On Languages Generated by Signed Grammars

Ömer Eğecioğlu
Department of Computer Science
University of California
Santa Barbara, CA 93106, USA
omer@cs.ucsb.edu

Benedek Nagy
Department of Mathematics, Eastern Mediterranean University
99628 Famagusta, North Cyprus, Mersin-10, Turkey
Department of Computer Science, Institute of Mathematics and Informatics,
Eszterházy Károly Catholic University, Eger, Hungary
nbenedek.inf@gmail.com

We consider languages defined by signed grammars which are similar to context-free grammars except productions with signs associated to them are allowed. As a consequence, the words generated also have signs. We use the structure of the formal series of yields of all derivation trees over such a grammar as a method of specifying a formal language and study properties of the resulting family of languages.

1 Introduction

We consider properties of signed grammars, which are grammars obtained from context-free grammars (CFGs) by allowing right hand sides of productions to have negative signs in front. The concept of generation for such grammars is somewhat different from that of context-free grammars. A signed grammar is said to generate a language $\mathscr{L}$ if the formal sum of the yields over all derivation trees over the grammar corresponds to the list of words in $\mathscr{L}$. For a signed grammar, the yields of derivation trees may have negative signs attached to them, but the requirement is that when the arithmetic operations are carried out in the formal sum, the only remaining words are those of $\mathscr{L}$, each appearing with multiplicity one.

The structure of context-free languages (CFLs) under a full commutation relation defined on the terminal alphabet is the central principle behind Parikh's theorem [24]. In partial commutation, the order of letters of some pairs of the terminal alphabet is immaterial, that is, if they appear consecutively, the word obtained by swapping their order is equivalent to the original one. These equivalence classes are also called traces and studied intensively in connection to parallel processes [18, 12, 21, 4]. Our motivation for this work is languages obtained by picking representatives of the equivalence classes in Σ^* under a partial commutativity relation, called Cartier-Foata languages [1]. In the description of these languages with Kleene-closure type expansions, words appear with negative signs attached to them. However such words are cancelled by those with positive signs, leaving only the sum of the words of the language. An example of this is $(a+b-ba)^*$ which is more familiarly denoted by the regular expression a^*b^*. The interesting aspect of Cartier-Foata languages is that the words with negative signs cancel out automatically, leaving only the representative words, each appearing exactly once.

Motivated by these languages, we consider grammars which are obtained from context-free grammars by allowing signed productions, i.e., normal productions (in the role of positive productions) and productions of the form $A \to -\alpha$ (negative productions). In this way, a derivation results in a signed word where the sign depends on the parity of the number of negative rules applied in the derivation. We consider those derivations equivalent that belong to the same derivation tree, and actually, the derivation tree itself defines the sign of the derived word. The language generated by such a grammar is obtained by taking all possible derivation trees for a given word (both its positive and negative derivations) and requiring that the sum of the yields of all derivation trees over the grammar simply is a list of the words in a language $\mathscr{L}$. This means that the simplified formal sum is of the form $\sum_{w\in\mathscr{L}} w$, each word of the language appearing with multiplicity one. (Without loss of generality, in this study, we restrict ourselves to grammars having finitely many parse trees for each of the derived words.)

On one hand, the requirements in the specification of a language generated by a signed grammar may seem too restrictive. But at the same time this class of languages includes all unambiguous context-free languages and it is closed under complementation, and consequently can generate languages that are not even context-free. Therefore it is of interest to consider the interplay between the restrictions and various properties of languages generated by signed grammars.

2 Preliminaries

Given a language $\mathscr{L}$ over an alphabet Σ, we identify $\mathscr{L}$ with the formal sum of its words denoted by $f(\mathscr{L})$:

$$f(\mathscr{L}) = \sum_{w\in\mathscr{L}} w\,. \tag{1}$$

The sum in (1) is also referred to as the *listing series* of $\mathscr{L}$. A *weighted series of* $\mathscr{L}$ is a formal series of the form $\sum_{w\in\mathscr{L}} n_w w$ where n_w are integers. Thus a weighted series of Σ^*

$$\sum_{w\in\Sigma^*} n_w w$$

is the listing series of some language $\mathscr{L}$ over Σ iff

$$n_w = \begin{cases} 1 & \text{if } w \in \mathscr{L} \\ 0 & \text{if } w \notin \mathscr{L}\,. \end{cases} \tag{2}$$

We are allowed ordinary arithmetic operations on weighted series in a natural way. The important thing is that a weighted series is the listing series of a language $\mathscr{L}$ iff the coefficients of the words in $\mathscr{L}$ in the weighted series are 1, and all the others are 0. So for example over $\Sigma = \{a,b,c\}$, the weighted series $a+b+c+ba$ is the listing series of the finite language $\mathscr{L} = \{a,b,c,ba\}$, whereas the weighted series $a+b+c-ba$ does not correspond to a language over Σ. This is because in the latter example n_w does not satisfy (2) for $w = ba$. As another example, the difference of the weighted series $2a+3b-c+ba$ and $a+2b-2c+ba$ corresponds to the language $\mathscr{L} = \{a,b,c\}$.

2.1 CFGs and degree of ambiguity

Next we look at the usual CFGs $G = (V,\Sigma,P,S)$. Here the start symbol is $S \in V$. Let T be a parse (derivation) tree over G with root label S and terminal letters as labels of the leaves of T. Let $Y(T) \in \Sigma^*$ be the *yield* of T. Then the language generated by G is

$$\mathscr{L}(G) = \{Y(T) \mid T \text{ is a parse tree over } G\}\,.$$

This is equivalent to $\mathscr{L}(G) = \{w \in \Sigma^* \mid S \overset{*}{\underset{G}{\Longrightarrow}} w\}$. For a CFG G, we can define the formal weighted sum

$$f(G) = \sum_{T\in\mathscr{T}_G} Y(T) = \sum_{w\in\Sigma^*} n_w w \tag{3}$$

where $\mathscr{T}_G$ denotes all parse trees over G. Various notions of ambiguity for CFLs can be interpreted as the nature of the coefficients n_w that appear in (3). Rewriting some of the definitions in Harrison [8, pp. 240-242] in terms of these coefficients, we have

1. Given $k \geq 1$, G is *ambiguous of degree* k if $n_w \leq k$ for all $w \in \mathscr{L}(G)$.
2. $\mathscr{L}$ is *inherently ambiguous of degree* $k \geq 2$ if $\mathscr{L}$ cannot be generated by any grammar that is ambiguous of degree less than k but can be generated by by a grammar that is ambiguous of degree k. In other words the degree of ambiguity of a CFL is the least upper bound for the number of derivation trees which a word in the language can have.
3. $\mathscr{L}$ is *finitely inherently ambiguous* if there is some k and some G for $\mathscr{L}$ so that G is inherently ambiguous of degree k.
4. A CFG G is *infinitely ambiguous* if for each $i \geq 1$, there exists a word in $\mathscr{L}(G)$ which has at least i parse trees. A language L is *infinitely inherently ambiguous* if every grammar generating L is infinitely ambiguous.

The CFL $\mathscr{A} = \{a^i b^j c^k \mid i = j \text{ or } j = k\}$ is inherently ambiguous of degree 2 [8, p. 240], $\mathscr{A}^m$ is inherently ambiguous of degree 2^m [8, Theorem 7.3.1], and $\mathscr{A}^*$ is infinitely inherently ambiguous [8, Theorem 7.3.3]. Another interesting CFL which is infinitely inherently ambiguous is Crestin's language [3] of double palindromes over a binary alphabet $\{w_1 w_2 \mid w_1, w_2 \in \{a,b\}^*, w_1 = w_1^R, w_2 = w_2^R\}$. Furthermore, for every $k \geq 1$, there exist inherently ambiguous CFLs of degree k. The behavior of the sequence n_w over all CFGs for a language was studied by Wich [25, 26].

3 Signed grammars

We consider *signed grammars* G which are like CFGs but with a sign associated with each production, that is, apart from the usual (say positive) productions, we allow productions of the form $A \rightarrow -\alpha$. In the derivation relation we use the signs as usual in a multiplicative manner: We start the derivation from the sentence symbol (with $+$ sign, but as usual we may not need to put it, as it is the default sign). The derivation steps, as rewriting steps, occur as they are expected in a CFG, the only extension is that we need to deal with also the sign. When a positive production is applied in a sentential form, its sign does not change, while whenever a negative production is applied, this derivation step switches the sign of the sentential form. Thus, in this case the yield of a parse tree of G is a word over Σ with a $\pm$ sign attached to it. Furthermore, the sign of a derived word depends only on the parity of the number of negative productions used during its derivation. Therefore, different derivation trees for the same word may lead to the word with different signs attached to it. We note that, in fact, any CFG is a signed grammar. For a signed grammar G, let $f(G)$ be defined as in (3), where again $\mathscr{T}_G$ denotes all parse trees over G. Without loss of generality, we may assume that in the grammar G there are only finitely many parse trees for any of the words generated by the grammar.

Definition 1 *We say that a signed grammar G* generates *a language $\mathscr{L}$ iff the weighted series $f(G)$ in (3) is the listing series of $\mathscr{L}$, i.e. $f(G) = f(\mathscr{L})$.*

3.1 Examples of languages generated by signed grammars

Example 1 For the signed grammar G_1 with start symbol A and productions $A \rightarrow -aA \mid \lambda$, we have

$$f(G_1) = \sum_{i \geq 0} a^{2i} - \sum_{i \geq 0} a^{2i+1}. \tag{4}$$

Therefore the signed grammar G with productions $S \to A \,|\, B$, $A \to -aA \,|\, \lambda$, $B \to aaB \,|\, a$ generates the regular language $(aa)^*$. As this is our first example, we provide details of the derivations in G:

- The empty word λ can be derived only in one way, by applying a positive production, thus it is in the language.
- By applying a negative and a positive production, $S \Rightarrow A \Rightarrow -aA \Rightarrow -a$ yields $-a$, and $S \Rightarrow B \Rightarrow a$ yields $+a$. These two are the only derivations over G for $\pm a$. This means that the word a is not in the language.
- For the word aa, the only derivation is $S \Rightarrow A \Rightarrow -aA \Rightarrow aaA \Rightarrow aa$. Consequently aa is in the generated language.
- Finally, by induction, one can see that an even number of a-s can only be produced by starting the derivation by $S \Rightarrow A$. Following this positive production, each usage of $A \to -aA$ introduces a negative sign. Therefore each word of the form a^{2i} is generated once this way with a $+$ sign. On the other hand there are two possible ways to produce a string a^{2i+1} of an odd number of a-s. One of these starts with $A \Rightarrow -aA$ as before and produces $-a^{2i+1}$ after an odd number of usages of $A \to -aA$; the other one starts with $S \Rightarrow B$ and produces a^{2i+1} after an even number of applications of $B \to aaB$, followed by $B \to a$. Therefore odd length words cancel each other out and are not in the language generated.

Another way to look at this is to note that for the (signed) grammar G_2 with the start symbol B and productions $B \to aaB \,|\, a$, we have

$$f(G_2) = \sum_{i \geq 0} a^{2i+1}, \tag{5}$$

and the words generated by G are given by the formal sum of (4) and (5).

Example 2 The signed grammar with productions $S \to aS \,|\, bS \,|\, -baS \,|\, \lambda$ generates the regular language denoted by the regular expression a^*b^*. First few applications of the productions give

$$\begin{aligned}&\lambda;\\ &a+b-ba;\\ &a^2+ab-aba+ba+b^2-b^2a-ba^2-bab+baba;\end{aligned}$$

in which the only immediate cancellation is of $-ba$, though all words carrying negative signs will eventually cancel out. This is a special case of the Cartier-Foata result [1], [5, Section 8.4].

Example 3 Over the decimal (or the binary) alphabet we can construct an unambiguous regular grammar G that generates all nonnegative even numbers, e.g., $S \to 9S \,|\, 8A \,|\, 7S \,|\, 6A \,|\, 5S \,|\, 4A \,|\, 3S \,|\, 2A \,|\, 1S \,|\, 0A$ and $A \to 9S \,|\, 8A \,|\, 7S \,|\, 6A \,|\, 5S \,|\, 4A \,|\, 3S \,|\, 2A \,|\, 1S \,|\, 0A \,|\, \lambda$. Let, further, a regular grammar G' be generating the numbers which are divisible by 6 (e.g., based on the deterministic finite automaton checking the sum of the digits to be divisible by 3 and the last digit must be even, we need states/nonterminals to count the sum of already read digits by mod 3 and take care to the last digit as we did for G).

Then $\mathscr{L}(G)$ consists of all even numbers and $\mathscr{L}(G')$ consists of all numbers divisible by 6. Now, from G', we may make a signed grammar G'' which allows us to derive every multiple of 6 with the sign $-$. Then by combining the two grammars G and G'', we can easily give a signed grammar that generates all even numbers that are not divisible by 3 (i.e., even numbers not divisible by 6).

Example 4 Over the alphabet $\{a,b\}$ consider the signed grammar with productions $S \to aSa \,|\, bSb \,|\, a \,|\, b$. This so far generates odd length palindromes. Let us add the productions $S \to -A,\ A \to -abAba \,|\, a$.

Then each odd length palindrome with the letter b in the middle has exactly one derivation tree with a $+$ sign. There are no cancellations for these and therefore all odd length palindromes with b in the middle are in the language. If the middle of an odd length palindrome w is a but not $ababa$, then w is not in $\mathscr{L}$ as it has also derivation tree with $-$ sign. Similarly, if the middle of w is $ababa$ but not $ababababa$, w is in $\mathscr{L}$. In general, if an odd length palindrome w has $(ab)^{2k-1}a(ba)^{2k-1}$ in the middle, but it does not have $(ab)^{2k}a(ba)^{2k}$ in its middle, then it is in $\mathscr{L}$. Here the number of derivation trees for a word with a $+$ sign is either equal to the number of derivation trees with a $-$ sign for the word, or it is exactly one more.

Example 5 For the following signed grammar

$$\begin{aligned} S_1 &\to -aA \,|\, Ba \,|\, a \\ A &\to -aA \,|\, Ba \,|\, a \\ B &\to -aB \,|\, Ba \,|\, -a \,|\, aa \end{aligned}$$

for n odd, there are 2^{n-1} parse trees for a^n and $2^{n-1}-1$ parse trees for $-a^n$. For n even, there are $2^{n-1}-1$ parse trees for a^n and 2^{n-1} parse trees for $-a^n$. In other words for the above grammar

$$\begin{aligned} f(G) &= \sum_{i\geq 0} 2^{2i}a^{2i+1} + \sum_{i\geq 0}(2^{2i}-1)a^{2i} - \sum_{i\geq 0}(2^{2i}-1)a^{2i+1} - \sum_{i\geq 0} 2^{2i}a^{2i} \\ &= \sum_{i\geq 0}(-1)^i a^{i+1}. \end{aligned}$$

If we add the productions $S \to S_1 \,|\, S_2,\ S_2 \to aaS_2 \,|\, aa$ then the resulting signed grammar generates the regular language $a(aa)^*$. Even though the language generated is very simple we see that signed grammars possess some interesting behavior.

4 Properties of languages generated by signed grammars

In this section our aim is twofold. On the one hand we give some closure properties of the class of languages generated by our new approach and, on the other hand, we give hierarchy like results by establishing where this family of languages is compared to various other classes.

We immediately observe that in the weighted sum (3) for a CFG G (i.e. a signed grammar G with no signed productions), the coefficient n_w is the number of parse trees for w over G, in other words the degree of ambiguity of w.

Proposition 1 *Any unambiguous CFL is generated by a signed grammar.*

Proof An unambiguous CFL $\mathscr{L}$ is generated by the signed grammar G where G is any unambiguous CFG for $\mathscr{L}$. ●

As the class of unambiguous CFLs contains all deterministic CFLs, $LR(0)$ languages, regular languages, subsets of $w_1^*w_2^*$ [7, Theorem 7.1], all of these languages are generated by signed grammars. Further, all these classes are proper subsets of the class of languages generated by signed grammars.

Now we present a closure property.

Proposition 2 *Languages generated by signed grammars are closed under complementation.*

Proof Take an unambiguous CFG for Σ^* with start symbol S_1. If $\mathscr{L}$ is generated by a signed grammar with start symbol S_2 (and no common nonterminal in the two grammars), then the productions of the two grammars together with $S \to S_1 \mid -S_2$ with a new start symbol S generates $\overline{\mathscr{L}}$. ●

We continue the section comparing our new class of languages with other well-known language class, the class of CFLs.

In 1966 Hibbard and Ullian constructed an unambiguous CFL whose complement is not a CFL [9, Theorem 2]. Recently Martynova and Okhotin constructed an unambiguous linear language whose complement is not context-free [14]. This shows that unambiguous linear CFLs are not closed under complementation while providing another proof of Hibbard and Ullian's result.

We know that languages generated by signed grammars are closed under complementation, and also every unambiguous CFL is generated by a signed grammar. A consequence of this is that signed grammars can generate languages that are not context-free.

Proposition 3 *There is a language generated by a signed grammar that is not context-free.*

Proof If $\mathscr{L}$ is the unambiguous CFL constructed by Hibbard and Ullian, then $\mathscr{L}$ and therefore $\overline{\mathscr{L}}$ are generated by signed grammars. But we know that $\overline{\mathscr{L}}$ is not context-free. ●

Actually, our last proposition shows that the generative power of signed grammars is surprisingly large, it contains, e.g., all deterministic and unambiguous CFLs and their complements. Thus, one can easily generate some languages that are not in the class of CFLs.

Continuing with closure properties, recall that disjoint union is an operation that is defined only on disjoint sets which produces their union.

Proposition 4 *Languages generated by signed grammars are closed under disjoint union* $\uplus$.

Proof Let $\mathscr{L}_1$ and $\mathscr{L}_2$ be two languages over an alphabet Σ such that $\mathscr{L}_1 \cap \mathscr{L}_2 = \emptyset$. Let $\mathscr{L}_1$ be generated by a signed grammar with start symbol S_1 and $\mathscr{L}_2$ be generated by a signed grammar with start symbol S_2, such that the sets of nonterminals of these two grammars are disjoint. Then the productions of the two grammars together with $S \to S_1 \mid S_2$ with a new start symbol S generates the disjoint union $\mathscr{L}_1 \uplus \mathscr{L}_2$. ●

Now, let us define the set theoretical operation "subset minus" ($\ominus$), as follows: let $A \subseteq B$, then $B \ominus A = B \setminus A$. This type of setminus operation is defined only for sets where the subset condition holds.

Proposition 5 *Languages generated by signed grammars are closed under subset minus* $\ominus$.

Proof Let $\mathscr{L}_1 \subseteq \mathscr{L}_2$ be two languages over a given alphabet Σ. Take the signed grammar for $\mathscr{L}_1$ with start symbol S_1. If $\mathscr{L}_2$ is generated by a signed grammar with start symbol S_2 (with no common nonterminals of the two grammars), then the productions of the two grammars together with $S \to S_1 \mid -S_2$ with a new start symbol S generates the language of $\mathscr{L}_2 \ominus \mathscr{L}_1$. ●

Let $\mathscr{L}_1, \mathscr{L}_2 \subseteq \Sigma^*$ be two languages and $\$ \notin \Sigma$. The \$-concatenation of $\mathscr{L}_1$ and $\mathscr{L}_2$ is the language $\mathscr{L}_1\$\mathscr{L}_2$ over the alphabet $\Sigma \cup \{\$\}$.

Proposition 6 *Languages generated by signed grammars are closed under* \$*-concatenation.*

Proof The language $\mathscr{L}_1\$$ has the prefix property (i.e. it is prefix-free) due to the special role of the marker \$. Let G_1 and G_3 be signed grammars with disjoint variables and start symbols S_1 and S_3 that generate $\mathscr{L}_1$ and $\mathscr{L}_2$, respectively. Consider also the signed grammar G_2 with the single production

$S_2 \to \$$. Then the signed grammar which have all the productions of G_1, G_2, G_3 together with the production $S \to S_1 S_2 S_3$ where S is a new start symbol generates the language $\mathscr{L}_1 \$ \mathscr{L}_2$. The proof follows by observing that for $u, u' \in \mathscr{L}_1$ and $v, v' \in \mathscr{L}_2$, $u\$v = u'\$v'$ iff $u = u'$ and $v = v'$, so that each word that appears in the expansion of

$$\left(\sum_{w \in \mathscr{L}_1} w \right) \$ \left(\sum_{w \in \mathscr{L}_2} w \right)$$

has coefficient 1. •

In a similar manner, it can also be seen that we have a similar statement for languages over disjoint alphabet, i.e., the class of languages generated by signed grammars is closed under "disjoint concatenation" $\boxdot$.

Proposition 7 *Let $\mathscr{L}_1 \subseteq \Sigma_1^*$ and $\mathscr{L}_2 \subseteq \Sigma_2^*$ be two languages that are generated by signed grammars, where $\Sigma_1 \cap \Sigma_2 = \emptyset$. Then, the language $\mathscr{L}_1 \boxdot \mathscr{L}_2 = \mathscr{L}_1 \mathscr{L}_2$ can be generated by a signed grammar.*

In the following proposition, $f(\mathscr{L})$ and $f(G)$ are as defined in (1) and (3).

Proposition 8 *Suppose $\mathscr{L}$ generated by a signed grammar. Then there are CFGs G_1 and G_2 such that $f(\mathscr{L}) = f(G_1) - f(G_2)$.*

Proof Given a signed grammar over Σ, add an extra letter t to Σ and replace all productions of the form $A \to -\alpha$ by $A \to t\alpha$. The words generated by this CFG over $\Sigma \cup \{t\}$ with an even number of occurrences of t is a CFL since it is the intersection of CFL and the regular language, i.e. all words over $\Sigma \cup \{t\}$ with an even number of occurrences of t. Similarly, the words generated with an odd number of occurrences of t is a CFL. We can then take homomorphic images of these two languages generated by replacing t by λ and obtain two CFLs generated by CFGs G_1 and G_2. The weighted series $f(G)$ is then the difference of two weighted series

$$f(G) = f(G_1) - f(G_2) = \sum_{w \in \Sigma^*} n_w w \; - \; \sum_{w \in \Sigma^*} n'_w w \; . \tag{6}$$

In (6), the coefficients n_w and n'_w are nonnegative integers for all $w \in \Sigma^*$ as they count the number of derivation trees for w over G_1 and G_2, respectively. •

Remark 1 *In Proposition 8, $f(G_1) - f(G_2)$ is the listing series of $\mathscr{L}$, and therefore $n_w - n'_w = 1$ or $n_w - n'_w = 0$ for all $w \in \Sigma^*$. In the first case $w \in \mathscr{L}$, and in the second $w \notin \mathscr{L}$. Note that these conditions do not imply that $\mathscr{L} = \mathscr{L}(G_1) \setminus \mathscr{L}(G_2)$.*

5 Partial commutativity

Addition of commutativity relations to CFGs was considered in [19]. Here we consider partial commutativity defined on Σ^* where $\Sigma = \{x_1, x_2, \ldots, x_m\}$. Given an $m \times m$ symmetric $\{0,1\}$-matrix $A = [a_{i,j}]$ with 1s down the diagonal, a pair of letters x_i, x_j is a commuting pair iff $a_{i,j} = 1$. This defines an equivalence relation and partitions Σ^* into equivalence classes, also known as traces. Thinking about the element of the alphabet as processes and traces as their scheduling, commuting processes are considered as independent from each other. In this way the theory of traces has been intensively studied in connection to parallel processes [11, 12]. A (linearization of a) trace language is a union of some of these equivalence classes. Trace languages based on regular, linear and context-free languages (adding a partial commutativity relation to the language) were studied and accepted by various types of automata with translucent letters in [21, 23, 22], respectively. Traces and trajectories are also analyzed in various grids [15, 16, 20]. On the other hand, the *Cartier–Foata language* $\mathscr{L}_A$ corresponding to the matrix A of a partial commutativity relation is constructed by picking a representative word from each equivalence class.

Let us define a set $F \subseteq \Sigma$ to be *commuting* if any pair of letters in F commute. Let $\mathscr{C}(A)$ denote the collection of all nonempty commuting sets. Denote by $w(F)$ the word obtained by juxtaposing the letters of F. The order in which these letters are juxtaposed is immaterial since all arrangements are equivalent.

The central result is that the listing series $f(\mathscr{L}_A)$ can be constructed directly from the matrix A:

$$f(\mathscr{L}_A) = \left(\sum_{F \in \mathscr{C}(A)} (-1)^{\#F} w(F) \right)^* = \sum_{n \geq 0} \left(\sum_{F \in \mathscr{C}(A)} (-1)^{\#F} w(F) \right)^n , \tag{7}$$

where $\#F$ denotes the number of elements of F.

Over $\Sigma = \{a,b\}$ where a and b commute, the Cartier-Foata theorem gives $\mathscr{L}_A$ as $(a+b-ba)^*$, which is to be interpreted as the weighted series $\lambda + (a+b-ba) + (a+b-ba)^2 + \cdots$ In this case the representatives of the equivalence classes are seen to be the words in a^*b^*. The essence of the theorem is that this is a listing series, so there is exactly one representative word from each equivalence class that remains after algebraic cancellations are carried out.

Similarly over $\Sigma = \{a,b,c\}$ with a,b and a,c commuting pairs, the listing series is $\lambda + (a+b+c-ba-ca) + (a+b+c-ba-ca)^2 + \cdots$

The words in this second language are generated by the signed grammar

$$S \to \lambda \,|\, aS \,|\, bS \,|\, cS \,|\, -baS \,|\, -caS \,.$$

6 Conclusions and a conjecture

Proposition 8 provides an expression for the listing series of a language generated by a signed grammar in terms of weighted listed series of two CFLs. However this result is short of a characterization in terms of CFLs. It is also possible to change the way signed grammars generate languages by requiring $n_w \geq 1$ in (2) instead of equality. In this way, every signed grammar would generate a language, and obviously, the class of generated languages would also change. However, our consideration in this paper to allow only 0 and 1 to be the signed sum, gives a nice and immediate connection to Cartier-Foata languages in the regular case by special regular like expressions.

Since by signed grammars, we generate languages based on counting the number of (signed) derivation trees, it is straightforward to see the connection between our grammars and unambiguous CFLs. On the other hand, there may be more than one derivation tree for a given word w, with the proviso that the algebraic sum of the yields of derivation trees for it has multiplicity $n_w \in \{0,1\}$. Therefore signed grammars may also generate ambiguous CFLs. In this sense, the bottom of the hierarchy, the unambiguous CFLs are included in the class we have investigated. On the other hand, if there are multiple derivation trees for a word generated by a grammar, by playing with their signs, we have a chance to somehow have their signed sum to be in $\{0,1\}$. Thus, it may be possible to generate languages that are higher in the hierarchy based on ambiguity. However, this is still an open problem.

We have shown that signed grammars can generate languages that are not context-free. It would be of interest to use the fact that the languages generated by signed grammars are closed under complementation to show that signed grammars can generate inherently ambiguous CFLs. One way to do this would be to start with an unambiguous CFL whose complement is an inherently ambiguous CFL. The standard examples of inherently ambiguous CFLs do not seem to have this property. By the Chomsky-Schützenberger theorem [2] the generating function of an unambiguous CFL is algebraic. Using the contrapositive and analytical methods, Flajolet [6] and later Koechlin [13] devised ingenious methods to show the transcendence of the generating function of a given language to prove its inherent ambiguity. However if the generating function of $\mathscr{L}$ is transcendental so is the generating function of its complement $\overline{\mathscr{L}}$. This means that one needs to look among inherently ambiguous languages with algebraic generating functions (e.g. $\{a^i b^j c^k \mid i = j \text{ or } j = k\}$, see [13, Proposition 14]) if the complement has any chance of being unambiguous.

So it would be nice to have an answer to the following question: *Is there an unambiguous CFL whose complement is an inherently ambiguous CFL?*

A related problem of showing the existence of an inherently ambiguous CFL whose complement is also an inherently ambiguous CFL was settled by Maurer [17].

References

[1] P. Cartier & D. Foata (1969): *Problèmes combinatoires de commutation et réarrangements*. Springer, doi:10.1007/BFb0079468.

[2] N. Chomsky & M. P. Schützenberger (1963): *The Algebraic Theory of Context-Free Languages*. In P. Braffort & D. Hirschberg, editors: *Computer Programming and Formal Systems*, *Studies in Logic and the Foundations of Mathematics* 35, Elsevier, pp. 118–161, doi:10.1016/S0049-237X(08)72023-8. Available at `https://www.sciencedirect.com/science/article/pii/S0049237X08720238`.

[3] J. P. Crestin (1972): *Un langage non ambigu dont le carré est d'ambiguité non bornée*. In Maurice Nivat, editor: *Automata, Languages and Programming, Colloquium, Paris, France, July 3-7, 1972*, North-Holland, Amsterdam, pp. 377–390. Available at `https://api.semanticscholar.org/CorpusID:44540005`.

[4] V. Diekert & G. Rozenberg, editors (1995): *The Book of Traces*. World Scientific, doi:10.1142/2563.

[5] Ö. Eğecioğlu & A. Garsia (2021): *Lessons in Enumerative Combinatorics*. Springer, Graduate Texts in Mathematics, doi:10.1007/978-3-030-71250-1.

[6] P. Flajolet (1987): *Analytic models and ambiguity of context-free languages*. *Theor. Comput. Sci.* 49(2), pp. 283–309, doi:10.1016/0304-3975(87)90011-9.

[7] S. Ginsburg & J. S. Ullian (1966): *Ambiguity in context free languages*. *J. ACM* 13, pp. 62–89, doi:10.1145/321341.321345.

[8] M. A. Harrison (1978): *Introduction to Formal Language Theory*. Addison-Wesley.

[9] T. N. Hibbard & J. S. Ullian (1966): *The independence of inherent ambiguity from complementedness among context-free languages*. *Journal of the ACM* 13(4), pp. 588–593, doi:10.1145/321356.321366.

[10] J. Hopcroft & J. Ullman (1979): *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley.

[11] Ryszard Janicki, Jetty Kleijn, Maciej Koutny & Lukasz Mikulski (2017): *Invariant Structures and Dependence Relations*. *Fundam. Informaticae* 155(1-2), pp. 1–29, doi:10.3233/FI-2017-1574.

[12] Ryszard Janicki, Jetty Kleijn, Maciej Koutny & Lukasz Mikulski (2019): *Classifying invariant structures of step traces*. *J. Comput. Syst. Sci.* 104, pp. 297–322, doi:10.1016/j.jcss.2017.05.002.

[13] F. Koechlin (2022): *New Analytic Techniques for Proving the Inherent Ambiguity of Context-Free Languages*. In Anuj Dawar & Venkatesan Guruswami, editors: *42nd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2022)*, *Leibniz International Proceedings in Informatics (LIPIcs)* 250, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany,

pp. 41:1–41:22, doi:10.4230/LIPIcs.FSTTCS.2022.41. Available at `https://drops.dagstuhl.de/opus/volltexte/2022/17433`.

[14] O. Martynova & A. Okhotin (2023): *Non-Closure under Complementation for Unambiguous Linear Grammars*. *Inf. Comput.* 292(C), doi:10.1016/j.ic.2023.105031.

[15] Alexandru Mateescu, Grzegorz Rozenberg & Arto Salomaa (1998): *Shuffle on Trajectories: Syntactic Constraints*. *Theor. Comput. Sci.* 197(1-2), pp. 1–56, doi:10.1016/S0304-3975(97)00163-1.

[16] Alexandru Mateescu, Kai Salomaa & Sheng Yu (2000): *On Fairness of Many-Dimensional Trajectories*. *J. Autom. Lang. Comb.* 5(2), pp. 145–157, doi:10.25596/jalc-2000-145.

[17] H. A. Maurer (1970): *A note on the complement of inherently ambiguous context-free languages*. *Commun. ACM* 13, p. 194, doi:10.1145/362052.362065.

[18] A. Mazurkiewicz (1977): *Concurrent Program Schemes and their Interpretations*. *DAIMI Report Series* 6(78), doi:10.7146/dpb.v6i78.7691. Available at `https://tidsskrift.dk/daimipb/article/view/7691`.

[19] Benedek Nagy (2009): *Languages generated by context-free grammars extended by type $AB \to BA$ rules*. *Journal of Automata, Languages and Combinatorics* 14, pp. 175–186, doi:10.25596/jalc-2009-175.

[20] Benedek Nagy & Arif A. Akkeles (2017): *Trajectories and Traces on Non-traditional Regular Tessellations of the Plane*. In Valentin E. Brimkov & Reneta P. Barneva, editors: *Combinatorial Image Analysis - 18th International Workshop, IWCIA 2017, Plovdiv, Bulgaria, June 19-21, 2017, Proceedings*, *Lecture Notes in Computer Science* 10256, Springer, pp. 16–29, doi:10.1007/978-3-319-59108-7_2.

[21] Benedek Nagy & Friedrich Otto (2010): *CD-Systems of Stateless Deterministic R(1)-Automata Accept All Rational Trace Languages*. In Adrian-Horia Dediu, Henning Fernau & Carlos Martín-Vide, editors: *Language and Automata Theory and Applications, 4th International Conference, LATA 2010, Trier, Germany, May 24-28, 2010. Proceedings*, *Lecture Notes in Computer Science* 6031, Springer, pp. 463–474, doi:10.1007/978-3-642-13089-2_39.

[22] Benedek Nagy & Friedrich Otto (2011): *An Automata-Theoretical Characterization of Context-Free Trace Languages*. In Ivana Cerná, Tibor Gyimóthy, Juraj Hromkovic, Keith G. Jeffery, Rastislav Królovic, Marko Vukolic & Stefan Wolf, editors: *SOFSEM 2011: Theory and Practice of Computer Science - 37th Conference on Current Trends in Theory and Practice of Computer Science, Nový Smokovec, Slovakia, January 22-28, 2011. Proceedings*, *Lecture Notes in Computer Science* 6543, Springer, pp. 406–417, doi:10.1007/978-3-642-18381-2_34.

[23] Benedek Nagy & Friedrich Otto (2020): *Linear automata with translucent letters and linear context-free trace languages*. *RAIRO Theor. Informatics Appl.* 54, p. 3, doi:10.1051/ita/2020002.

[24] R. J. Parikh (1961): *Language generating devices*. *MIT Res. Lab., Quarterly Progress Report* 60, pp. 199–212.

[25] K. Wich (2000): *Sublinear Ambiguity*. In Mogens Nielsen & Branislav Rovan, editors: *Mathematical Foundations of Computer Science 2000*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 690–698, doi:10.1007/3-540-44612-5_64.

[26] K. Wich (2005): *Sublogarithmic ambiguity*. *Theoretical Computer Science* 345(2), pp. 473–504, doi:10.1016/j.tcs.2005.07.024.