On Parallel Communicating Grammar Systems and Correctness Preserving Restarting Automata

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

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

Abstract. This paper contributes to the study of Freely Rewriting Restarting Automata (FRR-automata) and Parallel Communicating Grammar Systems (PCGS) as formalizations of the linguistic method of *analysis by reduction*. For PCGS we study two complexity measures called *generation complexity* and *distribution complexity*, and we prove that a PCGS Π, for which both these complexity measures are bounded by constants, can be simulated by a freely rewriting restarting automaton of a very restricted form. From this characterization it follows that the language $L(\Pi)$ is semi-linear, that its characteristic analysis is of polynomial size, and that this analysis can be computed in polynomial time.

1 Introduction

This paper contributes to the analysis of Freely Rewriting Restarting Automata (FRR-automata) and Parallel Communicating Grammar Systems (PCGS), see [1,2]. Here the main goal is the quest for constraints for FRRs and PCGSs, under which the corresponding classes of languages and their *analysis by reduction* are of interest from the point of view of computational linguistics. For example, this is the case if the languages obtained are semi-linear, and if their so-called *characteristic analysis* can be computed in polynomial time.

Freely rewriting restarting automata create a suitable tool for modelling the so-called *analysis by reduction*. In general, analysis by reduction explains basic types of so-called dependencies in sentences of natural languages. The Functional Generative Description for the Czech language developed in Prague (see, e.g., [3]) is based on this method.

In order to model analysis by reduction, 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

* Partially supported by the Slovak Grant Agency for Science (VEGA) under contract "1/0726/09 - Algorithmic and complexity issues in information processing".
** 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.

its sentences. We focus on restarting automata that ensure the *correctness preserving property* for the analysis, that is, after any restart within a computation starting with a word from the characteristic language, the content of the tape is again from that language. This property is required in order to introduce the notion of *characteristic analysis* in a linguistically adequate way. To achieve our goal we use a technique that is based on the notion of *skeletal set*, which is particularly useful for error recovery during a robust parsing or during a grammar-checking procedure.

We study two complexity measures for returning PCGSs with regular components: the *generation complexity*[4], which bounds the number of generative sections in a word generated by a PCGS, and the *distribution complexity*[5], which bounds the distribution of concurrently generated segments over the word generated. Our technical main result states the following. If Π is a PCGS, for which the generation complexity is bounded by a constant g and the distribution complexity is bounded by a constant d, then the language $L(\Pi)$ generated by Π is the proper language of a freely rewriting restarting automaton M of a very restricted form: M is correctness preserving, and it only performs rewrite operations of a very restricted type. In addition, the number of rewrites per cycle and the number of auxiliary symbols that occur in any word from the characteristic language of M are both bounded by constants that depend on the bounds g and d above. In fact, M even has a skeletal set of type (g, d). Based on these restrictions of M we obtain the following important results on the language $L(\Pi)$: it is semi-linear, its characteristic analysis is of the polynomial size, and it can be computed in polynomial time, where the degree of the polynomial time-bound also depends on the constants g and d. The latter two results answer questions that were left open in [1,2].

The structure of the paper is as follows. In Section 2 we give the (informal) definitions of FRR-automata, AuxRR-automata, and PCGS and present some basic facts about them. In Section 3, which constitutes the technical main part of the paper, we present our simulation result described above, and we then introduce the notion of skeletal set. Using this notion we derive the main results of the paper from the simulation given in the first part of this section. This section ends with some concluding remarks.

2 Basic notions

Restarting automata. A *freely rewriting restarting automaton* (FRR-automaton, for short) is a restarting automaton without rewriting constraints, that is, it is a nondeterministic machine with a flexible tape, a read/write window of a fixed size $k \geq 1$ that can move along this tape, and a finite-state control. Formally, it is described by an 8-tuple $M = (Q, \Sigma, \Gamma, ¢, \$, q_0, k, \delta)$. 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

[4] The generation complexity corresponds to the degree of (linguistic) independence.
[5] The distribution complexity models the degree of (linguistic) dependence (valence).

called *auxiliary symbols*. The additional symbols $\mathbb{c}, \$ \notin \Gamma$ 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 a finite set of transition steps to each pair of the form (q, u), where q is a state and u is a possible content 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 word v satisfying $|v| \leq |u|$, 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 $\mathbb{c}$, 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 $\beta \in \{\mathbb{c}\} \cdot \Gamma^* \cdot \{\$\}$ or $\alpha \in \{\mathbb{c}\} \cdot \Gamma^*$ 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 $q_0 \mathbb{c} w \$$, 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 that is strictly length-decreasing*. Thus, each cycle strictly 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 $q_0 \mathbb{c} u \$$ and ends with the restarting configuration $q_0 \mathbb{c} v \$$; 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 $q_0 \mathbb{c} w \$$, and ends with an application of an accept step. By $L_{\mathrm{C}}(M)$ we denote the so-called *characteristic language of M*, which is the language consisting of all words accepted by 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.

Motivated by linguistic considerations to model the analysis by reduction with parallel processing, we are 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 $u \in L_{\mathrm{C}}(M)$. Realize that the main difference between the input and the proper language lies in the way in which auxiliary symbols are inserted into the (terminal) words of the language.

An FRR-automaton M is called *linearized* if there exists a constant j such that $|w|_{\Gamma \setminus \Sigma} \leq j \cdot |w|_\Sigma + j$ for each $w \in L_C(M)$ [1,2]. Since a linearized FRR-automaton only uses linear space, we see immediately that the proper language of each linearized FRR-automaton is context-sensitive.

In a real process of analysis by reduction of a sentence of a natural language it is desired that whatever is done within the process does not change the correctness of the sentence. For restarting automata this property can be formalized as follows: An FRR-automaton M is *correctness preserving* if $u \in L_C(M)$ and $u \vdash_M^{c^*} v$ imply that $v \in L_C(M)$, too. While it is easily seen that each *deterministic* FRR-automaton is correctness preserving, there are FRR-automata which are not correctness preserving.

Let $M = (Q, \Sigma, \Gamma, \mathbb{c}, \$, q_0, k, \delta)$ be an FRR-automaton that is correctness preserving, and let $w \in \Sigma^*$. Then $A_C(w, M) = \{\, w_C \in L_C(M) \mid \ w_C$ is an extended version of $w \,\}$ is called the *characteristic analysis of w by M*.

Definition 1. *An* FRR*-automaton $M = (Q, \Sigma, \Gamma, \mathbb{c}, \$, q_0, k, \delta)$ is called* aux-rewriting *if, for each of its rewrite operations $(q', v) \in \delta(q, u)$, $\mathsf{Pr}^\Sigma(v)$ is obtained from $\mathsf{Pr}^\Sigma(u)$ by deleting some symbols, and $\mathsf{Pr}^{\Gamma \setminus \Sigma}(v)$ is obtained from $\mathsf{Pr}^{\Gamma \setminus \Sigma}(u)$ by replacing some symbol by another symbol.*

By AuxRR we denote the class of aux-rewriting FRR-automata that are correctness preserving. For each type X of restarting automata and each $t \in \mathbb{N}_+$, we use t-X to denote the class of X-automata that execute at most t rewrite steps in any cycle.

Parallel Communicating Grammar Systems. A returning PCGS *of degree m (≥ 1)* with regular components 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 $C = (x_1, \ldots, x_m)$, where $x_i = \alpha_i A_i$, $\alpha_i \in T^*$, and $A_i \in (N_i \cup \{\varepsilon\})$; we call x_i the *i-th component* of the configuration. 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 called an *NC-cut* and denoted by $NC(C)$.

A *derivation* of Π is a sequence of configurations $D = C_1, C_2, \ldots, C_t$, where C_{i+1} is obtained from C_i by one generative step or one communication *step*. If no communication symbol appears in any of the components, then we perform a *generative step*. It consists of synchronously performing a rewrite step 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, and 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, then w is the word that is generated by Π in this derivation. In this situation D is usually denoted as $D(w)$. 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 themselves 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. Communication steps are performed until no more communication symbols are present, or until the derivation is blocked because no communication symbol can be replaced. The maximal sub-sequence of communication steps forms a *communication section*.

A *generative section* is a non-empty sequence of generative steps between two consecutive communication sequences in D (or before the first or after the last communication step). Thus, the communication steps divide the derivation into generative and communication sections.

The *(terminal) language* $L(\Pi)$ generated by Π is the set of terminal words that appear in the component G_1, which is called the *master* of the system:

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

Several notions are associated with the derivation $D(w)$:

- $g(i,j)$ (or $g(i,j,D(w))$) denotes the (i,j)-*(generative) factor* of $D(w)$, which is the terminal word that is generated by G_i within the j-th generative section of $D(w)$;
- $n(i,j)$ (or $n(i,j,D(w))$) denotes the number of occurrences of $g(i,j)$ in w.
- The *communication structure* $CS(D(w))$ of $D(w)$ captures the connection between the terminal word w and its particular derivation $D(w)$):
  $CS(D(w)) = (i_1,j_1),(i_2,j_2),\ldots,(i_r,j_r)$, if $w = g(i_1,j_1)g(i_2,j_2)\ldots g(i_r,j_r)$.
- For $j \geq 1$ let $N(j,D(w)) = \sum_{i=1}^{m} n(i,j,D(w))$. Then, the so-called *degree of distribution* $DD(D(w))$ of $D(w)$ is the maximum over all $N(j,D(w))$.
- The *trace* of a (sub-)derivation D is the sequence $T(D)$ of nonterminal cuts of individual configurations of D: $T(D)$=$N(C_0), N(C_1), \ldots, N(C_t)$. Note that (in general) the trace does not unambiguously identify the derivation.
- The *communication sequence*, resp. the *NC-sequence*, is defined analogously: $NCS(D)$ is the sequence of all NC-cuts in the (sub-)derivation D. Realize that the communication sequence $NCS(D(w))$ unambiguously defines the communication structure of $D(w)$. Moreover, the set of words with the same communication sequence/structure might, in general, be infinite.

A *cycle in a derivation* D is a smallest (continuous) sub-derivation $\mathcal{C} = C_1, \ldots, C_j$ of D such that $N(C_1) = N(C_j)$. If *none* of the nonterminal cuts in $\mathcal{C}$ contains a communication symbol, then the whole cycle is contained in a generative section; we speak about a *generative cycle* in this case. If the first nonterminal cut contains communication symbols, which means that $N(C_1) = N(C_j)$ are NC-cuts, then the cycle is called a *communication cycle*.

If there is a cycle in the derivation $D(w)$, then manifold repetition[6] of the cycle is possible and the resulting derivation is again a derivation of some terminal word. Observe, however, that the repetition or deletion of a generative cycle does not change the communication structure of a derivation. We call a derivation $D(w)$ *reduced*, if every repetition of any of its cycles leads to a *longer* terminal word ω; $|w| < |\omega|$. Obviously, to every derivation $D(w)$ there is an equivalent

[6] Deletion of a cycle is also possible.

reduced derivation $D'(w)$ of the same word. In what follows, we consider only derivations that are reduced.

Finally, we define several complexity measures for PCGS. Informally, the *communication complexity* of a derivation D (denoted $com(D)$) is defined as the number of communications performed within the derivation D; analogously, the *distribution complexity* of a derivation D is the degree of distribution defined above, and the *generation complexity* of the derivation D is the number of generative sections in D. Then the communication/distribution/generation complexity of a language and the associated complexity class are defined in the usual way (always considering the corresponding maximum).

Here, we are mainly interested in those classes of languages for which all the complexity measures considered above are bounded from above by a constant. For natural numbers k, d, g, we denote the corresponding communication complexity class by COM(k), and the distribution and/or generation complexity class by d-DG, g-DD, and d-g-DDG, respectively. Some relevant observations characterizing the derivations of a PCGS with constant communication complexity are summarized in the following facts.

Fact 1 *Let Π be a* PCGS *with constant communication complexity. Then there are constants $d(\Pi)$, $\ell(\Pi)$, $s(\Pi)$, and $e(\Pi)$ such that*

1. *the number $n(i,j)$ of occurrences of individual $g(i,j)$'s in a* reduced *derivation is bounded by $d(\Pi)$; that is, $n(i,j) \leq d(\Pi)$;*
2. *the length of the communication structure for every (reduced) derivation is bounded by $\ell(\Pi)$;*
3. *the cardinality of the set of all possible communication structures corresponding to a reduced derivation by Π is bounded by $s(\Pi)$.*
4. *Let $D(w)$ be a* reduced *derivation of a terminal word w in Π. If more than $e(\Pi)$ generative steps are performed in the j-th generative section of $D(w)$, then at least one factor $g(i,j,D(w))$ has been changed.*

Based on pumping arguments the following observation now follows easily.

Proposition 1. *Let Π be a* PCGS *with constant communication complexity. Then the set of all derivations by Π without a generative cycle is finite.*

3 Analysis by Reduction for PCGSs

In [1] it is shown how to transform a PCGS with constant communication complexity into a deterministic linearized FRR-automaton. In what follows we will reduce the number of occurrences of auxiliary symbols in the words from the characteristic language of the corresponding restarting automaton to a constant by utilizing nondeterminism. In fact, the resulting automaton will be an AuxRR-automaton for which the positions at which rewrites are performed within a cycle are severely restricted. These restrictions will be formalized through the notion of a *skeletal set* in Definition 2.

Theorem 1. *For each $L \in g$-d-DDG, there is a d-AuxRR-automaton M such that $L = L_{\mathrm{P}}(M)$. Moreover, the number of auxiliary symbols in $w \in L_{\mathrm{C}}(M)$ is bounded from above by the constant $2 \cdot g \cdot d + 2$.*

Proof. Let $L \in g$-d-DDG, and let Π be a PCGS with m components that generates L with distribution complexity d and generation complexity g. Our construction is based on the fact that Π has only a finite number σ of cycle-free derivations. Let $Cf = \{\hat{D}_1(\hat{w}_1), \cdots, \hat{D}_\sigma(\hat{w}_\sigma)\}$ be the set of these derivations.

We describe a d-AuxRR-automaton M that, given a certain extended version w_C of a word w, performs the analysis by reduction which starts by considering a Π-derivation $D(w)$ of the word $w \in L$, and ends by checking that the Π-derivation $\hat{D}_k(\hat{w}_k)$ obtained is one of the cycle-free derivations listed above.

Let $w \in L$, let $D(w)$ be a derivation of w in Π, and let $g(i,j)$ be the terminal word generated by the component grammar G_i within the j-th generative section. Then w can be written as $w = g(i_1,j_1)g(i_2,j_2)\dots g(i_r,j_r)$. As Π has generation complexity g, there are at most g generative sections in the derivation $D(w)$, and as Π has distribution complexity d, there are at most d occurrences of factors $g(i_t,j_t)$ such that $j_t = j$ for any j. Hence, we have $r \leq d \cdot g$.

To reconstruct the derivation of a factor $g(i,j)$ in detail we utilize the following notion of an *extended j-trace*. Let

$$(A_1,\dots,A_m),(\alpha_{1,1}A_{1,1},\dots,\alpha_{1,m}A_{1,m}),\dots \\ (\alpha_{1,1}\alpha_{2,1}\dots\alpha_{s,1}A_{s,1},\dots,\alpha_{1,m}\alpha_{2,m}\dots\alpha_{s,m}A_{s,m})$$

be the sub-derivation corresponding to the j-th generative section of $D(w)$. It yields the following *extended version* of the trace of the j-th generative section:

$$\begin{pmatrix} A_1 \\ A_2 \\ \cdots \\ A_m \end{pmatrix} \begin{pmatrix} \alpha_{1,1}A_{1,1} \\ \alpha_{1,2}A_{1,2} \\ \cdots \\ \alpha_{1,m}A_{1,m} \end{pmatrix} \begin{pmatrix} \alpha_{2,1}A_{2,1} \\ \alpha_{2,2}A_{2,2} \\ \cdots \\ \alpha_{2,m}A_{2,m} \end{pmatrix} \cdots\cdots \begin{pmatrix} \alpha_{s,1}A_{s,1} \\ \alpha_{s,2}A_{s,2} \\ \cdots \\ \alpha_{s,m}A_{s,m} \end{pmatrix}.$$

This description is denoted $ex\text{-}T(D(w),j)$. It describes the sequence of generative steps of the j-th generative section. Assume that $D(w)$ has g_k generative sections. Then $ex\text{-}T(D(w)) = ex\text{-}T(D(w),1), ex\text{-}T(D(w),2),\dots,ex\text{-}T(D(w),g_k)$ is called the *extended trace* of $D(w)$. Let us note that $ex\text{-}T(D(w))$ can serve as an another representation of $D(w)$.

The restarting automaton M processes the word w_C as follows. In each cycle M first *nondeterministically chooses* an index j of a generative section, and then it consistently removes the *rightmost* generative cycle from each of the factors $g(i,j)$ of w. Simultaneously, it checks the consistency of its guess and makes necessary changes in the delimiters. M repeatedly executes such cycles until a word is obtained that does not contain any generative cycles anymore. From Proposition 1 we see that the set of words of this form is finite.

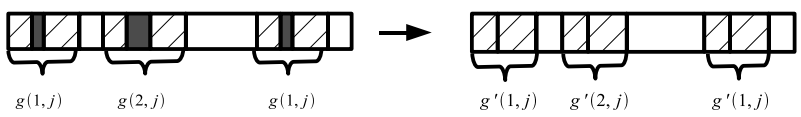

Fig. 1. The situation before and after the execution of a cycle that makes rewrites within the j-th generative section: the reduced parts are grey. Two occurrences of $g(1,j)$ were reduced to $g'(1,j)$; one occurrence of $g(2,j)$ was reduced to $g'(2,j)$.

We show that we only need to store a constant amount of information in the auxiliary symbols to realize this strategy. Accordingly, the word w_C is chosen as

$$w_C := \Delta_{0,k}\Delta_{1,k}\ g(i_1,j_1)\Lambda_{1,k}\Delta_{2,k}\ g(i_2,j_2)\Lambda_{2,k}\dots\Delta_{r,k}\ g(i_r,j_r)\Lambda_{r,k}\Delta_{r+1,k},$$

where $\Delta_{0,k},\dots,\Delta_{r+1,k}$ and $\Lambda_{1,k},\dots,\Lambda_{r,k}$ are auxiliary symbols. These symbols are not only used to separate the individual factors $g(i,j)$ from each other, but also to store relevant information about the derivations $D(w)$ and $\hat{D}_k(\hat{w}_k)$. In fact, the information stored in each symbol $\Delta_{t,k}$ $(0 \le t \le r+1)$ will be *fixed*, while the information stored in each symbol $\Lambda_{t,k}$ $(1 \le t \le r)$ is *temporary*. The information stored in $\Delta_{t,k}$ describes the factor $g(i_t,j_t)$ and the factor $\hat{g}(i_t,j_t)$. The information stored in $\Lambda_{t,k}$ will be changed whenever a deletion is executed in the left neighborhood of the particular delimiter; it describes a suffix of the relevant extended trace $ex\text{-}T(D(w),j_t)$. By $\Lambda_{t,k}(D(w))$ we will denote the particular symbol that corresponds to this information.

▶ If M has decided to try to execute a *cycle*, then it nondeterministically chooses a number $j \in \{1,\dots,g_k\}$ as the index of the generative section of $D(w)$ that it will reduce in this cycle. It stores j and $\Delta_{0,k}$ in its internal state and moves its head to the right until it reaches the first delimiter $\Delta_{t,k}$ for which $j_t = j$ holds, that is, the leftmost occurrence of a factor of the form $g(i,j)$ is found. Then M moves its window further to the right until $\Delta_{t,k}$ becomes the leftmost symbol inside the window. Now M is going to try to simulate a reduction of the factor $g(i_t,j_t)$ as described above.

(1.) From the description of $ex\text{-}T(\hat{D}_k(\hat{w}_k))$ stored in $\Delta_{0,k}$, M determines the nonterminal cut with which the extended j-trace $ex\text{-}T(D(w),j)$ begins.

(2.) Moving from left to right across the factor $g(i_t,j)$, M guesses the extended j-trace $ex\text{-}T(D(w),j)$ in such a way that it is consistent with the word $g(i_t,j)$; if no such guess is possible, the computation is blocked. Simultaneously, M always remembers the current suffix ℓ_t of length $2\cdot p(\Pi)$ of the part of $ex\text{-}T(D(w),j)$ considered so far. Here $p(\Pi)$ is a constant that is sufficiently large to ensure that any sub-word of any $ex\text{-}T(D(w),j)$ of length at least $p(\Pi)$ contains a generative cycle.

(3.) When the delimiter $\Delta_{t+1,k}$ occurs as a rightmost symbol in M's window, then M tries to execute a reduction of the suffix of $g(i_t,j)$; if none is possible, then the computation is blocked. To perform a reduction M checks whether the current suffix ℓ_t of $ex\text{-}T(D(w,j))$ contains a (generative) cycle, that is, whether the suffix $\ell_{t,2}$ of ℓ_t of length $p(\Pi)$ has the following form:

$$\begin{pmatrix} \alpha_{1,1}A_{1,1} \\ \alpha_{1,2}A_{1,2} \\ \cdots \\ \alpha_{1,m}A_{1,m} \end{pmatrix} \cdots \begin{pmatrix} \alpha_{\gamma,1}A_{\gamma,1} \\ \alpha_{\gamma,2}A_{\gamma,2} \\ \cdots \\ \alpha_{\gamma,m}A_{\gamma,m} \end{pmatrix} \cdots \begin{pmatrix} \alpha_{\gamma+\nu,1}A_{\gamma+\nu,1} \\ \alpha_{\gamma+\nu,2}A_{\gamma+\nu,2} \\ \cdots \\ \alpha_{\gamma+\nu,m}A_{\gamma+\nu,m} \end{pmatrix} \cdots \begin{pmatrix} \alpha_{s',1}A_{s',1} \\ \alpha_{s',2}A_{s',2} \\ \cdots \\ \alpha_{s',m}A_{s',m} \end{pmatrix}$$

such that $A_{\gamma,\mu} = A_{\gamma+\nu,\mu}$ for all $\mu = 1, \ldots, m$. If that is the case, and if $\ell_{t,2}$ coincides with the information stored in the symbol $\Lambda_{t,k}(D(w))$, then M removes the factor $\alpha_{\gamma+1,i_t} \cdots \alpha_{\gamma+\nu,i_t}$ from $g(i_t, j)$, it removes the corresponding cycle from ℓ_t, which yields the suffix ℓ'_t of an extended j-trace, and it replaces the information stored in $\Lambda_{t,k}$ by the suffix of ℓ'_t of length $p(\Pi)$. Observe that the factor $\alpha_{\gamma+1,i_t} \cdots \alpha_{\gamma+\nu,i_t}$ may well be empty, implying that this rewrite step simply replaces the symbol $\Lambda_{t,k}$ by a new symbol $\Lambda'_{t,k}$. Further, M stores ℓ_t in its finite control, in order to be able to verify at later occurrences of factors of the form $g(., j)$ that a consistent reduction is performed, and then M moves further to the right.

If no further factor of the form $g(., j)$ is encountered, then M restarts at the right end of the tape. If, however, another factor $g(i_{t'}, j_{t'}) = g(i', j)$ is found, then M tries to reduce this factor in a way consistent with the reduction applied to $g(i_t, j)$. Essentially, M processes the factor $g(i', j)$ in the same way as $g(i_t, j)$. However, on reaching the symbol $\Lambda_{t',k}$ it checks whether the current suffix $\ell_{t'}$ of the extended j-trace simulated coincides with the suffix ℓ_t stored in its finite control. In the affirmative it can then perform the same replacement in $\Lambda_{t',k}$ that it performed on $\Lambda_{t,k}$, and it can reduce the factor $g(i', j)$ in a way consistent with this replacement; otherwise, the computation is blocked.

► In an *accepting tail* M simply checks whether the current content w'_C of the tape belongs to the finite set of "shortest" characteristic words.

From the description above it follows that M is a nondeterministic aux-rewriting FRR-automaton, that the number of auxiliary symbols occurring in any restarting configuration of an accepting computation of M is bounded from above by the constant $2 \cdot g \cdot d + 2$, and that M performs at most d rewrite steps in any cycle of any computation. Further, it is quite clear that $L_{\mathrm{P}}(M) = L$ holds.

Finally, observe that M is in fact *correctness preserving*. For two different factors $g(i_t, j)$ and $g(i_{t'}, j)$ it may guess different extended j-traces, but because of the information stored in $\Lambda_{t,k} = \Lambda_{t',k}$, the suffixes of length $2 \cdot p(\Pi)$ of these traces coincide. Thus, as long as the corresponding suffix of the extended j-trace considered coincides with $\Lambda_{t,k}$, the reduction performed is consistent with a valid derivation $D(w)$. Further, if an inconsistency is discovered by M, then the computation is blocked immediately, that is, no further restart is performed. It follows that $w'_C \in L_{\mathrm{C}}(M)$ if $w_C \in L_{\mathrm{C}}(M)$ and $w_C \vdash^{\mathrm{c}}_M w'_C$ hold, that is, M is indeed correctness preserving. This completes the proof of Theorem 1. □

A detailed example with some additional explanations can be found in the technical report [4].

The AuxRR-automaton M described in the proof of Theorem 1 processes a given input by first choosing a particular derivation without cycles (and its communication structure) from among the finite set of possible derivations without cycles by inserting delimiters. Then, in each cycle a specific generative section j is chosen nondeterministically, and the rightmost generative cycle is removed from each factor $g(i,j)$. In fact, each rewrite operation of each cycle executed by M replaces an auxiliary symbol of the form $\Lambda(D(w))$ by another auxiliary symbol of the form $\Lambda(D'(w'))$, and there is at least one rewrite operation in each cycle that removes a non-empty factor consisting of input symbols[7]. These observations motivate the following definition of a *skeletal set*.

Definition 2. *Let $M = (Q, \Sigma, \Gamma, ¢, \$, q_0, k, \delta)$ be a t-AuxRR-automaton, $r, s \in \mathbb{N}_+$, and let SP be a subalphabet of Γ of cardinality $|SP| \leq s \cdot r \cdot t$. We call SP a* skeletal set of type (r,t)*, if there is an injection $\phi : SP \to \{1,\ldots,s\} \times \{1,\ldots,r\} \times \{1,\ldots,t\}$ such that the properties below are satisfied:*

1. *Elements of SP are neither inserted, nor removed, nor changed during any computation of M; accordingly, we call them* islands.
2. *For all $w \in L_{\mathrm{C}}(M)$ and all $\chi \in SP$, $|w|_\chi \leq 1$, that is, w contains at most one occurrence of χ.*
3. *For $1 \leq i \leq s$, let $SP(i) = \{\, \chi \in SP \mid \phi(\chi) = [i,a,b] \,\}$ be the i-th* skeleton *of SP. For each word $w \in L_{\mathrm{C}}(M)$, there exists a unique index $i \in \{1,\ldots,s\}$ such that $\mathsf{Pr}^{SP}(w) \subseteq SP(i)$ holds. Thus, w only contains islands of a single skeleton.*
4. *Each rewrite operation O of M has the form $xyz\gamma\chi \to xz\gamma'\chi$, where $xyz \in \Sigma^*$, $|y| \geq 0$, $\gamma, \gamma' \in (\Gamma \setminus (\Sigma \cup SP))$ are auxiliary symbols that are not islands, and $\chi \in SP$ is an island. The symbol χ is called the* island visited *by O.*
5. *For $1 \leq i \leq s$ and $1 \leq j \leq r$, let $SP(i,j) = \{\, \chi \in SP \mid \phi(\chi) = [i,j,b] \,\}$, which is the j-th* level *of the i-th skeleton of SP. Within a cycle of a computation of M, the level of a skeleton is kept fixed, that is, if a rewrite operation O is applied in a cycle such that the island visited by O is from $SP(i,j)$, then for* every *rewrite operation executed during this cycle the island visited belongs to $SP(i,j)$.*
6. *There exists a constant $\ell(M)$ such that, for each $w = xyz \in L_{\mathrm{C}}(M)$, where $|y| > \ell(M)$ and y does not contain any island, then starting from the restarting configuration corresponding to w, M will execute at least one cycle before it accepts.*

If SP is a *skeletal set* of type (r,t), then the auxiliary symbols in $\Gamma \setminus SP$ are called *variables* of M. Thus, Γ is partitioned into three disjoint subsets: the set of input symbols Σ, the skeletal set SP, and the set of variables.

Based on the properties of a skeletal set the following result can be derived similarly as in [1].

Corollary 1 (SP semi-linearity). *Let $t \in \mathbb{N}$ be a positive integer, and M be a t-AuxRR-automaton with a skeletal set. Then the languages $L_{\mathrm{C}}(M)$ and $L_{\mathrm{P}}(M)$ are semi-linear, that is, their Parikh images are semi-linear (see [5]).*

[7] All derivations considered are reduced.

Observe that the copy language $L_{copy} = \{ ww \mid w \in \{a,b\}^* \}$ can be generated by a returning PCGS with regular components and constant communication complexity, but that it cannot be generated by any centralized returning PCGS with regular components. Thus, Corollary 1 is not a special case of the corresponding result for the latter class of PCGSs given in [5].

Obviously, with almost no change the delimiters $\Delta_{0,k}$ and $\Delta_{1,k}$ in the proof of Theorem 1 can be shifted just before the delimiter $\Delta_{2,k}$. Thus, the set of delimiters of the form $\Theta_{1,k} = (\Delta_{0,k}, \Delta_{1,k}, \Delta_{2,k})$, and $\Theta_{t,k} = \Delta_{t+1,k}$ for $t > 1$, can serve as a skeletal set for a newly constructed automaton M'. The characteristic word w_C from $L_{\rm C}(M')$ will then be of the following form:

$$w_C := g(i_1, j_1)\Lambda_{1,k}\Theta_{1,k}\ g(i_2, j_2)\Lambda_{2,k}\Theta_{2,k} \ldots\ g(i_r, j_r)\Lambda_{r,k}\Theta_{r,k}.$$

Corollary 2. *For each $L \in g$-d-DDG, there exists a nondeterministic d-AuxRR-automaton M with a skeletal set SP of type (g,d) such that $L = L_{\rm P}(M)$. Each variable of $w \in L_{\rm C}(M)$ is positioned immediately to the left of an element of SP.*

The d-AuxRR-automaton M in Corollary 2 is inherently nondeterministic. To avoid this nondeterminism we now consider a slight generalization of the underlying FRR-automaton: the FRL-automaton. The FRL-automaton is obtained from the FRR-automaton by introducing *move-left* steps. For example, such an automaton can first scan its tape completely from left to right, then move back to the left end of the tape, and then perform some rewrite operations during a second left-to-right sweep. A d-AuxRL-automaton is an FRL-automaton that is aux-rewriting, correctness preserving, and that performs at most d rewrite operations in any cycle.

Obviously, the d-AuxRR-automaton M from Corollary 2 can be seen as a d-AuxRL-automaton of a restricted form. Hence, Theorem 3.4 in [6] applies, which states that there exists a deterministic d-FRL-automaton M_{det} that accepts the same characteristic language as M. In fact, if $w \vdash^c_{M_{det}} w'$, then also $w \vdash^c_M w'$ holds, and if $w \vdash^c_M w'$, then $w \vdash^c_{M_{det}} w''$ for some word w''. Since the transformation from M to M_{det} in the proof of Theorem 3.4 in [6] does not change the existence of a skeletal set, we can restate Corollary 2 as follows.

Corollary 3. *For each $L \in g$-d-DDG, there exists a deterministic d-AuxRL-automaton M with a skeletal set SP of type (g,d) such that $L = L_{\rm P}(M)$. Moreover, in each $w_C \in L_{\rm C}(M)$ each variable is positioned immediately to the left of an element of SP.*

Let M be a deterministic d-FRL-automaton. Given an input w of length n, M will execute at most n cycles, before it either accepts or rejects. Each of these cycles requires a number of steps that is linear in n. It follows that the membership problem for the language $L_{\rm C}(M)$ is decidable in quadratic time.

If M is a deterministic d-AuxRL-automaton with a skeletal set SP of type (g,d), then an input word w of length n belongs to the proper language $L_{\rm P}(M)$, if there exists an extended variant w_C of w that is in the characteristic language $L_{\rm C}(M)$. From the form of the skeletal set we see that w_C is obtained from

w by inserting at most $g \cdot d$ factors of the form $\lambda\delta$, where λ is a variable and δ is an island. Hence, there are $O(|\Gamma \setminus \Sigma|^{2 \cdot g \cdot d} \cdot n^{g \cdot d})$ many candidates for w_C. They can all be enumerated systematically, and then for each of them membership in $L_{\mathrm{C}}(M)$ can be tested in time $O((n + 2 \cdot g \cdot d)^2)$. Thus, we obtain the following.

Proposition 2. *Let M be a deterministic d-*AuxRL*-automaton with a skeletal set SP of the type (g, d) such that each variable in $w_C \in L_C(M)$ is positioned immediately to the left of an element of SP. Then, for each $w \in \Sigma^*$, the size of $A_{\mathrm{C}}(w, M)$, the characteristic analysis of w by M, is at most $O(|\Gamma \setminus \Sigma|^{2 \cdot g \cdot d} \cdot n^{g \cdot d})$, and this set can be computed in time $O(|\Gamma \setminus \Sigma|^{2 \cdot g \cdot d} \cdot n^{g \cdot d} \cdot (n + 2 \cdot g \cdot d)^2)$.*

This proposition together with Corollary 3 has the following consequence.

Corollary 4. *For each language $L \subseteq \Sigma^*$, if $L \in g$-d-*DDG*, then there exists a d-*AuxRR*-automaton M such that $L = L_{\mathrm{P}}(M)$, and for each $w \in \Sigma^*$, the size of the set $A_{\mathrm{C}}(w, M)$ is at most $O(|\Gamma \setminus \Sigma|^{2 \cdot g \cdot d} \cdot n^{g \cdot d})$, and it can be computed in time $O(|\Gamma \setminus \Sigma|^{2 \cdot g \cdot d} \cdot n^{g \cdot d} \cdot (n + 2 \cdot g \cdot d)^2)$.*

Concluding remarks. Similar results as in this paper can be derived for PCGSs with linear-grammar components. On the other hand, that will surely not be the case for PCGSs with context-free components. Let us note that a polynomial upper bound for the (simple) membership problem for returning PCGSs with regular components already follows from [7]. Some results illustrating the generative power of the classes of PCGSs considered here are given already in [2] by the use of a deterministic variant of FRR-automata.

References

1. Pardubská, D., Plátek, M.: Parallel communicating grammar systems and analysis by reduction by restarting automata. In Bel-Enguix, G., Jimenez-Lopez, M., eds.: ForLing 2008, Proc. Research Group on Mathematical Linguistics, Universitat Rovira i Virgili, Tarragona (2008) 81–98
2. Pardubská, D., Plátek, M., Otto, F.: On PCGS and FRR-automata. In Vojtáš, P., ed.: ITAT 2008, Proc. Pavol Jozef Šafárik University, Košice (2008) 41–47
3. Lopatková, D., Plátek, M., Sgall, P.: Towards a formal model for functional generative description - analysis by reduction and restarting automata. The Prague Bulletin of Mathematical Linguistics **87** (2007) 7–26
4. Pardubská, D., Plátek, M., Otto, F.: On parallel communicating grammar systems and correctness preserving restarting automata. Kasseler Informatikschriften, Universität Kassel (2008) https://kobra.bibliothek.uni-kassel.de/bitstream/urn:nbn:de:hebis:34-2008111825149/3/Technicalreport2008_4.pdf.
5. Czuhaj-Varjú, E., Dassow, J., Kelemen, J., Păun, G., eds.: Grammar Systems: A Grammatical Approach to Distribution and Cooperation. Gordon and Breach Science Publishers (1994)
6. Messerschmidt, H., Otto, F.: On determinism versus nondeterminism for restarting automata. Information and Computation **206** (2008) 1204–1218
7. Cai, L.: The computational complexity of PCGS with regular components. In Dassow, J., Rozenberg, G., Salomaa, A., eds.: DLT 1995, Proc. World Scientific, Singapore (1996) 209–219