IFIG

RESEARCH

REPORT

INSTITUT FÜR INFORMATIK

GRAMMARS WITH SCATTERED NONTERMINALS

Andreas Klein Martin Kutrib

IFIG RESEARCH REPORT 0202

FEBRUARY 2002

Institut für Informatik
JLU Gießen
Arndtstraße 2
D-35392 Giessen, Germany
Tel: +49-641-99-32141
Fax: +49-641-99-32149
mail@informatik.uni-giessen.de
www.informatik.uni-giessen.de

JUSTUS-LIEBIG-UNIVERSITÄT GIESSEN

IFIG Research Report
IFIG Research Report 0202, February 2002

Grammars with Scattered Nonterminals

Andreas Klein[1]

Institut für Mathematik, Universität Kassel
Heinrich Plett Straße 40, D-34132 Kassel, Germany

Martin Kutrib[2]

Institute für Informatik, Universität Giessen
Arndtstr. 2, D-35392 Giessen, Germany

Abstract. We investigate rewriting systems, grammars with scattered nonterminals, which extend the generative capacity of context-free grammars. They can be thought of as having sentential forms where some instances of a nonterminal may be coupled. The context-free-like productions replace a nonterminal together with its coupled instances where new couplings are only established between symbols of the derived subforms. A natural limitation is to bound the degree of synchronous rewriting. We present an infinite hierarchy of separated language families with the property that degree one meets the regular and degree two the context-free languages. It is shown that all generated languages are included in LOGCFL and hence they are context-sensitive and are included in P. Several closure properties are shown.

CR Subject Classification (1998): F.4.2, F.4.3

[1]E-mail: klein@mathematik.uni-kassel.de
[2]E-mail: kutrib@informatik.uni-giessen.de

1 Introduction

Context-free grammars are one of the most important and most developed parts of formal language theory. However, in many situations we are confronted with naturally non-context-free languages.

"The world is not context-free": a comprehensive discussion of this observation giving "seven circumstances where context-free grammars are not enough" can be found in [4]. So there is considerable interest in grammars based on context-free rewriting rules that extend the generative capacity but have similar properties. Several approaches restrict the use of productions in context-free derivations. They are considered in the framework of regulated rewriting. A detailed presentation is the monograph [4] and [5], which include references for the following cited related results.

Particularly related are grammars with partially parallel substitution mode. One development are Indian parallel grammars where a substitution step consists of the substitution of all instances of one nonterminal according to one and the same rule. It has turned out that the absence or presence of erasing rules does not affect their generative capacity, and that the generated language family is incomparable with context-free languages. The so-called k-grammars are parallel rewriting systems where the degree of synchronous rewriting is bounded by some constant k. Besides the first step in such systems exactly k nonterminals have to be rewritten during a derivation step. The rules and nonterminals may be different. Hierarchies depending on k have been shown for k-grammars with and without erasing productions.

The next restriction concerns the choice and the places of applications of the productions. In scattered context grammars the context-free rewriting rules are grouped together into matrices. The rules in a matrix must be applied simultaneously during one step. The nonterminals to be rewritten in the sentential form have to appear in the ordering given by the rules in the matrices. A slight modification where the ordering condition is relaxed yields the unordered scattered context grammars. The language families of unordered grammars with and without erasing rules are properly included in the languages generated by their ordered variants, respectively. Ordered grammars with scattered context and erasing rules are characterizing the recursively enumerable languages. There are several other developments, e.g. k-simple grammars where also a hierarchy depending on k is known.

Common to all of these restrictions is that the conditions for applying a production at some time can be verified by inspecting the sentential form at the same time. If the requirement of applying all rules in a matrix simultaneously is relaxed, unordered scattered context grammars become unordered vector grammars. The matrices are now called vectors, and the requirement is that once a production in a vector has been applied, all productions must be applied during the derivation. Thus, in general it cannot be decided whether a production can be applied or not by inspecting just one sentential form. This can be seen as a vertical context condition. Stronger definitions are given in [10]. Unordered vector grammars have been introduced in [3]. For a subclass without λ-productions and unit-productions it was shown in [12] that the generated languages are belonging to the complexity class LOGCFL. In [11] this result has been improved to arbitrary languages generated by unordered vector grammars.

Here we investigate grammars with scattered nonterminals which are a natural extention of regular and context-free grammars. Basically, the idea is to consider sentential forms of context-free-like grammars where some occurrences of one and the same nonterminal may be coupled. A derivation step replaces a nonterminal together with its coupled instances, whereby new couplings are only established between nonterminals of the derived subforms. The idea is motivated by the following observation. Some word $a_1 \cdots a_n$ can be seen as a set of couples $\{(a_1, 1), \ldots, (a_n, n)\}$ bringing together letters and positions. If one letter appears at several positions we could write $(a, i_1, \ldots, i_p)$ instead of $(a, i_1), \ldots, (a, i_p)$. From this point of view the representation is not unique since, e.g. $(a, i_1, i_2, i_p), (a, i_3, \ldots, i_{p-1})$ is another one. On the other hand, some tuples can be seen as in some sense generalized or coupled nonterminals that are to be rewritten in a context-free fashion.

A formal definition of this notion is given in the next section.

A natural condition is the limitation of the maximal number of coupled-up nonterminals. This restriction leads to grammars of a certain degree. In Section 3 it will be shown that there exists an infinite degree hierarchy of separated language families. The hierarchy has the considerable property that degree one meets the regular and degree two the context-free languages. So we obtain an in some sense unified generalization of both families. Moreover, all families will be shown to belong to the complexity class LOGCFL which is in P and in the context-sensitive languages.

Another approach is to limit the number of nonterminals which may simultaneously appear in a sentential form. These grammars of finite index have extensively be investigated. E.g., for matrix grammars with context-free rules an infinite hierarchy has been shown [4]. By means of different closure properties, which are investigated in Section 4, these grammars and grammars with scattered nonterminals are generating different languages.

2 Grammars with Scattered Nonterminals

We denote the positive integers $\{1, 2, ...\}$ by $\mathbb{N}$ and the set $\mathbb{N} \cup \{0\}$ by $\mathbb{N}_0$. The empty word is denoted by λ and the reversal of a word w by w^R. For the length of w we write $|w|$. The number of occurrences of a symbol a in w is denoted by $|w|_a$. The Parikh mapping associated with an alphabet $T = \{a_1, \ldots, a_n\}$ is a map ψ such that $\psi(w) = (|w|_{a_1}, \ldots, |w|_{a_n})$. We use $\subseteq$ for inclusions and $\subset$ if the inclusion is strict.

Due to the basic motivation and the underlying idea we have terminal and nonterminal symbols, but we do not regard couplings between terminals. More formally, let N and T be two finite sets and $w \in (N \cup T)^*$. In order to express the couplings between symbols from N in w we use a set of tuples whose components identify the corresponding positions. So $(a_1 \cdots a_n, C)$ is a word with coupled symbols from N if and only if $a_1 \cdots a_n \in (N \cup T)^*$ and C is a subset of the tuples which partition $\{1, \ldots, n\}$ such that for each tuple $(i_1, \ldots, i_q) \in C$ there is some $X \in N$ such that $a_{i_1} = \cdots = a_{i_q} = X$. The set of all couplings obeying these restrictions for a given word w is denoted by $C_N(w)$.

Now we are prepared to define grammars based on this concept.

Definition 1 *A* grammar $\mathcal{G}$ with scattered nonterminals *is a system* $\langle N, T, S, P \rangle$, *where*

1. N *is the finite set of* nonterminals,
2. T *is the finite set of* terminals,
3. $S \in N$ *is the* axiom (starting symbol),
4. P *is the finite set of* productions *each of one of the forms* $(S \to w, C)$ *where* $w \in (N \setminus \{S\} \cup T)^*$, $C \in C_N(w)$, *or* $(X \to w_1, \ldots, w_p, C)$ *where* $X \in N$, $p \in \mathbb{N}$, $w_j \in (N \setminus \{S\} \cup T)^*$ *for* $1 \leq j \leq p$, $C \in C_N(w_1 \cdots w_p)$ *such that all* $(i_1, \ldots, i_q) \in C$ *satisfy* $q \leq p$.

Since the starting symbol does not appear on the right-hand side of any production, productions of the form $(S \to w, C)$ are only possible during the first derivation step. They establish possibly some couplings. Productions of the form $(X \to w_1, \ldots, w_p, C)$ mean that the instances of a p-fold coupled symbol X are rewritten by $w_1, \ldots, w_p$. Since $C \in C_N(w_1 \cdots w_p)$ new couplings are only possible between nonterminals in the derived subforms. The condition $q \leq p$ implies that – besides the first derivation step – the maximal number of nonterminals in a coupling cannot grow. Thus, a production of the form $(X \to w_1, \ldots, w_p, C)$ is *applicable* to

$$u = (u_1 \cdots u_m, D), \quad u_j \in N \cup T,\ 1 \leq j \leq m, \quad D \in C_N(u_1 \cdots u_m)$$

if there exists $(i_1, \ldots, i_p) \in D$ such that $u_{i_1} = \cdots = u_{i_p} = X$.
The application yields $v = (v_1 \cdots v_n, E)$, where

$$v_1 \cdots v_n = u_1 \cdots u_{i_1 - 1} w_1 u_{i_1+1} \cdots u_{i_2-1} w_2 u_{i_2+1} \cdots w_p u_{i_p+1} \cdots u_m$$

and E contains exactly the couplings derived from C and $D \setminus \{(i_1, \ldots, i_p)\}$ by adjusting the symbol positions accordingly. As usual we write $u \Rightarrow v$ if u directly derives v and $\Rightarrow^*$ for the reflexive and transitive closure of $\Rightarrow$.

The language generated by $\mathcal{G}$ is

$$L(\mathcal{G}) = \{ w \mid w \in T^*, \quad (S, \emptyset) \Rightarrow^* (w, \emptyset) \}$$

The condition that a coupling is always between one and the same nonterminal makes life easier but is not really a restriction or even a limitation. Since derivations are in a context-free-like fashion a grammar without this property can always be transformed to an equivalent one obeying the condition. The only thing to do is to rename uniquely coupled-up nonterminals that appear on the left-hand side of a production.

Example 2 The grammar $\langle \{A, S\}, \{a, b, c\}, S, P\rangle$ with

$$P = \{\ (S \to AA, \{(1, 2)\}), \quad (A \to aAb, cA, \{(2, 5)\}), \quad (A \to ab, c, \emptyset\})\ \}$$

generates the language $\{a^n b^n c^n \mid n \in \mathbb{N}\}$. The derivation for $n = 4$ is

$$\begin{aligned}(S, \emptyset) &\Rightarrow (AA, \{(1, 2)\}) \Rightarrow (aAbcA, \{(2, 5)\}) \\ &\Rightarrow (aaAbbccA, \{(3, 8)\}) \Rightarrow (aaaAbbbcccA, \{(4, 11)\}) \\ &\Rightarrow (aaaabbbbcccc, \emptyset)\end{aligned}$$

More figurative the derivation together with the current couplings may be represented as:

$$S \Rightarrow AA \Rightarrow aAbcA \Rightarrow aaAbbccA \Rightarrow \cdots$$

Similarly, a grammar for the language $\{ww \mid w \in \{a, b\}^+\}$ can be constructed. A natural condition is the limitation of the maximal number of coupled-up nonterminals. This restriction leads to grammars of a certain degree.

Definition 3 *Let $m \in \mathbb{N}$ be a constant. A grammar with scattered nonterminals*

1. *is* of degree $2m$ *if for its productions $(S \to w, C)$ all $(i_1, \ldots, i_q) \in C$ satisfy $q \leq m$ (i.e. the maximal number of nonterminals in a coupling is bounded by m).*
2. *It is even* of degree $2m - 1$ *if in addition for its productions $(S \to w, C)$ resp. $(X \to w_1, \ldots, w_m, C)$ a coupling $(i_1, \ldots, i_m) \in C$ implies $i_m = |w|$ resp. $i_m = |w_1 \cdots w_m|$ (i.e. i_m is the position of the rightmost symbol on the right-hand side).*

Roughly speaking, the degree of a grammar determines the maximal number of subwords that may be lengthened during one derivation step in a coupled fashion (a nonterminal may have left and right neighboring subwords). For odd degrees the rightmost coupled nonterminal is forced to have only left neighbors if the number of symbols in that coupling is maximal. This is in some sense a generalization of right-linearity. In general, for a grammar of degree k the maximal number k of nonterminals in one coupling can be derived during the first transition step. In subsequent steps this number cannot be increased. In case of even degrees there may occur more than one coupling with k nonterminals. In case of odd degrees for maximal couplings it is requested that the rightmost symbol on the right-hand side of a production belongs to the coupling. This implies that there may occur at most one maximal coupling in any sentential form. But nevertheless there may occur other couplings which are not maximal.

For example, the language $\{a^n b^n c^n \mid n \in \mathbb{N}\}$ in Example 2 is generated by a grammar of degree three. The construction can be adapted such that for any $k \in \mathbb{N}$ the language $L_k = \{a_1^n \cdots a_k^n \mid n \in \mathbb{N}\}$ over the alphabet $\{a_1, \ldots, a_k\}$ is generated by a grammar of degree k.

The family of all languages which can be generated by grammars with scattered nonterminals is denoted by $\mathscr{L}(\mathrm{SN})$. If the productions are restricted to degree k we use the notion $\mathscr{L}(k\text{-SN})$.

Corollary 4 *Let $k \in \mathbb{N}$ be a constant, then $L_k \in \mathscr{L}(k\text{-SN})$.*

3 Generative Capacity

Taking a closer look at the definitions leads to the observation that in case of degree two on the right-hand sides of the productions there is always a single word. Moreover, the maximal number of nonterminals in a coupling is one. This means that there are no useful couplings at all and, hence, the grammar is a context-free one.

The situation in case of degree one is more restrictive. If a nonterminal appears on the right-hand side of some production, then it has to be the rightmost symbol since again the maximal number of symbols coupled together is one. This implies that there is at most one nonterminal on each right-hand side. Together it follows that the grammar is right-linear. We denote the regular languages by $\mathscr{L}(\mathrm{REG})$ and the context-free languages by $\mathscr{L}(\mathrm{CF})$.

Corollary 5 $\mathscr{L}(1\text{-SN}) = \mathscr{L}(\mathrm{REG})$ *and* $\mathscr{L}(2\text{-SN}) = \mathscr{L}(\mathrm{CF})$

The corollary relates the generative capacity of degree one and two to the regular and context-free languages, respectively. A fundamental result concerning arbitrary degrees is immediately derived from the definition and a well-known result about context-free languages. A language L is *semilinear* if its set of Parikh vectors $\{\psi(w) \mid w \in L\}$ is semilinear.

Lemma 6 *Each of the languages in $\mathscr{L}(\mathrm{SN})$ or $\mathscr{L}(k\text{-SN})$ for some $k \in \mathbb{N}$ is semilinear.*

Proof. Let $\mathcal{G}$ be some grammar under consideration. The right-hand sides of its productions can be rearranged such that coupled symbols are placed joint together. In general, the resulting grammar $\mathcal{G}'$ generates a different language but $L(\mathcal{G})$ and $L(\mathcal{G}')$ are letter-equivalent. Since the coupled nonterminals of $\mathcal{G}'$ appear joint together they can be replaced by just one single nonterminal, respectively, without changing the generated language. Now we have a context-free grammar $\mathcal{G}''$ such that $L(\mathcal{G}'')$ and $L(\mathcal{G})$ are letter-equivalent. In [8] it has been shown that every context-free language is semilinear. Trivially, letter-equivalence preserves semilinearity. □

3.1 Strictly Monotone Derivations

In order to investigate the relationships between degrees k and $k+1$, in particular, to prove an infinite hierarchy, we need a tool for proving negative results. This will be a pumping argument. In order to prove this and other results it is helpful to simplify an arbitrary grammar. A production $(X \to w_1, \ldots, w_p, C)$ is said to be *strictly monotone* if it inserts more symbols than it replaces, i.e., if $|w_1 \cdots w_p| > p$. Otherwise we call a production *non-increasing*.

Next we are going to show how to remove non-increasing productions possibly with the exceptions $(S \to a, \emptyset)$ for $a \in T \cup \{\lambda\}$ if the empty word or some words of length one have to be generated.

Lemma 7 *For every grammar $\mathcal{G}$ of degree $k \in \mathbb{N}$ there exists an equivalent grammar $\mathcal{G}'$ of degree k whose only non-increasing productions are of the form $(S \to a, \emptyset)$ where $a \in T \cup \{\lambda\}$.*

Proof. Let $\mathcal{G}$ be some grammar of degree k. At first we construct a grammar $\tilde{\mathcal{G}}$ from $\mathcal{G}$ as follows.

Let $r = (X \to w_1, \ldots, w_p, C)$ be a non-increasing production of $\mathcal{G}$. For all productions r_i whose right-hand sides contain a p-fold coupled X a new production $\tilde{r}_i$ is included in $\tilde{\mathcal{G}}$, where the p-fold coupled X is replaced by $w_1, \ldots, w_p$. The set of couplings of $\tilde{r}_i$ is simply given by C and the couplings of r_i. Besides the (new) productions $\tilde{r}_i$ the (old) productions r and r_i are also included in $\tilde{\mathcal{G}}$. This step is repeated until the resulting set of productions remains unchanged.

Since during the construction none of the old productions is deleted and the right-hand sides of the newly included productions never get longer the construction process terminates. (There exist only finitely many productions which may be included.)

The equivalence $L(\mathcal{G}) = L(\tilde{\mathcal{G}})$ follows immediately. In general, $\tilde{\mathcal{G}}$ has still non-increasing productions but, on the other hand, if w belongs to $L(\tilde{\mathcal{G}})$, then there exists a derivation $(S, \emptyset) \Rightarrow^* (w, \emptyset)$ in $\tilde{\mathcal{G}}$, where the only non-increasing step is the first one.

This claim can be shown as follows. Let

$$(S, \emptyset) = (v_0, C_0) \Rightarrow (v_1, C_1) \Rightarrow \cdots \Rightarrow (v_n, C_n) = (w, \emptyset)$$

be a derivation of w where the number n of steps is minimal under all derivations of w. Assume that in some step $(v_i, C_i) \Rightarrow (v_{i+1}, C_{i+1})$ for $1 \le i \le n-1$ a non-increasing production $(X \to w_1, \ldots, w_p, C)$ is applied. To this end the p-fold coupled X must have been generated of some step before. Let us say in step $(v_j, C_j) \Rightarrow (v_{j+1}, C_{j+1})$, $0 \le j \le i-1$, by the production $r = (Y \to w'_1, \ldots, w'_q, C')$. Due to the construction of $\tilde{\mathcal{G}}$ there exists a production $\tilde{r} = (Y \to w''_1, \ldots, w''_q, C'')$ where the coupled X on the right-hand side of r is replaced by $w_1, \ldots, w_p$. By applying $\tilde{r}$ instead of r, the latter step $(v_i, C_i) \Rightarrow$

(v_{i+1}, C_{i+1}) can be omitted and

$$(v_0, C_0) \Rightarrow^* (v_j, C_j) \Rightarrow (v'_{j+1}, C'_{j+1}) \Rightarrow^* (v'_{i-1}, C'_{i-1}) \Rightarrow (v_{i+1}, C_{i+1}) \Rightarrow^* (v_n, C_n)$$

is a derivation of w. Since the length of the derivation is $n-1$ we have a contradiction to the minimality of n. Therefore, the assumption is not true and the claim follows.

Now we can delete all non-increasing productions whose left-hand sides are not S without affecting the generated languages. It remains to remove productions $(S \to Y, C)$ where $Y \in N$. This can be achieved by inserting a new production $(S \to w, C)$ for every production $(Y \to w, C)$. Since the latter has been shown to be strictly monotone and S never appears on the right-hand side of a production, now $(S \to Y, C)$ can be deleted without changing the generated language. The resulting grammar $\mathcal{G}'$ is of the desired form. □

3.2 Hierarchy

The following pumping lemma is a useful tool for separating language families since it allows to prove negative results. It is in some sense weaker than others since it contains no statement about the usual ordering in which the repeated subwords appear.

Lemma 8 *Let $\mathcal{G}$ be a grammar of degree $k \in \mathbb{N}$. Then there exists a constant $n \in \mathbb{N}$ such that every $w \in L(\mathcal{G})$ with $|w| \geq n$ may be written as $x_0 y_1 x_1 y_2 \cdots y_k x_k$, where $1 \leq |y_1 y_2 \cdots y_k| \leq n$, and for all $i \in \mathbb{N}$ there exists a word $w' \in L(\mathcal{G})$ such that w' is in some order a concatenation of the (sub)words $x_0, \ldots, x_k$ and i times y_j for each $1 \leq j \leq k$.*

Proof. Since $\mathcal{G}$ has only finitely many productions for every long enough word w from $L(\mathcal{G})$ there exists a derivation such that some production(s) with the same left-hand side are applied at least twice. Moreover, the second application is on symbols derived from the first application (cf. Figure 1). Obviously, for a given grammar the necessary word length for such a situation can be calculated. It defines the constant n.

Assume for a moment the degree of $\mathcal{G}$ is even. Then for the core derivation we have

$$S \Rightarrow \cdots \Rightarrow u_0 A u_1 A u_2 \cdots u_{j-1} A u_j \Rightarrow \cdots \Rightarrow u_0 v_1 u_1 v_2 u_2 \cdots u_{j-1} v_j u_j \Rightarrow \cdots \Rightarrow w$$

where $A \in N$, $1 \leq j \leq \frac{k}{2}$, $u_0, \ldots, u_j \in T^*$ and $v_1 \cdots v_j$ contains a j-fold coupled A and some terminal symbols. Without loss of generality we may assume that u_i are terminal words and v_i are terminal words with the exception of the coupled A, since the derivation of other nonterminals can be finished without affecting the A.

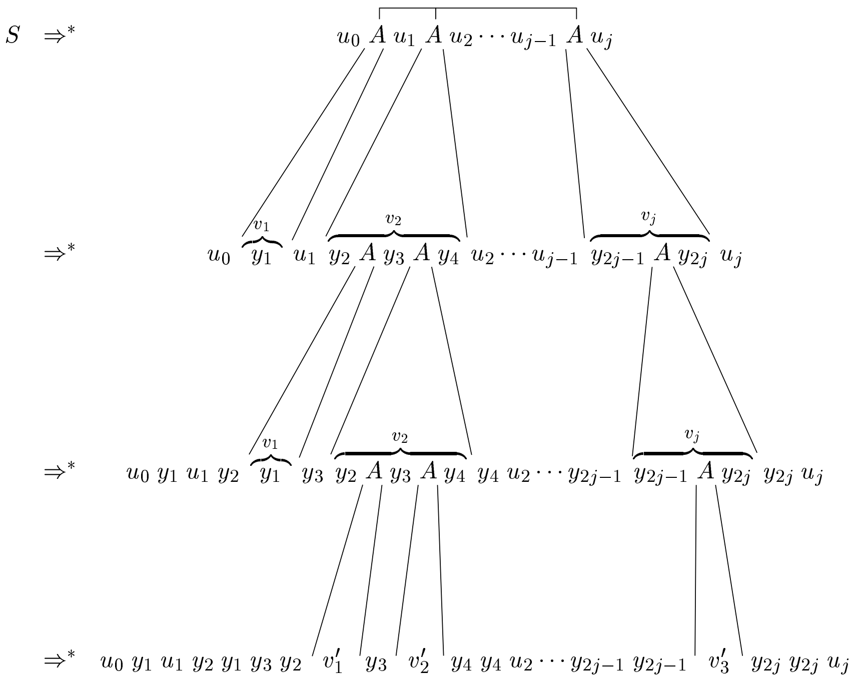

Figure 1: Derivation scheme of Lemma 8.

The word v_1 is derived from the first of the coupled A, v_2 from the second and so on. Since $v_1 \cdots v_j$ contain a j-fold coupled A there must exist derivations in which this loop appears arbitrary times. In order to determine the pumped portions of w we have to consider the positions of the A in the words $v_1, \ldots, v_j$.

We define subwords y_m, $1 \leq m \leq 2j$ of $v_1 \cdots v_j$ as follows. From left to right a y_m starts either at the beginning of a word v_l or immediately after a nonterminal A. A y_m ends either at the end of a word v_l or immediately before a nonterminal A. Clearly, some of the y_m may be empty. But due to the fact that we may assume strictly monotone productions at least one of the y_m is not empty.

Next we define the subwords x_l which are not pumped. These are the already derived subwords $u_0, \ldots, u_j$ and, in addition, the terminal subwords $v'_1, \ldots, v'_j$ to which the j-fold coupled A is finally derived. They are numbered from left to right according to their appearance. Again, some of the $x_0, \ldots, x_{2j}$ may be empty.

For the even degree case the lemma follows since j is at most $\frac{k}{2}$ and, thus, we found subwords $x_0, \ldots, x_k$ and $y_1, \ldots, y_k$ as claimed.

In case of an odd degree j is at most $\lceil \frac{k}{2} \rceil$. For $j < \lceil \frac{k}{2} \rceil$ the lemma has been shown. For $j = \lceil \frac{k}{2} \rceil$ the subword u_j must have been derived during the first step. Afterwards during the pump loops the rightmost symbol is always the unique nonterminal which is j-fold coupled.

Therefore, y_{2j} is always empty and at most $2j - 1$ portions can be pumped. Since k is odd we obtain $2j - 1 = 2\lceil \frac{k}{2} \rceil - 1 = 2\frac{k+1}{2} - 1 = k$, and the lemma follows for the odd degree case, too. □

We apply the pumping lemma to the languages $L_k = \{a_1^n \cdots a_k^n \mid n \in \mathbb{N}\}$ of Example 2 and Corollary 4.

Lemma 9 *Let $k \in \mathbb{N}$ be a constant, then L_{k+1} does not belong to $\mathscr{L}(k\text{-SN})$.*

Proof. Assume L_{k+1} belongs to $\mathscr{L}(k\text{-SN})$. Let n be the constant of Lemma 8 and consider the word $w = a_1^n \cdots a_{k+1}^n$. Since we may pump at most k portions of w the result would not be a word in L_{k+1}. □

The lemma immediately implies the hierarchy of grammars with scattered nonterminals.

Theorem 10 *Let $k \in \mathbb{N}$ be a constant, then $\mathscr{L}(k\text{-SN}) \subset \mathscr{L}((k+1)\text{-SN})$ and $\mathscr{L}(k\text{-SN}) \subset \mathscr{L}(\text{SN})$.*

Proof. The inclusions $\mathscr{L}(k\text{-SN}) \subseteq \mathscr{L}((k+1)\text{-SN}) \subseteq \mathscr{L}(\text{SN})$ are due to structural reasons. The strictness follows from Lemma 8, Example 2, Corollary 4 and the identity $\mathscr{L}(\text{SN}) = \bigcup_{k=1}^{\infty} \mathscr{L}(k\text{-SN})$. □

The languages L_k are witnesses for the hierarchy. On the other hand, for any $k \in \mathbb{N}$ the language L_k belongs to the family $\mathscr{L}(\text{SN})$. Thus, in some sense the infinite hierarchy converges to $\mathscr{L}(\text{SN})$:

$$\mathscr{L}(1\text{-SN}) \subset \mathscr{L}(2\text{-SN}) \subset \cdots \subset \mathscr{L}(k\text{-SN}) \subset \cdots \subset \mathscr{L}(\text{SN})$$

3.3 Comparison with LOGCFL

We now turn to the question where the hierarchy ends up. To this end we can compare $\mathscr{L}(\text{SN})$ with other families. Since $\mathscr{L}(1\text{-SN})$ and $\mathscr{L}(2\text{-SN})$ are equal to Chomsky families a natural candidate for comparisons is the family of context-sensitive languages $\mathscr{L}(\text{CS})$.

The inclusion $\mathscr{L}(\text{SN}) \subseteq \mathscr{L}(\text{CS})$ follows immediately from the fact that after the first step the derivations according to a grammar of arbitrary degree are strictly monotone. So they can be simulated by a linearly space bounded nondeterministic Turing machine what implies context-sensitivity.

The inclusion is even strict: $\mathscr{L}(\text{SN}) \subset \mathscr{L}(\text{CS})$. For example the language $L = \{a^n(b^n c^n)^+ \mid n \in \mathbb{N}\}$ is context-sensitive. By using the pumping lemma it is easy to see that L cannot be generated by any grammar of any degree $k \in \mathbb{N}$. Since L is not semilinear the same result follows alternatively from Lemma 6.

In order to strengthen this inclusion $\mathscr{L}(\text{SN})$ must be compared with other language families properly included in $\mathscr{L}(\text{CS})$. An interesting candidate is LOGCFL, the class of languages which are log-space reducible to $\mathscr{L}(\text{CF})$. LOGCFL is properly contained in P and $\mathscr{L}(\text{CS})$, respectively, and has several characterizations. A collection of problems in LOGCFL can be found in [2]. In [13] it has been characterized in terms of nondeterministic auxiliary pushdown automata working in polynomial time and logarithmic space. Such automata are nondeterministic devices having an unbounded pushdown store and, in addition, a logarithmically bounded read-write working tape.

Theorem 11 $\mathscr{L}(\mathrm{SN}) \subset \mathsf{LOGCFL}$

Proof. Let $\mathcal{G} = \langle N, T, S, P \rangle$ be an arbitrary grammar with scattered nonterminals, say of degree $k \in \mathbb{N}$. At first we construct a nondeterministic auxiliary pushdown automaton $\mathcal{A}$ working in polynomial time and logarithmic space, such that $\mathcal{A}$ accepts $L(\mathcal{G})$. Let $w = a_1, \ldots, a_n$ be an input. The main task of $\mathcal{A}$ is to compute predicates $\texttt{test}(A, s_1, l_1, \ldots, s_m, l_m)$ where $A \in N$, $m \in \mathbb{N}$, and $s_1, \ldots, s_m \in \mathbb{N}$, $l_1, \ldots, l_m \in \mathbb{N}_0$ such that $s_i + l_i \leq s_{i+1}$ and $s_m + l_m \leq n$ for $1 \leq i < m$.

The predicate $\texttt{test}$ is true if and only if an m-fold coupled $A \in N$ can be derived to the m portions of the input which are given by their starting positions s_i and their lengths l_i, respectively. I.e.,

$$(A^m, \{(1, \ldots, m)\}) \Rightarrow^* (v_1 \cdots v_m, \emptyset)$$

where for $1 \leq i \leq m$ we have $v_i = \lambda$ if $l_i = 0$ and $v_i = a_{s_i} \cdots a_{s_i+l_i-1}$ otherwise.

Assume for a moment the arguments of $\texttt{test}$ are written on the working tape. The positions s_i and lengths l_i are given in binary on $2 \cdot m$ tracks. Since the maximal number of symbols in a coupling is bounded by $\lceil \frac{k}{2} \rceil$ (with respect to the given grammar) the maximal number of tracks needed is also bounded. Therefore, the arguments are written in logarithmic space.

The first step of $\mathcal{A}$ is to guess an applicable production $(A \to w_1, \ldots, w_m, C)$ from P. Then $\mathcal{A}$ has successively to guess partitions $s_{i_1}, l_{i_1}, \ldots, s_{i_n}, l_{i_n}$ of the intervals $s_i, \ldots, s_i + l_i - 1$ according to the symbols in $w_i = u_{i_1} \cdots u_{i_p}$, $1 \leq i \leq m$. In order to guess partitions for i, $\mathcal{A}$ starts with $s_{i_1} = s_i$. If u_{i_1} is a terminal, then l_{i_1} is set to 1. If u_{i_1} is a nonterminal, then $\mathcal{A}$ guesses the length l_{i_1} of the terminal (sub)word derived from u_{i_1}. This can be done by sweeping over an empty track and guessing the bits. During another sweep the length can be added to s_{i_1} in order to obtain s_{i_2}. Now $\mathcal{A}$ proceeds until u_{i_p} where $s_{i_p} + l_{i_p} - 1 = s_i + l_i - 1$ is checked. Observe that this behavior works also fine in cases where parts of coupled nonterminals are derived to the empty word. In such situations l_i would be 0 and s_{i+1} equals s_i.

Since the number of symbols of any of the right-hand sides of the productions in P is bounded by a constant we may assume that there exist as many tracks as needed for the partitioning task. Thus, it can be performed in logarithmic space. Moreover, each computation of a bound in the partitions takes at most $O(\log(n))$ steps. Therefore, the whole partitioning takes $O(\log(n))$ time.

Once all partitionings are available on the working tape, $\mathcal{A}$ creates computation batch jobs for the predicate $\texttt{test}$.

Since the couplings between nonterminals in the words $w_1, \ldots, w_m$ are totally determined by the applied production, and this production is known, $\mathcal{A}$ proceeds symbolwise from left to right in $w_1 \cdots w_m$. If it finds a nonterminal A', then the guessed intervals $s'_1, l'_1, \ldots, s'_{m'}, l'_{m'}$ for the nonterminal itself and possibly for its coupled instances are arguments to a job $\mathtt{test}(A', s'_1, l'_1, \ldots, s'_{m'}, l'_{m'})$. This job is pushed onto the stack which takes $O(\log(n))$ time.

If $\mathcal{A}$ finds a terminal symbol a, then the corresponding guessed interval s'_1, l'_1 has length one and $\mathcal{A}$ verifies whether the input symbol at position s'_1 equals a. To this end the input head is moved to the first input symbol. Subsequently the value s'_1 is successively decremented whereby the input head is moved one position to the right, respectively. Since the decrementation takes $O(n)$ time steps [9], the time needed for the verification is $O(n)$.

So, altogether this phase of $\mathcal{A}$ takes $O(n)$ time steps.

Now the body part of the computation of the call of `test` is done. It remains to compute the jobs on the stack. Therefore, after finishing the last phase $\mathcal{A}$ clears its working tape and loads another job from its pushdown store. The computation of $\mathcal{A}$ stops successfully if no further jobs are available. If $\mathcal{A}$ is initialized with $\mathtt{test}(S, 1, n)$, then a successful computation implies that the input can be derived from the starting symbol S and leads to acceptance.

The initialization of $\mathcal{A}$ takes $O(n)$ time. Every body computation of `test` takes $O(n)$ time also. We may assume that the productions of $\mathcal{G}$ are strictly monotone. Therefore, at most n derivation steps are possible. Since for every step we have one body computation of `test`, $\mathcal{A}$ obeys the time complexity $O(n^2)$. Together with the space complexity $O(\log(n))$ this proves that $\mathscr{L}(\mathrm{SN})$ is included in LOGCFL. The strictness of the inclusion follows from the fact that the already mentioned language $\{a^n(b^nc^n)^+ \mid n \in \mathbb{N}\}$ belongs to LOGCFL but cannot be generated by any grammar of any degree $k \in \mathbb{N}$ and, thus, does not belong to $\mathscr{L}(\mathrm{SN})$. □

The following lemma can alternatively be used to prove the strictness of the inclusion $\mathscr{L}(\mathrm{SN}) \subset \mathsf{LOGCFL}$. In addition, it generalizes a well-known result for context-free languages and is a useful tool for proving negative results.

Lemma 12 *Every unary language belonging to* $\mathscr{L}(\mathrm{SN})$ *is regular.*

Proof. In the proof of Lemma 6 it has been shown that every language $L \in \mathscr{L}(\mathrm{SN})$ is letter-equivalent to some context-free language L'. For unary languages this implies $L = L'$. Since every unary context-free language is regular [6] the lemma follows. □

For example, the LOGCFL-language $\{a^{2^n} \mid n \in \mathbb{N}\}$ does not belong to $\mathscr{L}(\mathrm{SN})$. Finally, the lemma holds for all families $\mathscr{L}(k\text{-SN})$, too.

4 Closure Properties

Some closure properties of the families $\mathscr{L}(\mathrm{SN})$ and $\mathscr{L}(k\text{-SN})$ are investigated. It turns out that these properties are similar to the properties of context-free languages.

4.1 Closure

We start the investigation by showing the closure under union, concatenation, iteration and arbitrary homomorphisms.

Lemma 13 *Let $k \in \mathbb{N}$ be a constant, then $\mathscr{L}(k\text{-SN})$ and $\mathscr{L}(\mathrm{SN})$ are closed under union.*

Proof. Let $\mathcal{G} = \langle N, T, S, P\rangle$ and $\mathcal{G}' = \langle N', T', S', P'\rangle$ be two grammars in question. As usual we can construct a grammar $\mathcal{G}'' = \langle N \cup N' \cup \{S''\}, T \cup T', S'', P''\rangle$ for $L(\mathcal{G}) \cup L(\mathcal{G}')$. Thereby we assume that N and N' are disjoint and S'' is a new symbol. P'' is given by $P \cup P' \cup \{(S'' \to w, C) \mid (S \to w, C) \in P$ or $(S' \to w, C) \in P'\}$. Thus, the degree of $\mathcal{G}''$ is the maximum of the degrees of $\mathcal{G}$ and $\mathcal{G}'$. □

In general, for proving positive closure properties constructively, we have to cope with the problem of preserving the degree. Thereby special attention has to be paid for odd degrees.

Theorem 14 *Let $k \in \mathbb{N}$ be a constant, then $\mathscr{L}(k\text{-SN})$ and $\mathscr{L}(\mathrm{SN})$ are closed under concatenation.*

Proof. Let $\mathcal{G} = \langle N, T, S, P\rangle$ and $\mathcal{G}' = \langle N', T', S', P'\rangle$ be two grammars in question. The idea of introducing a new starting symbol S'' and the production $(S'' \to SS', \emptyset)$ does not help, since the maximal number of nonterminals in a coupling must not grow.

But for even degrees we can concatenate the right-hand sides of the productions $(S \to w, C) \in P$ and $(S' \to w', C') \in P'$ to $(S'' \to ww', D)$, where D are the couplings according to C and C'. The construction preserves the degree and yields to a grammar for $L(\mathcal{G})L(\mathcal{G}')$.

In case of odd degrees at most one maximal coupling is allowed since an instance of the nonterminals in maximal couplings must be the rightmost symbol on the right-hand side. In order to cope with this problem we utilize the following observations.

If a maximal coupling exists in any sentential form, then the rightmost symbol of the form belongs to the coupling. Considering an arbitrary derivation, then either during the first step only non-maximal couplings are created or there exists exactly one step that produces a sentential form without maximal coupling from a sentential form having a maximal coupling.

The idea is to modify such productions such that the rightmost nonterminal is rewritten by its original word concatenated by a right-hand side of start productions of the second grammar. Formally, let $m = \lceil \frac{k}{2} \rceil$, then $\mathcal{G}'' = \langle N \cup N', T \cup T', S, P'' \cup P' \rangle$ where $P'' = P_1 \cup P_2 \cup P_3 \cup P_4$ with

$$
\begin{aligned}
P_1 &= \{(X \to w_1, \dots, w_p, C) \in P \mid p < m\} \\
P_2 &= \{(S \to w, C) \in P \mid \text{there exists } (i_1, \dots, i_q) \in C :\ q = m\} \\
&\quad \cup \{(X \to w_1, \dots, w_m, C) \in P \mid \text{there exists } (i_1, \dots, i_q) \in C :\ q = m\} \\
P_3 &= \{(S \to ww', D) \mid (S \to w, C) \in P \setminus P_2 \text{ and } (S' \to w', C') \in P' \text{ and} \\
&\qquad D \text{ are couplings according to } C \text{ and } C'\} \\
P_4 &= \{(X \to w_1, \dots, w_m w', D) \mid (X \to w_1, \dots, w_m, C) \in P \setminus P_2 \text{ and} \\
&\qquad (S' \to w', C') \in P' \text{ and } D \text{ are couplings according to } C \text{ and } C'\}
\end{aligned}
$$

The construction preserves odd degrees and yields to a grammar for $L(\mathcal{G})L(\mathcal{G}')$.
□

For odd degrees, the closure under iteration can be shown by a straightforward modification of the construction for concatenation. The problem for even degrees is that the rightmost symbol in sentential forms is not necessarily a nonterminal. On the other hand, we may insert as many maximal couplings as needed.

Theorem 15 *Let $k \in \mathbb{N}$ be a constant, then $\mathscr{L}(k\text{-SN})$ and $\mathscr{L}(\text{SN})$ are closed under iteration.*

Proof. Let $\mathcal{G} = \langle N, T, S, P \rangle$ be a grammar in question. For odd degrees a grammar $\mathcal{G}'$ for $(L(\mathcal{G}))^*$ is constructed as for concatenation. The necessary modifications are straightforward.

For even degrees we provide a new nonterminal A which is always maximally coupled. Moreover, its positions are always adjacent at the right end of a sentential form. So let $m = \frac{k}{2}$, then $\mathcal{G}' = \langle N \cup \{A\}, T, S, P' \rangle$ with

$$
\begin{aligned}
P' = \{&(S \to \lambda, \emptyset), (S \to A^m, \{(1, \dots, m)\})\} \\
&\cup \{(A \to w, \lambda, \dots, \lambda, C) \mid (S \to w, C) \in P\} \\
&\cup \{(A \to wA^m, D) \mid (S \to w, C) \in P \text{ and } D \text{ are the couplings of } C \\
&\qquad \text{joint with } (i_1, \dots, i_m) \text{ where } i_1, \dots, i_m \text{ are the positions of } A^m\}
\end{aligned}
$$

preserves an odd degree and generates $(L(\mathcal{G}))^*$. □

The next operations for which we obtain positive closure properties are homomorphisms.

Theorem 16 *Let $k \in \mathbb{N}$ be a constant, then $\mathscr{L}(k\text{-SN})$ and $\mathscr{L}(\text{SN})$ are closed under arbitrary homomorphisms.*

Proof. Since all productions are context-free-like, we only need to modify the right-hand sides of the productions such that terminals are replaced by their homomorphic images (Of course, the couplings need to be adjusted, too). □

4.2 Non-Closure

Now we turn to the remaining Boolean operations complementation and intersection and show the non-closure of the families.

Theorem 17 *Let* $k \geq 2$ *be a constant, then* $\mathscr{L}(k\text{-SN})$ *and* $\mathscr{L}(\text{SN})$ *are not closed under intersection.*

Proof. All families in question contain the context-free languages. Moreover, they are closed under homomorphisms. Since every recursively enumerable language can be represented as the homomorphic image of the intersection of two context-free languages [7] the closure under intersection would contradict Theorem 11. □

Theorem 18 *Let* $k \geq 2$ *be a constant, then* $\mathscr{L}(k\text{-SN})$ *and* $\mathscr{L}(\text{SN})$ *are not closed under complement.*

Proof. Since the families are closed under union by $\overline{\overline{L_1} \cup \overline{L_2}} = L_1 \cap L_2$ the closure under complement would imply the closure under intersection what contradicts Theorem 17. □

Corollary 19 *Let* $k \geq 2$ *be a constant, then* $\mathscr{L}(k\text{-SN})$ *and* $\mathscr{L}(\text{SN})$ *are not closed under set difference.*

The non-closure under complement yields another proof of the properness of the inclusion $\mathscr{L}(\text{SN}) \subset \mathsf{LOGCFL}$ by means of different closure properties. In [1] the closure of LOGCFL under complement has been shown.

References

[1] Borodin, A., Cook, S. A., Dymond, P. W., Ruzzo, W. L., and Tompa, M. *Two applications of inductive counting for complementation problems*. SIAM J. Comput. 18 (1989), 559–578.

[2] Cook, S. A. *A taxonomy of problems with fast parallel algorithms*. Inform. Control 64 (1985), 2–22.

[3] Cremers, A. B. and Mayer, O. *On matrix languages*. Inform. Control 23 (1973), 86–96.

[4] Dassow, J. and Păun, G. *Regulated Rewriting in Formal Language Theory*. Springer, Berlin, 1989.

[5] Dassow, J., Păun, G., and Salomaa, A. *Grammars with controlled derivations*. In Rozenberg, G. and Salomaa, A. (eds.), *Handbook of Formal Languages 2*. Springer, Berlin, pp. 101–154.

[6] Ginsburg, S. and Rice, H. G. *Two families of languages related to ALGOL*. J. Assoc. Comput. Mach. 9 (1962), 350–371.

[7] Ginsburg, S., Greibach, S. A., and Harrison, M. A. *One-way stack automata*. J. Assoc. Comput. Mach. 14 (1967), 389–418.

[8] Parikh, R. J. *On context-free languages*. J. Assoc. Comput. Mach. 13 (1966), 570–581.

[9] Paul, W. J. *Komplexitätstheorie*. Teubner, Stuttgart, 1978.

[10] Rambow, O. *Imposing vertical context conditinos on derivations*. Developments in Language Theory II (DLT 1995), World Scientific, Singapore, 1996, pp. 257–266.

[11] Satta, G. *The membership problem for unordered vector languages*. Developments in Language Theory II (DLT 1995), World Scientific, Singapore, 1996, pp. 267–275.

[12] Sudborough, I. H. *The complexity of the membership problem for some extensions of context-free languages*. Internat. J. Comput. Math. 6 (1977), 191–215.

[13] Sudborough, I. H. *On the tape complexity of deterministic context-free languages*. J. Assoc. Comput. Mach. 25 (1978), 405–414.