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), which both are useful models in computational linguistics. For PCGS we study two complexity measures called *generation complexity* and *distribution complexity*, and we prove that a PCGS Π, for which the generation complexity and the distribution complexity are both bounded by constants, can be transformed into 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, see [8]) and Parallel Communicating Grammar Systems (PCGS, see, e.g., [2,3,12]). The motivation for our study is the quest for a formal transformation of PCGSs into a suitable formal model of analysis by reduction. Here the main goal is the identification of constraints for FRRs and PCGSs, under which the corresponding classes of languages can be used to model important phenomena of natural languages. We are mainly interested in formal languages which are semi-linear and which have a possibly simple syntactic analysis. The type of syntactic analysis we focus on is the *characteristic analysis* formally introduced in this paper.

Freely rewriting restarting automata create a suitable tool for modelling the so-called *analysis by reduction*. In general, analysis by reduction explains basic

⋆ 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.

types of so-called dependencies in sentences of natural languages. The Functional Generative Description for the Czech language developed in Prague (see, e.g., [4]) 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 its sentences. By requiring that the automata considered are linearized we restrict the number of auxiliary symbols allowed in each sentence by a function linear in the number of input symbols in that sentence. We mainly 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 a word from the characteristic language, and conversely, after any restart within a computation starting with a word from the complement of the characteristic language, the content of the tape is again from this complement. This property is necessary in order to obtain a type of characteristic analysis which has the properties of manual syntactic analysis of traditional central-European linguistics.

Here we use formalized analysis by reduction to show that Parallel Communicating Grammar Systems inherently define two important types of linguistic phenomena: sentence segments with common *valencies* (*parallel dependencies*) and full independence of sentence segments. 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 PCGSs: the *generation complexity*, which bounds the number of generative sections in a word generated by a PCGS, and the *distribution complexity*, 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 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 even be computed in polynomial time, where the degree of the polynomial time-bound also depends on the constants g and d. This is an essential improvement of the results from [10,11].

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 in Section 4 we

introduce the notion of skeletal set. Using this notion we then derive the main results of the paper from the simulation given in the previous section. Finally, some closing remarks are found in Section 5.

2 Basic notions

Here we informally introduce FRRs and PCGSs, define some necessary notions and abbreviations, and list relevant properties of both models.

2.1 Restarting automata

A *freely rewriting restarting automaton*, abbreviated as FRR-automaton, 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 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 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. However, some restrictions apply in that neither a move-right step nor a rewrite step can shift the read/write window across the right sentinel \$. 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 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 $q_0 ¢ u \$$ 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 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.

In recent papers (see, e.g., [7]) restarting automata were mainly used as acceptors. The main focus was on the so-called (*input*) *language* of a restarting automaton M, that 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 $u \in L_{\mathrm{C}}(M)$. 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. For words from the input language, auxiliary symbols can only be inserted by the automaton itself in the course of a computation, while for words from the proper language, the auxiliary symbols are provided beforehand by an outside source, e.g., a linguist.

Based on the number of auxiliary symbols that are allowed in a word two different classes of FRR automata have been considered in the literature –lexicalized and linearized FRR-automata [8,10,11]. An FRR-automaton M is called *lexicalized* if there exists a constant j such that, for every substring $u \in (\Gamma \setminus \Sigma)^*$ of every word $w \in L_C(M)$, $|u| \leq j$ holds. M is called *linearized* if there exists a constant j such that $|w|_{\Gamma \setminus \Sigma} \leq j \cdot |w|_\Sigma + j$ for each $w \in L_C(M)$. Since a linearized FRR-automaton only uses linear space, the following upper bound on its computational power is obvious.

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

Definition 1. (Correctness Preserving Property.) *An* FRR*-automaton M is* correctness preserving *if $u \in L_{\mathrm{C}}(M)$ and $u \vdash_M^{c^*} v$ imply that $v \in L_{\mathrm{C}}(M)$, too.*

While it is easily seen that each *deterministic* FRR-automaton is correctness preserving, there are FRR-automata which are not correctness preserving.

The characteristic analysis is a formalization of the traditional syntactic analysis of central-European languages. In fact, it is taught manually in middle-schools in a very similar way.

Definition 2. *Let $M = (Q, \Sigma, \Gamma, ¢, \$, q_0, k, \delta)$ be an* FRR*-automaton that is correctness preserving, and let $w \in \Sigma^*$. Then the set*

$$A_C(w, M) := \{\, w_C \in \Gamma^* \mid w_C \in L_C(M) \text{ and } \mathsf{Pr}^{\Sigma}(w_C) = w \,\}$$

is called the characteristic analysis of w by M. *The size of the set $A_C(w, M)$ is denoted as the* characteristic ambiguity of w by M.

Note that the assumption of the correctness preserving property is quite important in the previous definition. It ensures the so-called 'syntactic completeness' of categories used in the characteristic analysis; in our approach we use auxiliary symbols to model these categories.

Definition 3. *An* FRR*-automaton $M = (Q, \Sigma, \Gamma, ¢, \$, q_0, k, \delta)$ 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.*

Below we will mainly be interested in proper languages of aux-rewriting FRR-automata that are correctness preserving; in particular, we denote by AuxRR 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.

2.2 Parallel Communicating Grammar Systems

A PCGS Π *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 $C = (x_1, \ldots, x_m)$, 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 the PCGS 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*. Further, those components that are used to replace communication symbols are reset to their start symbol. Obviously, in one communication step at most $m-1$ communications can be performed. If some communication symbol is 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 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 steps (resp. communication sequences) in D, resp. before the first and/or after the last communication step in D. Thus, the communication steps divide the derivation into generative and communication sections.

The *(terminal) language* $L(\Pi)$ generated by the PCGS Π is the set of terminal words that appear in the component G_1 (this component is also 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 can be associated with the derivation $D(w)$ which help to analyze the derivation of Π and to unambiguously describe w. We start with those describing the structure of w first:

- $g(i,j)$, resp. $g(i,j,D(w))$, denotes the terminal word that is generated by G_i within the j-th generative section of $D(w)$ – we call it the (i,j)-(generative) factor (by $D(w)$);
- $n(i,j)$, resp. $n(i,j,D(w))$, denotes the number of occurrences of $g(i,j)$ in w. Note that $n(i,j) > 1$ is possible. For example, this happens when there are two or more different component grammars that require the content of the same component grammar *at the same time*.

The *communication structure* $CS(D(w))$ of $D(w)$ captures the connection between the terminal word w and its particular derivation $D(w)$) – it determines, how w is composed from the individual $g(i,j)$'s:

$$CS(D(w)) = (i_1,j_1),(i_2,j_2),\ldots,(i_r,j_r), \text{ if } w = g(i_1,j_1)g(i_2,j_2)\ldots g(i_r,j_r).$$

The *set* of tuples of indices of $CS(D(w))$ is denoted $I(D(w))$. Realize that a communication has been involved if $(i,j) \in I(D(w))$ for $i \neq 1$. 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 last couple of notions is mostly connected with the derivations themselves.

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. Recall that any NC-cut contains at least one communication symbol. Realize also 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 $\mathcal{C}$ of D, $\mathcal{C} = C_1, \ldots, C_j$, such that the corresponding first and last nonterminal cuts are identical; $N(C_1) = N(C_j)$. If *none* of the nonterminal cuts involved in $\mathcal{C}$ contains a communication symbol, then, obviously, 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*.

Note that, if there is a cycle in the derivation $D(w)$, then manifold repetition[4] 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 any of its cycles leads to a *longer* terminal word ω; $|w| < |\omega|$. Obviously, to every derivation $D(w)$ there is an equivalent reduced derivation $D'(w)$ of the same word. In what follows, we consider only derivations that are reduced. Observe the following basic fact.

Fact 2 *Repetition/deletion of a generative cycle does not change the communication sequence or the communication structure of a derivation.*

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 the language and the associated complexity class are defined in the usual way (always considering the corresponding maximum). For a function $f : \mathbb{N} \to \mathbb{N}$, the *class of languages* with communication complexity f is denoted COM(f(n)). For the distribution and generation complexities, the corresponding complexity classes are denoted $DD(f)$ and $DG(f)$, respectively.

[4] Deletion of a cycle is also possible.

Here we are mainly interested in the classes of languages for which all the complexity measures considered above are bounded from above by a constant. For natural numbers s, d, g, we denote the corresponding communication complexity class by $\mathsf{COM}(s)$, 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 (see [9] for more information) are summarized in the following facts.

Fact 3 *Let Π be a* PCGS *with constant communication complexity. Then there are constants $d(\Pi), \ell(\Pi), s(\Pi)$ such that*

1. *the number $n(i,j)$ of occurrences of individual $g(i,j)$'s in a* reduced *derivation 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 $s(\Pi)$.*

Fact 4 *Let Π be a* PCGS *with constant communication complexity, and let $D(w)$ be a* reduced *derivation of a terminal word w in Π. Then there is a constant $e(\Pi)$ such that, if more than $e(\Pi)$ generative steps of the j-th generative section are performed in $D(w)$, then at least one factor $g(i,j,D(w))$ has been changed.*

Based on pumping arguments the following observation follows easily.

Proposition 1. *Let Π be a* PCGS *with constant communication complexity. Then the set of all derivations by Π that do not contain any generative cycle is finite.*

3 Analysis by Reduction for PCGSs

As already mentioned in the introduction we are looking for restrictions of FRR-automata and PCGSs that imply that the resulting (proper) languages are semi-linear with polynomial-time computable characteristic analysis (of polynomial size). In the following we will place restrictions on the positions at which rewrites within a cycle can be performed by a restarting automaton. These restrictions will be formalized through the notion of a *skeletal set* in Definition 4. They are motivated by the way in which a PCGS with constant communication complexity is transformed into an AuxRR-automaton (modelling analysis by reduction) in the proof of Theorem 1.

In [10] we have presented a transformation of a PCGS with constant communication complexity into a deterministic linearized FRR-automaton. In that transformation we use auxiliary symbols to merge the description of a derivation of a word w into the word itself. 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 the computations of which have some additional nice properties.

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 of the j-th generative section of the derivation $D(w)$ is denoted by $ex\text{-}T(D(w),j)$. 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 the 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.

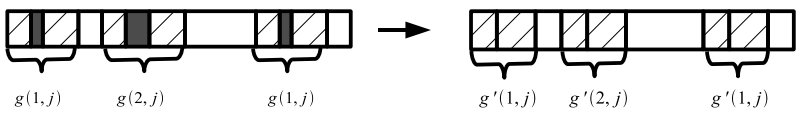

Fig. 1. The situation before and after the execution of a cycle that makes rewrites within the j-th generative section: the reduced parts are grey. Two occurrences of $g(1,j)$ were reduced to $g'(1,j)$; one occurrence of $g(2,j)$ was reduced to $g'(2,j)$.

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 this 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.

The description of $g(i,j)$: Consider an extended trace $ex\text{-}T(D(w),j)$ that corresponds to a generative section j of $D(w)$, and assume that there is a cycle within this trace. The removal of such a cycle from $ex\text{-}T(D(w),j)$ leads to a relevant change in *all* occurrences of factors $g(i,j)$ in w. We can repeat this reduction of the extended trace considered until no more cycles occur in it. The resulting reduced form of $g(i,j)$ is denoted by $\hat{g}(i,j)$, and by our assumption $ex\text{-}T(\hat{D}_k(\hat{w}_k),j)$ is the extended trace of the corresponding cycle-free derivation of $\hat{g}(i,j)$. Further, by our assumption

$$ex\text{-}T(\hat{D}_k(\hat{w}_k)) = ex\text{-}T(\hat{D}_k(\hat{w}_k),1), ex\text{-}T(\hat{D}_k(\hat{w}_k),2),\dots,ex\text{-}T(\hat{D}_k(\hat{w}_k),g_k).$$

Then $ex\text{-}T(\hat{D}_k(\hat{w}_k))$ represents the cycle free derivation $\hat{D}_k(\hat{w}_k)$. Now, in all delimiters $\Delta_{t,k}$ for which $g(i_t,j_t) = g(i,j)$, we store t,k, the pair (i,j), and the factor $\hat{g}(i,j)$. Moreover, the complete $ex\text{-}T(\hat{D}_k(\hat{w}_k))$ will be stored in $\Delta_{0,k}$. Remember that the set of all possible values (i,j), $\hat{g}(i,j)$, and $ex\text{-}T(\hat{D}(\hat{w}))$ is finite, and that its cardinality only depends on Π.

The communication structure: Since Π is a g-d-DDG, the length of the communication structure $CS(w)$ for $D(w)$ is bounded by $d \cdot g$. Thus, we can store the complete communication structure $CS(w)$ together with the extended trace $ex\text{-}T(\hat{D}_k(\hat{w}_k))$ in the first delimiter $\Delta_{0,k}$. We see that all factors $\hat{g}(i,j)$ can simply be obtained from $ex\text{-}T(\hat{D}_k(\hat{w}_k))$. This part of information in the delimiter is fixed. Remember that the set of all communication structures of Π is finite, its cardinality depending on Π only.

The relevant suffix of an extended trace: Based on Fact 4 we know that there is a constant $p(\Pi)$ such that, whenever an extended trace $ex\text{-}T(D(w), j)$ is of length at least $p(\Pi)$, then it contains a (generative) cycle. Accordingly in the symbol $\Lambda_{t,k}$ we store the suffix of length $p(\Pi)$ of the corresponding extended trace $ex\text{-}T(D(w), j_t)$. Observe that this information is the same for all values of s' satisfying $j_{s'} = j_t$, and it is consistent with the factor $g(i_t, j_t)$. When reducing the factors $g(i, j)$, where $j = j_t$, then the right-most generative cycle is removed from $ex\text{-}T(D(w), j)$, which results in an extended trace $ex\text{-}T(D'(w'), j)$ of a shortened derivation, the corresponding factor is removed from $g(i_t, j_t)$, resulting in the word $g'(i_t, j_t)$, and $\Lambda_{t,k}$ is updated accordingly. These reductions must be consistent, that is,

- there is a prefix $pref$ and a suffix suf of $ex\text{-}T(D, j)$ such that $pref \cdot suf = ex\text{-}T(\hat{D}(\hat{w}), j)$,
- a suffix of $\Lambda_{t,k}(D'(w'))$ corresponds to a suffix of $ex\text{-}T(D', j)$, and
- the words $g(i_t, j_t)$ and $g'(i_t, j_t)$ are related to each other in accordance with the factor removed from $\Lambda_{t,k}(D(w))$.

Obviously, the information stored in $\Lambda_{t,k}$ is temporary. However, the set of all possible values of $\Lambda_{t,k}$ is bounded from above by a constant that only depends on Π.

Thus, we can summarize the definition of the auxiliary symbols of M for each value of $k \in \{1, \ldots, \sigma\}$ as follows:

$$\begin{aligned}
\Delta_{t,k} &= [t, k, (i_t, j_t), \hat{g}(i_t, j_t)], \quad \text{for all } 1 \leq t \leq r,\\
\Delta_{0,k} &= [0, k, CS, ex\text{-}T(\hat{D}_k(\hat{w}_k))],\\
\Delta_{r+1,k} &= [r+1, k, \varepsilon, \varepsilon],\\
\Lambda_{t,k}(D(w)) &= [s], \text{ where } s \text{ is the suffix of the extended trace } ex\text{-}T(D(w), j_t)\\
&\qquad \text{satisfying } |s| = p(\Pi).
\end{aligned}$$

Now we can describe the behaviour of M in some more detail.

▶ If M has decided to try to execute a *cycle*, then it nondeterministically chooses a number $j \in \{1, \ldots, g_k\}$ 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.

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. These words are characterized by the following properties:

- the communication structure $CS(w')$ implicitly stored in the delimiters of w'_C coincides with the one stored in $\Delta_{0,k}$;
- there is a Π-derivation $D(w')$ of $w' = Pr^{\Sigma}(w'_C)$ that is consistent with this communication structure, and that does not contain any generative cycles;
- the factors $\hat{g}(i, j)$ stored on the tape are equal to the corresponding factors $g(i, j, D(w'))$.

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. □

We now illustrate the above construction by a detailed example.

Example 1. Let $\Pi_{ab}(2)$ be the following PCGS:

$$\begin{array}{lll} G_1 : S_1 \to aS_1 | aQ_2, & G_2 : S_2 \to bZ_2, & G_3 : S_3 \to Z_3, \\ \quad Z_2 \to aZ_2 | aQ_3, & \quad Z_2 \to bZ_2, & \quad Z_3 \to bZ_3. \\ \quad Z_3 \to b, & & \end{array}$$

It is easily seen that $\Pi_{ab}(2)$ generates the language

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

Consider the following (outline of) a derivation $D(w)$ in $\Pi_{ab}(2)$:

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

From this derivation we obtain the following extended trace:

$$\left|\begin{array}{l} S_1 \\ S_2 \\ S_3 \end{array}\right| \left|\begin{array}{l} aS_1 \\ bZ_2 \\ Z_3 \end{array}\right| \left|\begin{array}{l} aS_1 \\ bZ_2 \\ bZ_3 \end{array}\right| \left|\begin{array}{l} aQ_2 \\ bZ_2 \\ bZ_3 \end{array}\right| \left\|\begin{array}{l} Z_2 \\ S_2 \\ Z_3 \end{array}\right| \left|\begin{array}{l} aZ_2 \\ bZ_2 \\ bZ_3 \end{array}\right| \left|\begin{array}{l} aQ_3 \\ bZ_2 \\ bZ_3 \end{array}\right| \left\|\begin{array}{l} Z_3 \\ Z_2 \\ S_3 \end{array}\right| \left|\begin{array}{l} b \\ bZ_2 \\ Z_3 \end{array}\right|,$$

which yields the following generative factors:

$$\begin{array}{lll} g(1,1) = a^3, & g(1,2) = a^2, & g(1,3) = b, \\ g(2,1) = b^3, & g(2,2) = b^2, & g(2,3) = b, \\ g(3,1) = b^2, & g(3,2) = b^2, & g(3,3) = \varepsilon. \end{array}$$

Thus,

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

the communication structure of $D(w)$ is

$$CS(D(w)) = (1,1)(2,1)(1,2)(3,1)(3,2)(1,3),$$

and the corresponding NC-sequence is

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

The derivation $D(w)$ contains a generative cycle of length 1. By removing this generative cycle we obtain the following derivation $\hat{D}_1(\hat{w})$, which does not contain any (generative) cycle anymore:

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

From this derivation we obtain the following extended trace:

$$\left|\begin{array}{l|ll||l|l|l||l|l} S_1 & aS_1 & aQ_2 & Z_2 & aZ_2 & aQ_3 & Z_3 & b \\ S_2 & bZ_2 & bZ_2 & S_2 & bZ_2 & bZ_2 & bZ_2 & bZ_2 \\ S_3 & Z_3 & bZ_3 & Z_3 & bZ_3 & bZ_3 & S_3 & Z_3 \end{array}\right|,$$

which yields the following generative factors:

$$\begin{array}{lll} g'(1,1) = a^2, & g'(1,2) = a^2, & g'(1,3) = b, \\ g'(2,1) = b^2, & g'(2,2) = b^2, & g'(2,3) = b, \\ g'(3,1) = b, & g'(3,2) = b^2, & g'(3,3) = \varepsilon. \end{array}$$

These, in turn, give the factorization

$$\hat{w} = a^2b^2a^2b^5 = g'(1,1)g'(2,1)g'(1,2)g'(3,1)g'(3,2)g'(1,3).$$

Observe that $CS(\hat{D}_1(\hat{w})) = CS(D(w))$ and $NCS(\hat{D}_1(\hat{w})) = NCS(D(w))$ hold.

Now we describe the representation of the words of $L_{ab}(2)$ through the words of the characteristic language of an AuxRR-automaton M that corresponds to $\Pi_{ab}(2)$. First we describe the auxiliary symbols $\Delta_{0,1}, \Delta_{1,1}, \Delta_{2,1}, ...\Delta_{7,1}$ and $\Lambda_{1,1}, ..., \Lambda_{6,1}$ for the derivations $D(w)$ and $\hat{D}_1(\hat{w})$:

$$\Delta_{0,1} \quad = \quad [0, 1, ((1,1)(2,1)(1,2)(3,1)(3,2)(1,3)), \left|\begin{array}{l|ll||l|l|l||l|l} S_1 & aS_1 & aQ_2 & Z_2 & aZ_2 & aQ_3 & Z_3 & b \\ S_2 & bZ_2 & bZ_2 & S_2 & bZ_2 & bZ_2 & bZ_2 & bZ_2 \\ S_3 & Z_3 & bZ_3 & Z_3 & bZ_3 & bZ_3 & S_3 & Z_3 \end{array}\right|],$$

$$\Delta_{1,1} \quad = [1, 1, (1,1), a^2],$$

$$\Lambda_{1,1}(D) = \left|\begin{array}{llll} S_1 & aS_1 & aS_1 & aQ_2 \\ S_2 & bZ_2 & bZ_2 & bZ_2 \\ S_3 & Z_3 & bZ_3 & bZ_3 \end{array}\right|, \text{ and } \Lambda_{1,1}(\hat{D}_1) = \left|\begin{array}{lll} S_1 & aS_1 & aQ_2 \\ S_2 & bZ_2 & bZ_2 \\ S_3 & Z_3 & bZ_3 \end{array}\right|,$$

$$\begin{aligned}
\Delta_{2,1} &= [2,1,(2,1),b^2],\\
\Lambda_{2,1}(D) &= \begin{vmatrix} S_1 & aS_1 & aS_1 & aQ_2 \\ S_2 & bZ_2 & bZ_2 & bZ_2 \\ S_3 & Z_3 & bZ_3 & bZ_3 \end{vmatrix}, \text{ and } \Lambda_{2,1}(\hat{D}_1) = \begin{vmatrix} S_1 & aS_1 & aQ_2 \\ S_2 & bZ_2 & bZ_2 \\ S_3 & Z_3 & bZ_3 \end{vmatrix},\\
\Delta_{3,1} &= [3,1,(1,2),a^2],\\
\Lambda_{3,1}(D) &= \begin{vmatrix} Z_2 & aZ_2 & aQ_3 \\ S_2 & bZ_2 & bZ_2 \\ Z_3 & bZ_3 & bZ_3 \end{vmatrix} = \Lambda_{3,1}(\hat{D}_1),\\
\Delta_{4,1} &= [4,1,(3,1),b],\\
\Lambda_{4,1}(D) &= \begin{vmatrix} S_1 & aS_1 & aS_1 & aQ_2 \\ S_2 & bZ_2 & bZ_2 & bZ_2 \\ S_3 & Z_3 & bZ_3 & bZ_3 \end{vmatrix}, \text{ and } \Lambda_{4,1}(\hat{D}_1) = \begin{vmatrix} S_1 & aS_1 & aQ_2 \\ S_2 & bZ_2 & bZ_2 \\ S_3 & Z_3 & bZ_3 \end{vmatrix},\\
\Delta_{5,1} &= [5,1,(3,2),b^2],\\
\Lambda_{5,1}(D) &= \begin{vmatrix} Z_2 & aZ_2 & aQ_3 \\ S_2 & bZ_2 & bZ_2 \\ Z_3 & bZ_3 & bZ_3 \end{vmatrix} = \Lambda_{5,1}(\hat{D}_1),\\
\Delta_{6,1} &= [6,1,(1,3),b],\\
\Lambda_{6,1}(D) &= \begin{vmatrix} Z_3 & b \\ Z_2 & bZ_2 \\ S_3 & Z_3 \end{vmatrix} = \Lambda_{6,1}(\hat{D}_1), \text{ and } \Delta_{7,1} = [7,1,\varepsilon,\varepsilon].
\end{aligned}$$

The characteristic word w_C corresponding to w and the characteristic word $\hat{w}_C$ corresponding to $\hat{w}$ are obtained as follows, where $\Lambda_{i,1}^{(1)}$ denotes $\Lambda_{i,1}(D)$ and $\Lambda_{i,1}^{(2)}$ denotes $\Lambda_{i,1}(\hat{D}_1)$:

$$\begin{aligned}
w_C &= \Delta_{0,1}\Delta_{1,1}a^3\Lambda_{1,1}^{(1)}\Delta_{2,1}b^3\Lambda_{2,1}^{(1)}\Delta_{3,1}a^2\Lambda_{3,1}^{(1)}\Delta_{4,1}b^2\Lambda_{4,1}^{(1)}\Delta_{5,1}b^2\Lambda_{5,1}^{(1)}\Delta_{6,1}b\Lambda_{6,1}^{(1)}\Delta_{7,1},\\
\hat{w}_C &= \Delta_{0,1}\Delta_{1,1}a^2\Lambda_{1,1}^{(2)}\Delta_{2,1}b^2\Lambda_{2,1}^{(2)}\Delta_{3,1}a^2\Lambda_{3,1}^{(2)}\Delta_{4,1}b^2\Lambda_{4,1}^{(2)}\Delta_{5,1}b\Lambda_{5,1}^{(2)}\Delta_{6,1}b\Lambda_{6,1}^{(2)}\Delta_{7,1}.
\end{aligned}$$

The construction in the proof of Theorem 1 shows that M reduces the word w_C to the word $\hat{w}_C$ in a single cycle, and that it accepts the latter word in an accepting tail.

4 Skeletal automata

Consider the way in which the AuxRR-automaton M described in the proof of Theorem 1 processes a given input. First, a particular derivation without cycles (and its communication structure) is chosen (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 $g(i,j)$. This is in fact an interesting formal example of the mutual independence of segments in the linguistic sense (in the sense of dependency theory). In fact, each rewrite operation of each cycle executed by M replaces an auxiliary symbol of the form $\Lambda_{t,k}(D(w))$ by another auxiliary symbol of the form $\Lambda_{t,k}(D'(w'))$, and there is at least one rewrite operation in each cycle that removes a non-empty factor consisting of terminals (input symbols). These observations motivate the following definition of a *skeletal set*.

Definition 4. *Let $M = (Q, \Sigma, \Gamma, ¢, \$, q_0, k, \delta)$ be a t-*AuxRR*-automaton, let $r, s \in \mathbb{N}_+$, 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 all the properties listed 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 $w \in L_{\mathrm{C}}(M)$, if $w = xyz$, 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.

Let M be a t-AuxRR-automaton with a skeletal set SP, and let $w \in L_{\mathrm{C}}(M)$. If $w \vdash_M^{\mathrm{c}} w_1$, where $w = x_0y_1x_1y_2 \cdots y_tx_t$, and $w_1 = x_0x_1x_2 \cdots x_t$, then we see from condition (4) above that each rewrite operation can be interpreted as a kind of suffix rewrite on a syllable ending with an island. Further, condition (6) implies that the set of words of $L_{\mathrm{C}}(M)$ that are accepted without a restart

is finite. As suffix rewrites preserve regularity, it follows that all Σ-syllables of words from $L_{\mathrm{C}}(M)$ satisfy some regularity constraints. In particular this implies the following important result.

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.*

Now we reconsider the construction given in the proof of Theorem 1. 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 $L_{\mathrm{C}}(M')$ that corresponds to the word $w = g(i_1, j_1)g(i_2, j_2) \dots g(i_r, j_r)$ will then be of the following form:

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

Finally, the symbols of the form $\Lambda_{t,k}$ are the variables of M'. Thus, we have the following refinement of Theorem 1.

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 $L = L_{\mathrm{P}}(M)$. Each variable of $w \in L_C(M)$ is positioned immediately to the left of an element of SP. Moreover, whenever $w \vdash_M^c w_1$ holds, where $w = x_0y_1x_1y_2 \dots y_tx_t$ and $w_1 = x_0x_1x_2 \dots x_t$, then*

$$x_0y_1^\ell x_1y_2^\ell \cdots y_s^\ell x_s \in L_{\mathrm{C}}(M)$$

for all non-negative integers ℓ.

The d-AuxRR-automaton M in Corollary 2 is inherently nondeterministic, as in each cycle it must guess the value of the parameter j (see the proof of Theorem 1). 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 and 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, [6] Theorem 3.4 applies, which states that there exists a deterministic d-FRL-automaton M_{det} that accepts the same characteristic language as M. In fact, if $w \vdash_{M_{det}}^c w'$, then also $w \vdash_M^c w'$ holds, and if $w \vdash_M^c w'$, then $w \vdash_{M_{det}}^c w''$ for some word w''. Since the transformation from M to M_{det} in the proof of [6] Theorem 3.4 does not interfere with 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 $L = L_{\mathrm{P}}(M)$. Each variable of $w \in L_C(M)$ is positioned immediately to the left of an element of SP.*

Let M be 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 $L_{\mathrm{C}}(M)$ 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 $L_{\mathrm{P}}(M)$, if there exists an extended variant w_C of w that is in the characteristic language $L_{\mathrm{C}}(M)$. 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. There are $\binom{n}{g \cdot d} = O(n^{g \cdot d})$ many different ways to insert these factors, of which there are mostly $|\Gamma \setminus \Sigma|^2$ many different ones. 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 result.

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 $w \in L_C(M)$ 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 $O(|\Gamma \setminus \Sigma|^{2 \cdot g \cdot d} \cdot n^{g \cdot d} \cdot (n + 2 \cdot g \cdot d)^2)$.*

5 Conclusion

Here we have presented a transformation of PCGSs that are restricted by two types of parameters into an adequate model of analysis by reduction with two corresponding parameters. Based on these parameters we have established a polynomial size restriction for the characteristic analysis for this model (which, in fact, means an upper bound for the characteristic ambiguity of this model), and a polynomial upper bound for the time required for the characteristic analysis. That is an interesting result compared to the situation for context-free languages, for which any reasonable type of ambiguity has an exponential upper bound. Let us note that a polynomial upper bound for the (simple) membership problem already follows from [1].

References

1. L. Cai. The computational complexity of PCGS with regular components. In: *Developments in Language Theory*, 1995, 209–219.
2. E. Czuhaj-Varjú, J. Dassow, J. Kelemen and G. Paun (eds.). *Grammar Systems: A Gramatical Approach to Distribution and Cooperation*. Gordon and Breach Science Publishers, 1994, ISBN 2-88124-957-4.
3. J. Hromkovič, J. Kari, L. Kari, and D. Pardubská. Two lower bounds on distributive generation of languages. In: I. Prívara, B. Rovan, and P. Ružička (eds.), *Mathematical Foundations of Computer Science 1994, MFCS'94, Proc.*, LNCS 841, Springer, Berlin, 1994, 423–432.
4. 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.
5. V. Kuboň, M. Lopatková, M. Plátek, and P. Pognan. Segmentation of Complex Sentences. In: P. Sojka, I. Kopeček, and K. Pala (eds.), *Text, Speech and Dialog, TSD 2006, Proc.*, LNCS 4188, Springer, Berlin, 2006, 151-158.
6. H. Messerschmidt and F. Otto. On determinism versus nondeterminism for restarting automata. *Information and Computation* 206 (2008) 1204–1218.
7. 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.
8. F. Otto and M. Plátek. A two-dimensional taxonomy of proper languages of lexicalized FRR-automata. In: C. Martin-Vide, F. Otto, and H. Fernau (eds.), *Language and Automata Theory and Applications, LATA 2008, Proc.*, LNCS 5196, Springer, Berlin, 2008, 409–420.
9. D. Pardubská. Communication complexity hierarchy of parallel communicating grammar system. In: *Developments in Theoretical Computer Science*. Gordon and Breach Science Publishers, Yverdon, 1994, 115–122.
10. D. Pardubská and M. Plátek. Parallel communicating grammar systems and analysis by reduction by restarting automata. In: G. Bel-Enguix and M.D. Jimenez-Lopez (eds.), *International Workshop on Non-Classical Formal Languages in Linguistics, ForLing 2008, Proc.*, Tarragona, 2008, 81–98.
11. D. Pardubská, M. Plátek, and F. Otto. On PCGS and FRR-automata. In: P. Vojtáš (ed.), *Information Technologies – Applications and Theory, ITAT 2008, Proc.*, Department of Computer Science, Faculty of Science, Pavol Jozef Šafárik University, Košice, 2008, 41–47.
12. Gh. Păun and L. Santean. Parallel communicating grammar systems: the regular case. *Ann. Univ. Buc. Ser. Mat.-Inform.* 37 (1989) 55–63.