

On PCGS and FRR-automata

Dana Pardubská^{*1}, Martin Plátek^{**2}, and Friedrich Otto³

¹ Dept. of Computer Science, Comenius University, Bratislava
pardubska@dcs.fmph.uniba.sk

² Dept. of Computer Science, Charles University, Prague
Martin.Plátek@mff.cuni.cz

³ Fachbereich Elektrotechnik/Informatik, Universität Kassel, Kassel
otto@theory.informatik.uni-kassel.de

Abstract. *This paper presents the second part of the technical report [7] in which the study of the relation between Parallel Communicating Grammar Systems (PCGS) and Freely Rewriting Restarting Automata (FRR) has been initiated. The first part of [7] is presented in [6]. Here, the distribution and generation complexity for PCGS are introduced and studied. It is shown that analysis by reduction for PCGS with distribution complexity bounded by a constant k and generation complexity bounded by some other constant j can be implemented by strongly linearized deterministic FRR-automata with k rewrites per cycle. We show infinite hierarchies of classes of languages based on the parameters k, j and on the notion of skeleton.*

1 Introduction

This paper deals with the comparison of Freely Rewriting Restarting Automata (FRR, [4]) and Parallel Communicating Grammar Systems (PCGS, [1,8]). Namely, the so-called linearized FRR-automaton is used 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*. Analysis by reduction in general 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 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 in an accepting computation the content of the tape is a word from the characteristic language. In fact, we mainly consider *strongly lexicalized* restarting automata. This additional restriction requires that all rewrite operations are deletions.

Parallel Communicating Grammar Systems are able to handle creations of copies of generated strings and their regular mappings in a natural way. This ability strongly resembles the generation of coordinations in Czech (and some other natural languages) sentences, where coordinations are certain contiguous segments (not only lexicalized elements). However, the synonymy of coordinations has not yet been modelled appropriately.

In this paper the notions of distribution and generation complexity for PCGS are introduced and studied. It is shown that analysis by reduction for PCGS with distribution complexity bounded by a constant k and generation complexity bounded by some other constant j can be implemented by strongly linearized deterministic FRR-automata with k rewrites per cycle. We show infinite hierarchies of classes of languages based on the parameters k, j and on the notion of *skeleton*. The notion of skeleton is introduced in order to model the principle of so-called segments in (Czech) sentences (or in text). The elements of skeletons are so-called *islands*, which serve 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 $M = (Q, \Sigma, \Gamma, \epsilon, \$, q_0, k, \delta)$. 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 $\epsilon, \$ \notin \Gamma$

* 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.

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 \mathfrak{c} , 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 $\beta \in \{\mathfrak{c}\} \cdot \Gamma^* \cdot \{\mathfrak{s}\}$ or $\alpha \in \{\mathfrak{c}\} \cdot \Gamma^*$ and $\beta \in \Gamma^* \cdot \{\mathfrak{s}\}$; 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 $q_0 \mathfrak{c} w \mathfrak{s}$, 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 $u \vdash_M^{\mathfrak{c}} v$ to denote a cycle of M that begins with the restarting configuration $q_0 \mathfrak{c} u \mathfrak{s}$ and ends with the restarting configuration $q_0 \mathfrak{c} v \mathfrak{s}$; the relation $\vdash_M^{\mathfrak{c}}$ is the reflexive and transitive closure of $\vdash_M^{\mathfrak{c}}$.

A word $w \in \Gamma^*$ is *accepted* by M , if there is a computation which starts from the restarting configuration $q_0 \mathfrak{c} w \mathfrak{s}$, and ends with an application of an accept step. By $L_C(M)$ 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 $v := \text{Pr}^\Sigma(w)$, then v is the Σ -*projection* of w , and w is an *expanded version* of v . For a language $L \subseteq \Gamma^*$, $\text{Pr}^\Sigma(L) := \{\text{Pr}^\Sigma(w) \mid w \in L\}$.

In recent papers restarting automata were mainly used as acceptors. The (*input*) *language* accepted by a restarting automaton M is the set $L(M) := L_C(M) \cap \Sigma^*$. 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 $L_P(M) := \text{Pr}^\Sigma(L_C(M))$. Hence, a word $v \in \Sigma^*$ belongs to $L_P(M)$ if and only if there exists an expanded version u of v such that $u \in L_C(M)$.

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

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

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

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

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

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 [4].

Definition 1. Let $M = (Q, \Sigma, \Gamma, \mathfrak{c}, \mathfrak{s}, q_0, k, \delta)$ be an FRR-automaton, $|x|_K$ denotes the number of occurrences of symbols from K in word x .

- The FRR-automaton M is called *linearized* if there exists a constant $j \in \mathbb{N}_+$ such that $|w|_{\Gamma \setminus \Sigma} \leq j \cdot |w|_\Sigma + j$ for each $w \in L_C(M)$.
- M is called *strongly linearized* if it is linearized, and if each of its rewrite operations just deletes some symbols.

Since linearized FRR automata use linear space only, we have the following:

Corollary 1. If M is a linearized FRR-automaton, then the proper language $L_P(M)$ 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 by $t\text{-}\mathcal{A}$ the subclass of \mathcal{A} -automata which 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 $\Pi = (G_1, \dots, G_m, K)$, where for all $i \in \{1, \dots, m\}$, $G_i = (N_i, T, S_i, P_i)$, so-called component grammars, are regular grammars satisfying $N_i \cap T = \emptyset$ and $K \subseteq \{Q_1, \dots, Q_m\} \cap \bigcup_{i=1}^m N_i$ is a set of special symbols, called *communication symbols*.

A *configuration* is an m -tuple $C = (x_1, \dots, x_m)$, $x_i = \alpha_i A_i$, $\alpha_i \in T^*$, $A_i \in (N_i \cup \varepsilon)$; we call x_i the i -th *component of the configuration* (resp. component). The *nonterminal cut* of configuration C is the m -tuple $N(C) = (A_1, A_2, \dots, A_m)$. If the nonterminal cut $N(C)$ contains at least one communication symbol, it is denoted $NC(C)$ and called an NC-cut.

We say that a *configuration* $X = (x_1, \dots, x_m)$ *directly derives a configuration* $Y = (y_1, \dots, y_m)$, and write $X \Rightarrow Y$, 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

$$\begin{aligned} x_i &\Rightarrow^{G_i} y_i \text{ for } x_i \in T^* N_i \text{ and} \\ y_i &= x_i \text{ for } x_i \in T^+. \end{aligned}$$
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 is true:
 - (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 has not occurred in any of the x_i 's, we put $y_t = x_t$.

Now, we describe the derivations in PCGSs. A *derivation* of a PCGS Π is a sequence of configurations $D = C_1, C_2, \dots, C_t$, 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* and *communication steps*, too.

If no communication symbol appears in any of the component grammars, 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 component grammars contains a nonterminal that cannot be rewritten, the derivation is blocked. If the first grammar G_1 contains 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 a set of the terminal words generated by G_1 (in cooperation with the other grammars):

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

Let $D = D(w) = C_0, C_1, \dots, C_t$ 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 $T(D) = N(C_0)N(C_1) \dots N(C_t)$ of the nonterminal cuts of the individual configurations of D .

The *NC-sequence* is defined analogously; $NCS(D)$ is the sequence of the NC-cuts of the configurations 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 $N(C)$, $N(C_1)$, \dots , $N(C_j)$, $N(C)$ of nonterminal cuts of the derivation⁴ in which the first and the last cuts ($N(C)$) are the same. 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 each 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

⁴ More precisely it is a subsequence of trace of the derivation.

$D'(w)$.

A *generative section* is a non-empty sequence of generative steps between two consecutive communication steps in $D(w)$ ⁵, 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 the following we consider only **PCGS without communication cycles**, i.e., $DG(D(w))$ is bounded by a constant depending only on Π .

$\mathbf{g}(i, j)$ ($\mathbf{g}(i, j, \mathbf{D}(w))$) 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 $D(w)$);

$\mathbf{n}(i, j)$ ($\mathbf{n}(i, j, \mathbf{D}(w))$) denotes the number of occurrences of $g(i, j)$ in w ;

$\mathbf{g}(i, j, l)$ denotes the l -th occurrence of $g(i, j)$ in w , we call it the (i, j, l) -(generative) factor.

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

$\mathbf{N}(j, \mathbf{D}(w)) = \sum_i n(i, j, D(w))$, 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 word \rightsquigarrow distribution complexity of a language L (denoted $DD(L)$) as a function of the length of the word $\rightsquigarrow f(n) - DD$ as class of languages whose distribution complexity is bounded by $f(n)$.

The generation complexity is introduced analogously. Here, we are mainly interested in the classes of languages with t-DD and/or with j-DG for some natural numbers j, t . We denote by j-t-DDG the class of languages such that, to any language L of this class, there is a PCGS Π such that $L(\Pi) = L$, and $DD(L(\Pi)) = t$, $DG(L(\Pi)) = j$.

⁵ Note that if some communication cut contains more than one communication symbol, then there might be no generative step between two communication steps.

Relevant observations about derivations of PCGS (see [5] for more information) are summarized in the following facts:

Fact 1 *Let Π be a PCGS without a communication cycle. Then there are constant $d(\Pi), \ell(\Pi), s(\Pi)$ 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 $s(\Pi)$.*

Fact 2 *Let Π be a PCGS without a communication cycle, $D(w)$ a reduced derivation of a terminal word w . Then there is a constant $e(\Pi)$ such that, if more than $e(\Pi)$ generative steps of one generative section are performed, then at least one $g(i, j, D(w))$ is changed (see Example 1 in [7]).*

3 Bounded Degree of Distribution

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

In fact, the result follows from the analysis of the proof of Theorem 1 ([7]). For better understanding and to motivate the notions defined below we sketch the mentioned proof (from [7]).

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 Π -description of w .

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

$(A_1, \dots, A_m), (\alpha_{1,1} A_{1,1}, \dots, \alpha_{1,m} A_{1,m}), \dots$

$(\alpha_{1,1} \alpha_{2,1} \dots \alpha_{s,1} A_{s,1}, \dots, \alpha_{1,m} \alpha_{2,m} \dots \alpha_{s,m} A_{s,m})$ 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 $g(i, j, l)$:

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

Such a 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 $w, D(w), ex-g(i, j, l)$, 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 :

$$\Pi D(D(w)) = NCS(D(w))CS(D(w))ex-w.$$

Observations. Let $\Pi D(D(w))$ be the 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) From $\Pi D(D(w))$ the terminal word w is easily obtained 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 .

Analyzing the proof of Theorem 1 from [7] we have the following consequence. 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 [7].

Corollary 2. For all $k \in \mathbb{N}$, k -DD $\subseteq \mathcal{L}_P(k$ -SLnRR).

For $t \in \mathbb{N}_+$, separation of PCGSs of 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 the language

$$L_t := \{c_1wd \dots c_twd \mid w \in \{a, b\}^*\},$$

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

Proposition 1. For all $t \in \mathbb{N}_+$,

$$L_t \in \mathcal{L}(t$$
-DD) $\setminus \mathcal{L}_P((t - 1)$ -NLnRR).

Proof. It is not hard to show that $L_t \in \mathcal{L}(t$ -DD). We use a PCGS with $t + 1$ component grammars for that: $(S_1, S_2, \dots, S_{t+1}) \Rightarrow^*$

$$\Rightarrow^* (c_1Q_2, c_2Q_3, \dots, c_tQ_{t+1}, wd)$$

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

For the lower-bound part we use a similar technique as in [4].

Let $M = (Q, \Sigma, \Gamma, \mathfrak{c}, \$, q_0, k, \delta)$ be a nondeterministic linearized FRR-automaton that executes at most $t - 1$ rewrites per cycle, where $\Sigma := \Sigma_0 \cup \Sigma_1$. Assume that $L_P(M) = L_t$ holds. Consider the word $w := c_1a^n b^n d \dots c_t a^n b^n d \in L_t$, where n is a large integer. Then there exists an expanded version $W \in \Gamma^*$ of w such that $W \in L_C(M)$. Let W be a shortest expanded version of w in $L_C(M)$. 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 $w_1 := \text{Pr}^\Sigma(W_1) \in L_t$. As $|W_1| < |W|$, we see from our choice of W that $w_1 \neq w$, that is, $w_1 = c_1x_1d \dots c_t x_1d$ 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 $a^n b^n$ into the same word x_1 . It follows that $w_1 \notin L_t$, implying that $L_t \notin \mathcal{L}_P((t - 1)$ -NLnRR). \square

As $\mathcal{L}(t$ -DD) $\subseteq \mathcal{L}_P(t$ -SLnRR), we obtain the following hierarchies from Proposition 1, where $\mathcal{L}_P(t$ -DD) just denotes the class $\mathcal{L}(t$ -DD).

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

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

4 Skeletons

In this part we define the notions of *skeleton* and *islands* whose introduction has been 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 serve for modelling the independence of segments. A technical example how to construct skeletons is given by the construction in the proof of [7] Theorem 1. In fact, skeletons are only defined for t -SLnRR-automata that fulfill certain additional requirements.

Definition 2. Let $M = (Q, \Sigma, \Gamma, \mathfrak{c}, \$, q_0, k, \delta)$ 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 $\Gamma' = \Gamma \cup \{\mathfrak{c}, \$\}$. For each $j \in \{1, \dots, s\}$, let $SK(s, j) = \{c_{1,j}, \dots, c_{t,j}\}$ 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 $\mathcal{L}_P(t\text{-SK}(s))$ (resp. by $\mathcal{L}_P(t\text{-LSK}(s))$) we denote the class of proper languages of $t\text{-SLnRR}$ -automata with s -skeletons (resp. with left s -skeletons). The corresponding classes of characteristic languages are denoted by $\mathcal{L}_C(t\text{-SK}(s))$ (resp. by $\mathcal{L}_C(t\text{-LSK}(s))$).

Observe that the symbols of the form $[b, i, s, l]$ in the construction of an $s\text{-SLnRR}$ -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 $s\text{-t-DDG}$ denotes the class of PCGSs that have simultaneously generation degree s and distribution degree t .

Corollary 3.

For all $s, t \in \mathbb{N}_+$, $\mathcal{L}(s\text{-t-DDG}) \subseteq \mathcal{L}_P(t\text{-LSK}(s))$.

To separate PCGSs of generation complexity t and distribution complexity s from the class of proper languages of $(t-1)\text{-LSK}(s)$ -automata we define language $L_{(t,s)}$. This 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 $\Sigma_1, \dots, \Sigma_s, \Delta$ are new alphabets with empty intersection with $\{a, b\}$. Then,

$$L_{(t,s)} := (L_{(t)})^s.$$

Proposition 2. For all $s, t \in \mathbb{N}_+$,

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

Sketch of the proof. Note that $L_t = L_{(t)} = L_{(t,1)}$ when $|\Sigma_1| = \dots = |\Sigma_t| = |\Delta| = 1$.

(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}, \dots, G_{s+t} plays the role of G_2, \dots, G_{t+1} from the proof of Proposition 1, while the component grammars G_1, \dots, G_s play the role of grammar G_1 from that proof. At the end of the p -th generative section, there is a word ω_p present in component grammar G_{s+1} , where $\omega_p = c_{1,p} w_p d_p \dots c_{t,p} w_p d_p$ 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 appropriate⁶ way in component grammar G_1 .

(b) Assume that M is a $t\text{-SK}(s-1)$ -automaton such that $L_P(M) = L_{(t,s)}$. 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 $i = 1, \dots, s$, $w_i \in L_{t,i}$, 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 $j \in \{1, \dots, s-1\}$ 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 1 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 $W \in L_C(M)$ 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 $t\text{-SK}(s-1)$ -automaton.

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

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

⁶ The construction of PCGS heavily utilizes nondeterminism. In case of “wrong” nondeterministic choices the derivation is blocked.

Proposition 3. $L_{pe} \in \mathcal{L}_P(2\text{-SK}(1))$.

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

Proposition 4. $\forall s, t \in \mathbb{N}_+ : L_{pe} \notin \mathcal{L}_P(t\text{-LSK}(s))$.

Proof. Assume that M is a $t\text{-LSK}(s)$ -automaton such that $L_P(M) = L_{pe}$, that is, M has a left skeleton $SK(s) = \{c_{i,j} \mid 1 \leq i \leq t, 1 \leq j \leq s\}$. Let $w = (a^n b^n)^m$, where $n, m \in \mathbb{N}_+$ are sufficiently large, and let $z = w c w^R \in L_{pe}$. Then there exists a (shortest) expanded version $Z \in \Gamma^+$ of z such that $Z \in L_C(M)$. 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 $V \in L_C(M)$ and $|V| < |Z|$. Thus, $v = \text{Pr}^{\Sigma}(V) \in L_{pe}$, 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 $j \in \{1, \dots, s\}$. 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, $L_{pe} \notin \mathcal{L}_P(t\text{-LSK}(s))$. \square

The results above yield the following consequences.

Theorem 2. For all $X \in \{\text{LSK}, \text{SK}\}$, and all $s, t \geq 1$, we have the following proper inclusions:

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

5 Conclusion

The study of the relation between PCGS and FRR was motivated by computational linguistics; both models seem to be useful in this field. While in [6] the basic relation between the computational power of these two models was established, the aim of this paper was to introduce and study the relevant complexity measures of PCGS and restrictions on computation of FRR.

We have succeeded in showing infinite hierarchies both for PCGSs and FRRs. The question of whether $j\text{-k-DDG}$ is equal to $\mathcal{L}_P(j\text{-LSK}(k))$ or not remains open.

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

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

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 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. Tirnăuță (eds.), Tarragona 2008, 419 – 430.
5. 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.
6. D. Pardubská and M. Plátek. Parallel Communicating Grammar Systems and Analysis by Reduction by Restarting Automata. Submitted to ForLing 2008.
7. Dana Pardubská, Martin Plátek, and Friedrich Otto. On the Correspondence between Parallel Communicating Grammar Systems and Restarting Automata. Technical Reports in Informatics, TR-2008-015, Comenius University, Bratislava (<http://kedrigern.dcs.fmph.uniba.sk/reports/>)
8. Gh. Păun and L. Santean. Parallel communicating grammar systems: the regular case. *Ann. Univ. Buc. Ser. Mat.-Inform.* 37 vol.2 (1989) 55–63.