]$.
   – If the visited symbol is of the form $[b,i_I,j,l_L]$, where $l_L \geq 2$, M should have some information for (i',j) stored in its state and is therefore able to check the desired correctness of the prefix. If the prefix fits the stored patterns, M reduces the relevant cycle in the same way as in the case for $[b,i_I,j,1]$.

   – If the visited symbol is the right sentinel and no inconsistence has been found, then M restarts.

Rejecting tail computation:

M computes in the same way as described for a cycle until some inconsistency (error) is found; in this situation M rejects. Let us recall that ω is a correct word belonging to $L_C(M)$ iff it has the form $\omega = NCS(D(w))CS(D(w))ex\text{-}w$. The (main) inconsistencies are of the following types:

a) There is no prefix of ω which corresponds to a $NCS(D)$ by Π.
b) Let $\omega = NCS(D)\alpha$. There is no prefix of α which corresponds to a $CS(D)$ by Π.
c) Let $\omega = NCS(D)CS(D_1)\alpha$. D and D_1 cannot be equal.
d) Let $\omega = NCS(D)CS(D)\alpha$. The string α disturbs the (regular part of) the syntax of a sequence of factors given by $NCS(D)$ and $CS(D)$.
e) Some factor F of α does not contain a factor cycle which should be deleted accordingly to the other factors of α (see the cycle part).

Using its finite control M is able to check all inconsistencies of the previous types.

Accepting tail computation:

An accepting tail computation occurs when none of the above described cases occurs. In this situation, M only checks whether the word on the tape ω_c is a correct member of the finite subset of $L_C(M)$ which does not contain any factor cycle.

It is not hard to see that M is a deterministic k-SLnRR-automaton accepting the language $L_C(M)$ described above. □

Let M be as in the proof of Theorem 1. Obviously, we can obtain $L_P(M)$ from $L_C(M)$ by a projection of $L_C(M)$ onto the terminal symbols of Π. That means $L_P(M) = L(\Pi)$.

Corollary 2. *Let $k \in \mathbb{N}$ be a constant. Then to every* PCGS *Π of degree $m > 1$ with communication complexity k, there is a* t-SLnRR*-automaton M such that $L(\Pi) = L_P(M)$, and $t \leq k + (m-1)^k$.*

Considering the previous proof,
– the length of a terminal word generated without a (generative or communication) cycle is bounded by a constant depending on Π only;
– whenever there is a generative cycle in the derivation $D(w)$ for $w \in L(\Pi)$, then that cycle can be iterated. This yields the following pumping lemma for PCGS.

Corollary 3. (PCGS Pumping and Cutting Lemma) *Let $k \in \mathbb{N}$ be a constant. Then to every* PCGS *Π of degree $m > 1$ with communication complexity k, there are constants t, p, $0 < t \leq m^k$, such that any $w \in L(\Pi)$, $|w| > p$, can be written in the form $w = y_0 x_1 y_1 \cdots x_t y_t$, where $p \geq |x_1 x_2 \cdots x_t| > 0$, and $y_0 x_1^i y_1 \cdots x_t^i y_t \in L(\Pi)$ for any non-negative integer i.*

The next corollary follows directly from the cutting part of the previous corollary.

Corollary 4. (PCGS semi-linearity) *Let $k \in \mathbb{N}$ be a constant, and Π is a* PCGS *with communication complexity k. Then the language $L(\Pi)$ is semi-linear.*

The constant communication complexity of PCGS Π is important to guarantee the semi-linearity of $L(\Pi)$. As demonstrated by Example 2 (taken from [7]), unconstrained PCGSs are able to generate also non-semi-linear languages.

Example 2. This example illustrates how a communication cycle can be used to generate the language $L = \{\, a^{2^n} \mid n \in \mathbb{N} \,\}$. Note that the only generated symbol a is generated within the first generative section:

$$\begin{array}{ll} P_1: & \{S_1 \to aB \mid Q_2,\ B_1 \to B \mid \varepsilon\}, \\ P_2: & \{S_2 \to Q_1,\ B \to Q_3\}, \\ P_3: & \{S_3 \to Q_1,\ B \to B_1\}, \end{array}$$

$$\begin{array}{l|l||l|l|l|r|l||r|r|r|r|r||r} G_1: & S_1 & aB & S_1 & Q_2 & Q_2 & aaB_1 & aaB & S_1 & Q_2 & Q_2 & aaaaB_1 & aaaa \\ G_2: & S_2 & Q_1 & aB & aQ_3 & aaB_1 & S_2 & Q_1 & aaB & aaQ_3 & aaaaB_1 & S_2 & Q_1 \\ G_3: & S_3 & Q_1 & aB & aB_1 & S_3 & S_3 & Q_1 & aaB & aaB_1 & S_3 & S_3 & Q_1 \end{array}$$

□

Let $L_{k-copy} := \{\, (w\#)^k\# \mid w \in \{a,b\}^+ \,\}$. We use L_{k-copy} to show that the result of Theorem 1 is not far from optimal concerning the number of rewrites per cycle.

Proposition 1. $L_{k-copy} \in \mathcal{L}(\mathsf{k\text{-}SLnRR}) - \mathcal{L}(\mathsf{(k\text{-}1)\text{-}NLnRR})$, *while* $L_{k-copy} \in \mathsf{COM}((\log k))$.

Proof. First, we sketch the upper bound part for PCGS. Using quite a sophisticated construction that involves nondeterminism and timing we obtain a PCGS Π_{copy} that generates L_{k-copy} with communication complexity $O(\log k)$.

The PCGS generating L_{k-copy} consists of several groups of grammars:

$\mathbf{G_2}$	is used to generate w;
$\mathbf{C_1, C_2, C_3}$	are used to prepare strings of the form $(w\#)^{2^i} W_i$ one-by-one: $(w\#)^{2^i} W_i \quad \rightsquigarrow \quad (w\#)^{2^{i+1}} W_{i+1}$;
$\mathbf{P_i}$	is used to store $(w\#)^{2^i} W_i, 0 \leq i \leq \lfloor \log k \rfloor + 1$;
$\mathbf{Stop_3}$	is used to restart grammar C_3 whenever necessary.

Let $b_m \dots b_0, m = \lfloor \log k \rfloor + 1$, be the binary representation of k. Then, $\omega = (w\#)^k$ is composed in grammar G_1 with the help of communication as a concatenation of "pieces" stored in those P_{i_j}'s for which the corresponding value of b_{i_j} is 1.

While w is generated in G_2 at the beginning of the derivation, all other grammars are "idling", nondeterministically choosing $S_X \to S_X$ as long as necessary:

$$\begin{array}{|l|l|l|l|l|l|l|l|}
\hline
G_2 & S_2 & w_1 S_2 & & & w\#W_1 & S_2 & \\
C_1 & S_{C_1} & S_{C_1} & & & Q_{G_2} & w\#W_1 & \\
C_2 & S_{C_2} & S_{C_2} & \cdots & & S_{C_2} & Q_{C_1} & \cdots \\
C_3 & S_{C_3} & S_{C_3} & & & S_{C_3} & Q_{C_1} & \\
\hline
\end{array}$$

Then G_2 decides to idle and all the other grammars start working in a cycle:

$$\begin{array}{|l|l|l|l|l|l|}
\hline
C_1 & (w\#)^{2^i} W_i & S_{C_1} & Q_{C_2} & Q_{C_2} & (w\#)^{2^{i+1}} W'_{i+1} & (w\#)^{2^{i+1}} W_{i+1} \\
C_2 & Q_{C_1} & (w\#)^{2^i} W_i & (w\#)^{2^i} Q_{C_3} & (w\#)^{2^{i+1}} W'_{i+1} & S_{C_2} & Q_{C_1} \\
C_3 & Q_{C_1} & (w\#)^{2^i} W_i & (w\#)^{2^i} W'_{i+1} & S_{C_3} & S_{C_3} & Q_{C_1} \\
P_i & & & Q_{C_2} & Q_{C_2} & (w\#)^{2^{i+1}} W'_{i+1} & (w\#)^{2^{i+1}} W_{i+1} \\
Stop_3 & & & & Q_{C_3} & & \\
\hline
\end{array}$$

At the end G_1 nondeterministically wakes up and finishes generation by concatenating parts that are necessary to be concatenated. For an illustration let $k = 5$; written in binary $k = 101$

$$\begin{array}{|l|l|l|l|l|l|}
\hline
G_1 & Q_{P_1} & (w\#)^{2^0} W_0 & (w\#)^{2^0} Q_{P_2} & (w\#)^{2^0} (w\#)^{2^2} W_2 & (w\#)^{2^0} (w\#)^{2^2} \# = \boxed{(w\#)^{101}\#} \\
P_0 & (w\#)^{2^0} W_0 & S_{P_1} & S_{P_1} & S_{P_1} & S_{P_1} \\
P_1 & (w\#)^{2^1} W_1 & & & & \\
P_2 & (w\#)^{2^2} W_2 & (w\#)^{2^2} W_2 & (w\#)^{2^2} W_2 & S_{P_2} & S_{P_2} \\
\hline
\end{array}$$

Now we analyze the recognition of L_{k-copy} by FRR automata. It is not hard to construct an k-SLnRR-automaton M_{cp} such that $L_P(M_{cp}) = L_{k-copy}$. In fact, a very similar construction was presented by [5].

For the lower-bound part let $k > 1$. Assume that M is a $(k-1)$-NSLnRR-automaton on Γ such that $L_{\mathrm{P}}(M) = L_{k-copy}$. Let $A(m,n,k) := ((a^m b^m)^n \#)^k \#$, where $m, n \in \mathbb{N}_+$ are sufficiently large numbers. Obviously, $A(m,n,k) \in L_{k-copy}$. Hence, there exists an expanded version $w \in \Gamma^*$ of $A(m,n,k)$ such that $w \in L_{\mathrm{C}}(M)$. Assume that w is a shortest expanded version of $A(m,n,k)$ in $L_{\mathrm{C}}(M)$ and consider an accepting computation C of M on input w. Based on the Pumping Lemma it is easily seen that this computation cannot just consist of an accepting tail computation. Thus, it begins with a cycle of the form $w \vdash_M^{\mathrm{c}} x$. From the Weak Correctness Preserving Property [Section 2] and from the assumption that C is accepting it follows that $x \in L_{\mathrm{C}}(M)$, which in turn implies that $\mathsf{Pr}^{\Sigma}(x) \in L_{k-copy}$. As all rewrite steps of M are length-reducing, $|x| < |w|$ follows. Thus, our choice of $w \in L_{\mathrm{C}}(M)$ as a shortest expanded version of $A(m,n,k)$ implies that x is not an expanded version of $A(m,n,k)$. Since C executes at most $k-1$ rewrite steps in the above cycle, it follows that $\mathsf{Pr}^{\Sigma}(x) \notin L_{k-copy}$. Hence, $L_{\mathrm{P}}(M) \neq L_{k-copy}$, which implies that $L_{k-copy} \notin \mathcal{L}_{\mathrm{P}}((k-1)\text{-NSLnRR})$. □

3.1 Other Separation Results

In what follows we show several results relating communication and distribution complexity of a language to the number of rewrites in one cycle used to recognize it as a proper language.

First, the language $L_{ab}(k) := \{\, a^{i_1}b^{i_1}a^{i_2}b^{i_1+i_2}\dots a^{i_k}b^{i_1+\dots+i_k} \mid i_1,\dots,i_k \geq 1 \,\}$ is used to show hierarchies for both PCGS and SLnRR automata. While an infinite hierarchy of communication complexity is known [7], a new infinite hierarchy for linearized FRR-automata is given in Proposition 3.

Proposition 2. [7] *For all* $j \in \mathbb{N}_+$, $L_{\rm ab}(j) \in \mathsf{COM(j)} - \mathsf{COM(j\text{-}1)}$.

Proposition 3. *For all* $j \in \mathbb{N}_+$, $L_{\rm ab}(j) \in \mathcal{L}_{\rm P}(j\text{-}\mathsf{SLnRR}) - \mathcal{L}_{\rm P}((j-1)\text{-}\mathsf{NLnRR})$

Proof. The construction of a strongly linearized deterministic j-FRR-automaton $M_{\rm ab}^{(j)}$ that recognizes $L_{\rm ab}(j)$ is obvious. In its cycles, moving from left to right, the automaton (i) finds the leftmost substring $aabb$, (ii) rewrites it with ab, (iii) deletes one b from all following groups of b's, (iv) while checking that the word considered has the form $(a^+b^+)^j$. The only word accepted by this automaton in a tail computation is $ababb\dots ab^j$.

We see that $M_{\rm ab}^{(j)}$ is able to work in the way outlined above with a window of the size 4, and that $M_{\rm ab}^{(j)}$ recognizes $L_C(M_{\rm ab}^{(j)})$ by using at most j rewritings in a cycle. From that it easily follows that $L_{\rm ab}(j) \in \mathcal{L}_{\rm P}(j\text{-}\mathsf{SLnRR})$.

Now let $j > 1$, and assume that M is nondeterministic $(j-1)$-NLnRR-automaton on Γ such that $L_{\rm P}(M) = L_{\rm ab}(j)$. Let us consider words of the shape $C(m,j) := a^m b^m a^m b^{2m} \cdots a^m b^{j\cdot m}$ where $m \in \mathbb{N}_+$ are sufficiently large. Obviously, $C(m,j) \in L_{\rm ab}(j)$. Hence, there exists an expanded version $\omega \in \Gamma^*$ of $C(m,j)$ such that $\omega \in L_{\rm C}(M)$. Assume that ω is a shortest expanded version of $C(m,j)$ in $L_{\rm C}(M)$. Then there is an accepting computation C of M on input ω, but based on the Pumping Lemma it is easily seen that this computation cannot just consist of an accepting tail computation. Thus, it begins with a cycle of the form $\omega \vdash_M^{\rm c} x$. From the Weak Correctness Preserving Property it follows that $x \in L_{\rm C}(M)$, which in turn implies that $\mathsf{Pr}^{\Sigma}(x) \in L_{\rm ab}(j)$. As all rewrite steps of M are length-reducing, $|x| < |\omega|$ follows. Thus, our choice of $\omega \in L_{\rm C}(M)$ as a shortest expanded version of $C(m,j)$ implies that x is not an expanded version of $C(m,j)$. Since M executes at most $j-1$ rewrite steps in the above cycle, it follows that $\mathsf{Pr}^{\Sigma}(x) \notin L_{\rm ab}(j)$. Hence, $L_{\rm P}(M) \neq L_{\rm ab}(j)$, which implies that $L_{\rm ab}(j) \notin \mathcal{L}_{\rm P}((j-1)\text{-}\mathsf{NLnRR})$. □

For $t \in \mathbb{N}_+$, separation of PCGS with distribution complexity t from the proper languages of nondeterministic linearized FRR-automata with at most $t-1$ rewrites per cycle is done with the help of language L_t

$$L_t := \{\, c_1 w d \cdots c_t w d \mid w \in \{a,b\}^* \,\},$$

where $\Sigma_1 := \{c_1,\dots,c_t,d\}$ is a new alphabet disjoint from $\Sigma_0 := \{a,b\}$.

Proposition 4. *For all* $t \in \mathbb{N}_+$, $L_t \in \mathcal{L}(t\text{-}\mathsf{DD}) \setminus \mathcal{L}_{\rm P}((t-1)\text{-}\mathsf{NLnRR})$.

Proof. To show $L_t \in \mathcal{L}(t\text{-DD})$ it is enough to take the following PCGS with $t+1$ component grammars:

$$(S_1,S_2,\dots,S_{t+1}) \Rightarrow^* (c_1Q_2, c_2Q_3, \dots, c_tQ_{t+1}, wd) \Rightarrow^* (c_1wdc_2wd\dots c_twd, S_2,\dots,S_t,S_{t+1}).$$

For the lower-bound part we use a technique similar to that in [5]. Let $M = (Q,\Sigma,\Gamma,\mathrm{¢},\$,q_0,k,\delta)$ be a nondeterministic linearized FRR-automaton that executes at most $t-1$ rewrites per cycle, where $\Sigma := \Sigma_0 \cup \Sigma_1$. Assume that $L_{\rm P}(M) = L_t$ holds. Consider the word $w := c_1a^nb^nd\cdots c_ta^nb^nd \in L_t$, where n is a large enough integer. Then there exists an expanded version $W \in \Gamma^*$ of w such that $W \in L_{\rm C}(M)$. Let W be a shortest expanded version of w in $L_{\rm C}(M)$. Consider an accepting computation of M on input W. Clearly this cannot just be an accepting tail, and hence, it begins

with a cycle of the form $W \vdash_M^c W_1$. From the Correctness Preserving Property it follows that $W_1 \in L_C(M)$, which implies that $w_1 := \mathsf{Pr}^{\Sigma}(W_1) \in L_t$. As $|W_1| < |W|$, we see from our choice of W that $w_1 \neq w$, that is, $w_1 = c_1x_1d\cdots c_tx_1d$ for some word $x_1 \in \Sigma_0^*$ of length $|x_1| < 2n$. However, in the above cycle M executes at most $t-1$ rewrite steps, that is, it cannot possibly rewrite each of the t occurrences of a^nb^n into the same word x_1. It follows that $w_1 \notin L_t$, implying that $L_t \notin \mathcal{L}_P((t-1)\text{-NLnRR})$. □

As $\mathcal{L}(t\text{-DD}) \subseteq \mathcal{L}_P(t\text{-SLnRR})$, we obtain the following hierarchies from Proposition 4, where $\mathcal{L}_P(t\text{-DD})$ just denotes the class $\mathcal{L}(t\text{-DD})$.

Theorem 2. *For all* $\mathsf{X} \in \{\mathsf{DD}, \mathsf{LnRR}, \mathsf{SLnRR}, \mathsf{NLnRR}, \mathsf{NSLnRR}\}$*, and all* $t \geq 1$*,*

$$\mathcal{L}_P(t\text{-}\mathsf{X}) \subset \mathcal{L}_P((t+1)\text{-}\mathsf{X}) \subset \bigcup_{t\geq 1} \mathcal{L}_P(t\text{-}\mathsf{X}) \subset \mathcal{L}_P(\mathsf{X}).$$

4 Skeletons

In this part we define the notions of *skeleton* and *islands*, whose introduction is motivated by our attempt to model two basic kinds of coordinated segments in (Czech, German, Slovak) sentences. The islands in a level of skeleton serve to denote places of coordinated segments which are coordinated in a mutually dependent (bound) way. The different levels of islands serves for modelling of the independence of segments. A technical example how to construct skeletons is given by the construction in the proof of Theorem 1. In fact, skeletons are only defined for t-SLnRR-automata that fulfill certain additional requirements.

Definition 6. *Let* $M = (Q, \Sigma, \Gamma, ¢, \$, q_0, k, \delta)$ *be a* t-SLnRR*-automaton for some* $t \in \mathbb{N}_+$*, and let* $s \in \mathbb{N}_+$*. Let* $SK(s) = \{ c_{i,j} \mid 1 \leq i \leq t, 1 \leq j \leq s \}$ *be a subalphabet of cardinality* $t \cdot s$ *of* $\Gamma' = \Gamma \cup \{¢, \$\}$*. For each* $j \in \{1, \ldots, s\}$*, let* $SK(s,j) = \{c_{1,j}, \ldots, c_{t,j}\}$ *be the* j*-th* level *of* $SK(s)$*. We say that* $SK(s)$ *is an* s-skeleton (skeleton) *of* M *if the following holds:*

1. *For all* $w \in L_C(M)$ *and all* $c \in SK(s)$*,* $|w|_c \leq 1$*, that is,* w *contains at most one occurrence of* c*.*
2. *Each rewrite operation of* M *deletes a single continuous factor from the actual contents of the window, and at that point the window must contain exactly one occurrence of a symbol from* $SK(s)$*. This symbol is either in the first or in the last position of the window.*
3. *If a cycle* C *of* M *contains a rewrite operation during which a symbol* $c_{i,j} \in S(s,j)$ *is in the first or last position of the window, then every rewrite operation during* C *is executed with some element of* $S(s,j)$ *in the first or last position of the window.*
4. *If* $w \in L_C(M)$*,* $w = xyz$*, such that* $|y| > k$*, and* y *does not contain any element of* $SK(s)$*, then starting from the restarting configuration corresponding to* w*,* M *will execute at least one cycle before it accepts.*

The elements of $SK(s)$ *are called* islands *of* M*. We say that* $SK(s)$ *is a* left skeleton *of* M*, if* M *executes rewrite operations only with an island in the leftmost position of its window.*

Thus, in each cycle M performs up to t rewrite (that is, delete) operations, and during each of these operations a different island $c_{i,j}$ of the same level $SK(j)$ is inside the window. As there are s such levels, we see that there are essentially just s different ways to perform the rewrite steps of a cycle.

By $\mathcal{L}_{\mathrm{P}}(t\text{-}\mathsf{SK}(s))$ (resp. by $\mathcal{L}_{\mathrm{P}}(t\text{-}\mathsf{LSK}(s))$) we denote the class of proper languages of t-SLnRR-automata with s-skeletons (resp. with left s-skeletons). The corresponding classes of characteristic languages are denoted by $\mathcal{L}_{\mathrm{C}}(t\text{-}\mathsf{SK}(s))$ (resp. by $\mathcal{L}_{\mathrm{C}}(t\text{-}\mathsf{LSK}(s))$).

Observe that the symbols of the form $[b, i, s, l]$ in the construction of an s-SLnRR-automaton M accepting the language $L_C(M) = \{\, \Pi d(D(w)) \mid w \in L(\Pi) \,\}$ play the role of islands for M, and their complete set is a left skeleton for M. This observation serves as the basis for the proof of the next corollary. Recall that s-t-DDG denotes the class of PCGSs that have simultaneously generation degree s and distribution degree t.

Corollary 5. *For all* $s, t \in \mathbb{N}_+$, $\mathcal{L}(s\text{-}t\text{-}\mathsf{DDG}) \subseteq \mathcal{L}_{\mathrm{P}}(t\text{-}\mathsf{LSK}(s))$.

To separate PCGSs of generation complexity t and distribution complexity s from the class of proper languages of $(t-1)$-LSK(s)-automata we define language $L_{(t,s)}$. The language is based on a kind of bounded concatenation of L_t. For $s, t \in \mathbb{N}_+$ and $i \leq s$, let

$$L_{(t)} := \{\, c_1 w d \cdots c_t w d \mid w \in \{a, b\}^*, c_i \in \Sigma_i, d \in \Delta \,\},$$

where $\Sigma_1, \ldots \Sigma_s, \Delta$ are new alphabets disjoint from $\{a, b\}$. Then we take $L_{(t,s)} := (L_{(t)})^s$.

Proposition 5. *For all* $s, t \in \mathbb{N}_+$,

(a) $L_{(t,s)} \in \mathcal{L}(s\text{-}t\text{-}\mathsf{DDG})$,
(b) $L_{(t,s)} \notin \mathcal{L}_{\mathrm{P}}(t\text{-}\mathsf{SK}(s-1))$ *for* $s > 1$, *and*
(c) $L_{(t,s)} \notin \mathcal{L}_{\mathrm{P}}((t-1)\text{-}\mathsf{SK}(s))$ *for* $t > 1$.

Sketch of the proof. Note that $L_t = L_{(t)} = L_{(t,1)}$, when $|\Sigma_1| = \cdots = |\Sigma_t| = |\Delta| = 1$.
(a) For the upper-bound part we use a PCGS with $(t{+}s)$ component grammars, which realize s phases corresponding to s generative sections. The group of grammars $G_{s+1}, \ldots, G_{s+t}$ plays the role of $G_2, \ldots, G_{t+1}$ from the proof of Proposition 4, while the component grammars $G_1, \ldots, G_s$ play the role of the grammar G_1 from that proof. At the end of the p-th generative section, there is a word $\omega_p i$ present in component grammar G_{s+1}, where $\omega_p = c_{1,p} w_p d_p \ldots c_{t,p} w_p d_p$ is a terminal word and i, $1 \leq i \leq s$, is a nonterminal symbol indicating that G_i is the grammar into which ω_p should be communicated. Finally, the synchronized communication concatenates all ω's in an appropriate[3] way in component grammar G_1.

(b) Assume that M is a t-SK$(s-1)$-automaton such that $L_{\mathrm{P}}(M) = L_{(t,s)}$. Thus, M has a $(s-1)$-skeleton $SK(s-1) = \{\, c_{i,j} \mid 1 \leq i \leq t,\ 1 \leq j \leq s-1 \,\}$. Now assume that, for $i = 1, \ldots, s$, $w_i \in L_{t,i}$, that is, $w := w_1 w_2 \cdots w_s \in L_{(t,s)}$. Further, let W be an expanded version of w. For each cycle of M in an accepting computation on input W, there exists an index $j \in \{1, \ldots, s-1\}$ such that each rewrite step of this cycle is executed with an island $c_{i,j}$ in the left- or rightmost position of the window. From the proof of Proposition 4 we see that, for each of the factors $L_{t,j}$, t rewrite steps per cycle are required. Thus, each of the factors W_i must contain t islands, that is, W must contain at least $t \cdot s$ islands. However, as the word $W \in L_{\mathrm{C}}(M)$ contains at most a single occurrence of each symbol of the set $SK(s-1)$, and as $|SK(s-1)| = t \cdot (s-1)$, W can contain at most $t \cdot (s-1)$ islands. This contradicts the observation above, implying that $L_{(t,s)}$ is not the proper language of any t-SK$(s-1)$-automaton.

[3] The construction of PCGS heavily utilizes nondeterminism. In case of "wrong" nondeterministic choices the derivation is blocked.

(c) For the lower-bound part recall Proposition 4, where $L_t \notin \mathcal{L}_{\mathrm{P}}((t-1)\text{-NLnRR})$ is shown. From that proof it follows that $L_{(t,s)} \notin \mathcal{L}_{\mathrm{P}}((t-1)\text{-NLnRR})$. As $(t-1)\text{-SK}(s)$-automata are a special type of $(t-1)$-SLnRR-automata, the non-inclusion result in (c) follows. □

Next we consider the language $L_{pe} := \{\, wcw^R \mid w \in \{0,1\}^* \,\}$. By taking the symbol c as an island, we easily obtain the following result.

Proposition 6. $L_{pe} \in \mathcal{L}_{\mathrm{P}}(2\text{-SK}(1))$.

On the other hand, this language cannot be accepted, if we restrict our attention to left skeletons.

Proposition 7. $\forall s,t \in \mathbb{N}_+ : L_{pe} \notin \mathcal{L}_{\mathrm{P}}(t\text{-LSK}(s))$.

Proof. Assume that M is a t-LSK(s)-automaton such that $L_{\mathrm{P}}(M) = L_{pe}$, that is, M has a left skeleton $SK(s) = \{\, c_{i,j} \mid 1 \le i \le t,\ 1 \le j \le s \,\}$. Let $w = (a^n b^n)^m$, where $n, m \in \mathbb{N}_+$ are sufficiently large, and let $z = wcw^R \in L_{pe}$. Then there exists a (shortest) expanded version $Z \in \Gamma^+$ of z such that $Z \in L_{\mathrm{C}}(M)$. Hence, the computation of M on input Z is accepting, but because of the Pumping Lemma it cannot just consist of an accepting tail, that is, it begins with a cycle $Z \vdash_M^c V$, where $V \in L_{\mathrm{C}}(M)$ and $|V| < |Z|$. Thus, $v = \mathsf{Pr}^{\Sigma}(V) \in L_{pe}$, but $v \neq z$. In this cycle M performs up to t delete operations that each delete a continuous factor of Z to the right of an island $c_{i,j}$ for some level $j \in \{1,\ldots,s\}$. It follows that $v = w_1 c w_1^R$ for some word $w_1 \in \{a,b\}^*$ satisfying $|w_1| < |w|$, and that w_1 is obtained from w by deleting some factors, and w_1^R is obtained from w^R by deleting the corresponding reverse factors. When deleting a factor x within the prefix w to the right of an island $c_{i,j}$, then this means that this island "moves" to the right inside w, that is, from $c_{i,j}xy$ the factor $c_{i,j}y$ is obtained. Here we just consider the projection of Z onto $(SK(s,j) \cup \{a,b\})^*$. Now when the corresponding factor x^R is deleted from w^R, then it is to the right of an island $c_{i',j}$, that is, from $y^R c_{i',j} x^R$ the factor $y^R c_{i',j}$ is obtained. Thus, while for deleting the factor y of w the same island $c_{i,j}$ could be used in a later cycle, an island different from $c_{i',j}$ is needed for y^R. The same argument applies to the case that the roles of w and w^R are interchanged. This means that in the process of synchronously processing w and w^R, the same island can be used repeatedly in subsequence cycles within one of the two parts, but the corresponding deletions in the other part require new islands in each cycle. If w is of sufficient length, then it follows that $t \cdot s$ islands will not suffice. Hence, $L_{pe} \notin \mathcal{L}_{\mathrm{P}}(t\text{-LSK}(s))$. □

The results above yield the following consequences.

Theorem 3. *For all* $\mathsf{X} \in \{\mathsf{LSK}, \mathsf{SK}\}$*, and all* $s,t \ge 1$*, we have the following* proper *inclusions:*

(a) s-t-DDG $\subset$ $(s+1)$-t-DDG.
(b) s-t-DDG $\subset$ s-$(t+1)$-DDG.
(c) $\mathcal{L}_{\mathrm{P}}(t\text{-X}(s)) \subset \mathcal{L}_{\mathrm{P}}((t+1)\text{-X}(s))$.
(d) $\mathcal{L}_{\mathrm{P}}(t\text{-X}(s)) \subset \mathcal{L}_{\mathrm{P}}(t\text{-X}(s+1))$.
(e) s-t-DDG $\subseteq \mathcal{L}_{\mathrm{P}}(t\text{-LSK}(s)) \subseteq \mathcal{L}_{\mathrm{P}}(t\text{-SK}(s))$.
(f) $\mathcal{L}_{\mathrm{P}}(t\text{-LSK}(s)) \subset \mathcal{L}_{\mathrm{P}}(t\text{-SK}(s))$ *for* $t \ge 2$.

5 Conclusion

The study of the relation between PCGS and FRR is motivated by computational linguistics, as both models are useful in that field. Here we have established the basic relation between the computational power of these two models, and we have studied the relevant complexity measures of PCGS

and restrictions on computation of FRR. In fact, we have succeed in showing infinite hierarchies both for PCGSs and FRRs. However, the question of whether j-k-DDG is equal to $\mathcal{L}_{\mathrm{P}}(j\text{-LSK(k)})$ or not remains open.

We also believe that properly using nondeterminism the next conjecture can be shown.

Conjecture 1. For any $L \in$ j-k-DDG, there is a correctness preserving k-NSLnRR-automaton M with a left j-skeleton $SK(j)$ such that $L = L_P(M)$, and M has no auxiliary symbols outside of $SK(j)$.

In the close future we will use PCGS, and the corresponding models of restarting automata, to express formal translations (transformations). In this way we will support the more precise characterization of the basic task of a formal description of a natural language, namely of the Formal Generative Description (of the grammar of Czech), see, e.g., [2]. The basic task of such a system is to exactly describe the so-called synonymy-homonymy relation (characteristic relation). This is a relation between the sentences of a natural language (e.g. Czech) and their encoded meanings. How to create such a relation by the method of analysis by reduction is described in [2]. But in [2] the important phenomenon of coordinations in (Czech) sentences is not fully considered. We believe that PCGS and the corresponding models of restarting translators are very promising tools for the description of the synonymy (and homonymy) of coordinations, one of the most recent tasks connected with the Formal Generative Description.

References

1. J. Hromkovič, J. Kari, L. Kari, and D. Pardubská. Two lower bounds on distributive generation of languages. In *Proc. 19th International Symposium on Mathematical Foundations of Computer Science 1994*, *LNCS* vol. 841, Springer-Verlag, London, 423–432.
2. M. Lopatková, M. Plátek, and P. Sgall. Towards a formal model for functional generative description: Analysis by reduction and restarting automata. *The Prague Bulletin of Mathematical Linguistics* 87 (2007) 7–26.
3. V. Kuboň, M. Lopatková, M. Plátek, and P. Pognan. Segmentation of Complex Sentence. In:Lecture Notes In Computer Science 4188, 2006, 151-158.
4. F. Otto. Restarting automata. In: Z. Ésik, C. Martin-Vide, and V. Mitrana (eds.), *Recent Advances in Formal Languages and Applications*, Studies in Computational Intelligence, Vol. 25, Springer, Berlin, 2006, 269–303.
5. F. Otto and M. Plátek. A two-dimensional taxonomy of proper languages of lexicalized FRR-automata. *Pre-proc. LATA 2008*, S.Z. Fazekas, C. Martin-Vide, and C. Tirnăucă (eds.), Tarragona 2008, 419 – 430.
6. D. Pardubská. Communication complexity hierarchy of parallel communicating grammar system. In: *Developments in Theoretical Computer Science*. Yverdon: Gordon and Breach Science Publishers, 1994. - 115–122. - ISBN 2-88124-961-2.
7. D. Pardubská. Communication complexity hierarchy of parallel communicating grammar system. TR University Paderborn, 93 – 133.
8. D. Pardubská and M. Plátek. Parallel Communicating Grammar Systems and Analysis by Reduction by Restarting Automata. Submitted to Second International Workshop on NON-CLASSICAL FORMAL LANGUAGES IN LINGUISTICS 2008.
9. Dana Pardubská, Martin Plátek, and Friedrich Otto. On PCGS and FRR-automata. Accepted for ITAT 2008.
10. Gh. Păun and L. Santean. Parallel communicating grammar systems: the regular case. *Ann. Univ. Buc. Ser. Mat.-Inform.* 37 vol.2 (1989) 55–63.