ELSEVIER

Theoretical Computer Science 223 (1999) 87–120

Theoretical Computer Science

www.elsevier.com/locate/tcs

Independent parallelism in finite copying parallel rewriting systems

Owen Rambow[a], Giorgio Satta[b,*]

[a] *CoGenTex, Inc., 840 Hanshaw Road, Suite 11, Ithaca, NY 14850-1589, USA*

[b] *Dipartimento di Elettronica ed Informatica, Università di Padova, via Gradenigo, 6/A, I-35131 Padova, Italy*

Received March 1994; revised February 1997

Communicated by G. Rozenberg

Abstract

We consider the class of parallel rewriting systems and investigate the interaction between two complexity measures, that in the literature have been called synchronized parallelism and independent parallelism. It is shown that, when the degree of synchronized parallelism is bounded by some constant greater than one, the degree of independent parallelism induces an infinite non-collapsing hierarchy within the family of generated languages. The result is obtained using an original characterization of parallel rewriting systems. Other language-theoretic properties of parallel rewriting systems are proved in this work, that together with our main result provide an answer to some questions that were left open in the literature.

Keywords: Parallel rewriting systems; Local unordered scattered context grammars; Formal languages; Descriptional complexity

1. Introduction

Since the early 1970s, many rewriting systems have been presented in the formal language literature that extend the generative power of the class of context-free grammars. A family of these formalisms, called parallel rewriting systems, has been extensively investigated; the reader is referred to [9] for an excellent review of many important results about this family.

Two different kinds of parallelism are realized in parallel rewriting systems, that in [9, p. 151] have been called the *synchronized* parallelism and the *independent* parallelism. The synchronized parallelism allows derivations of substrings to proceed in a synchronous way, i.e., a sentence in the generated language may include substrings

* Corresponding author. Tel.: +39(49)827 7831; fax: +39(49)827 7699; e-mail: satta@dei.unipd.it.

0304-3975/99/$ - see front matter
PII: S0304-3975(97)00190-4

that have been obtained by a common underlying derivation process. The independent parallelism reflects the capability of the system to instantiate independent derivation processes that are combined together to form the generated string. Context-free grammars are the canonical example of a system with only independent parallelism, while the ETOL systems of Rozenberg [18] are an example of parallel rewriting systems with only synchronized parallelism. Examples of rewriting systems using both kinds of parallelism include the generalized syntax-directed translation (GSDT) of Aho and Ullman [2] and the top-down tree-to-string transducers (yT) of Engelfriet et al. [9]. (A transducer can be regarded as a controlled generative device.) Both classes generate the same languages. An interesting correspondence between the class of parallel rewriting systems and the class of two-way machines has been established in [9], where equivalence in generative power is shown using a generalized model called checking tree-pushdown transducer.

If we restrict the synchronized parallelism to a finite degree, that is if we allow only a bounded number of subderivations to be synchronized in a given grammar, we obtain a subfamily of parallel rewriting systems that includes the so called finite copying top-down tree-to-string transducers (yT_{fc}) of Engelfriet et al. [9], the string generating context-free hypergraph grammars (CFHG) of Bauderon and Courcelle [4] and Habel and Kreowski [11], the multiple context-free grammars (MCFG) of Kasami et al. [14] and Seki et al. [22] and the string-based linear context-free rewriting systems (LCFRS) of Vijay-Shanker et al. [24] and Weir [25]. All these rewriting systems are weakly equivalent, as shown in [8, 26]. We will call *finite copying* parallel rewriting systems the family of parallel rewriting systems with finite degree of synchronized parallelism. As far as the correspondence with two-way machines is concerned, the family of parallel rewriting systems with finite synchronized parallelism generates the same languages as the class of deterministic tree-walking transducers (DTWT) of Aho and Ullman [2].

At the same time, the size of the longest production in a given parallel rewriting system always imposes a finite bound on the number of independent derivation processes that can be interleaved. Hence, the degree of independent parallelism is always bounded by some constant in parallel rewriting systems. In what follows, we will regard the two kinds of parallelism introduced above as complexity measures.

Independent investigations of different formalisms in the family of finite copying parallel rewriting systems have shown that the degree of synchronized parallelism establishes an infinite, non-collapsing hierarchy in the generated languages; i.e., by increasing the degree of synchronized parallelism we gain additional generative power (see, for instance, [9, 11, 22]). As an example, if the degree of synchronized parallelism is bounded by an integer $f \geqslant 1$, a grammar can "count" up to $2f$, but cannot generate the language $L = \{a_1^n a_2^n \cdots a_{2f+1}^n \mid n \geqslant 0\}$. However, not much was known to date about the second complexity measure, i.e. independent parallelism or maximum production size. The major result of this paper is that, within parallel rewriting systems with synchronized parallelism bounded by a fixed (but arbitrary) constant $f \geqslant 2$, the degree of independent parallelism induces an infinite non-collapsing hierarchy for the generated languages. However, as we show, independent parallelism can be "traded" for

synchronized parallelism, in the sense that, given a finite copying parallel rewriting system, a weakly equivalent system in the same family can be obtained with a reduced degree of independent parallelism, but with an increased degree of synchronized parallelism. We investigate language theoretic properties of the hierarchy of rewriting systems of fixed degree of synchronized parallelism, and solve in the negative a question left open in [9], about whether parallel rewriting systems with degree of synchronized parallelism bounded by a fixed constant constitute a full principal AFL. When the degree of synchronized parallelism is bounded by $f = 1$, that is when synchronized parallelism is inhibited, the above rewriting systems can be cast in a normal form defined by some bound on the size of the productions, that is some bound on the degree of independent parallelism. (As already mentioned, in this case the generated languages are exactly the context-free languages and the above fact is related to the existence of two-normal forms for context-free grammars.) Our result shows that, when $f \geqslant 2$, such normal forms are not admitted. This solves a question left open in [2] and has interesting consequences for the design of algorithms for the recognition problem for these rewriting systems.

In a sense, our result is a generalization of a result in [1], concerning non-simple syntax-directed translation schemata, that may be viewed as a restricted kind of parallel rewriting systems with degree of synchronized parallelism bounded by $f = 2$. There, the existence of an infinite non-collapsing hierarchy induced by the degree of independent parallelism is shown for such systems. We generalize this formalism by introducing a new class of rewriting systems called local unordered scattered-context grammar (LUSCG) which has the same generative power as finite copying parallel rewriting systems. The definition of LUSCG is based on a rewriting restriction, called locality, that turns out to provide an exact characterization of finite copying parallel rewriting systems. We show our result by working on LUSCG in such a way that we can then transfer our result to all other formalisms mentioned above. The choice of LUSCG renders the proof of our result more intuitive, due to the intrinsic parallelism of these systems.

This paper is organized in the following way. In Section 2 we introduce local unordered scattered context grammars and define two parameters for this class that are related to synchronized and independent parallelism. In Section 3 we show that the degree of independent parallelism induces an infinite hierarchy when the degree of synchronized parallelism is bounded by a constant, and in Section 4 we prove some language theoretic properties for members of such a hierarchy. In Section 5 we use an equivalence result reported in the appendix to show how all our results can be transferred to other formalisms in the class of parallel rewriting systems. Finally, in Section 6 we discuss some other consequences of the presented results.

2. Definitions

In this section we introduce a new class of parallel rewriting systems, which we will call local unordered scattered context grammar, and show how valid derivations in

these systems can be represented by means of trees generated by context-free grammars. We then define two (independent) parameters for this class, called *fan-out* and *rank*. These two parameters are related to synchronized and independent parallelism, and their mutual interaction will be investigated throughout this paper. In what follows, we will use standard notational conventions. For an alphabet V, we denote by V^* the set of all finite strings over V. Let $a \in V$ and $w \in V^*$; $\#_a(w)$ denotes the number of occurrences of a in w. As usual, for a class $\mathscr{C}$ of generative devices, $\mathscr{L}(\mathscr{C})$ denotes the class of all languages generated by $\mathscr{C}$.

A class of rewriting systems called scattered context grammars was introduced by Greibach and Hopcroft [10]; an unordered version was proposed by Milgram and Rosenfeld [17] and Mayer [16]. The following definition is based on Salomaa [20, p. 259] and Dassow and Păun [6, p. 135].

Definition 1. An *unordered scattered context grammar* (*USCG* for short) is a quadruple $G = (V_N, V_T, P, S)$ where V_N, V_T are finite, disjoint sets of nonterminal and terminal symbols, respectively, $S \in V_N$ is the start symbol and P is a finite set of productions having the form $(A_1, \ldots, A_n) \to (\alpha_1, \ldots, \alpha_n)$, where $n \geqslant 1$, $A_i \in V_N$, $\alpha_i \in (V_N \cup V_T)^*$, $1 \leqslant i \leqslant n$.

We write $\gamma \Rightarrow_G \delta$ whenever there exist $p = (A_1, \ldots, A_n) \to (\alpha_1, \ldots, \alpha_n) \in P$ and an arbitrary permutation π of $\{1, \ldots, n\}$ such that

$$\gamma = \gamma_0 A_{\pi(1)} \gamma_1 A_{\pi(2)} \cdots \gamma_{n-1} A_{\pi(n)} \gamma_n,$$

$$\delta = \gamma_0 \alpha_{\pi(1)} \gamma_1 \alpha_{\pi(2)} \cdots \gamma_{n-1} \alpha_{\pi(n)} \gamma_n,$$

where $\gamma_i \in (V_N \cup V_T)^*$, $0 \leqslant i \leqslant n$.

The class of all unordered scattered context grammars is also denoted USCG. USCGs are known to be weakly equivalent to several other regulated rewriting systems, including context-free matrix grammars and state grammars. The reader is referred to Dassow and Păun [6] for details.

We now introduce a restriction on the derivation relation for USCG which we will call *locality*. Informally, locality forces each production to rewrite only symbols which were previously introduced together in a single step of the derivation. As a result, we have that in a local rewriting system the set of all derivations can be characterized by a recognizable set of trees in the sense of Thatcher [23], i.e., each derivation can be represented by a tree generated by a (fixed) context-free grammar. The notion of locality was first discussed in [25] and can be used to characterize the class of finite copying parallel rewriting systems, as it will be discussed in Section 5.

We need to introduce some additional notation. In what follows, strings γ in $(V_N \cup V_T)^*$ will be viewed as sequences of instances of symbols in $V_N \cup V_T$. Let G be an USCG and let γ and δ be two strings in $(V_N \cup V_T)^*$ such that $\gamma \Rightarrow_G \delta$. Intuitively speaking, if we view $\gamma \Rightarrow_G \delta$ as a rewriting step, then there is a one-to-one correspondence between instances of nonterminals in δ that have not been newly introduced by the step and instances of nonterminals in γ that have not been replaced. More formally,

assume that there exist B, n and i, $B \in V_N$, $n \geqslant 1$ and $0 \leqslant i \leqslant n$, such that

$$\gamma = \gamma_0 A_1 \gamma_1 \cdots A_i \gamma_i B \gamma_i' A_{i+1} \cdots \gamma_{n-1} A_n \gamma_n,$$

$$\delta = \gamma_0 \alpha_1 \gamma_1 \cdots \alpha_i \gamma_i B \gamma_i' \alpha_{i+1} \cdots \gamma_{n-1} \alpha_n \gamma_n.$$

Then we say that the indicated instance of nonterminal B in δ *corresponds to* the indicated instance of B in γ.

An equivalence relation I is said to be *associated with* a string $\gamma \in (V_N \cup V_T)^*$ if I is defined on the set of elements of γ that are instances of symbols in V_N. We introduce associated equivalence relations in the definition of the derive relation for USCG to define a new class of rewriting systems. (We overload symbol $\Rightarrow_G$.)

Definition 2. A *local unordered scattered context grammar* (*LUSCG* for short) is an unordered scattered context grammar $G = (V_N, V_T, P, S)$ for which a binary relation $\Rightarrow_G$ is defined over pairs consisting of a string in $(V_N \cup V_T)^*$ and an associated equivalence relation. We write $(\gamma, I_\gamma) \Rightarrow_G (\delta, I_\delta)$ if and only if:

(i) there exist $p = (A_1, \ldots, A_n) \rightarrow (\alpha_1, \ldots, \alpha_n) \in P$ and an arbitrary permutation π of $\{1, \ldots, n\}$ such that

$$\gamma = \gamma_0 A_{\pi(1)} \gamma_1 A_{\pi(2)} \cdots \gamma_{n-1} A_{\pi(n)} \gamma_n,$$

$$\delta = \gamma_0 \alpha_{\pi(1)} \gamma_1 \alpha_{\pi(2)} \cdots \gamma_{n-1} \alpha_{\pi(n)} \gamma_n,$$

where $\gamma_i \in (V_N \cup V_T)^*$, $0 \leqslant i \leqslant n$,

(ii) the indicated instances of nonterminals $A_{\pi(1)}, \ldots, A_{\pi(n)}$ in γ are equivalent under I_γ, and

(iii) all instances of nonterminals in the subsequences $\alpha_{\pi(1)}, \ldots, \alpha_{\pi(n)}$ indicated in δ are equivalent under I_δ, as are any instances of nonterminals in δ that correspond to instances in γ equivalent under I_γ; no other instances are equivalent under I_δ. (This uniquely determines I_δ.)

Note that, according to the above definition, string δ is obtained from γ as in Definition 1, with the additional requirement that the indicated instances of $A_{\pi(1)}, A_{\pi(2)}, \ldots, A_{\pi(n)}$ in γ are equivalent in the associated relation I_γ. Furthermore, the obtained equivalence relation I_δ makes equivalent all and only the instances of nonterminals newly introduced by the derivation step, and "preserves", with respect to I_γ, equivalences between instances of nonterminals that have not been newly introduced (see Example 1 below). The class of all local unordered scattered context grammars is also denoted LUSCG. The relationship between the non-local and local versions of USCG will be discussed in Section 6.

We introduce additional notation to be used in the following. Given a string of the form $\gamma_0 A_{\pi(1)} \gamma_1 \cdots \gamma_{n-1} A_{\pi(n)} \gamma_n$, $n \geqslant 1$ and π some permutation of $\{1, \ldots, n\}$, we denote with $I^{\langle A_1, \ldots, A_n \rangle}$ any associated relation that contains every pair of instances of nonterminals $A_1, \ldots, A_n$ indicated in γ. If $p : (A_1, \ldots, A_n) \rightarrow (\alpha_1, \ldots, \alpha_n)$ belongs to P, we say that $(A_1, \ldots, A_n)$ is the *left-hand tuple* of p and $(\alpha_1, \ldots, \alpha_n)$ is the *right-hand*

$$
\begin{aligned}
G_L &= (V_N, V_T, P, S);\\
V_N &= \{S, A, B\}\\
V_T &= \{[,]\}\\
P &= \{p_1: \quad (S) \to (AA),\\
&\qquad p_2: \quad (A,A) \to ([A],[A]),\\
&\qquad p_3: \quad (A,A) \to ([\,],[\,]),\\
&\qquad p_4: \quad (A,A) \to (AB,AB),\\
&\qquad p_5: \quad (B,B) \to (A,A)\}
\end{aligned}
$$

Fig. 1. An LUSCG for language $L = \{ww \mid w \in D_1\}$.

tuple of p. Relation $\Rightarrow_G$ will sometimes be written $\stackrel{p}{\Rightarrow}_G$ to indicate that production p was used in the rewriting. In order to represent derivations in G, we use the reflexive and transitive closure of $\Rightarrow_G$, written $\stackrel{*}{\Rightarrow}_G$. The language generated by a LUSCG G is $L(G) = \{w \mid (S, I^{\langle S\rangle}) \stackrel{*}{\Rightarrow}_G (w, \emptyset)\}$ (note that relation $I^{\langle S\rangle}$ associated with S is uniquely defined).

Example 1. Let D_1 be the Dyck language of strings of properly balanced parentheses and let $L = \{ww \mid w \in D_1\}$. Language L can be derived by the LUSCG G_L specified in Fig. 1. In order to show a derivation in G_L, we number instances of nonterminals in a string from left to right, and specify equivalence relations by giving their equivalence classes. Then string $[[]][[]][[]][] \in L$ can be derived in G_L as follows:

$$
\begin{aligned}
(S, \{\{1\}\}) &\stackrel{p_1}{\Rightarrow}_G (AA, \{\{1,2\}\})\\
&\stackrel{p_4}{\Rightarrow}_G (ABAB, \{\{1,2,3,4\}\})\\
&\stackrel{p_2}{\Rightarrow}_G ([A]B[A]B, \{\{1,3\},\{2,4\}\})\\
&\stackrel{p_5}{\Rightarrow}_G ([A]A[A]A, \{\{1,3\},\{2,4\}\})\\
&\stackrel{p_3}{\Rightarrow}_G ([[]]A[[]]A, \{\{1,2\}\})\\
&\stackrel{p_3}{\Rightarrow}_G ([[]][][[]][], \emptyset).
\end{aligned}
$$

As already mentioned, the locality restriction makes it possible to represent the underlying structure of a derivation by means of a tree from a recognizable set of trees. The following definition specifies how this can be done. Let $G = (V_N, V_T, P, S)$ be a LUSCG. Define $P^{(0)} = \{p \mid p \in P$, there are no nonterminals in the right-hand tuple of $p\}$ and $P^{(1)} = P - P^{(0)}$. We assume an arbitrary canonical (linear) ordering $\leqslant$ of the productions in P. Without loss of generality, we also assume that p_S is the unique production in P that rewrites S (i.e., has left-hand tuple (S)) and $p_S \in P^{(1)}$.

Definition 3. The *derivation grammar of* a LUSCG $G = (V_N, V_T, P, S)$, denoted $\mathrm{der}(G)$, is the context-free grammar $(P^{(1)}, P^{(0)}, \Pi, p_S)$ where $P^{(1)}$ and $P^{(0)}$ are the set of nonterminal and terminal symbols, respectively, p_S is the initial symbol and Π contains

$$
\begin{aligned}
\mathrm{der}(G_L) = {} & (P^{(1)}, P^{(0)}, \Pi, p_1);\\
P^{(1)} = {} & \{p_1, p_2, p_4, p_5\},\\
P^{(0)} = {} & \{p_3\},\\
\Pi = {} & \{\ p_1 \to p_2, \quad p_1 \to p_3, \quad p_1 \to p_4,\\
& \quad p_2 \to p_2, \quad p_2 \to p_3, \quad p_2 \to p_4,\\
& \quad p_4 \to p_2\,p_5, \quad p_4 \to p_3\,p_5, \quad p_4 \to p_4\,p_5,\\
& \quad p_5 \to p_2, \quad p_5 \to p_3, \quad p_5 \to p_4 \quad \}
\end{aligned}
$$

Fig. 2. The derivation grammar of G_L.

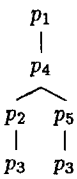


Fig. 3. The derivation tree in $\mathrm{der}(G_L)$ corresponding to the derivation in G_L of string [[]][][[]][].

all and only productions of the form $p \to p_1 \cdots p_n$, where $p, p_1, \ldots, p_n \in P$ and $n \geqslant 1$, such that $p_1 \leqslant p_2 \leqslant \cdots \leqslant p_n$ and the multiset of instances of nonterminals in the right-hand tuple of p equals the multiset of instances of nonterminals in the left-hand tuples of $p_1, \ldots, p_n$.

We remark that, as a consequence of the canonical ordering of the productions in P in Definition 3, two productions in $\mathrm{der}(G)$ cannot differ only in the order of the right-hand symbols. Note also that to every derivation in G corresponds exactly one derivation in $\mathrm{der}(G)$, in a straightforward way, and to every derivation in $\mathrm{der}(G)$ corresponds at least one derivation in G.

Example 1 (*continued*). Assume the canonical ordering of productions of G_L such that $p_i \leqslant p_j$ for $1 \leqslant i \leqslant j \leqslant 5$. Then the derivation grammar $\mathrm{der}(G_L)$ is given in Fig. 2. Note how production $p_4 \to p_2\,p_5$ in Π, for instance, satisfies Definition 3, since productions p_2 and p_5 together rewrite all instances of nonterminal symbols of G introduced by the right-hand tuple of p_4. The (leftmost) derivation in $\mathrm{der}(G_L)$ corresponding to the derivation of string [[]][][[]][] previously presented is $p_1 \Rightarrow p_4 \Rightarrow p_2\,p_5 \Rightarrow p_3\,p_5 \Rightarrow p_3\,p_3$ and is represented as a derivation tree in Fig. 3.

We conclude the present section with the definition of two parameters associated with grammars in the class LUSCG. In the next sections these parameters will be considered as complexity measures and their interaction will be investigated.

Definition 4. Let $G = (V_N, V_T, P, S)$ be a LUSCG, $p \in P$, and let $\mathrm{der}(G) = (P^{(1)}, P^{(0)}, \Pi, p_S)$ be the derivation grammar of G. The *fan-out* of production p, written $\varphi(p)$, is the length of its tuples. The fan-out of G is defined as $\varphi(G) = \max_{p \in P} \varphi(p)$. The

rank of production p, written $\rho(p)$, is defined as $\rho(p)=\max_{(p\to\alpha)\in\Pi}|\alpha|$ (assuming $\max\emptyset=0$). The rank of G is defined as $\rho(G)=\max_{p\in P}\rho(p)$.

Example 1 (*continued*). For the sample grammar G_L, we have $\varphi(p_1)=1$ and $\varphi(p_i)=2$ for $2\leqslant i\leqslant 5$; hence we have $\varphi(G_L)=2$. Furthermore, from the specification of $\mathrm{der}(G_L)$ we see that $\rho(p_4)=\max_{(p_4\to\alpha)\in\Pi}|\alpha|=2$. This implies $\rho(G_L)=2$, since all the remaining productions in P have rank smaller than two.

For integers $f\geqslant 1$ and $r\geqslant 0$, LUSCG(f) will denote the class of all LUSCG having fan-out bounded by f and r-LUSCG will denote the class of all LUSCG with rank bounded by r; r-LUSCG(f) will denote the intersection of the two.

3. A rank hierarchy

This section presents the main result of the paper. We show that the rank parameter defines an infinite (non-collapsing) hierarchy within each class LUSCG(f), $f\geqslant 2$. The technique we have adopted has been inspired by a technique used in [1] to prove the existence of an infinite hierarchy induced by the rank in non-simple syntax-directed translation schemata (SDTS). The result in [1] concerns the class of translations between two context-free grammars, i.e., between parallel rewriting systems with degree of synchronized parallelism bounded by $f=1$. This class may be viewed as a (proper) subclass of the parallel rewriting systems with $f=2$. Indeed, the definition we have given for the derive relation for LUSCG is very similar to the definition of derivation in SDTS.

Let $G=(V_N,V_T,P,S)$ be a LUSCG. We first introduce some notions that describe productions of G in terms of derivations in which they can participate.

Definition 5. Let $\mathscr{A}\subseteq V_T$. A production p in P of the form $(A_1,\ldots,A_t)\to(\alpha_1,\ldots,\alpha_t)$, $t\geqslant 1$, *covers* $\mathscr{A}$ if and only if for every integer $d\geqslant 1$ there exists a derivation η such that the following conditions are satisfied:

(i) η has the form

$$\begin{aligned}(S,I^{\langle S\rangle}) &\overset{*}{\Rightarrow}_G (u_0A_{\pi(1)}u_1\cdots u_{t-1}A_{\pi(t)}u_t, I_1^{\langle A_1,\ldots,A_t\rangle})\\ &\overset{p}{\Rightarrow}_G (u_0\alpha_{\pi(1)}u_1\cdots u_{t-1}\alpha_{\pi(t)}u_t, I_2)\\ &\overset{*}{\Rightarrow}_G (u_0v_1u_1\cdots u_{t-1}v_tu_t,\emptyset),\end{aligned}\tag{1}$$

where $u_i,v_j\in V_T^*$, $0\leqslant i\leqslant t$, $1\leqslant j\leqslant t$, and π is some permutation of $\{1,\ldots,t\}$;

(ii) string $v_1\cdots v_t$ includes more than d instances of each terminal in $\mathscr{A}$ and string $u_0v_1u_1\cdots u_{t-1}v_tu_t$ includes more than d instances of each terminal in V_T.

In the following we will use symbol $\lhd$ to denote the covering relation, and will write $p\lhd a$ to abbreviate $p\lhd\{a\}$. Furthermore, we will write $A\lhd\mathscr{A}$ *by means of* p

if there exists a production p of the form $(A_1,\ldots,A_t)\rightarrow(\alpha_1,\ldots,\alpha_t)$, $t\geqslant 1$, such that $A=A_l$ for some l, $1\leqslant l\leqslant t$, and for every integer $d\geqslant 1$ there exists a derivation η of the form (1), such that string v_l includes more than d instances of each terminal in $\mathscr{A}$ and string $u_0v_1u_1\cdots u_{t-1}v_tu_t$ includes more than d instances of each terminal in V_T. Note that if $A\lhd\mathscr{A}$ by means of p, then $p\lhd\mathscr{A}$. Note also that $p\lhd\mathscr{A}$ $(A\lhd\mathscr{A})$ implies $p\lhd\mathscr{B}$ $(A\lhd\mathscr{B})$ for every $\mathscr{B}\subseteq\mathscr{A}$.

We present a sufficient condition for a production p to cover a subset of V_T, that will be used throughout this section. Let $p:(A_1,\ldots,A_t)\rightarrow(\alpha_1,\ldots,\alpha_t)$, $t\geqslant 1$, be a production in P, and let $\mathscr{A}\subseteq V_T$. If p does not cover $\mathscr{A}$ (written $p\not\lhd\mathscr{A}$) there must be a constant $M_{p,\mathscr{A}}$ such that, for every derivation of the form in (1), either $M_{p,\mathscr{A}}$ bounds the number of instances of a symbol from $\mathscr{A}$ appearing in $v_1v_2\cdots v_t$, or $M_{p,\mathscr{A}}$ bounds the number of instances of a symbol from V_T appearing in $u_0v_1u_1\cdots v_tu_t$. If $p\lhd\mathscr{A}$, we let $M_{p,\mathscr{A}}=-1$. Let M_G be the maximum among all $M_{p,\mathscr{A}}$, $p\in P$ and $\mathscr{A}\subseteq V_T$. Then, whenever $p\in P$ is used in a derivation of $w\in L(G)$, in such a way that p derives more than M_G instances of each symbol in some set $\mathscr{A}\subseteq V_T$ and w itself includes more than M_G instances of each symbol in V_T, we can conclude that p covers $\mathscr{A}$.

The following fact will also be used several times in this section.

Lemma 1. *Let $p\in P$ be a production of the form $(A_1,\ldots,A_t)\rightarrow(\alpha_1,\ldots,\alpha_t)$, $t\geqslant 1$. If p covers some $\mathscr{A}\subseteq V_T$, then there exist sets $\mathscr{A}_i$, $1\leqslant i\leqslant t$, such that $\bigcup_{i=1}^{t}\mathscr{A}_i=\mathscr{A}$ and, for every integer $d\geqslant 1$, there exists a derivation η such that the following conditions are both satisfied:*

(i) *η has the form (1); and*

(ii) *for each i, $1\leqslant i\leqslant t$, string v_i includes more than d instances of each terminal in $\mathscr{A}_i$ and string $u_0v_1u_1\cdots u_{t-1}v_tu_t$ includes more than d instances of each terminal in V_T.*

Proof. Let $m\geqslant 1$ be an integer. Since $p\lhd\mathscr{A}$, there exists a derivation η of the form (1) such that string $v_1\cdots v_t$ includes more than mt instances of each terminal in $\mathscr{A}$ and string $u_0v_1u_1\cdots u_{t-1}v_tu_t$ includes more than m instances of each terminal in V_T. Let $\mathscr{P}_m=\langle\mathscr{A}_{m,1},\mathscr{A}_{m,2},\ldots,\mathscr{A}_{m,t}\rangle$ be a partition of $\mathscr{A}$ specified as follows. For each $a\in\mathscr{A}$, choose the least l, $1\leqslant l\leqslant t$, such that v_l includes more than m instances of a, and let $a\in\mathscr{A}_{m,l}$ (there is always such an l by a counting argument; some of the $\mathscr{A}_{m,i}$, $1\leqslant i\leqslant t$, may remain empty). We thus have an infinite sequence of partitions $\sigma=\mathscr{P}_1,\mathscr{P}_2,\ldots$. Since there are finitely many partitions of $\mathscr{A}$, there exists some partition $\mathscr{P}$ that occurs infinitely often in σ. Clearly, the members of $\mathscr{P}$ meet the requirements on the $\mathscr{A}_i$'s set out in the proposition. □

We first prove a separation result for classes r-LUSCG(f), $r\geqslant 3$. In order to do so, we define a particular family of languages to which we will henceforth restrict our attention.

Definition 6. Let r and f be two integers, $r, f \geqslant 1$. Let also $V_{\mathrm{T}}^{(r,f)} = \{a_{i,j} \mid 1 \leqslant i \leqslant r, 1 \leqslant j \leqslant f\}$ and π_r a permutation of $\{1,2,\ldots,r\}$ defined as follows. If r is even:

$$\pi_r(i) = \begin{cases} 2i-1, & i \in \{1,\ldots,r/2\}; \\ 2i-r, & i \in \{r/2+1,\ldots,r\}. \end{cases}$$

If r is odd:

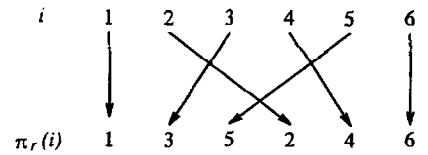
$$\pi_r(i) = \begin{cases} i, & i \in \{1,r\}; \\ r-1, & i = (r+1)/2; \\ r-2(i-1), & i \in \{2,\ldots,(r+1)/2-1\}; \\ 2i-r-1, & i \in \{(r+1)/2+1,\ldots,r-1\}. \end{cases}$$

Language $L_{r,f}$ is specified as follows:

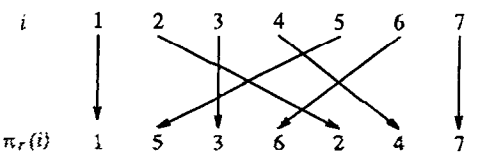
$$L_{r,f} = \{w_1 w_2 \cdots w_f \mid w_1 = a_{1,1}^{i_1} \cdots a_{r,1}^{i_r},\ w_h = a_{\pi_r(1),h}^{i_{\pi_r(1)}} \cdots a_{\pi_r(r),h}^{i_{\pi_r(r)}},\ 2 \leqslant h \leqslant f,\ i_j \geqslant 1,\ 1 \leqslant j \leqslant r\}.$$

Note that if $r = 2n$ then π_r is the permutation $[1,3,5,\ldots,2n-1,2,4,6,\ldots,2n]$, if $r = 2n-1$ then π_r is the permutation $[1,2n-3,2n-5,\ldots,5,3,2n-2,2,4,\ldots,2n-4, 2n-1]$. The effect of π_r in the cases $r=6$ and $r=7$ is shown for illustrative purposes in Fig. 4. Below we often use the fact that for all $r \geqslant 1$, $\pi_r(1) = 1$ and $\pi_r(r) = r$.

Observe that there is an order $c_1, c_2, \ldots, c_{rf}$ of $V_{\mathrm{T}}^{(r,f)}$ such that $L_{r,f} \subseteq c_1^+ c_2^+ \cdots c_{rf}^+$. In the following we will call *segment* each substring w_h, $1 \leqslant h \leqslant f$, in the definition of a string in $L_{r,f}$. For $1 \leqslant s \leqslant r$, we will also use $\overline{a_s}$ to denote the set $\{a_{s,1}, a_{s,2} \ldots, a_{s,f}\}$, which we will refer to as a *terminal group* for $L_{r,f}$. Let $\overline{\mathscr{A}}$ be a set of terminal



$r = 6$



$r = 7$

Fig. 4. The permutation π_r for $r = 6$ and $r = 7$.

$$
\begin{aligned}
G &= (V_N, V_T^{(r,f)}, P, S);\\
P &= \{p_i \mid 1 \leqslant i \leqslant 3+2r\};\\
V_N &= \{S\} \cup \{Q_j, R_j \mid 1 \leqslant j \leqslant f\} \cup\\
&\qquad \{A_{i,j} \mid 1 \leqslant i \leqslant r, 1 \leqslant j \leqslant f\};
\end{aligned}
$$

$$
\begin{aligned}
p_1 &: (S) \to (A_{1,1}Q_1 \cdots A_{1,f}Q_f);\\
p_2 &: (Q_1, \ldots, Q_f) \to (R_1 A_{r,1}, \ldots, R_f A_{r,f});\\
p_3 &: (R_1, \ldots, R_f) \to (\alpha^{(1)}, \ldots, \alpha^{(f)}), \quad \alpha^{(1)} = A_{2,1}A_{3,1}\cdots A_{r-1,1},\\
&\qquad \alpha^{(j)} = A_{\pi_r(2),j}A_{\pi_r(3),j}\cdots A_{\pi_r(r-1),j}, \quad 2 \leqslant j \leqslant f;\\
p_{3+j} &: (A_{j,1}, \ldots, A_{j,f}) \to (a_{j,1}A_{j,1}, \ldots, a_{j,f}A_{j,f}), \quad 1 \leqslant j \leqslant r;\\
p_{3+r+j} &: (A_{j,1}, \ldots, A_{j,f}) \to (a_{j,1}, \ldots, a_{j,f}) \quad 1 \leqslant j \leqslant r;
\end{aligned}
$$

Fig. 5. A $(r-2)$-LUSCG(f) grammar for $L_{r,f}$.

groups for $L_{r,f}$. If $p \lhd \{a \mid a \in \overline{a_s},\ \overline{a_s} \in \overline{\mathscr{A}}\}$, then we write $p \lhd \overline{\mathscr{A}}$. For $r \geqslant 3$, the set $\{\overline{a_s} \mid 2 \leqslant s \leqslant r-1\}$ will be denoted $\mathscr{B}^{(r,f)}$.

Example 2. Language $L_{r,f}$ can be derived by a grammar in $(r-2)$-LUSCG(f), for $f \geqslant 1$ and $r \geqslant 4$: such a grammar is defined in Fig. 5. We have $\rho(p_1) = \rho(p_2) = 2$, $\rho(p_3) = r-2$ and $\rho(p_{3+j}) = 1$, $\rho(p_{3+r+j}) = 0$ for $1 \leqslant j \leqslant r$; the rank of p_3 determines the rank of G. Observe that p_3 covers $\mathscr{B}^{(r,f)}$.

In Lemma 5 below we will show a basic fact about our family of languages, namely that, for any $r \geqslant 6$ and $f \geqslant 2$, any grammar in LUSCG that derives $L_{r,f}$ cannot have a production that covers more than one, but fewer than $r-2$, terminal groups in $\mathscr{B}^{(r,f)}$. To do this, we need first some intermediate results. In the following discussion, we will be referring to an implicit LUSCG of fan-out f generating $L_{r,f}$; hence, for example, whenever we mention a symbol $a_{s,q}$, the ranges of s and q are implicitly stated.

The next lemma shows that for languages $L_{r,f}$, the properties of covering a set of terminal symbols and of covering the set of associated terminal groups cannot be distinguished.

Lemma 2. *If a production p covers some $\mathscr{A} \subseteq V_T^{(r,f)}$, then p covers $\{\overline{a_s} \mid a_{s,q} \in \mathscr{A}\}$.*

Proof. Let $\mathscr{A}' = \bigcup_{a_{s,q} \in \mathscr{A}} \overline{a_s}$. If $\mathscr{A} = \mathscr{A}'$ the statement trivially holds. Otherwise, let $b \in \mathscr{A}'$ be such that $p \lhd \mathscr{A}$ but $p \not\lhd \mathscr{A} \cup \{b\}$. Consider a derivation of the form

$$
\begin{aligned}
(S, I^{\langle S \rangle}) &\overset{*}{\Rightarrow}_G (u_0 A_1 u_1 \cdots u_{t-1} A_t u_t, I_1^{\langle A_1, \ldots, A_t \rangle})\\
&\overset{p}{\Rightarrow}_G (u_0 \alpha_1 u_1 \cdots u_{t-1} \alpha_t u_t, I_2)\\
&\overset{*}{\Rightarrow}_G (u_0 v_1 u_1 \cdots u_{t-1} v_t u_t, \emptyset), \qquad (2)
\end{aligned}
$$

where $u_i, v_j \in (V_T^{(r,f)})^*$, and $t \geqslant 1$. (Note that such a derivation exists since p covers $\mathscr{A}$.) Let m be the number of instances of b in $u_0 u_1 \cdots u_t$. Since p covers $\mathscr{A}$, there exists

a second derivation

$$\begin{aligned}(S, I^{\langle S\rangle}) &\overset{*}{\Rightarrow}_G (u'_0 A_{\pi(1)} u'_1 \cdots u'_{t-1} A_{\pi(t)} u'_t, I_3^{\langle A_1,\ldots,A_t\rangle}) \\ &\overset{p}{\Rightarrow}_G (u'_0 \alpha_{\pi(1)} u'_1 \cdots u'_{t-1} \alpha_{\pi(t)} u'_t, I_4) \\ &\overset{*}{\Rightarrow}_G (u'_0 x_{\pi(1)} u'_1 \cdots u'_{t-1} x_{\pi(t)} u'_t, \emptyset), \end{aligned} \tag{3}$$

with $u'_i, x_j \in (V_T^{(r,f)})^*$ and π some permutation of $\{1,\ldots,t\}$, such that the number of instances of each $a \in \mathscr{A}$ in string $x_1 x_2 \cdots x_t$ exceeds $M_G + m$ and the number of instances of each $a \in V_T^{(r,f)}$ in string $u'_0 x_{\pi(1)} u'_1 \cdots u'_{t-1} x_{\pi(t)} u'_t$ exceeds M_G. Since $\mathscr{A} \cup \{b\}$ is not covered by p, the overall number of instances of b distributed within $x_1 x_2 \cdots x_t$ must be bounded by M_G. We can combine (2) and (3) to obtain a third string $u_0 x_1 u_1 \cdots u_{t-1} x_t u_t$ in $L_{r,f}$, for which the number of instances of some $a_{s,q} \in \mathscr{A}$ with $b \in \overline{a_s}$ differs from that of b, contradicting the definition of $L_{r,f}$. We conclude that no such a b could exist. □

Next we show (in Lemma 4) that whenever (an instance of) a nonterminal A in the left-hand tuple of a production p covers $\{a_{s,q}, a_{s,q'}\}$, i.e. a set of two symbols belonging to the same terminal group but to different segments, then p covers the whole of $V_T^{(r,f)}$. We prove the result in two steps.

Let a and b be two symbols in $V_T^{(r,f)}$. Observe that the set of all terminal symbols occurring between a and b (including a and b) is the same for all strings in $L_{r,f}$. We will call this set the *in-between set* of a and b.

Lemma 3. *Let p be a production such that an instance of a nonterminal symbol A in the left-hand tuple of p covers (by means of p) the set $\{a,b\} \subseteq V_T^{(r,f)}$. Then p covers the in-between set of a and b.*

Proof. Let $\mathscr{A}$ be the in-between set of a and b. If $\mathscr{A} = \{a,b\}$, the lemma holds trivially. Otherwise, let $\mathscr{A}' = \mathscr{A} - \{a,b\}$ ($\mathscr{A}' \neq \emptyset$). Since $A \lhd \{a,b\}$, there must be a derivation of the form

$$\begin{aligned}(S, I^{\langle S\rangle}) &\overset{*}{\Rightarrow}_G (u_0 A_1 u_1 \cdots u_{k-1} A_k u_k \cdots u_{t-1} A_t u_t, I_1^{\langle A_1,\ldots,A_t\rangle}) \\ &\overset{p}{\Rightarrow}_G (u_0 \alpha_1 u_1 \cdots u_{k-1} \alpha_k u_k \cdots u_{t-1} \alpha_t u_t, I_2) \\ &\overset{*}{\Rightarrow}_G (u_0 v_1 u_1 \cdots u_{k-1} v_k u_k \cdots u_{t-1} v_t u_t, \emptyset), \end{aligned} \tag{4}$$

where $A = A_k$, $u_i, v_j \in (V_T^{(r,f)})^*$ and $t \geqslant 1$, such that more than M_G instances of symbols a and b are found in v_k and more than M_G instances of each symbol in $V_T^{(r,f)}$ are distributed within the string $u_0 v_1 \cdots u_{k-1} v_k u_k \cdots v_t u_t$. From the definition of $L_{r,f}$ it follows that all instances of the symbols in $\mathscr{A}'$ must occur in v_k. But then p covers $\mathscr{A}$. □

We can now prove the previously mentioned result.

Lemma 4. *Let $f \geqslant 2$ and $r \geqslant 1$. If a nonterminal A from the left-hand tuple of a production p covers $\{a_{s,q}, a_{s,q'}\} \subseteq V_{\mathrm{T}}^{(r,f)}$ for some s and $q < q'$, then p covers $V_{\mathrm{T}}^{(r,f)}$.*

Proof. Since $\pi_r(r) = r$ and $\pi_r(1) = 1$, $a_{r,q}$ and $a_{1,q+1}$ are in the in-between set of $a_{s,q}$ and $a_{s,q'}$. Then p must cover $\{a_{r,q}, a_{1,q+1}\}$ by Lemma 3. By Lemma 2, p must cover $\{\overline{a_1}, \overline{a_r}\}$. Let B be a nonterminal that covers $a_{1,1}$ by means of p (which should exist by Lemma 1 applied to $\mathscr{A} = \overline{a_1} \cup \overline{a_r}$). If B covers $\{a_{1,1}, a\}$, a a member of $\overline{a_r}$, then by Lemma 3, $p \lhd \{a_{j,1} \mid 1 \leqslant j \leqslant r\}$, and we are done by Lemma 2. Suppose instead that B covers no member of $\overline{a_r}$. Since every $a_{r,i}$ is covered by at least one nonterminal from the left-hand tuple of p (again by Lemma 1 applied to $\mathscr{A} = \overline{a_1} \cup \overline{a_r}$) and since we assume that p has at most f nonterminals, there must be a nonterminal from the left-hand tuple of p which covers two members of $\overline{a_r}$, say $a_{r,u}$ and $a_{r,u'}$ with $u < u'$. Again by Lemma 3, $p \lhd \{a_{j,u'} \mid 1 \leqslant j \leqslant r\}$, and we are done by Lemma 2. □

We now use the previous results to derive a basic property of G that will be used to show the major result.

Lemma 5. *Let $f \geqslant 2$ and $r \geqslant 6$. If a production p covers more than one terminal group in $\mathscr{B}^{(r,f)}$, then p covers $\mathscr{B}^{(r,f)}$.*

Proof. Assume that $(A_1, \ldots, A_t)$, $t \geqslant 1$, is the left-hand tuple of p. First we show that, if $p \lhd \{\overline{a_s}, \overline{a_{s'}}\}$ for $\overline{a_s}, \overline{a_{s'}} \in \mathscr{B}^{(r,f)}$, $s < s'$, then the only interesting case for us is $t = f$ and $A_i \lhd \{a_{s,i}, a_{s',i}\}$ for $1 \leqslant i \leqslant f$. By Lemma 1 (applied to $\mathscr{A} = \overline{a_s} \cup \overline{a_{s'}}$) symbols $a_{s,q}$ and $a_{s,q'}$, $q \neq q'$, must be covered by some (instances of) nonterminals in the left-hand tuple of p. If any nonterminal A in the left-hand tuple of p covers $\{a_{s,q}, a_{s,q'}\}$, then by Lemma 4, p covers all symbols in $V_{\mathrm{T}}^{(r,f)}$. A similar statement holds for s' instead of s. The remaining possibility is that $t = f$ and for all i, $1 \leqslant i \leqslant f$, A_i covers a set including exactly one terminal in $\overline{a_s}$ and exactly one terminal in $\overline{a_{s'}}$. W.l.o.g. we may assume that $A_i \lhd a_{s,i}$, $1 \leqslant i \leqslant f$. Assume also that l is the least index such that $A_l \lhd \{a_{s,l}, a_{s',j}\}$, for some $j > l$. Then we would have that instances of $a_{s',l}$ follow instances of $a_{s',j}$ in the derived string, contrary to the definition of $L_{r,f}$. Thus, in the following, we will deal only with the case $A_i \lhd \{a_{s,i}, a_{s',i}\}$ for $1 \leqslant i \leqslant f$.

Since A_1 covers $\{a_{s,1}, a_{s',1}\}$ by means of p, by Lemma 3 we conclude that p must also cover $\{a_{s,1}, a_{s+1,1}\}$, and hence $\{\overline{a_s}, \overline{a_{s+1}}\}$ by Lemma 2. Again we restrict our attention to the only interesting case in which $A_i \lhd \{a_{s,i}, a_{s+1,i}\}$, $1 \leqslant i \leqslant f$. By investigating the case $i = 2$, we now show that $p \lhd \{a_{r-1,2}, a_{2,2}\}$. We distinguish three cases.

Case 1: r is even. It can be seen from the definition of $L_{r,f}$ that $a_{r-1,2}$ and $a_{2,2}$ are in the in-between set of $a_{s,2}$ and $a_{s+1,2}$. We have that p covers $\{a_{r-1,2}, a_{2,2}\}$ by Lemma 3.

Case 2: r is odd and $s \neq r-2$. It again follows from the definition of $L_{r,f}$ and from Lemma 3 that $p \lhd \{a_{r-1,2}, a_{2,2}\}$.

Case 3: r is odd and $s = r-2$. Then $A_2 \lhd \{a_{r-2,2}, a_{r-1,2}\}$. By Lemma 3 and from the definition of $L_{r,f}$, p must also cover $\{a_{r-2,2}, a_{3,2}\}$; by Lemma 2 p covers $\{\overline{a_3}, \overline{a_{r-2}}\}$.

One more time we restrict our attention to the case in which $A_1 \lhd \{a_{3,1}, a_{r-2,1}\}$. Since $r \geqslant 6$, we can apply the same reasoning to see that p covers $\{\overline{a_3}, \overline{a_4}\}$. But since $3 \neq r-2$, we are now in Case 2.

We may conclude that $p \lhd \{a_{r-1,2}, a_{2,2}\}$. By Lemma 2, $\{a_{2,1}, a_{r-1,1}\}$ must also be covered by p. The only interesting case is if this set is covered by A_1. But then by Lemma 3 $p \lhd \{a_{j,1} \mid 2 \leqslant j \leqslant r-1\}$, and we are done by Lemma 2. □

The following lemma presents a property of derivations in G that will be used to "factorize" derivations for some sentences in $L_{r,f}$. We need to introduce two additional notions. Let p be a production whose left-hand tuple is $(A_1, \ldots, A_t)$, $t \leqslant f$. Assume the existence of a derivation of the form

$$(S, I^{\langle S \rangle}) \overset{*}{\Rightarrow}_G (u_0 A_{\pi(1)} u_1 \cdots u_{t-1} A_{\pi(t)} u_t, I^{\langle A_1, \ldots, A_t \rangle}),$$

where $u_i \in (V_T^{(r,f)})^*$ and π is some permutation of $\{1, \ldots, t\}$. Then $u_0 A_{\pi(1)} u_1 \cdots u_{t-1} A_{\pi(t)} u_t$ is called a *p-factorized* sentential form. Let a, b, c be different symbols in $V_T^{(r,f)}$. We say that b is *isolated* in the above sentential form whenever, for strings $x, y, v, z \in (V_T^{(r,f)})^*$, one of the following conditions is realized: (i) $u_0 = xbycv$, (ii) $u_j = xaybvcz$ for some j, $1 \leqslant j \leqslant t-1$, or (iii) $u_t = xaybv$. Note that whenever a terminal symbol b is isolated in a p-factorized sentential form, then p cannot generate b (because all occurrences of b in a sentence of $L_{r,f}$ are consecutive).

Lemma 6. *Let $f \geqslant 2$, $r \geqslant 6$. Let p be a production such that $p \lhd \mathscr{B}^{(r,f)}$ and let $u_0 A_1 u_1 \cdots u_{t-1} A_t u_t$, $t \leqslant f$, be a p-factorized sentential form. Then for every terminal group $\overline{a_s} \in \mathscr{B}^{(r,f)}$ there exists a terminal symbol $a \in \overline{a_s}$ such that a is not found in the string $u_0 u_1 \cdots u_t$.*

Proof. For the sake of contradiction, assume that $u_0 u_1 \cdots u_t$ contains instances of every terminal symbol in some $\overline{a_s} \in \mathscr{B}^{(r,f)}$. First of all, we claim that no u_i, $0 \leqslant i \leqslant t$, can contain two different terminals from $\overline{a_s}$. From the definition of $L_{r,f}$ it can be seen that for any $a_{s,q}$, $a_{s,q'}$ in $\overline{a_s}$, $q \neq q'$, the in-between set of $a_{s,q}$ and $a_{s,q'}$ contains at least one terminal b from some $\overline{a_{s'}} \in \mathscr{B}^{(r,f)}$, $s' \neq s$. If $a_{s,q}$ and $a_{s,q'}$ are included in u_i, then b will be isolated in the p-factorized sentential form and p could not generate b, contrary to the hypotheses. This proves our claim.

Let $l = \pi_r(r-1)$, i.e., $l = r-2$ if r is even, $l = r-3$ if r is odd. To prove the lemma, we will distinguish three cases.

Case 1: $s \notin \{2, l\}$. If any $a_{s,q} \in \overline{a_s}$ is included in u_0, then $a_{2,1}$ to its left will be isolated and p could not generate $a_{2,1}$, contrary to the hypotheses. Similarly, if any $a_{s,q}$ is included in u_t, then $a_{l,f}$ to its right will be isolated and p could not generate $a_{l,f}$, again a contradiction. We conclude therefore that the terminals in $\overline{a_s}$ are all contained within $u_1 \cdots u_{t-1}$. Since $t \leqslant f$, there will be some u_i, $1 \leqslant i \leqslant t-1$, which contains two different terminals from $\overline{a_s}$, contradicting our claim.

Case 2: $s = 2$. If any $a_{2,q} \in \overline{a_2}$ is contained in u_t, then $a_{l,f}$ to its right will be isolated and p could not generate such a symbol. From our claim and the definition of $L_{r,f}$, it

follows that $t=f$ and each $a_{2,q}$ is contained in u_{q-1} for $1\leqslant q\leqslant f$. Consider now A_1. Since p covers $\mathscr{B}^{(r,f)}$ and u_1 contains $a_{2,2}$, A_1 must cover at least the set $\{a_{2,1},a_{3,2}\}$ (whose elements occur to the left of $a_{2,2}$ in strings of $L_{r,f}$). By Lemma 3 p covers the in-between set of $a_{2,1}$ and $a_{3,2}$, which includes $a_{1,2}$; therefore p covers $\overline{a_1}$ by Lemma 2. But this is impossible, since u_0 contains an instance of $a_{2,1}$, which would be to the left of the instances of $a_{1,1}$ generated by p, contradicting the definition of $L_{r,f}$.

Case 3: $s=l$. If any $a_{l,q}\in\overline{a_l}$ is contained in u_0, then $a_{2,1}$ to its left will be isolated and p could not generate such a symbol. Again it follows from our claim that $t=f$ and each $a_{l,q}$ is contained in u_q, for $1\leqslant q\leqslant f$. Consider A_2. Since p covers $\mathscr{B}^{(r,f)}$, u_1 contains $a_{l,1}$, and u_2 contains $a_{l,2}$, A_2 must cover at least the set $\{a_{r-1,1},a_{3,2}\}$ (whose elements occur in between $a_{l,1}$ and $a_{l,2}$ in strings of $L_{r,f}$). By Lemma 3, p covers the in-between set of $a_{r-1,1}$ and $a_{3,2}$, which includes $a_{r,1}$; therefore p covers $\overline{a_r}$ by Lemma 2. This is impossible, because u_f contains an instance of $a_{l,f}$, which would be to the right of the instances of $a_{r,f}$ generated by p, contradicting the definition of $L_{r,f}$. □

The proof of the following theorem, which refers to all previous results, shows that, for all sentences w in some subset of $L_{r,f}$, any derivation in G of w can be partitioned into two parts. In a sense to be made more precise below, the first part of the derivation cannot generate all terminal symbols in any terminal group in $\mathscr{B}^{(r,f)}$, while the second part of the derivation uses productions that do cover $\mathscr{B}^{(r,f)}$.

Theorem 1. *Let* $f\geqslant 2$, $r\geqslant 6$. *Then we have* $L_{r,f}\in\mathscr{L}((r-2)\text{-LUSCG}(f))-\mathscr{L}((r-3)\text{-LUSCG}(f))$.

Proof. A grammar in $(r-2)$-LUSCG(f) that derives $L_{r,f}$ has been presented in Example 2. To prove the statement, we show that the assumption of the existence of $G\in(r-3)$-LUSCG(f) such that $L(G)=L_{r,f}$ leads to a contradiction.

Let Δ_G be the maximum number of terminal symbols in the right-hand tuple of a production of G. Let w be a sentence in $L_{r,f}$ such that $\#_a(w)>(r-3)\cdot M_G+\Delta_G$ for every $a\in V_T^{(r,f)}$, and let also η be a derivation in G for w. Let p_1 be the first production used in η, i.e., η has the form $(S,I^{\langle S\rangle})\overset{p_1}{\Rightarrow}_G(\alpha,I')\overset{*}{\Rightarrow}_G(w,\emptyset)$. S is a p_1-factorized sentential form and, by the choice of w, p_1 covers $\mathscr{B}^{(r,f)}$. Let $p_{1,1},\ldots,p_{1,k_1}$, $1\leqslant k_1\leqslant r-3$, be the sequence of productions used in η to rewrite the right-hand tuple of p_1. (Hence, for some string ξ which is a permutation of $p_{1,1},\ldots,p_{1,k_1}$, $p\to\xi$ is a production in Π of der(G).) If among these productions there are some that cover $\mathscr{B}^{(r,f)}$, we arbitrarily choose one and call it p_2. We iterate the step until either we arrive at a production p_l, $l\geqslant 1$, whose right-hand tuple contains only terminal symbols, or we arrive at some production p_l, $l\geqslant 1$, used in η such that p_l covers $\mathscr{B}^{(r,f)}$ and none of the productions that are used in η to rewrite the right-hand tuple of p_l covers $\mathscr{B}^{(r,f)}$ (see Fig. 6).

Let $(A_1,\ldots,A_t)$, $t\leqslant f$, be the left-hand tuple of p_l, let π be some permutation of $\{1,\ldots,t\}$ and let $w=u_0v_{\pi(1)}u_1\cdots u_{t-1}v_{\pi(t)}u_t$, where v_i is the substring derived

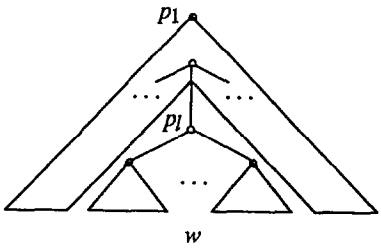


Fig. 6. A derivation of w in G, represented as a derivation tree in der(G). The part of the tree above production p_l cannot generate all terminal symbols of any of the terminal groups in $\mathscr{B}^{(r,f)}$. By the construction of w, these symbols must therefore be covered by the productions that are the daughters of p_l.

under η by the nonterminal A_i in the left-hand tuple of p_l, $1 \leqslant i \leqslant t$. Then string $u_0 A_{\pi(1)} u_1 \cdots u_{t-1} A_{\pi(t)} u_t$ is a p_l-factorized sentential form. By Lemma 6 we have that, for every terminal group $\overline{a_s} \in \mathscr{B}^{(r,f)}$, there exists a terminal a_{s,q_s} that is not contained within string $u_0 u_1 \cdots u_t$. Hence, more than $(r-3) \cdot M_G + \Delta_G$ instances of each a_{s,q_s}, $2 \leqslant s \leqslant r-1$, are generated under η from the nonterminals in the left-hand tuple of p_l. Thus, the right-hand tuple of p_l cannot contain only terminal symbols. Now let $p_{l,1}, \ldots, p_{l,k_l}$, $1 \leqslant k_l \leqslant r-3$, be the sequence of productions used in η to rewrite the right-hand tuple of p_l. (Again, for some string ξ which is a permutation of $p_{l,1}, \ldots, p_{l,k_l}$, $p_l \to \xi$ is a production in Π of der(G).) The right-hand tuple of p_l itself cannot contain more than Δ_G instances of each a_{s,q_s}, and therefore $p_{l,1}, \ldots, p_{l,k_l}$ must generate more than $(r-3) \cdot M_G$ instances of each a_{s,q_s}. Since $k_l \leqslant r-3$, by a counting argument we conclude that for each s, $2 \leqslant s \leqslant r-1$, there must be at least one $p_{l,i}$, $1 \leqslant i \leqslant k_l$, such that $p_{l,i}$ generates under η more than M_G instances of a_{s,q_s}. Again by a counting argument, we derive that at least one $p_{l,i}$, $1 \leqslant i \leqslant k_l$, generates more than M_G instances of two symbols a_{s,q_s} and $a_{s',q_{s'}}$, $s \neq s'$. Hence $p_{l,i} \lhd \{a_{s,q_s}, a_{s',q_{s'}}\}$. Then we have that $p_{l,i}$ covers two terminal groups in $\mathscr{B}^{(r,f)}$, by Lemma 2, and $p_{l,i}$ covers $\mathscr{B}^{(r,f)}$ by Lemma 5. This contradicts the choice of production p_l: we conclude that there can be no derivation in G for w, that is, grammar G does not exist. □

We now turn to subclasses 2-LUSCG(f) and 3-LUSCG(f), $f \geqslant 2$. We first show that for $f=2$, they collapse.

Theorem 2. $\mathscr{L}$(2-LUSCG(2)) = $\mathscr{L}$(3-LUSCG(2)).

Proof. We show how to convert a grammar $G \in$ 3-LUSCG(2) − 2-LUSCG(2) into $G' \in$ 2-LUSCG(2) such that $L(G) = L(G')$. Let p be a production of G of rank three. Assume first that $\varphi(p) = 2$ and let p be of the form $(A_1, A_2) \to (\alpha_1, \alpha_2)$ with $\alpha_1 = u_0 B_1 u_1 \cdots u_{l-1} B_l u_l$ and $\alpha_2 = v_0 C_1 v_1 \cdots v_{r-1} C_r v_r$, where $B_i, C_i \in V_N$, $u_i, v_i \in V_T^*$, and where l, r are nonnegative integers with $3 \leqslant l + r \leqslant 6$. Let $\Pi_p = \{p_1, p_2, p_3\}$ be any multiset of productions of G that rewrites the right-hand tuple of p. We distinguish three cases.

Case 1: $l,r>1$. By a counting argument, there must be $p_h \in \Pi_p$ such that $\varphi(p_h)=2$ and p_h rewrites two nonterminals among B_1, B_l, C_1 and C_r. Assume p_h rewrites B_l and C_1. For new nonterminal symbols $[p, p_h, 1]$ and $[p, p_h, 2]$, construct productions

$$(A_1, A_2) \to ([p, p_h, 1]B_l u_l,\ \ v_0 C_1 [p, p_h, 2]),$$
$$([p, p_h, 1], [p, p_h, 2]) \to (u_0 B_1 \cdots B_{l-1} u_{l-1},\ \ v_1 C_2 \cdots C_r v_r) \tag{5}$$

to be used by G'. By assumption, the productions in (5) have rank not greater than two. The remaining cases for p_h are handled in a similar way.

Case 2: $l=1$ or $r=1$. Assume $l=1$. Choose production $p_h \in \Pi_p$ such that p_h rewrites B_1. If $\varphi(p_h)=2$, p_h also rewrites nonterminal C_q for some $1 \leqslant q \leqslant r$. If $q \notin \{1, r\}$, then for new nonterminal symbols $[p, p_h, 1]$ and $[p, p_h, 2]$ construct productions

$$(A_1, A_2) \to (u_0 B_1 u_1,\ \ [p, p_h, 1] C_q [p, p_h, 2]),$$
$$([p, p_h, 1], [p, p_h, 2]) \to (v_0 C_1 \cdots C_{q-1} v_{q-1},\ \ v_q C_{q+1} \cdots C_r v_r). \tag{6}$$

Productions in (6) have rank not greater than two. If $q=1$, $q=r$ or $\varphi(p_h)=1$, we have subcases that can be handled with just one new nonterminal.

Case 3: $l=0$ or $r=0$. Assume $l=0$. Choose production $p_h \in \Pi_p$ such that p_h rewrites C_1. If $\varphi(p_h)=2$, p_h also rewrites nonterminal C_q for some $2 \leqslant q \leqslant r$. If $q \notin \{2, r\}$, then for new nonterminal symbols $[p, p_h, 1]$ and $[p, p_h, 2]$ construct productions

$$(A_1, A_2) \to (u_0,\ \ v_0 C_1 [p, p_h, 1] C_q [p, p_h, 2]),$$
$$([p, p_h, 1], [p, p_h, 2]) \to (v_1 C_2 \cdots C_{q-1} v_{q-1},\ \ v_q C_{q+1} \cdots C_r v_r). \tag{7}$$

Again, productions in (7) have rank not greater than two. If $q=2$, $q=r$ or $\varphi(p_h)=1$, we have two subcases that can be handled with just one new nonterminal. This exhausts all cases in which $\varphi(p)=2$.

Finally, if p is of the form $(A) \to (\alpha)$, we can proceed as in Case 3 above. □

Next we will show that for any integer $f \geqslant 3$, the class 2-LUSCG(f) is properly included in 3-LUSCG(f). The family of languages $L_{5,f}$ studied at the beginning of this section cannot be used in order to prove this separation result, and we have to define new languages to which we will restrict our attention in what follows.

Definition 7. Let f be an integer, $f \geqslant 3$, and let $V_T^{(f)} = \{a_{1,h}, a_{2,h}, a_{3,h}, a_{4,h}, a_{5,h} \mid 1 \leqslant h \leqslant f\}$. Language Q_f is specified as follows:

$$Q_f = \{w_1 w_2 \cdots w_f \mid w_1 = a_{1,1}^{i_1} a_{2,1}^{i_2} a_{3,1}^{i_3} a_{4,1}^{i_4} a_{5,1}^{i_5},\ \ w_2 = a_{1,2}^{i_1} a_{3,2}^{i_3} a_{2,2}^{i_2} a_{4,2}^{i_4} a_{5,2}^{i_5},$$
$$w_h = a_{1,h}^{i_1} a_{2,h}^{i_2} a_{4,h}^{i_4} a_{3,h}^{i_3} a_{5,h}^{i_5},\ \ 3 \leqslant h \leqslant f, i_j \geqslant 1,\ \ 1 \leqslant j \leqslant 5\}.$$

As in the case of languages $L_{r,f}$, we will call *segment* each substring w_h, $1 \leqslant h \leqslant f$, in the definition of a string in Q_f. We will also use the terminal group notation

$\overline{a_s} = \{a_{s,1}, a_{s,2} \ldots, a_{s,f}\}$, $1 \leqslant s \leqslant 5$. Finally, the set $\{\overline{a_2}, \overline{a_3}, \overline{a_4}\}$ will be denoted $\mathscr{B}^{(f)}$. We now study some properties that are common to all grammars in LUSCG that derive languages Q_f. (Henceforth, we will always assume $f \geqslant 3$.) In what follows, there is a strong similarity with the properties of languages $L_{r,f}$ that have been investigated so far; for this reason, sometimes proofs will be omitted; in the remaining cases, our arguments will be simpler than those used for languages $L_{r,f}$, due to the fact that languages Q_f depend upon only one parameter.

Assume that $G \in$ LUSCG is a grammar of fan-out f deriving some language Q_f. We start with three properties of G that correspond to Lemmas 2–4. Let a and b be two symbols in $V_T^{(f)}$. For any string w in Q_f, the set of all terminal symbols occurring between a and b in w (including a and b) is always the same. Again this set will be called the *in-between set* of a and b.

Lemma 7. *For every production p of G, the following statements hold:*

(i) *if p covers some $\mathscr{A} \subseteq V_T^{(f)}$ then p covers $\{\overline{a_s} \mid a_{s,q} \in \mathscr{A}\}$;*

(ii) *if a nonterminal symbol A in the left-hand tuple of p covers (by means of p) the set $\{a, b\}$, then p covers the in-between set of a and b;*

(iii) *if a nonterminal symbol A in the left-hand tuple of p covers (by means of p) the set $\{a_{s,q}, a_{s,q'}\}$, $q \neq q'$, then p covers $V_T^{(f)}$.*

Proof. Statements (i) and (ii) can be proved using the same arguments found in the proofs of Lemmas 2 and 3. To prove statement (iii), observe from the definition of Q_f that $a_{5,q}$ and $a_{1,q+1}$ are in the in-between set of $a_{s,q}$ and $a_{s,q'}$. We then proceed exactly as in the proof of Lemma 4, taking $r = 5$. □

We now derive a basic property of productions in G. What follows is the analogue of Lemma 5 for languages Q_f.

Lemma 8. *If a production p of G covers more than one terminal group in $\mathscr{B}^{(f)}$, then p covers $\mathscr{B}^{(f)}$.*

Proof. Let p cover groups $\overline{a_s}$ and $\overline{a_{s'}}$ in $\mathscr{B}^{(f)}$, $s < s'$; assume also that $(A_1, \ldots, A_t)$, $1 \leqslant t \leqslant f$, is the left-hand tuple of p. As in the proof of Lemma 5 we may restrict ourselves to the case that $t = f$ and $A_i \lhd \{a_{s,i}, a_{s',i}\}$ for $1 \leqslant i \leqslant f$. There is a finite number of cases for the pair s, s': from the definition of Q_f we see that in all cases a terminal symbol $a_{s'',i}$ is included in the in-between set of $a_{s,i}$ and $a_{s',i}$ for some i, where $\overline{a_{s''}} \in \mathscr{B}^{(f)}$ and $s'' \notin \{s, s'\}$. By Lemma 7, p must cover $\{a_{s,i}, a_{s',i}, a_{s'',i}\}$ and therefore $\mathscr{B}^{(f)}$. □

The notion of p-factorized sentential form for a production p and the associated notion of isolated symbol have been introduced in the discussion preceding Lemma 6. These notions will also be used in the following statement, which represents for languages Q_f the analogue of Lemma 6.

Lemma 9. *Let p be a production of G such that $p \lhd \mathscr{B}^{(f)}$ and let $u_0 A_1 u_1 \cdots u_{t-1} A_t u_t$, $t \leqslant f$, be a p-factorized sentential form. Then for every terminal group $\overline{a_s} \in \mathscr{B}^{(f)}$ there exists a terminal symbol $a \in \overline{a_s}$ such that a is not found in the string $u_0 u_1 \cdots u_t$.*

Proof. We assume that $u_0 u_1 \cdots u_t$ contains instances of every terminal symbol in some $\overline{a_s} \in \mathscr{B}^{(f)}$ and derive a contradiction. First, we claim that no u_i, $0 \leqslant i \leqslant t$, can contain two different terminals from $\overline{a_s}$. Assume the contrary. From the definition of Q_f it can be seen that at least one terminal from some $\overline{a_{s'}} \in \mathscr{B}^{(f)}$, $s' \neq s$, will be isolated in the p-factorized sentential form. Then p could not generate it, contrary to the hypotheses. To prove the lemma, we then proceed by distinguishing three cases.

Case 1: $s = 4$. If any $a_{4,q} \in \overline{a_4}$ is included in u_0, then $a_{2,1}$ to its left will be isolated and p could not generate $a_{2,1}$, contrary to the hypotheses. A similar argument applies if any $a_{4,q}$ is included in u_t. Since $t \leqslant f$, we conclude that there is some u_i, $1 \leqslant i \leqslant t-1$, which contains at least two different terminals from $\overline{a_4}$. But this contradicts the above claim.

Case 2: $s = 2$. If any $a_{2,q} \in \overline{a_2}$ is contained in u_t, then $a_{3,f}$ to its right will be isolated and p could not generate such a symbol. If $t < f$ we establish a contradiction using again the claim above. Assume therefore $t = f$ and each $a_{2,q}$ is contained in u_{q-1} for $1 \leqslant q \leqslant f$. Since p covers $\mathscr{B}^{(f)}$, by Lemma 1, $a_{3,1}$ and $a_{3,2}$ must be covered by some nonterminals in the left-hand tuple of p. Since u_0 contains $a_{2,1}$, u_1 contains $a_{2,2}$ and symbols $a_{3,1}$ and $a_{3,2}$ occur between symbols $a_{2,1}$ and $a_{2,2}$ in strings in Q_f, no nonterminal other than A_1 in the left-hand tuple of p can cover $a_{3,1}$ and $a_{3,2}$. We conclude that A_1 must cover $\{a_{3,1}, a_{3,2}\}$. By statement (iii) of Lemma 7, p covers $V_T^{(f)}$ and then $\overline{a_1}$. But this is impossible, since u_0 must contain at least one instance of $a_{1,1}$ to the left of $a_{2,1}$, which is therefore isolated.

Case 3: $s = 3$. If any $a_{3,q} \in \overline{a_3}$ is contained within u_0, then $a_{2,1}$ to its left will be isolated and p could not generate such a symbol. Again we deal with the case $t = f$ and $a_{3,q}$ in u_q for $1 \leqslant q \leqslant f$. With an argument similar to Case 2, we can argue that p covers $\overline{a_5}$. Again this is not possible, because u_f must contain at least one instance of $a_{5,f}$ to the right of $a_{3,f}$, which is therefore isolated. □

The technique used in the proof of Theorem 1 along with the above lemmas can be used to show the following result. The proof is omitted because of its strong similarity with the one of Theorem 1.

Theorem 3. *Let f be an integer, $f \geqslant 3$. Then we have $Q_f \in \mathscr{L}(3\text{-LUSCG}(f)) - \mathscr{L}(2\text{-LUSCG}(f))$.*

To conclude this section and to complete our picture of the rank hierarchy for fixed values of the fan-out parameter, we give a last result that compares the subclasses of LUSCG of ranks one and two. The proof of the result, however, must be deferred to Section 5, where we will use results of Section 4 along with an equivalence result that allows us to transfer to LUSCG some facts that are already known for the class of finite copying parallel rewriting systems.

Theorem 4. *Let* $f \geqslant 1$. *Then* $\mathscr{L}(1\text{-LUSCG}(f))$ *is properly included in* $\mathscr{L}(2\text{-LUSCG}(f))$.

Section 5 will complete our investigation of the interaction between the fan-out and rank complexity measures by transferring the rank hierarchy results of this section to parallel rewriting systems and combining them with a fan-out hierarchy result that is well known for the latter class.

4. Closure properties

This section investigates some language-theoretic properties of classes r-LUSCG(f), $r, f \geqslant 1$. We will use these results in Section 5.

A family of languages is called an *abstract family of languages* (for short AFL), if it is closed under union, concatenation, ε-free Kleene closure, ε-free homomorphism, inverse homomorphism and intersection with regular languages. A full AFL is an AFL which is also closed under arbitrary homomorphism.

Theorem 5. *For integers* $r \geqslant 2$ *and* $f \geqslant 1$, $\mathscr{L}(r\text{-LUSCG}(f))$ *is a substitution-closed full* AFL.

Proof. Since for $r \geqslant 2$ and $f \geqslant 1$, r-LUSCG(f) contains all regular languages, it is sufficient to show closure under substitution and under intersection with regular languages [20, p. 126].

To show closure under substitution, consider a grammar $G = (V_N, V_T, P, S)$ in r-LUSCG(f). For each a in V_T, let $L^{(a)}$ be a language in $\mathscr{L}(r\text{-LUSCG}(f))$ and let $G^{(a)} = (V_N^{(a)}, V_T^{(a)}, P^{(a)}, S^{(a)})$ be a grammar in r-LUSCG(f) that generates $L^{(a)}$. We assume that $V_N^{(a)}$ and $V_N^{(b)}$ are disjoint for all $a, b \in V_T$, $a \neq b$, and that $V_N^{(a)}$ and V_N are disjoint for all $a \in V_T$. We construct a new grammar

$$G' = \left(V_N' \cup \bigcup_{a \in V_T} V_N^{(a)}, \bigcup_{a \in V_T} V_T^{(a)}, P' \cup \bigcup_{a \in V_T} P^{(a)}, [S] \right),$$

where V_N' and P' are defined as follows. Let δ_G be the length of the longest sequence of consecutive terminal symbols introduced by a rule in P. Let

$$V_N' = \{[uAv], [uAv, p] \mid A \in V_N, u, v \in V_T^*, \ |u|, |v| \leqslant \delta_G \text{ and } p \in P\}.$$

To construct P', let p be a production in P of the form $(A_1, \ldots, A_t) \to (\alpha_1, \ldots, \alpha_t)$, where $1 \leqslant t \leqslant f$ and for each k, $1 \leqslant k \leqslant t$, we have $\alpha_k = u_{k,0} B_{k,1} u_{k,1} \cdots u_{k,l_k-1} B_{k,l_k} u_{k,l_k}$, $l_k \geqslant 0$, $B_{k,i} \in V_N$ for $1 \leqslant i \leqslant l_k$ and $u_{k,j} \in V_T^*$ for $0 \leqslant j \leqslant l_k$. For every tuple $\tau = (u_1, v_1, \ldots, u_t, v_t)$ such that $u_i, v_i \in V_T^*$ and $|u_i|, |v_i| \leqslant \delta_G$, $1 \leqslant i \leqslant t$, we add to P' the production

$$p_\tau : ([u_1 A_1 v_1], \ldots, [u_t A_t v_t]) \to ([u_1 A_1 v_1, p], \ldots, [u_t A_t v_t, p]).$$

Furthermore, for every τ as above and for every $a \in V_T$ and $1 \leqslant h \leqslant t$, we add to P' the production

$$p_{\tau,h,a} : ([A_1, p], \ldots, [A_{h-1}, p], [au_h A_h v_h, p], \ldots, [u_t A_t v_t, p])$$
$$\rightarrow ([A_1, p], \ldots, [A_{h-1}, p], S^{(a)}[u_h A_h v_h, p], \ldots, [u_t A_t v_t, p])$$

and the production

$$p'_{\tau,h,a} : ([A_1, p], \ldots, [A_{h-1}, p], [u_h A_h v_h a, p], \ldots, [u_t A_t v_t, p])$$
$$\rightarrow ([A_1, p], \ldots, [A_{h-1}, p], [u_h A_h v_h, p] S^{(a)}, \ldots, [u_t A_t v_t, p]).$$

Finally, we add to P' the production

$$p' : ([A_1, p], \ldots, [A_t, p]) \rightarrow (\alpha'_1, \ldots, \alpha'_t),$$

where $\alpha'_k = [u_{k,0} B_{k,1}][u_{k,1} B_{k,2}] \ldots [u_{l_k - 1} B_{k,l_k} u_{k,l_k}]$, $1 \leqslant k \leqslant t$. The construction is carried out for every p in P.

It is straightforward to show that G' generates the language whose strings are obtained from strings in $L(G)$ by replacing each $a \in V_T$ with some string in $L^{(a)}$; we omit the details. We have $\rho(p') = \rho(p)$, $\rho(p_\tau) = 1$ and $\rho(p_{\tau,h,a}) = \rho(p'_{\tau,h,a}) = 2$, for every p in P and every choice of τ, h and a. Thus, $\rho(G') = \rho(G)$. Since all productions in P' derived from p in P preserve the fan-out of p, we have $G' \in r\text{-LUSCG}(f)$. This proves closure under substitution.

As far as intersection with regular languages is concerned, we anticipate here some of the contents of the next section (Theorem 6), where an equivalence result is presented between classes r-LUSCG(f) and classes r-MCFG(f) studied in [22]. In [22, Theorem 3.9] it is shown that, for every $f \geqslant 1$, $\bigcup_{r \geqslant 1} r\text{-MCFG}(f)$ is closed under intersection with regular languages; their proof preserves parameter r. Hence our result follows from Theorem 6. □

We obtain the following two corollaries, the first of which was proven (more simply) in [9, Theorem 5.1] (for finite copying top-down tree-to-string transducers) and in [22, Theorem 3.9] (for multiple context-free grammars).

Corollary 1. *For $f \geqslant 1$, $\mathscr{L}$(LUSCG(f)) is a substitution-closed full* AFL.

Corollary 2. *For $r \geqslant 2$, $\mathscr{L}(r\text{-LUSCG})$ is a substitution-closed full* AFL.

5. Implications for parallel rewriting systems

In this section we provide an overview over some classes of finite copying parallel rewriting systems that have been defined in the literature. We start by proving a generative equivalence relation between these formalisms and the class LUSCG. The

importance of such a result is that it provides an original characterization of finite copying parallel rewriting systems in terms of the locality restriction that was introduced in Section 2. At the same time, the equivalence result maps the fan-out and rank parameters defined for LUSCG into synchronized parallelism and independent parallelism, respectively, as defined for the parallel rewriting systems we consider here. In this way we can transfer the results presented so far and show how independent parallelism induces an infinite non-collapsing hierarchy in parallel rewriting systems with degree of synchronized parallelism bounded by a constant greater than one. This hierarchy is orthogonal to the non-collapsing hierarchy induced by synchronized parallelism that has been previously discussed in the literature. The hierarchy result, combined with the results of Section 4, also provides an answer to a question that was left open in the literature. In what follows, we will use the term *rank* of a context-free grammar to refer to the greatest number of nonterminal symbols that can be found in the right-hand side of the productions of the grammar.

We start by relating the class LUSCG to a class of rewriting systems known as multiple context-free grammars (MCFG) introduced in [14, 22]. For notational convenience, we present MCFG through a notational variant of this class that in [26] is called *string-based linear context-free rewriting system*. This variant requires the "information-lossless" condition (see [22]) while MCFG does not. However, Seki et al. [22] show that this does not affect the generative power of the class (their Lemma 2.2).[1] We discuss the relationship between LUSCG and MCFG in some detail, since existing results will then allow us to relate LUSCG to other known formalisms as well.

Let V_T be an alphabet of terminal symbols; in the following we will be interested in functions mapping tuples of strings in V_T^* into tuples of strings in V_T^*. For integers r and f, $r \geqslant 0$ and $f \geqslant 1$, we say that g is an r-ary function if there exist integers $f_i \geqslant 1$, $1 \leqslant i \leqslant r$, such that g is defined on $(V_T^*)^{f_1} \times (V_T^*)^{f_2} \times \cdots \times (V_T^*)^{f_r}$; we say that g has fan-out f if the range of g is a subset of $(V_T^*)^f$. Let y_h, x_{ij}, $1 \leqslant h \leqslant f$, $1 \leqslant i \leqslant r$ and $1 \leqslant j \leqslant f_i$, be string-valued variables. A function g as above is said to be *linear regular* if it is defined by an equation of the form

$$g(\langle x_{1,1}, \ldots, x_{1,f_1} \rangle, \ldots, \langle x_{r,1}, \ldots, x_{r,f_r} \rangle) = \langle y_1, \ldots, y_f \rangle, \tag{8}$$

where $\langle y_1, \ldots, y_f \rangle$ represents some grouping into f sequences of all and only the variables appearing in the left-hand side of (8) (without repetitions) along with some additional terminal symbols (with possible repetitions). The following definition is based on [26, p. 137] and [22, p. 196], and can easily seen to be a notational variant of either.

[1] String-based linear context-free rewriting system is a member of the family of linear context-free rewriting system (LCFRS) introduced in [24, 25] independently of MCFG. This family groups together a large class of rewriting systems that operate on different types of objects, such as strings, tuples of strings, trees, graphs, and so on. The result of rewriting is then associated with terminal strings by "yield functions", in order to generate string languages.

$$
\begin{aligned}
G'_L &= (V_N, V_T, S, P);\\
V_N &= \{S, A, B\}\\
V_T &= \{[,]\}\\
P &= \{p_1: \quad S \to g(A),\\
&\qquad p_2: \quad A \to f_1(),\\
&\qquad p_3: \quad A \to f_2(A),\\
&\qquad p_4: \quad A \to f_3(A,A)\ \}\\
g(\langle x_{11}, x_{12}\rangle) &= \langle x_{11}x_{12}\rangle,\\
f_1() &= \langle [\,],[\,]\rangle,\\
f_2(\langle x_{11}, x_{12}\rangle) &= \langle [x_{11}],[x_{12}]\rangle,\\
f_3(\langle x_{11}, x_{12}\rangle, \langle x_{21}, x_{22}\rangle) &= \langle x_{11}x_{21}, x_{12}x_{22}\rangle.
\end{aligned}
$$

Fig. 7. A multiple context-free grammar for language $L = \{ww \mid w \in D_1\}$.

Definition 8. A *multiple context-free grammar* (MCFG) is a quadruple $G = (V_N, V_T, P, S)$ where V_N and V_T are defined as for an unordered scattered context grammar, every symbol $A \in V_N$ is associated with an integer $\varphi(A) \geqslant 1$, S is a symbol in V_N such that $\varphi(S) = 1$, and P is a finite set of productions of the form $p: A \to g(B_1, B_2, \ldots, B_{\rho(p)})$, where $\rho(p) \geqslant 0$, $A, B_i \in V_N$, $1 \leqslant i \leqslant \rho(p)$ and where g is a linear regular function having arity $\rho(p)$ and fan-out $\varphi(A)$, defined on $(V_T^*)^{\varphi(B_1)} \times \cdots \times (V_T^*)^{\varphi(B_{\rho(p)})}$.

For every $A \in V_N$ and $t \in (V_T^*)^{\varphi(A)}$, we write $A \Rightarrow_G t$, if one of the following conditions is met:

(i) $A \to g() \in P$ and $g() = t$;

(ii) $A \to g(B_1, \ldots, B_{\rho(p)}) \in P$, $B_i \Rightarrow_G t_i$ for every $1 \leqslant i \leqslant \rho(p)$, where $t_i \in (V_T^*)^{\varphi(B_i)}$, and $g(t_1, \ldots, t_{\rho(p)}) = t$.

We emphasize that in MCFG the rewrite relation always relates a single nonterminal symbol to a tuple of terminal strings. For $A \in V_N$, we call $\varphi(A)$ the *fan-out* of A; for $p \in P$, we call $\rho(p)$ the *rank* of p and we write $\varphi(p) = \varphi(A)$ whenever A is the left-hand side symbol of p. For $G \in$ MCFG, we define $\varphi(G) = \max_{A \in V_N} \varphi(A)$ and $\rho(G) = \max_{p \in P} \rho(p)$. For $r \geqslant 0$ and $f \geqslant 1$, the class of all linear context-free rewriting systems with rank bounded by r and fan-out bounded by f is denoted r-MCFG(f).[2] The language derived by G is the set of strings $L(G) = \{w \mid S \Rightarrow_G \langle w\rangle\}$.

Example 3. Let L be the language considered in Example 1. A grammar $G'_L \in$ 2-MCFG(2) that generates L is defined in Fig. 7. Observe that nonterminal A generates all tuples of the form $\langle w, w\rangle$, with $w \in D_1$.

The following theorem establishes a strong (rank- and fan-out-preserving) equivalence relation between LUSCG and MCFG. The proof is conceptually straightforward but notationally complex; we defer it to the appendix.

[2] Seki et al. [22] use the notation f-MCFG to refer to MCFG of fan-out f, while we use r-MCFG to refer to MCFG of rank r.

Theorem 6. *Let r, f be integers such that $r, f \geqslant 1$. Then we have $\mathscr{L}(r\text{-MCFG}(f)) = \mathscr{L}(r\text{-LUSCG}(f))$.*

We can immediately obtain the following rank hierarchy result for MCFG.

Theorem 7. *For each $f \geqslant 2$, the rank parameter induces a non-collapsing hierarchy in class* MCFG(f).

Proof. The statement directly follows from our main result and from Theorem 6. □

Next we switch to other finite copying parallel rewriting systems that have been defined in the literature, and use Theorem 6 to transfer our main result to these formalisms. Deterministic tree-walking transducers (DTWT) were introduced by Aho and Ullman [2] (called TAT there). A DTWT is an automaton with a finite state control, that visits in checking mode an input tree generated by a context-free grammar and outputs a translation string. Since this (sequential) device can visit a given subtree more than once, the output tree will contain separated substrings that are "homomorphic" to (a string representation of) that structure. Two complexity measures can be defined for the class DTWT, usually called the *crossing number* and the *rank*. The crossing number of a DTWT represents the maximum number of times the automaton crosses (enters and exits) any subtree in the input tree language; because of the determinism, this number is always finite (see [2]). The rank of a DTWT is the rank of the context-free grammar that generates the input language and is finite by definition. For $f \geqslant 1$ and $r \geqslant 0$, let us denote by r-DTWT(f) the subclass of all DTWT with crossing number bounded by f and rank bounded by r. If we regard DTWT as generative devices controlled by some tree language, we have the following result.

Theorem 8. *For each $f \geqslant 2$, the rank parameter induces a non-collapsing hierarchy in class* DTWT(f).

Proof. In [26] it is shown that, for every $f \geqslant 1$, MCFG(f) has the same generative power as DTWT(f). The proof preserves the rank parameter. The claim then immediately follows from our main result and Theorem 6. □

Note that in [2, p. 473] the authors mention the existence of an analogue of the Chomsky normal form for the class DTWT, that is, the language produced by a DTWT of any rank can also be obtained by some DTWT of rank two. We remark that this does not contradict Theorem 8, since the conversion into the normal form increases the crossing number. In Theorem 11 below we give a formal proof of their statement, by showing that if the rank is greater than two, it is always possible to decrease the rank at the expense of increasing the crossing number.

Top-down tree-to-string transducers (yT) have been introduced in [9] as a model of the generalized syntax-directed translation (GSDT) of Aho and Ullman [2] and, in case the degree of independent parallelism is bounded by one, as a model of the

controlled ET0L systems of Rozenberg [18]. These parallel rewriting devices take a tree as input, and convert it through a series of rewrite steps into a string. Each rewrite step consumes the root node of a tree in the sentential form, and rearranges the subtrees that are immediately dominated by this node, interleaving them with terminal strings; these subtrees may also be copied. Rewriting is controlled by states which are explicitly represented in the sentential form.

In what follows, we regard yT as a class of generative devices controlled by the family of tree languages that can be generated by context-free grammars. With this assumption, two parameters can be defined for these systems. If in a derivation the number of copies of a subtree of the input that a tree-to-string transducer can generate is finite, we say that the transducer has *finite copying* degree. Furthermore, the *rank* of a transducer is the rank of the context-free grammar that generates the controlling tree language. We denote as $r\text{-}yT_{\mathrm{fc}(f)}$ the class of all devices in yT with finite copying degree bounded by f and rank bounded by r, $f, r \geqslant 1$. The following result can now be easily established.

Theorem 9. *For each $f \geqslant 2$, the rank parameter induces a non-collapsing hierarchy in class $yT_{\mathrm{fc}(f)}$.*

Proof. In [9, Theorem 4.9] it is shown that, for each $f \geqslant 1$, $\mathscr{L}(yT_{\mathrm{fc}(f)}) = \mathscr{L}(\mathrm{DTWT}(f))$ (the result is achieved using a model called deterministic checking tree transducer). The proof preserves the rank parameter for both classes. Hence the statement follows from Theorem 8. □

The class of ET0L systems of finite index ($\mathrm{ET0L_{FIN}}$) was introduced by Rozenberg and Vermeir [19] and Latteux [15]. In [9, Theorem 3.2.2] it is shown that, for each $f \geqslant 1$, the family of languages generated by $\mathrm{ET0L_{FIN}}(f)$ and $1\text{-}yT_{\mathrm{fc}(f)}$ are the same. This gives us the following corollary.

Corollary 3. *For every integer $f \geqslant 1$, $1\text{-LUSCG}(f) = \mathrm{ET0L_{FIN}}(f)$.*

Using this result, we can now supply the missing proof for Theorem 4 in Section 3, whose statement is repeated here.

Theorem 4. *Let $f \geqslant 1$. Then $\mathscr{L}(1\text{-LUSCG}(f))$ is properly included in $\mathscr{L}(2\text{-LUSCG}(f))$.*

Proof. Inclusion holds trivially. We have seen that, for every $f \geqslant 1$, $\mathscr{L}(2\text{-LUSCG}(f))$ is a substitution-closed AFL (Theorem 5), and thus closed under concatenation. In [15] it is shown that $\mathscr{L}(\mathrm{ET0L_{FIN}}(f))$ is not closed under concatenation. Properness of the inclusion follows then from Corollary 3. □

An alternative proof of the above result can be obtained using the well known fact that there exists a context-free language that is not contained in any of the

subclasses $\mathscr{L}(\mathrm{ET0L}_{\mathrm{FIN}(f)})$, $f \geqslant 1$ (see [9]). But we have already observed that the subclass 2-LUSCG(1) generates all and only the context-free languages.

We can also answer an open question raised in [9, p. 189], about whether family $\mathscr{L}(yT_{\mathrm{fc}(f)})$ is full principal for each $f \geqslant 2$.[3]

Corollary 4. *For each $f \geqslant 2$, $\mathscr{L}(yT_{\mathrm{fc}(f)})$ is not full principal.*

Proof. This follows immediately from the fact that, for every $r \geqslant 2$ and for every $f \geqslant 2$, $\mathscr{L}(r\text{-LUSCG}(f))$ is a full AFL (Theorem 5) and from our rank hierarchy result. □

Context-free hypergraph grammars (CFHG) are rewriting systems that derive sets of edge-labeled hypergraphs; these systems were introduced as a generalization of edge rewriting graph grammars (see, for instance, [4]). In a CFHG, each production specifies some replacement of a labeled hyperedge with a hypergraph, along with particular conditions that allow the replacing hypergraph to be embedded within the host hypergraph. In this way a derivation proceeds sequentially, by replacing hyperedges in a sentential hypergraph; but due to the many tentacles associated with each hyperedge, the derivation can mimic some sort of parallelism. In CFHG, edge rewriting is performed in a context-free fashion, so that the locality restriction is observed. It turns out that each derivation can be associated with an underlying tree structure that can be generated by a context-free grammar (see [8]). Two independent parameters can be identified for a CFHG. The first one is the maximum *number of tentacles* associated with a hyperedge in the grammar. The second parameter is the *rank* of the underlying context-free grammar associated with the CFHG. For integers $r, f \geqslant 1$, we denote by r-CFHG(f) the subclass of all context-free hypergraph grammars with rank bounded by r and maximum number of tentacles bounded by f.

If we restrict our attention to CFHG generating string languages, that is chain-like hypergraphs, we find the same generative power as the class DTWT, as shown in [8], thus relating CFHG to parallel rewriting systems. We can use their result to transfer our rank hierarchy to CFHG.

Theorem 10. *For each $f \geqslant 1$, the rank parameter induces a non-collapsing hierarchy in class* CFHG(f).

Proof. In [8] it is shown that, for each $f \geqslant 1$, classes DTWT(f) and (string language generating) CFHG($2f-1$) $\cup$ CFHG($2f$) generate the same family of languages. The proof fails to preserve the rank parameter only in their Lemma 5.3 (p. 349). However, in the proof of Theorem 6.5 (p. 356) the authors provide an alternative proof of Lemma 5.3 which is in fact rank-preserving. The result then directly follows from our main result and Theorem 8. □

[3] We are grateful to Joost Engelfriet for drawing our attention to the relevance of our result to this issue.

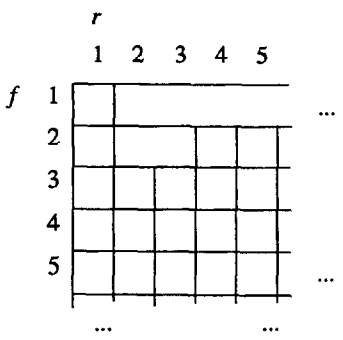


Fig. 8. Inclusion relations of languages generated by r-LUSCG(f). The rank parameter corresponds to columns, the fan-out parameter corresponds to rows in the array. Proper inclusion between adjacent entries is indicated by a separation line; no separation line signifies equality.

To conclude the present section, we combine our rank hierarchy result with well-known facts about parallel rewriting systems, in order to investigate how synchronized parallelism and independent parallelism interact. In [9] it is observed that each subclass r-$yT_{\mathrm{fc}(1)}$, $r \geqslant 2$, generates all and only the context-free languages, while subclass 1-$yT_{\mathrm{fc}(1)}$ generates all and only the linear context-free languages. It is well known that the family of linear context-free languages is strictly included in the family of context-free languages (see, for instance, [12]). In [9, Theorem 3.2.5] it is also shown that, for each $f \geqslant 1$, there exists a language generated by subclass 1-$yT_{\mathrm{fc}(f+1)}$ that cannot be generated by the class $yT_{\mathrm{fc}(f)}$. We are then led to the conclusion that the two complexity measures investigated in this work induce two orthogonal non-collapsing hierarchies for finite copying parallel rewriting systems. The two hierarchies are schematically represented in Fig. 8 by means of an array.

The two hierarchies state that there is proper inclusion of $\mathscr{L}(r_1\text{-LUSCG}(f_1))$ in $\mathscr{L}(r_2\text{-LUSCG}(f_2))$ if $f_1 < f_2$ and $r_1 \leqslant r_2$, or if $r_1 < r_2$ and $f_1 \leqslant f_2$. The question arises whether we can "trade" the two types of parallelism, i.e, what the relation between $\mathscr{L}(r_1\text{-LUSCG}(f_1))$ and $\mathscr{L}(r_1\text{-LUSCG}(f_2))$ is when $f_1 < f_2$ and $r_1 > r_2$, or when $f_1 > f_2$ and $r_1 < r_2$. As we have already mentioned, Engelfriet et al. [9, Theorem 3.2.5] show that, for each $f \geqslant 1$, there exists a language in $\mathscr{L}(1\text{-LUSCG}(f+1))$ that cannot be generated by any grammar in r-LUSCG(f), for any $r \geqslant 1$. Therefore, synchronized parallelism cannot be traded for independent parallelism. In the following theorem, we show that the converse is not true. The theorem, which is stated for class MCFG, shows that the rank parameter can be traded with the fan-out parameter. In the proof, we will use the following convention: a sequence $X_i, \ldots, X_j$ denotes the empty sequence whenever $j < i$.

Theorem 11. *Let* $f \geqslant 1$ *and* $r \geqslant 3$. *Then for* $1 \leqslant k \leqslant r-2$ *we have* $\mathscr{L}(r\text{-MCFG}(f)) \subseteq \mathscr{L}((r-k)\text{-MCFG}((k+1)f))$.

Proof. Let $G=(V_N, V_T, P, S)$ be a grammar in r-MCFG(f) and let k be an integer, $1 \leqslant k \leqslant r-2$. We will exhibit a grammar G' in $(r-k)$-MCFG($(k+1)f$) such that $L(G')=L(G)$. Let $G'=(V'_N, V_T, P', S)$, where

$$V'_N = V_N \cup \{A_{p,i} \mid p \in P \text{ and } 0 \leqslant i \leqslant k-1\}.$$

We define P' as follows. For each production $p \in P$, $p: A \to g(B_1, \ldots, B_t)$ with $t = \rho(p) > r-k$, we add the following productions to P':

$$\begin{aligned}
&p': A \to g'(B_1, \ldots, B_{t-k-1}, A_{p,0}),\\
&p_0: A_{p,0} \to g_0(B_{t-k}, A_{p,1}),\\
&p_1: A_{p,1} \to g_1(B_{t-k+1}, A_{p,2}),\\
&\vdots\\
&p_{k-2}: A_{p,k-2} \to g_{k-2}(B_{t-2}, A_{p,k-1}),\\
&p_{k-1}: A_{p,k-1} \to g_{k-1}(B_{t-1}, B_t).
\end{aligned}$$

The functions g_i introduced above simply form larger and larger tuples from their arguments, without appending any strings:

$$\begin{aligned}
&g_{k-1}(\langle x_{1,1}, \ldots, x_{1,\varphi(B_{t-1})}\rangle, \langle x_{2,1}, \ldots, x_{2,\varphi(B_t)}\rangle)\\
&\quad = \langle x_{1,1}, \ldots, x_{1,\varphi(B_{t-1})}, x_{2,1}, \ldots, x_{2,\varphi(B_t)}\rangle,\\
&g_{k-2}(\langle x_{1,1}, \ldots, x_{1,\varphi(B_{t-2})}\rangle, \langle x_{2,1}, \ldots, x_{2,\varphi(B_{t-1})+\varphi(B_t)}\rangle)\\
&\quad = \langle x_{1,1}, \ldots, x_{1,\varphi(B_{t-2})}, x_{2,1}, \ldots, x_{2,\varphi(B_{t-1})+\varphi(B_t)}\rangle,\\
&\vdots\\
&g_1(\langle x_{1,1}, \ldots, x_{1,\varphi(B_{t-k+1})}\rangle, \langle x_{2,1}, \ldots, x_{2,\varphi(B_{t-k+2})+\cdots+\varphi(B_t)}\rangle)\\
&\quad = \langle x_{1,1}, \ldots, x_{1,\varphi(B_{t-k+1})}, x_{2,1}, \ldots, x_{2,\varphi(B_{t-k+2})+\cdots+\varphi(B_t)}\rangle,\\
&g_0(\langle x_{1,1}, \ldots, x_{1,\varphi(B_{t-k})}\rangle, \langle x_{2,1}, \ldots, x_{2,\varphi(B_{t-k+1})+\cdots+\varphi(B_t)}\rangle)\\
&\quad = \langle x_{1,1}, \ldots, x_{1,\varphi(B_{t-k})}, x_{2,1}, \ldots, x_{2,\varphi(B_{t-k+1})+\cdots+\varphi(B_t)}\rangle.
\end{aligned}$$

Thus, for $0 \leqslant i \leqslant k-1$, we have the following relation:

$$\varphi(g_i) = \sum_{h=0}^{k-i} \varphi(B_{t-h}) \leqslant (k+1)f.$$

Now let us turn to function g' used in production p'. In order to define this function, we first introduce a homomorphism h from $\{x_{i,j} \mid 1 \leqslant i \leqslant t, 1 \leqslant j \leqslant \varphi(B_i)\} \cup V_T$ to $\{x_{i,j} \mid 1 \leqslant i \leqslant t-k-1, 1 \leqslant j \leqslant \varphi(B_i)\} \cup \{x_{t-k,j} \mid 1 \leqslant j \leqslant \varphi(B_{t-k}) + \cdots + \varphi(B_t)\} \cup V_T$. Homomorphism h is specified as follows:

$$h(\xi) = \begin{cases} a & \text{if } \xi = a,\ a \in V_T,\\ x_{i,j} & \text{if } \xi = x_{i,j},\ 1 \leqslant i \leqslant t-k-1,\\ x_{t-k,\varphi(B_{t-k})+\cdots+\varphi(B_{t-1})+j} & \text{if } \xi = x_{i,j},\ t-k \leqslant i \leqslant t. \end{cases}$$

Now assume that g is defined by a relation of the kind

$$g(\langle x_{1,1},\ldots,x_{1,\varphi(B_1)}\rangle,\ldots,\langle x_{t,1},\ldots,x_{1,\varphi(B_t)}\rangle)=\langle\alpha_1,\ldots,\alpha_{\varphi(p)}\rangle.$$

Then we have

$$g'(\langle x_{1,1},\ldots,x_{1,\varphi(B_1)}\rangle,\ldots,\langle x_{t-k-1,1},\ldots,x_{t-k-1,\varphi(B_{t-k-1})}\rangle,$$
$$\langle x_{t-k,1},\ldots,x_{t-k,\varphi(B_{t-k})+\cdots+\varphi(B_t)}\rangle)=\langle h(\alpha_1),\ldots,h(\alpha_{\varphi(p)})\rangle$$

(g' is linear regular because g is and because h is a bijection). Thus, $\rho(G')\leqslant r-k$, and $\varphi(G')\leqslant(k+1)f$. It can easily be seen that $L(G')=L(G)$. □

The above result transfers in the obvious way to the other parallel rewriting systems discussed in this section. We remark that, in the above theorem, the containment is proper, since we have already observed that there exist languages generated by 1-LUSCG($f+1$) that cannot be generated by class LUSCG(f), for each $f\geqslant 1$. It is not known whether the result is optimal, i.e., whether there exist r, f and k as in the hypotheses of Theorem 11 such that $\mathscr{L}(r\text{-MCFG}(f))$ is not included in $\mathscr{L}((r-k)\text{-MCFG}((k+1)f-1))$. Such a result would complete our knowledge about the class of languages generated by LUSCG, and remains as an open research issue.

6. Remarks

We have characterized finite copying parallel rewriting systems by imposing a restriction, called locality, in the definition of derivation for the class USCG. This significantly alters the formal properties of USCG. While LUSCG is known to generate only semi-linear languages [9, 25, 22], USCG can also generate non-semi-linear languages [6]. And while the class $\mathscr{L}$(LUSCG) is only composed of languages in P, that is languages whose sentences can be recognized in deterministic polynomial time,[4] [7, 24, 22] USCG can generate NP-complete languages [5]. As shown in this work, rewriting systems in LUSCG generate an infinite non-collapsing hierarchy with respect to the fan-out and rank parameters. The result implies that these rewriting systems do not admit normal forms that are defined by some bound on both complexity measures. In contrast, two-normal forms are admitted for grammars in USCG, with respect to both parameters. This result has been shown for matrix grammars (see, for instance, [6]) and it unproblematically transfers to USCG. Furthermore, it has been conjectured that language $L=\{ww\mid w\in D_1\}$ (see Section 2) is not in $\mathscr{L}$(USCG) [6, p. 42]; if this conjecture in fact holds, then $\mathscr{L}$(USCG) and $\mathscr{L}$(LUSCG) are incomparable.

The results of Section 3 also have interesting consequences for the recognition/parsing problem of the generated languages, as discussed in the following. Tabular

[4] In fact, [7] shows the stronger result that OUT(SAG), the class of output languages generated by a string-valued attribute grammar, is in LOG(CFL).

methods for the solution of the recognition problem for the class MCFG have been presented in [22], generalizing in this way the well known Cocke–Kasami–Younger tabular method for the recognition of context-free languages [27, 3]. It is worth observing that, in contrast with the Cocke–Kasami–Younger method, these methods do not behave "uniformly" on MCFG, in the sense that for each grammar $G \in \text{MCFG}$ a method using a recognition matrix with a number of dimensions proportional to d_G is needed, where $d_G = \varphi(G) \cdot (\rho(G) + 1)$ is called the *degree* of G. The existence of a k-rank normal form G' for any G in the class MCFG, such that G' can be obtained in polynomial deterministic time from G, would have entailed that the uniform recognition problem for MCFG could be solved in deterministic polynomial time. This is quite unlikely, since in [13, 21] NP-completeness results were independently shown for the uniform recognition problem for classes MCFG(f), $f \geqslant 2$. (These results easily transfer to classes LUSCG(f).) Assuming $\text{P} \neq \text{NP}$, the existence of a k-rank normal form obtainable in an amount of time not bounded by any polynomial in the size of the input grammar was still an open issue, leading to a possible solution to the uniform recognition of these languages. The result presented in this paper shows that tabular methods of the kind usually employed in context-free language recognition are not a viable solution to the problem.

Appendix: Equivalence of LUSCG and MCFG

Class MCFG has been introduced in Definition 8 and an equivalence relation between MCFG and LUSCG has been stated in Theorem 6; this appendix provides the proof of Theorem 6. We have already remarked that the recursive definition of the rewrite relation in MCFG observes the locality restriction. As a consequence, we find that in MCFG derivations can be associated with underlying trees that can be generated by context-free grammars. We develop here this idea and introduce concepts analogous to those presented in Definition 3.

For a given context-free grammar G_c, we call *complete* any derivation of the form $A \overset{*}{\Rightarrow}_{G_c} \eta$, η a string of terminals of G_c. As done previously in this paper, we represent derivations in G_c by means of trees whose nodes are labeled by symbols of G_c. We write $T(G_c)$ to denote the set of trees representing all complete derivations in G_c. Let $G = (V_N, V_T, P, S)$ be a multiple context-free grammar. Define $P^{(0)} = \{p \mid \rho(p) = 0\}$ and $P^{(1)} = P - P^{(0)}$. (We are overloading symbols $P^{(0)}$ and $P^{(1)}$; it will always be clear from the context whether these symbols denote subsets of productions of a grammar in LUSCG or of a grammar in MCFG.) Without loss of generality, we assume that p_S is the only production in P with left-hand side S and $p_S \in P^{(1)}$.

Definition A.1. The *derivation grammar* of an MCFG G, written $\text{der}(G)$, is a context-free grammar $(P^{(1)}, P^{(0)}, \Pi, p_S)$, where $P^{(1)}$ and $P^{(0)}$ are the sets of nonterminal and terminal symbols, respectively, p_S is the initial symbol and Π is a (finite) set of productions specified as follows. For every $p : A \to g(B_1, \ldots, B_{\rho(p)})$ in P and for

every sequence $p_1,\ldots,p_{\rho(p)}$ of productions such that the left-hand side of p_i is B_i, $1\leqslant i\leqslant\rho(p)$, production $p\rightarrow p_1\cdots p_{\rho(p)}$ belongs to Π.

It should be clear that any instance $A\Rightarrow_G\langle y_1,\ldots,y_{\varphi(A)}\rangle$ of the derivation relation in G can be associated with a complete derivation in $\mathrm{der}(G)$ of the form $p\Rightarrow^*_{\mathrm{der}(G)}\eta$ for some $p\in P$ and $\eta\in(P^{(0)})^*$, that is with a derivation tree in $T(\mathrm{der}(G))$ with root node labeled by p.

The main idea in the next theorem is to compare underlying context-free derivations in MCFG with underlying context-free derivations in LUSCG. To do so, we need to extend the rewrite relation in LUSCG to string tuples. Let $G=(V_N,V_T,P,S)$ be a grammar in LUSCG; in what follows we write $(\langle\gamma_1,\ldots,\gamma_n\rangle,I_1)\Rightarrow_G(\langle\delta_1,\ldots,\delta_n\rangle,I_2)$, $n\geqslant 1$, whenever $(\gamma_1\#\gamma_2\#\cdots\#\gamma_n,I_1)\Rightarrow_G(\delta_1\#\delta_2\#\cdots\#\delta_n,I_2)$ holds (where # is a new symbol). Let p be a production in G having left-hand tuple $(A_1,\ldots,A_n)$, $n\geqslant 1$. Similar to the case of MCFG, any derivation in G having the form $(\langle A_1,\ldots,A_n\rangle,I^{\langle A_1\cdots A_n\rangle})\Rightarrow^*_G(\langle w_1,\ldots,w_n\rangle,\emptyset)$, $w_i\in V_T^*$ for $1\leqslant i\leqslant n$, can be associated with a complete derivation in $\mathrm{der}(G)$ of the form $p\Rightarrow^*_{\mathrm{der}(G)}\eta$, $\eta\in(P^{(0)})^*$, that is with a derivation tree in $T(\mathrm{der}(G))$ with root node labeled by p. We are now ready to prove Theorem 6, whose statement is repeated here.

Theorem 6. *Let r,f be integers such that $r,f\geqslant 1$. Then we have $\mathscr{L}(r\text{-MCFG}(f))=\mathscr{L}(r\text{-LUSCG}(f))$.*

Proof. ($\subseteq$) Let $G=(V_N,V_T,P,S)$ be in r-MCFG(f). We construct G' in r-LUSCG(f) such that $L(G)=L(G')$. In what follows, let $p:A\rightarrow g(B_1,\ldots,B_{\rho(p)})$ be a production in P and let g be defined by an equation of the form

$$g(\langle x_{1,1},\ldots,x_{1,\varphi(B_1)}\rangle,\ldots,\langle x_{\rho(p),1},\ldots,x_{\rho(p),\varphi(B_{\rho(p)})}\rangle)=\langle y_1,\ldots,y_{\varphi(A)}\rangle.$$

Assume also that symbol $p_{S'}$ does not denote any production in P. We define

$$V'_N=\{[p,i,j]\mid p\in P,1\leqslant i\leqslant\rho(p),1\leqslant j\leqslant\varphi(B_i)\}\cup\{[p_{S'},0,0]\}$$

and $G'=(V'_N,V_T,P',[p_{S'},0,0])$, where set P' is constructed as follows. We associate with each p, specified as above, a homomorphism h_p mapping set $\{x_{i,j}\mid 1\leqslant i\leqslant\rho(p),1\leqslant j\leqslant\varphi(B_i)\}\cup V_T$ into set $V'_N\cup V_T$ and defined as follows:

$$h_p(\xi)=\begin{cases}[p,i,j] & \text{if } \xi=x_{i,j},\\ \xi & \text{if } \xi\in V_T.\end{cases}$$

Assume that p' is a production in P containing in the kth position of its right-hand side the symbol A in the left-hand side of p. We add to P' the production

$$([p',k,1],\ldots,[p',k,\varphi(A)])\rightarrow(h_p(y_1),\ldots,h_p(y_{\varphi(A)})).$$

We iterate the process for every pair p, p' as above. In addition, let $p_S : S \to g(B_1, \ldots, B_{\rho(p_S)})$ be defined by an equation (recall that $\varphi(p_S) = 1$)

$$g(\langle x_{1,1}, \ldots, x_{1,\varphi(B_1)}\rangle, \ldots, \langle x_{\rho(p_S),1}, \ldots, x_{\rho(p_S),\varphi(B_{\rho(p_S)})}\rangle) = \langle y_1\rangle.$$

We add to P' the production

$$([p_{S'}, 0, 0]) \to (h_{p_s}(y_1)).$$

Note that $\rho(G) = \rho(G')$ and $\varphi(G) = \varphi(G')$.

We claim that $A \Rightarrow_G \langle y_1, \ldots, y_{\varphi(A)}\rangle$, $y_i \in V_T^*$, $1 \leqslant i \leqslant \varphi(A)$, if and only if

$$(\langle [p', k, 1], \ldots, [p', k, \varphi(A)]\rangle, I^{\langle [p',k,1],\ldots,[p',k,\varphi(A)]\rangle}) \overset{*}{\Rightarrow}_{G'} (\langle y_1, \ldots, y_{\varphi(A)}\rangle, \emptyset),$$

for any p' and k such that the kth nonterminal in the right-hand side of p' is A. The claim can be easily established by associating with the derivations above the corresponding trees in $T(\mathrm{der}(G))$ and $T(\mathrm{der}(G'))$ (which have the same height) and then proceeding by induction on the height of these trees. Relation $L(G) = L(G')$ immediately follows from the claim.

($\supseteq$) Let $G = (V_N, V_T, P, S)$ be in r-LUSCG(f) and let $\mathrm{der}(G) = (P^{(1)}, P^{(0)}, \Pi, p_S)$ be the derivation grammar associated with G. We construct $G' \in r$-MCFG(f) such that $L(G) = L(G')$. Define

$$V'_N = \{[A_1, \ldots, A_{\varphi(p)}] \mid (A_1, \ldots, A_{\varphi(p)}) \text{ is the left-hand tuple of } p \in P\}$$

and let $G' = (V'_N, V_T, P', [S])$, where P' is specified as follows.

Let p_0 be a production in P of the form

$$p_0 : (A_1, \ldots, A_{\varphi(p_0)}) \to (\alpha_1, \ldots, \alpha_{\varphi(p_0)})$$

and, for $1 \leqslant i \leqslant \varphi(p_0)$, let $\alpha_i = u_{i,0} C_{i,1} u_{i,1} \cdots C_{i,k_i} u_{i,k_i}$, where $k_i \geqslant 0$, $u_{i,j} \in V_T^*$, $0 \leqslant j \leqslant k_i$ and $C_{i,j} \in V_N$, $1 \leqslant j \leqslant k_i$. Let also p be a production in Π of the form $p_0 \to p_1 \cdots p_n$, and let $(B_{i,1}, \ldots, B_{i,\varphi(p_i)})$ be the left-hand tuple of p_i, $1 \leqslant i \leqslant n$. Let σ be a bijection from the set $\{(i,j) \mid 1 \leqslant i \leqslant \varphi(p_0), 1 \leqslant j \leqslant k_i\}$ to the set $\{x_{i,j} \mid 1 \leqslant i \leqslant n, 1 \leqslant j \leqslant \varphi(p_i)\}$, such that $\sigma((i,j)) = x_{i',j'}$ always implies $C_{i,j} = B_{i',j'}$. The existence of at least one such a bijection follows from the definition of Π, since $p_1, \ldots, p_n$ together must rewrite exactly the nonterminals introduced by p_0. Then, for $1 \leqslant i \leqslant \varphi(p_0)$, let $y_i = u_{i,0} \sigma((i,1)) u_{i,1} \cdots \sigma((i,k_i)) u_{i,k_i}$.

We add to P' the production

$$[A_1, \ldots, A_{\varphi(p_0)}] \to g_\sigma([B_{1,1}, \ldots, B_{1,\varphi(p_1)}], \ldots, [B_{n,1}, \ldots, B_{n,\varphi(p_n)}]),$$

where g_σ is a linear regular function of arity $\varphi(p_0)$ and rank n defined by the equation

$$g_\sigma(\langle x_{1,1}, \ldots, x_{1,\varphi(p_1)}\rangle, \ldots, \langle x_{n,1}, \ldots, x_{n,\varphi(p_n)}\rangle) = \langle y_1, \ldots, y_{\varphi(p_0)}\rangle.$$

In the construction of P', this process is iterated for every possible choice of σ as above, and then for every possible choice of $p_0 \in P$ and $p \in \Pi$. Note that $\rho(G) = \rho(G')$ and $\varphi(G) = \varphi(G')$.

We claim that $(\langle A_1,\ldots,A_{\varphi(A)}\rangle, I^{\langle A_1,\ldots,A_{\varphi(A)}\rangle}) \overset{*}{\Rightarrow}_G (\langle y_1,\ldots,y_{\varphi(A)}\rangle, \emptyset)$, $y_i \in V_T^*$, $1 \leqslant i \leqslant \varphi(A)$, if and only if there exists a derivation in G' of the form $[A_1,\ldots,A_{\varphi(A)}] \Rightarrow_{G'} \langle y_1,\ldots,y_{\varphi(A)}\rangle$. As before, this can be easily established by induction on the common height of trees in $T(\mathrm{der}(G))$ and $T(\mathrm{der}(G'))$ associated with the above derivations. Again, relation $L(G)=L(G')$ immediately follows from the claim. □

Acknowledgements

We are grateful to Joost Engelfriet, Ryuichi Nakanisi and David Weir for helpful discussion on topics related to this paper. We would also like to thank an anonymous reviewer for extremely helpful and detailed comments. He or she pointed out the need for Lemma 1, and an error he or she found in a previous draft made us change some key definitions. This research was conducted while Rambow was with the Department of Computer and Information Science of the University of Pennsylvania, and Satta was a post-doctoral fellow at the Institute for Research in Cognitive Science at the University of Pennsylvania. The research was sponsored by the following grants: ARO DAAL 03-89-C-0031; DARPA N00014-90-J-1863; NSF IRI 90-16592; and Ben Franklin 91S.3078C-1. Rambow was also supported by the North Atlantic Treaty Organization under a Grant awarded in 1993 while at TALANA, Université Paris 7.

References

[1] A.V. Aho, J.D. Ullman, Syntax directed translations and the pushdown assembler, J. Comput. System Sci. 3(1) (1969) 37–56.
[2] A.V. Aho, J.D. Ullman, Translations on a context-free grammar, Inform. Control 19 (1971) 439–475.
[3] A.V. Aho, J.D. Ullman, The Theory of Parsing, Translation and Compiling, vol. 1, Prentice-Hall, Englewood Cliffs, NJ, 1972.
[4] M. Bauderon, B. Courcelle, Graph expressions and graph rewritings, Math. Systems Theory 20 (1987) 83–127.
[5] E. Dahlhaus, M.K. Warmuth, Membership for growing context-sensitive grammars is polynomial, J. Comput. System Sci. 33 (1986) 456–472.
[6] J. Dassow, G. Păun, Regulated Rewriting in Formal Language Theory, Springer, Berlin, 1989.
[7] J. Engelfriet, The complexity of languages generated by attribute grammars, SIAM J. Comput. 15(1) (1986) 70–86.
[8] J. Engelfriet, L. Heyker, The string generating power of context-free hypergraph grammars, J. Comput. System Sci. 43 (1991) 328–360.
[9] J. Engelfriet, G. Rozenberg, G. Slutzki, Tree transducers, L systems, and two-way machines, J. Comput. System Sci. 20 (1980) 150–202.
[10] S. Greibach, J. Hopcroft, Scattered context grammars, J. Comput. System Sci. 3 (1969) 233–247.
[11] A. Habel, H.J. Kreowski, Some structural aspects of hypergraph languages generated by hyperedge replacement, in Proc. STACS, Lecture Notes in Computer Science, vol. 247, Springer, Berlin, 1987, pp. 207–219.
[12] J.E. Hopcroft, J.D. Ullman, Introduction to Automata Theory, Languages and Computation, Addison-Wesley, Reading, MA, 1979.
[13] Y. Kaji, R. Nakanisi, H. Seki, T. Kasami, The universal recognition problems for multiple context-free grammars and for linear context-free rewriting systems, IEICE Trans. Inform. Systems E75-D(1): 78–88, 1992.

[14] T. Kasami, H. Seki, M. Fujii, Generalized context-free grammars, multiple context-free grammars and head grammars, Tech. Report, Dept. Information and Computer Science, Osaka University, 1987.
[15] M. Latteux, Substitutions dans les EDT0L-systèmes ultralinéaires, Inform. Control 42 (1979) 194–260.
[16] O. Mayer, Some restrictive devices for context-free grammars, Inform. Control 20 (1972) 69–92.
[17] D. Milgram, A. Rosenfeld, A note on scattered context grammars, Inform. Processing Lett. 1 (1971) 47–50.
[18] G. Rozenberg, Extension of tabled 0L-system and languages, Int. J. Comput. Inform. Sci. 2 (1973) 311–336.
[19] G. Rozenberg, D. Vermeir, On ET0L systems of finite index, Inform. Control 38 (1978) 103–133.
[20] A. Salomaa, Formal Languages, Academic Press, Orlando, FL, 1973.
[21] G. Satta, Recognition of linear context-free rewriting systems, in 30th Meeting of the Association for Computational Linguistics (ACL'92), 1992.
[22] H. Seki, T. Matsumura, M. Fujii, T. Kasami, On multiple context-free grammars, Theoret. Comput. Sci. 88 (1991) 191–229.
[23] J.W. Thatcher, Tree automata: an informal survey, in: A.V. Aho (Ed.), Currents in the Theory of Computing, Ch. 4, Prentice-Hall, Englewood Cliffs, NJ, 1973, pp. 143–172.
[24] K. Vijay-Shanker, D.J. Weir, A.K. Joshi, Characterizing structural descriptions produced by various grammatical formalisms, in: 25th Meeting of the Association for Computational Linguistics (ACL'87), 1987.
[25] D.J. Weir, Characterizing mildly context-sensitive grammar formalisms, Ph.D. Thesis, Department of Computer and Information Science, University of Pennsylvania, 1988.
[26] D.J. Weir, Linear context-free rewriting systems and deterministic tree-walk transducers, in: Proc. 30th Meeting of the Association for Computational Linguistics (ACL'92), Newark, Delaware, 1992.
[27] D.H. Younger, Recognition and parsing of context-free languages in time n^3, Inform. Control 10 (1967) 189–208.