

FROM INDEXED GRAMMARS TO GENERATING FUNCTIONS*

JARED ADAMS¹, ERIC FREDEN¹ AND MARNI MISHNA²

Abstract. We extend the DSV method of computing the growth series of an unambiguous context-free language to the larger class of indexed languages. We illustrate the technique with numerous examples.

Mathematics Subject Classification. 68Q70, 68R15.

1. INTRODUCTION

1.1. INDEXED GRAMMARS

Indexed grammars were introduced in the thesis of Aho in the late 1960s to model a natural subclass of context-sensitive languages, more expressive than context-free grammars with interesting closure properties [1, 16]. The original reference for basic results on indexed grammars is [1]. The complete definition of these grammars is equivalent to the following reduced form.

Definition 1.1. A *reduced indexed grammar* is a 5-tuple $(\mathcal{N}, \mathcal{T}, \mathcal{I}, \mathcal{P}, \mathbf{S})$, such that

1. \mathcal{N} , \mathcal{T} and \mathcal{I} are three mutually disjoint finite sets of symbols: the set \mathcal{N} of *non-terminals* (also called *variables*), \mathcal{T} is the set of *terminals* and \mathcal{I} is the set of *indices* (also called *flags*);
2. $\mathbf{S} \in \mathcal{N}$ is the start symbol;
3. \mathcal{P} is a finite set of productions, each having the form of one of the following:
(a) $\mathbf{A} \rightarrow \alpha$

Keywords and phrases. Indexed grammars, generating functions, functional equations, DSV method.

* *The third author gratefully acknowledges the support of NSERC Discovery Grant funding (Canada), and LaBRI (Bordeaux) for hosting during the completion of the work.*

¹ Department of Mathematics, Southern Utah University, Cedar City, 84720, UT, USA.

² Department of Mathematics, Simon Fraser University, Burnaby V5A 1S6, BC, Canada.
mmishna@sfu.ca

- (b) $\mathbf{A} \rightarrow \mathbf{B}_f$ (*push*)
 - (c) $\mathbf{A}_f \rightarrow \beta$ (*pop*)
- where $\mathbf{A}, \mathbf{B} \in \mathcal{N}$, $f \in \mathcal{I}$ and $\alpha, \beta \in (\mathcal{N} \cup \mathcal{T})^*$.

Observe the similarity to context-free grammars which are only defined by production rules of type (3a). The language defined by an indexed grammar is the set of all strings of terminals that can be obtained by successively applying production rules beginning with the rule that involves the start symbol \mathbf{S} . A key distinction from context free grammars is that rather than expand non-terminals, we expand non-terminal/stack pairs: (\mathbf{A}, ι) , $\iota \in \mathcal{I}^*$, $\mathbf{A} \in \mathcal{N}$. Here, the start symbol \mathbf{S} is shorthand for the pair (\mathbf{S}, ϵ) , where ϵ denotes the empty stack.

Production rules in \mathcal{P} are interpreted as follows. The stack is implicit, and is copied when the production is applied. For example, the type (3a) production rule $\mathbf{A} \rightarrow a\mathbf{BC}$ is shorthand for $(\mathbf{A}, \iota) \rightarrow a(\mathbf{B}, \iota)(\mathbf{C}, \iota)$, for $\mathbf{A}, \mathbf{B}, \mathbf{C} \in \mathcal{N}$, $a \in \mathcal{T}$ and $\iota \in \mathcal{I}^*$.

A production rule of form (3b) encodes a push onto the stack, and a production rule of the form (3c) encodes a pop off of the stack. For example, the production rule $\mathbf{A} \rightarrow \mathbf{B}_f$ applied to (\mathbf{A}, ι) expands to (\mathbf{B}, ι') where ι' is the stack ι with the character f pushed on. Likewise, $\mathbf{A}_f \rightarrow \beta$ can only be applied to (\mathbf{A}, ι) if the top of the stack string ι is f . The result is β such that any nonterminal $\mathbf{B} \in \beta$ is of the form (\mathbf{B}, ι'') , where ι'' is the stack ι with the top character popped off.

To lighten the notation the stack is traditionally written as a subscript. Note the difference: the presence of a subscript in a production rule is shorthand for an infinite collection of production rules, whereas in a derivation the stack is viewed as part of the symbol. Furthermore, it is also useful to introduce an end of stack symbol, which we write $\$$. This symbol is reserved strictly for the last position in the stack. This permits us to expand a non-terminal into a terminal only when the stack is empty. These subtleties are best made clear through an example.

Example 1.2. The class of indexed languages is strictly larger than the class of context-free languages since it contains the language $\mathcal{L} = \{a^n b^n c^n : n > 0\}$. This language is generated by the indexed grammar $(\{\mathbf{S}, \mathbf{T}, \mathbf{A}, \mathbf{B}, \mathbf{C}\}, \{a, b, c\}, \{f\}, \mathcal{P}, \mathbf{S})$ with

$$\mathcal{P} = \{\mathbf{S} \rightarrow \mathbf{T}_\$, \quad \mathbf{T} \rightarrow \mathbf{T}_f, \quad \mathbf{T} \rightarrow \mathbf{ABC},$$

$$\mathbf{A}_f \rightarrow a\mathbf{A}, \quad \mathbf{A}_\$ \rightarrow a, \quad \mathbf{B}_f \rightarrow b\mathbf{B}, \quad \mathbf{B}_\$ \rightarrow b, \quad \mathbf{C}_f \rightarrow c\mathbf{C}, \quad \mathbf{C}_\$ \rightarrow c\}.$$

A typical derivation is as follows. We begin with \mathbf{S} and derive $aaabbbccc$:

$$\mathbf{S} \rightarrow \mathbf{T}_\$ \rightarrow \mathbf{T}_{f\$} \rightarrow \mathbf{T}_{ff\$} \rightarrow \mathbf{A}_{fff}\mathbf{B}_{ff\$}\mathbf{C}_{ff\$} \rightarrow a\mathbf{A}_{f\$}\mathbf{B}_{ff\$}\mathbf{C}_{ff\$} \rightarrow$$

$$aa\mathbf{A}_\$ \mathbf{B}_{ff\$}\mathbf{C}_{ff\$} \rightarrow aaa\mathbf{B}_{ff\$}\mathbf{C}_{ff\$} \rightarrow aaab\mathbf{B}_f\mathbf{C}_{ff\$} \rightarrow \dots \rightarrow aaabbbccc.$$

The generic structure of a derivation has the following three phases: first, there is an initial push phase to build up the index string; this is followed by a transfer

stage where the stack is copied; finally, there is a pop stage which converts indices into terminal symbols. Most of our examples have this same structure.

This particular grammar is easy to generalize to a language with some fixed number k of symbols repeated n times: $\{a_1^n a_2^n \dots a_k^n : n > 0\}$.

In the case of reduced grammars, at most one index symbol is loaded or unloaded in any production. We use two additional properties which do not restrict the expressive power. A grammar is in *strongly reduced form* as per our definition and if there are no useless non-terminals. That is, every non-terminal \mathbf{V} satisfies both $S \xrightarrow{*} \alpha \mathbf{V}_\sigma \alpha'$ and $\mathbf{V}_\sigma \xrightarrow{*} w$ for $\sigma \in \mathcal{I}^*$, $w \in \mathcal{T}^*$ and $\alpha, \alpha' \in (\mathcal{N} \cup \mathcal{T})^*$. A grammar is ε -free if the only production involving the empty string ε is $\mathbf{S} \rightarrow \varepsilon$. Indexed grammars \mathfrak{G}_1 and \mathfrak{G}_2 are *equivalent* if they produce the same language \mathfrak{L} .

Theorem 1.3 ([21]). *Every indexed grammar \mathfrak{G} is equivalent to some strongly reduced, ε -free grammar \mathfrak{G}' . Furthermore, there is an effective algorithm to convert \mathfrak{G} to \mathfrak{G}' .*

Consequently we can assume all grammars are already strongly reduced (most of our examples are). On the other hand, we have found that ε -productions are a useful crutch in designing grammars (several of our examples are not ε -free).

1.2. THE SET OF INDEXED LANGUAGES

The set of all languages generated by indexed grammars forms the set of indexed languages. As alluded to above, this is a full abstract family of languages which is closed under union, concatenation, Kleene closure, homomorphism, inverse homomorphism and intersection with regular sets. The set of indexed languages, however is not closed under intersection or complement. The standard machine type that accepts the class of indexed languages is the nested stack automaton.

This class of languages properly includes all context-free languages. These are generated by grammars such that \mathcal{I} is empty. One way to view indexed grammars is as an extension of context-free grammars with an infinite number of non-terminals, however the allowable productions are quite structured. Furthermore, indexed languages form a proper subset of the class of context-sensitive languages. For instance $\{(ab^n)^n : n \geq 0\}$ is context-sensitive but not indexed [12].

Formal language theory in general and indexed languages in particular have applications to group theory. Two good survey articles are [25] and [13]. Bridson and Gilman [3] have exhibited indexed grammar combings for fundamental 3-manifold groups based on Nil and Sol geometries (see Example 4.3 below). More recently [15] showed that the language of words in the standard generating set of the Grigorchuk group that do not represent the identity (the so-called *co-word* problem) forms an indexed language. The original DSV method (attributed to Delest, Schützenberger, and Viennot [6]) of computing the growth of a context-free language was successfully exploited [11] to compute the algebraic but non-rational growth series of a family of groups attributed to Higman. One of our goals is to extend this method to indexed grammars to deduce results on growth series.

1.3. ORDINARY GENERATING FUNCTIONS

Generating functions are well-suited to answer enumerative questions about languages over finite alphabets, in particular, the number of words of a given length. For any language \mathfrak{L} with finite alphabet \mathcal{T} , let L_n be the number of words of length n . The *ordinary generating function* of the language is the formal power series $L(z) = \sum_{n \geq 0} L_n z^n$. We use this terminology interchangeably with *growth series*. Note that each L_n is bounded by the number of words of length n in the free monoid \mathcal{T}^* . Consequently, the generating function $L(z)$ has a positive radius of convergence.

One motivation for our study is to understand the enumerative nature of classes of languages beyond context-free, and simultaneously to understand the combinatorial nature of function classes beyond algebraic. To be more precise, it is already known that regular languages have generating functions that can be expressed as rational functions, *i.e.* the Taylor series of a function $P(z)/Q(z)$ where P and Q are polynomials. Furthermore, unambiguous context free languages have algebraic generating functions, that is, they satisfy $P(L(z), z)$ for some bivariate polynomial $P(x, y)$ with integer coefficients. This kind of generating function property has consequences on the asymptotic form, and can be used to exclude languages from the companion classes, by analytic arguments on the generating function. For example, Flajolet [8] proves the inherent ambiguity of several context-free languages by demonstrating the transcendence of their generating functions.

There are two natural contenders for function classes that may capture indexed grammars: *D-finite* and *differentiably algebraic*. A series is said to be *D-finite* if it satisfies a homogeneous linear differential equation with polynomial coefficients. A series $L(z)$ is said to be *differentiably algebraic* if there is a non-trivial $k+1$ -variate polynomial $P(x_0, x_1, \dots, x_k)$ with integer coefficients such that

$$P\left(L(z), \frac{d}{dz}L(z), \frac{d^2}{dz^2}L(z), \dots, \frac{d^k}{dz^k}L(z)\right) \equiv 0.$$

We prove that neither of these classes capture indexed grammars. In fact, many of our examples of indexed grammars have lacunary generating functions, with a natural boundary at the unit circle, because they are so sparse. This is perhaps unsatisfying, but it also illustrates a key difference between computational complexity and analytic complexity; a distinction which is not evident after studying only context-free and regular languages.

That said, the expressive power of growth series derived from indexed languages has been broached by previous authors. In [17], the authors consider a limitation on possible productions, which is close to one of the restrictions we consider below. They are able to describe the recurrence type satisfied by all sequences $u(n)$ for which the corresponding language $\{a^{u(n)}\}$ is generated by this restricted set of indexed grammars.

Furthermore, we mention that other characterizations of indexed languages are equally amenable to analysis. In particular, indexed languages are equivalent to

sequences of level 2 in the sense of [26], and have several different descriptions. Notably, they satisfy particular systems of catenative recurrent relations, to which methods comparable to what we present here may apply. By taking combinations of such sequences, Fratani and Senizergues [10] can give a characterization of D-finite functions with rational coefficients.

A second motivation for the present work is to verify that various growth rates are achievable with an indexed grammar. To that end, we often start with a language where the enumeration is trivial, but the challenge is to provide an indexed grammar that generates it. Our techniques verify that the desired growth rate has been achieved.

1.4. SUMMARY

Ideally, we would like to describe an efficient algorithm to determine the generating function of an indexed language given only a specification of its indexed grammar. Towards this goal we first describe a process in the next section that works under some conditions including only one stack symbol (excluding the end of stack symbol). Proposition 2.7 summarizes the conditions, and the results. We have several examples to illustrate the procedure. In Section 3 this is generalized to multiple stack symbols that are pushed in order. In this section we also illustrate the inherent obstacles in the case of multiple stack symbols. This is followed by some further examples from number theory in Section 4 and a discussion in Section 5 on inherent ambiguity in indexed grammars.

1.5. NOTATION

Throughout, we use standard terminology with respect to formal language theory. The expression $x|y$ denotes “ x exclusive-or y ”. We use epsilon “ ε ” to denote the empty word. The Kleene star operator applied to x , written x^* means make zero or more copies of x . A related notation is x^+ which means make one or more copies of x . The word reversal of w is indicated by w^R . The length of the string x is denoted $|x|$. We print grammar variables in upper case bold letters. Grammar terminals are in lower case italic. We use the symbol $\xrightarrow{*}$ to indicate the composition of two or more grammar productions.

2. THE CASE OF ONE INDEX SYMBOL

2.1. GENERALIZING THE DSV PROCESS

A central objective of this work is to answer enumerative questions about the number of words of a given length in a language defined by an indexed grammar. It turns out that in many cases the classic translation of production rules into functional equations satisfied by generating functions works here. This type of strategy was first applied to formal languages in [5], and the ideas have been expanded to handle more general combinatorial equations [9]. We summarize the

process below. The challenge posed by indexed grammars is that an infinite number of equations are produced by the process. Nonetheless, we identify some sufficiency criteria which allow us to solve the system in some reasonable way.

First, let us recall the translation process for context-free grammars. Let \mathfrak{G} be a context-free grammar specification for a non-empty language \mathfrak{L} , with start symbol \mathbf{S} . For each production, replace each terminal with the formal variable z and each non-terminal \mathbf{A} with a formal power series $A(z)$. Translate the grammar symbols $\rightarrow, |, \varepsilon$ into $=, +, 1$, respectively, with juxtaposition becoming commutative multiplication. Thus the grammar is transformed into a system of equations. We summarize the main results of this area as follows.

Theorem 2.1 (Chomsky–Schützenberger). *Each formal power series $A(z) = \sum A_n z^n$ in the above transformation is an ordinary generating function where A_n is an integer representing the number of word productions of length n that can be realized from the non-terminal \mathbf{A} . In particular, if the original context-free grammar is unambiguous, then $S(z)$ is the growth series for the language \mathfrak{L} , in which case $S(z)$ is an algebraic function.*

A context-free grammar has only finitely many non-terminals. In the case of an indexed grammar, we treat a single variable \mathbf{A} as housing infinitely many non-terminals, one for each distinct index string carried by \mathbf{A} (although only finitely many are displayed in parsing any given word). To generalize the DSV procedure to indexed grammars we apply the same transformation scheme to the grammar, under the viewpoint that every production rule is shorthand for an infinite set of productions, where non-terminals are paired with index strings. The generating functions are thus also similarly indexed, for example $\mathbf{A}_{gfgghgfs}$ gives rise to $A_{gfgghgfs}(z)$.

Initially, this is most unsatisfying, since the transformation recipe produces a system of *infinitely many equations in infinitely many functions!* We are unable to describe a general scheme to solve these equations, and even in some small cases we do not obtain satisfying expressions (see Example 2.9 below). However, if there is only one index symbol (disregarding the end of stack symbol), and some other conditions are satisfied, we can outline a procedure to reduce the system. When there is only one stack symbol, it is sufficient to identify only the size of the stack in a non-terminal, stack pair. For example, in the translation to functions $\mathbf{A}_{ffffffs}$ becomes $A_6(z)$.

Example 2.2. The language $\mathfrak{L}_{\text{sqr}} = \{a^{2^n} : n \geq 0\}$ is generated by an indexed grammar. The enumeration for this example is trivial but the example serves a pedagogical purpose, as it sets up the process in the case of one index symbol, and provides an example of an indexed language whose generating function is not differentially algebraic.

As usual, we use $\$$ to indicate the bottom-most index symbol. Disregarding this, there is only one index symbol actually used, and so in the translation process we note only the size of the stack, not its contents. Furthermore we identify $S_0(z) = S(z)$.

$$\begin{aligned}
 \mathbf{S} &\rightarrow \mathbf{T}_{\mathfrak{S}} & \mathbf{T} &\rightarrow \mathbf{T}_f | \mathbf{D} & \mathbf{D}_f &\rightarrow \mathbf{D}\mathbf{D} & \mathbf{D}_{\mathfrak{S}} &\rightarrow a \\
 S_0(z) &= T_0(z) & T_n(z) &= T_{n+1}(z) + D_n(z) & D_{n+1}(z) &= D_n(z)^2 & D_0(z) &= z.
 \end{aligned}$$

Observe that indices are loaded onto \mathbf{T} then transferred to \mathbf{D} which is then repeatedly doubled (\mathbf{D} is a mnemonic for “duplicator”). After all doubling, each instance of $\mathbf{D}_{\mathfrak{S}}$ becomes an a .

Immediately we solve $D_n(z) = D_0(z)^{2^n} = z^{2^n}$, and the system of grammar equations becomes

$$S_0(z) = T_0(z) = T_1(z) + D_0(z) = T_2 + D_1 + D_0 = \dots = \sum_{n \geq 0} D_n(z) = \sum_{n \geq 0} z^{2^n}.$$

We observe that the sequence of partial sums converges as a power series inside the unit circle, and that the $T_n(z)$ are incrementally eliminated. We refer to this process, summarized in Proposition 2.4 below, as *pushing the $T_n(z)$ off to infinity*.

The function $S(z)$ satisfies the functional equation $S(z) = z + S(z^2)$. The series diverges at $z = 1$, and hence $S(z)$ is singular at $z = 1$. However, by the functional equation it also diverges at $z = -1$. By repeated application of this argument, we can show that $S(z)$ is singular at every $2^n - th$ root of unity. Thus it has an infinite number of singularities, and it cannot be D-finite. In fact, $S(z)$ satisfies no algebraic differential equation [19]. Consequently, the class of generating functions for indexed languages is not contained in the class of differentially algebraic functions.

2.2. A STRAIGHTFORWARD CASE: BALANCED INDEXED GRAMMARS

Next, we describe a condition that allows us to guarantee that this process will result in a simplified generating function expression for $S(z)$.

Definition 2.3. An indexed grammar is *balanced* provided there are constants $C, K \geq 0$, depending only on the grammar, such that the longest string of indices associated to any non-terminal in any sentential form \mathcal{W} has length at most $C|w| + K$ where w is any terminal word produced from \mathcal{W} . (Note: in all our balanced examples we can take $C = 1$ and $K \in \{0, 1\}$).

Proposition 2.4. Let $\mathfrak{G} = (\mathcal{N}, \mathcal{T}, \mathcal{I}, \mathcal{P}, \mathbf{S})$ be an unambiguous, balanced, indexed grammar in strongly reduced form for some non-empty language \mathfrak{L} with $\mathcal{I} = \{f\}$. Furthermore, suppose that $\mathbf{V} \in \mathcal{N}$ is the only non-terminal that loads f and that the only allowable production in which \mathbf{V} appears on the right side is $\mathbf{S} \rightarrow \mathbf{V}_{\mathfrak{S}}$. Then in the generalized DSV equations for \mathfrak{G} , the sequence of functions $V_n(z) \equiv V_{f^n}(z)$ can be eliminated (pushed to infinity). Under these hypotheses, the system of equations defining $S(z)$ reduces to finitely many bounded recurrences with initial conditions whose solution is the growth function for \mathfrak{L} .

Proof. By hypothesis, there is only one load production and it has form $\mathbf{V} \rightarrow \mathbf{V}_f$. Without loss of generality we may assume there is no type (3c) rule $\mathbf{V}_f \rightarrow \beta$ (in fact, such a rule can be eliminated by the creation of a new variable \mathbf{U} and adding new productions $\mathbf{V} \rightarrow \mathbf{U}$ and $\mathbf{U}_f \rightarrow \beta$). Necessarily there will be at least one context-free type rule $\mathbf{V} \rightarrow \beta$ (else \mathbf{V} is a useless symbol).

Consider all productions in \mathfrak{G} that have \mathbf{V} on the left side. Converting these productions into the usual functional equations and solving for $V_n(z)$ gives an equation of form

$$V_n(z) = V_{n+1}(z) + W_{n\pm e}(z)$$

where $W_{n\pm e}(z)$ denotes an expression that represents all other grammar productions having \mathbf{V} on the left side and $e \in \{0, 1, -1\}$.

We make the simplifying assumption that $e = 0$ for the remainder of this paragraph. Starting with $n = 0$ and iterating $N \gg 0$ times yields $V_0(z) = V_N(z) + W_0(z) + W_1(z) + \dots + W_N(z)$. By the balanced hypothesis, there exists a constants $C, K \geq 0$ such that all terminal words produced from \mathbf{V}_f^N have length at least $N/C - K \gg 0$. This means that the first $N/C - K$ terms in the ordinary generating function for $V_0(z)$ are unaffected by the contributions from $V_N(z)$ and depend only on the fixed sum $W_0(z) + W_1(z) + \dots + W_N(z)$. Therefore the $(N/C - K)^{th}$ partial sum defining the generating function for $V_0(z)$ is stabilized as soon as the iteration above reaches $V_N(z)$. This is true for all big N , so we may take the limit as $N \rightarrow \infty$ and express $V_0(z) = \sum_{n \geq 0} W_n(z)$.

Allowing $e = \pm 1$ in the previous paragraph merely shifts indices in the sum and does not affect the logic of the argument. Therefore, in all cases the variables $V_n(z)$ for each $n > 0$ are eliminated from the system of equations induced by the grammar \mathfrak{G} . We assumed that \mathbf{V} was the only variable loading indices, so all other grammar variables either unload/pop indices or are terminals. Consequently, the remaining functions describe a finite triangular system, with finitely many finite recurrences of bounded depth and known initial conditions. The solution of this simplified system is $S(z)$. □

We observe that the expression for $V_0(z)$ derived above actually converges as an analytic function in a neighborhood of zero (see Sect. 1.3 above). It turns out that the balanced hypothesis used above is already satisfied.

Lemma 2.5. *Suppose the indexed grammar \mathfrak{G} is unambiguous, strongly reduced, ϵ -free, and has only one index symbol f (other than $\$$). Then \mathfrak{G} is balanced.*

Proof. If the language produced by \mathfrak{G} is finite, there is nothing to prove so we may assume the language is infinite. Let us define a special sequence of grammar productions used in producing a terminal word. Suppose a sentential form contains several variables, each of which is ready for unloading of the index symbol f . A *step* will consist of unloading a single f from each of these non-terminals, starting from the leftmost variable. After the step, each of these variables will hold an index string that is exactly one character shorter than before.

Consider a sentential form F containing one or more non-terminals in the unloading stage and each of whose index strings are of length at least $N \gg 0$. These symbols can only be unloaded at the rate of one per production rule (this is the reduced hypothesis) and we'll only consider changing F by steps.

On the other hand there, are only finitely many production rules to do this unloading of finitely many variables. Thus for large N there is a cycle of production rules as indices are unloaded, with cycle length bounded by a global constant $C > 0$ which depends only on the grammar. Furthermore, this cycle is reached after at most $K < C$ many productions. Let F' denote the sentential form that results from F as one such cycle is begun and let F'' be the sentential form after the cycle is applied to F' .

Consider lengths of sentential forms (counting terminals and variables but ignoring indices). Since the grammar is reduced and ε -free, each grammar production is a non-decreasing function of lengths. Thus $|F''| \geq |F'| \geq |F|$. Discounting any indices, the equality of sentential forms $F'' = F'$ is not possible because this implies ambiguity.

We claim that either F'' is longer than F' or that F'' has more terminals than F' . If not, then F'' has exactly the same terminals as F' , and each has the same quantity of variables. There are only finitely many arrangements of terminals and variables for this length and for large N we may loop stepwise through the production cycle arbitrarily often and thus repeat exactly a sentential form (discounting indices). This implies our grammar is ambiguous contrary to hypothesis.

Thus after C steps the sentential forms either obtain at least one new terminal or non-terminal. In the latter case, variables must convert into terminals on $\$$ (*via* the reduced, unambiguous, ε -free hypotheses). There will be at least one terminal per step in the final output word w . We obtain the inequality $(N - K)/C \leq |w|$ which establishes the lemma. \square

2.3. A COLLECTION OF EXAMPLES

We illustrate the method, its power and its limitations with three examples. The first two examples exhibit languages with intermediate growth and show that some of the hypotheses of Proposition 2.4 can be relaxed. We are unable to resolve the third example to our satisfaction.

The first example is originally due to [14] and features the indexed language

$$\mathfrak{L}_{G/M} = \{ab^{i_1}ab^{i_2} \dots ab^{i_k} : 0 \leq i_1 \leq i_2 \leq \dots \leq i_k\}$$

with intermediate growth (meaning that the number of words of length n ultimately grows faster than any polynomial in n but more slowly than 2^{kn} for any constant $k > 0$). The question of whether a context-free language could have this property was asked in [8] and answered in the negative [4, 18]. Grigorchuk and Machí constructed their language based on the generating function of Euler's partition function. A word of length n encodes a partition sum of n . For instance, the partitions of $n = 5$ are $1 + 1 + 1 + 1 + 1$, $1 + 1 + 1 + 2$,

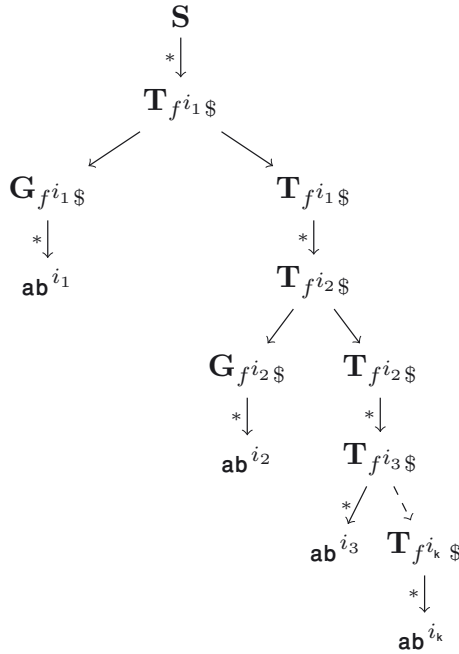


FIGURE 1. A typical parse tree in the grammar
 $S \rightarrow T_\$$ $T \rightarrow T_f|GT|G$ $G_f \rightarrow Gb$ $G_\$ \rightarrow a$.

1 + 2 + 2, 1 + 1 + 3, 2 + 3, 1 + 4, 5. The corresponding words in $\mathfrak{L}_{G/M}$ are *aaaaa*, *aaaab*, *aabab*, *aaabbb*, *ababb*, *aabbb*, *abbbb*, respectively. The derivation below is ours.

Example 2.6. An unambiguous grammar for $\mathfrak{L}_{G/M}$ is

$$S \rightarrow T_\$ \quad T \rightarrow T_f|GT|G \quad G_f \rightarrow Gb \quad G_\$ \rightarrow a$$

The latter two productions imply that $G_{f^m\$} \xrightarrow{*} ab^m$ or in terms of functions $G_m(z) = z^{m+1}$. A typical parse tree is illustrated in Figure 1.

The second grammar production group transforms to

$$T_m(z) = T_{m+1}(z) + G_m(z)T_m(z) + G_m(z).$$

Substitution and solving for T_m gives

$$T_m(z) = \frac{z^{m+1} + T_{m+1}(z)}{1 - z^{m+1}}.$$

Iterating this recurrence yields a kind of inverted continued fraction:

$$S(z) = T_0(z) = \frac{z + T_1(z)}{1 - z} = \frac{z + \frac{z^2 + T_2(z)}{1 - z^2}}{1 - z} = \frac{z + \frac{z^2 + \frac{z^3 + \dots}{1 - z^3}}{1 - z^2}}{1 - z}.$$

Equivalently, this recurrence can be represented as

$$\begin{aligned} \frac{z + T_1(z)}{1 - z} &= \frac{z(1 - z^2) + z^2 + T_2(z)}{(1 - z)(1 - z^2)} \\ &= \frac{z(1 - z^2)(1 - z^3) + z^2(1 - z^3) + z^3 + T_3(z)}{(1 - z)(1 - z^2)(1 - z^3)} \end{aligned}$$

or

$$\begin{aligned} S(z) &= \frac{z}{1 - z} + \frac{z^2}{(1 - z)(1 - z^2)} + \frac{z^3}{(1 - z)(1 - z^2)(1 - z^3)} \\ &\quad + \dots + \frac{z^k + T_k(z)}{\prod_{n=1}^k (1 - z^n)}. \end{aligned}$$

Even though this grammar allows the index loading variable \mathbf{T} to appear on the right side of the production $\mathbf{T} \rightarrow \mathbf{GT}$ (contrary to one of the hypotheses in Prop. 2.4) the convergence proof of Proposition 2.4 is applicable to the expression above allowing us to push $T_k(z)$ off to infinity:

$$S(z) = \sum_{j \geq 1} \frac{z^j}{(1 - z)(1 - z^2) \dots (1 - z^j)}.$$

Here we recognize a classic combinatorial summation of partitions in term of their largest part [9], Example I.7. Thus, we have recovered the ordinary generating function for partitions, $S(z) = \sum_{n \geq 1} p(n)z^n$, where the coefficients belong to Euler’s partition sequence $p(n)$. Since we can also write

$$\sum_{n \geq 1} p(n)z^n = \prod_{n \geq 1} \frac{1}{1 - z^n},$$

it is true that $S(z)$ has a dense set of singularities on the unit circle and is not D-finite.

In general, allowing an index loading variable \mathbf{V} to appear on both sides of a context-free rule implies that the corresponding DSV system of equations will have an algebraic (but not necessarily linear) equation expressing $V_n(Z)$ in terms of z and $V_{n+1}(z)$. However, multiple occurrences of \mathbf{V} on the right side of such a production yield an ambiguous grammar with meaningless $S(z)$ in the corresponding DSV reduction! In fact, suppose

$$\mathbf{S} \rightarrow \mathbf{V}\$ \quad \mathbf{V} \rightarrow \mathbf{V}_f | \alpha \mathbf{V} \beta \mathbf{V} \gamma$$

comprise part of a reduced indexed grammar where $\alpha, \beta, \gamma \in (\mathcal{T} \cup \mathcal{N} \cup \varepsilon)^*$. Then the distinct production chains

$$\mathbf{S} \xrightarrow{*} \mathbf{V}_{ff}\$ \rightarrow \alpha \mathbf{V}_{ff}\$ \beta \mathbf{V}_{ff}\$ \gamma \tag{+}$$

and

$$\mathbf{S} \xrightarrow{*} \mathbf{V}_{f\$} \rightarrow \alpha \mathbf{V}_{f\$} \beta \mathbf{V}_{f\$} \gamma \rightarrow \alpha \mathbf{V}_{ff\$} \beta \mathbf{V}_{f\$} \gamma \rightarrow \alpha \mathbf{V}_{fff\$} \beta \mathbf{V}_{ff\$} \gamma \quad (++)$$

yield identical output streams.

On the other hand, Example 2.6 shows that a single instance of the loading symbol can appear on the right side of a context-free rule without leading to ambiguity of the underlying grammar. It is not hard to see that the resulting DSV equation is always linear in $V_n(z)$. We have proved

Proposition 2.7. *Let $\mathfrak{G} = (\mathcal{N}, \mathcal{T}, \mathcal{I}, \mathcal{P}, \mathbf{S})$ be an unambiguous, balanced (or ε -free), indexed grammar in strongly reduced form for some non-empty language \mathfrak{L} with $\mathcal{I} = \{f\}$. Suppose that $\mathbf{V} \in \mathcal{N}$ is the only non-terminal that loads f . Then the DSV system of equations defining $S(z)$ reduces to finitely many bounded recurrences with initial conditions whose solution is the growth function for \mathfrak{L} .*

Example 2.8. Another series with intermediate growth can be realized as the ordinary generating function of the following indexed grammar:

$$\begin{aligned} \mathbf{S} &\rightarrow \mathbf{C} | \mathbf{CT}_{\$} & \mathbf{C} &\rightarrow b\mathbf{C} | \varepsilon & \mathbf{T} &\rightarrow \mathbf{T}_f | \mathbf{W} & \mathbf{W}_f &\rightarrow \mathbf{VW}\mathbf{X} \\ \mathbf{V}_f &\rightarrow aa\mathbf{V} & \mathbf{V}_{\$} &\rightarrow aa & \mathbf{W}_{\$} &\rightarrow a & \mathbf{X} &\rightarrow a|b \end{aligned}$$

As usual, we use index $\$$ to indicate the bottom of the stack, with f being the only actual index symbol. (The reader may notice that the rule $\mathbf{S} \rightarrow \mathbf{CT}_{\$}$ is yet another relaxation of the hypotheses of Prop. 2.4 that does not affect its conclusion). A typical parse tree is given in Figure 2. From this we see that the language generated (unambiguously) is

$$\mathfrak{L}_{\text{int}} = \left\{ b^* \left(\varepsilon \mid a^{n^2+n+1} (a|b)^n \right) : n \geq 0 \right\}.$$

We can derive the generating function in the usual fashion. Note the shortcuts $\mathbf{X}_{fn\$} \rightarrow (a|b)$ (regardless of indices) and $\mathbf{V}_{fn\$} \xrightarrow{*} a^{2n} \mathbf{V}_{\$} \rightarrow a^{2n+2}$. Starting with

$$\mathbf{W}_{ff\$} \rightarrow \mathbf{V}_{f\$} \mathbf{W}_{f\$} \mathbf{X}_{f\$} \xrightarrow{*} a^4 \mathbf{W}_{f\$} (a|b) \rightarrow a^4 \mathbf{V}_{\$} \mathbf{W}_{\$} \mathbf{X}_{\$} (a|b) \xrightarrow{*} a^4 a^2 a (a|b)^2$$

one can use induction to derive

$$\mathbf{W}_{fn\$} \xrightarrow{*} a^{2n} \dots a^4 a^2 a (a|b)^n = a^{n(n+1)} a (a|b)^n = a^{n^2+n+1} (a|b)^n.$$

In terms of generating functions these shortcuts imply $W_n(z) = z^{n^2+n+1} 2^n z^n = 2^n z^{(n+1)^2}$; also $C(z) = \frac{1}{1-z}$. Put this all together to get

$$\begin{aligned} S(z) &= C(z) + C(z)T_0(z) = C(1 + T_0) = C(1 + T_1 + W_0) = \\ &= C(1 + T_2 + W_0 + W_1) = \dots = C\left(1 + \sum_{n=0}^{\infty} W_n\right) \\ &= \frac{1}{1-z} \left(1 + \sum_{n=0}^{\infty} 2^n z^{(n+1)^2}\right). \end{aligned}$$

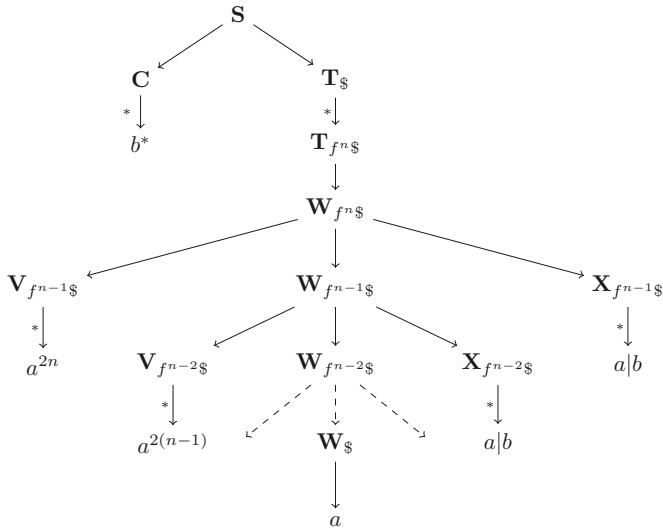


FIGURE 2. A typical parse tree in the grammar
 $S \rightarrow C|CT_\$$ $C \rightarrow bC|\varepsilon$ $T \rightarrow T_f|W$ $W_f \rightarrow VWX$
 $V_f \rightarrow aaV$ $V_\$ \rightarrow aa$ $W_\$ \rightarrow a$ $X \rightarrow a|b$.

Write as a sum of rational functions and expand each geometric series:

$$\begin{aligned}
 S(z) &= \frac{1}{1-z} + \frac{z}{1-z} + \frac{2z^4}{1-z} + \frac{4z^9}{1-z} + \frac{8z^{16}}{1-z} + \dots \\
 &= 1+z+z^2+z^3+z^4+z^5+z^6+z^7+z^8+z^9+\dots \\
 &\quad +z+z^2+z^3+z^4+z^5+z^6+z^7+z^8+z^9+\dots \\
 &\quad \quad +2z^4+2z^5+2z^6+2z^7+2z^8+2z^9+\dots \\
 &\quad \quad \quad +4z^9+\dots \\
 &\quad \quad \quad \quad \ddots
 \end{aligned}$$

and so forth. Sum the columns and observe that the coefficient of each z^n is a power of 2, with new increments occurring when n is a perfect square. Thus

$$S(z) = \sum_{n=0}^{\infty} 2^{\lfloor \sqrt{n} \rfloor} z^n$$

and the coefficient of z^n grows faster than any polynomial (as $n \rightarrow \infty$) but is sub-exponential.

The indexed grammars used in applications (combinations of groups, combinatorial descriptions, etc) tend to be reasonably simple and most use one index symbol. Despite the success of our many examples, Propositions 2.4 / 2.7 do *not* guarantee that an explicit closed formula for $S(z)$ can always be found.

Example 2.9. Consider the following balanced grammar:

$$\mathbf{S} \rightarrow \mathbf{T}_\$ \quad \mathbf{T} \rightarrow \mathbf{T}_f|\mathbf{N} \quad \mathbf{N}_f \rightarrow a\mathbf{N}|b^2\mathbf{M} \quad \mathbf{M}_f \rightarrow ab\mathbf{N}\mathbf{M} \quad \mathbf{M}_\$, \mathbf{N}_\$ \rightarrow \varepsilon$$

The hypotheses of Proposition 2.7 are satisfied so we can push T_n to infinity and obtain

$$S(z) = T_0 = N_0 + T_1 = N_0 + N_1 + T_2 = \dots = \sum_{n \geq 0} N_n(z).$$

However, the recursions defining $N_n(z)$ are intertwined and formidable:

$$N_n(z) = zN_{n-1} + z^2M_{n-1} \quad \text{and} \quad M_n(z) = z^2N_{n-1}M_{n-1} \quad \forall n \geq 1$$

with $N_0 = 1 = M_0$. It is possible to eliminate M but the resulting nonlinear recursion

$$N_n(z) = zN_{n-1} + z^2N_{n-1}N_{n-2} - z^3N_{n-2}^2$$

does not appear to be a bargain (it is possible that a multivariate generating function as per Example 4.3 may be helpful).

3. THE CASE OF SEVERAL INDEX SYMBOLS

Multiple index symbols increase the expressive power, and under certain conditions, and by grouping stacks into equivalence classes we can apply a similar technique.

Our next example uses two index symbols (in addition to $\$$) in an essential way.

Example 3.1. Define $\mathcal{L}_{\text{serial}} = \left\{ (ab^i c^j)^+ : 1 \leq i \leq j \right\}$. Consider the grammar:

$$\begin{aligned} \mathbf{S} &\rightarrow \mathbf{T}_\$ & \mathbf{T} &\rightarrow \mathbf{T}_g|\mathbf{U}_f & \mathbf{U} &\rightarrow \mathbf{U}_f|\mathbf{V}\mathbf{R}|\mathbf{V} & \mathbf{R} &\rightarrow \mathbf{V}\mathbf{R}|\mathbf{V} \\ \mathbf{V} &\rightarrow a\mathbf{B}\mathbf{C} & \mathbf{B}_f &\rightarrow \mathbf{B}b & \mathbf{B}_g &\rightarrow \varepsilon & \mathbf{C}_f &\rightarrow \mathbf{C}c & \mathbf{C}_g &\rightarrow c\mathbf{C} & \mathbf{C}_\$ &\rightarrow \varepsilon. \end{aligned}$$

Observe that the two index symbols are loaded serially: all g 's are loaded prior to any f so each valid index string will be of the form $f^+g^*\$$. We also have the shortcuts $\mathbf{C}_{f^m g^n \$} \xrightarrow{*} c^m \mathbf{C}_{g^n \$} \xrightarrow{*} c^{m+n}$ and $\mathbf{B}_{f^m g^n \$} \xrightarrow{*} b^m \mathbf{B}_{g^n \$} \rightarrow b^m \varepsilon = b^m$ and consequently $\mathbf{V}_{f^m g^n \$} \rightarrow a\mathbf{B}_{f^m g^n \$} \mathbf{C}_{f^m g^n \$} \xrightarrow{*} ab^m c^{m+n}$. A typical parse tree is given in Figure 3.

The special form of the index strings ensures that such a string is uniquely identified solely by the number of f 's and number of g 's it carries. Consequently, the induced function $V_{f^m g^n \$}(z)$ can be relabelled more simply as $V_{m,n}(z)$, and similarly with functions T, U, R . Working in the reverse order of the listed grammar productions, we have the identities

$$V_{m,n}(z) = z^{2m+n+1} \quad \text{and} \quad R_{m,n} = \frac{V_{m,n}}{1 - V_{m,n}} = \frac{z^{2m+n+1}}{1 - z^{2m+n+1}}.$$

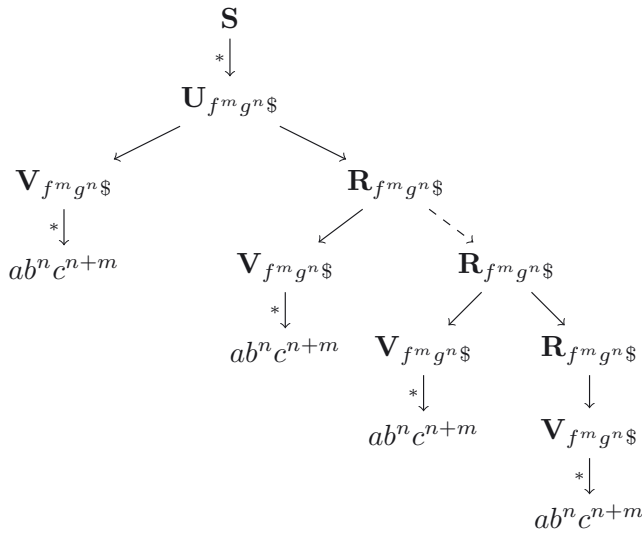


FIGURE 3. A typical parse tree in the grammar
 $S \rightarrow T_\$$ $T \rightarrow T_g|U_f$ $U \rightarrow U_f|VR|V$ $R \rightarrow VR|V$
 $V \rightarrow aBC$ $B_f \rightarrow Bb$ $B_g \rightarrow \varepsilon$ $C_f \rightarrow Cc$ $C_g \rightarrow cC$ $C_\$ \rightarrow \varepsilon$.

The grammar production $U \rightarrow U_f|VR|V$ implies for fixed $n > 0$ that

$$U_{1,n}(z) = U_{2,n} + (V_{1,n}R_{1,n} + V_{1,n}) = U_{3,n} + (V_{2,n}R_{2,n} + V_{2,n}) + (V_{1,n}R_{1,n} + V_{1,n}).$$

The hypotheses of Proposition 2.7 are satisfied in that we are dealing with a balanced grammar where currently only one index symbol is being loaded onto one variable. Therefore for fixed n we can push $U_{m,n}(z)$ off to infinity and obtain

$$U_{1,n}(z) = \sum_{m \geq 1} (V_{m,n}R_{m,n} + V_{m,n}) = \sum_{m \geq 1} R_{m,n}(z) = \sum_{m \geq 1} \frac{z^{2m+n+1}}{1 - z^{2m+n+1}}.$$

Our general derivation proceeds as follows:

$$S(z) = T_{0,0} = T_{0,1} + U_{1,0} = T_{0,2} + U_{1,1} + U_{1,0} = \dots = T_{0,k+1} + \sum_{n=1}^k U_{1,n}.$$

Proposition 2.7 can be invoked again to eliminate T . We find that

$$S(z) = \sum_{n \geq 1} \sum_{m \geq 1} \frac{z^{2m+n+1}}{1 - z^{2m+n+1}} = \sum_{j \geq 1} \frac{z^{3j}}{(1 - z^j)(1 - z^{2j})} = \sum_{i \geq 1} \frac{z^{3i} + z^{4i}}{(1 - z^{2i})^2}$$

with the latter two summations realized by expanding geometric series and/or changing the order of summation in the double sum. In any event, $S(z)$ has infinitely many singularities on the unit circle and is not D-finite.

3.1. “ENCODE FIRST, THEN COPY”

It is worth noting the reason why the copying schema used above succeeds in indexed grammars but fails for context-free grammars. The word $ab^i c^j$ is first encoded as an index string attached to \mathbf{V} and only then copied to \mathbf{VR} (the grammar symbol \mathbf{R} is a mnemonic for “replicator”). This ensures that $ab^i c^j$ is faithfully copied. Slogan: “encode first, then copy”. Context-free grammars are limited to “copy first, then express” which does not allow for fidelity in copying.

We would like to generalize the previous example. The key notion was the manner in which the indices were loaded.

Definition 3.2. Suppose that \mathfrak{G} is an unambiguous indexed grammar with index alphabet $\mathcal{I} = \{f_1, f_2, \dots, f_n\}$ such that every every index string σ has form $f_n^* f_{n-1}^* \dots f_1^* \$$. We say the indices are *loaded serially* in \mathfrak{G} .

Corollary 3.3. Assume \mathfrak{G} is an unambiguous balanced indexed grammar with variable set \mathcal{V} , non-terminal alphabet \mathcal{T} , and index alphabet $\mathcal{I} = \{f_1, \dots, f_n\}$. Suppose all indices are loaded serially onto respective variables $\mathbf{T}^{\{1\}}, \dots, \mathbf{T}^{\{n\}}$ and in the indicated order. Then each function family $T^{\{j\}}(z)$ can be eliminated (“pushed to infinity”) and the system of equations defining $S(z)$ can be reduced to finitely many recursions as per the conclusion of Proposition 2.4.

Proof. We have assumed that $\mathbf{T}^{\{n\}}$ is the last variable to load indices and is loaded with f_n only, so $\mathbf{T}^{\{n\}}$ carries an index string σ of form $f_n^* \dots f_2^* f_1^* \$$. Indeed, any grammar production having $\mathbf{T}^{\{n\}}$ on the left side will have a right side of two types: a string $\mathcal{U}_1 \in (\mathcal{V}|\mathcal{T})^*$ that loads f_n onto $\mathbf{T}^{\{n\}}$ or a string $\mathcal{U}_2 \in (\mathcal{V}|\mathcal{T})^*$ without any loading of indices. Neither of these types will include any variable $\mathbf{T}^{\{j\}}$ for $j < n$ (by the serial loading hypothesis) nor will there be any unloading of indices (by the unambiguous hypothesis). Consequently, the equations having $T_k^{\{n\}}(z)$ on the left side have on their right side products and sums involving no $T^{\{j\}}(z)$ for $j < n$ but only $T_k^{\{n\}}(z)$ and functions that define finite recurrences. The hypotheses of Proposition 2.4 apply to this situation and $T^{\{n\}}$ can be pushed to infinity.

The previous paragraph is both the basis step and induction step of an obvious argument that eliminates $T^{\{n-1\}}$ then $T^{\{n-2\}}$ and so on until $T^{\{1\}}$. □

In the case of a grammar with multiple index symbols, we would like to be able to replace an unwieldy expression like $A_{gfghgfs}(z)$ with $A_{2,3,1}(z)$ where the subscripts indicate two occurrences of f , three of g , and one h . This is certainly possible for a grammar with only one index symbol (excluding the end of stack marker $\$$) or several symbols loaded serially as per the previous Corollary, but is not possible in general.

Example 3.4. (Ordering matters) Consider the language $\mathfrak{L}_{\text{ord}}$ generated by the indexed grammar below.

$$\mathbf{S} \rightarrow \mathbf{T}_\$ \quad \mathbf{T} \rightarrow \mathbf{T}_\alpha | \mathbf{T}_\beta | \mathbf{N} \quad \mathbf{N}_\alpha \rightarrow a\mathbf{N} \quad \mathbf{N}_\beta \rightarrow b\mathbf{N}b\mathbf{N}b \quad \mathbf{N}_\$ \rightarrow \varepsilon$$

When applying the DSV transformations to this grammar we would like to write $N_{1,1}(z)$ as the formal power series corresponding to the grammar variable \mathbf{N} with any index string having one α index and one β index, followed by the end of stack marker $\$$. Note the derivations $\mathbf{S} \xrightarrow{*} \mathbf{N}_{\alpha\beta\$} \xrightarrow{*} abbb$ and $\mathbf{S} \xrightarrow{*} \mathbf{N}_{\beta\alpha\$} \xrightarrow{*} babab$. Even though both intermediate sentential forms have one of each index symbol, followed by the end of stack marker $\$$, they produce distinct words of differing length. Thus using subscripts to indicate the quantity of stack indices cannot work in general without some consideration of index ordering.

We note that the grammar is reduced and balanced. It is also unambiguous, which can be verified by induction. In fact, if $\sigma \in (\alpha|\beta)^* \$$ is an index string such that $\mathbf{N}_\sigma \xrightarrow{*} w$ where w is a terminal word of length n , then $\mathbf{N}_{\alpha\sigma} \xrightarrow{*} aw$ and $\mathbf{N}_{\beta\sigma} \xrightarrow{*} bwbb$ where $|aw| = n + 1$ and $|bwbb| = 2n + 3$. Suppose that all words $w \in \mathcal{L}_{\text{ord}}$ of length n or less are produced unambiguously. Consider a word v of length $n + 1$. Either $v = aw$ or $v = bw'bw'b$ for some shorter words $w, w' \in \mathcal{L}_{\text{ord}}$ that were produced unambiguously by hypothesis. Clearly neither of these forms for v can be confused since one starts with a and the other with b .

The proof of Proposition 2.4 can be applied to eliminate the $T_\sigma(z)$. Solving for $S(z)$ via the generalized DSV procedure gives

$$S(z) = \sum_{\sigma \in I} N_\sigma(z)$$

where the sum is over all index strings σ . It is unfeasible to simplify further because the number of grammar functions $N_\sigma(z)$ grows exponentially in the length of σ without suitable simplifying recursions.

The previous example showed two non-terminals having index strings with the same quantity of respective symbols but in different orders leading to two distinct functions. We can define a condition that ensures such functions are the same.

Definition 3.5. Let $\mathcal{A} = \{\alpha_1, \alpha_2, \dots, \alpha_n\}$ denote a finite alphabet. The *Parikh vector* associated to $\sigma \in \mathcal{A}^*$ records the number of occurrences of each α_i in σ as x_i in the vector $[x_1, x_2, \dots, x_n]$ (see [22]). Define two strings $\sigma, \tau \in \mathcal{A}^*$ to be *Parikh equivalent* if they map to the same Parikh vector (in other words τ is a permutation of σ). When \mathcal{A} is the index alphabet for a grammar, we extend this idea to non-terminals and say \mathbf{V}_σ is Parikh equivalent to \mathbf{V}_τ if σ and τ map to the same Parikh vector.

The following lemma gives sufficient conditions that allow simplifying the ordering difficulty for function subscripts. Its proof is immediate.

Lemma 3.6. *Assume \mathfrak{G} is a nontrivial balanced indexed grammar. Suppose each pair of Parikh equivalent index strings σ, τ appended to a given grammar variable \mathbf{V} result in identical induced functions $V_\sigma(z) \equiv V_\tau(z)$. Then the functions induced from \mathbf{V} can be consolidated into equivalence classes (where we replace index string subscripts by their respective Parikh vectors) without changing the solution $S(z)$ of the system of DSV equations.*

We have already used this lemma in Example 3.1 above. We illustrate with another example.

Example 3.7.

Consider the non-context-free language $\mathcal{L}_{\text{double}} = \{ww : w \in (a|b)^*\}$ produced by the grammar

$$\mathbf{S} \rightarrow \mathbf{T}_{\S} \quad \mathbf{T} \rightarrow \mathbf{T}_{\alpha}|\mathbf{T}_{\beta}|\mathbf{R}\mathbf{R} \quad \mathbf{R}_{\alpha} \rightarrow a\mathbf{R} \quad \mathbf{R}_{\beta} \rightarrow b\mathbf{R} \quad \mathbf{R}_{\S} \rightarrow \varepsilon.$$

Suppose $\sigma, \tau \in (\alpha|\beta)^*\$$ are Parikh equivalent index strings of length n . It is clear that $R_{\sigma}(z) = z^n = R_{\tau}(z)$. In fact, every string $u \in (\alpha|\beta)^n\$$ implies $R_u(z) = z^n$, regardless of the particular distribution of α and β in u . Instead of using the equivalence classes $R_{i,j}(z)$ where $[i, j]$ is the Parikh vector for u , let $R_n(z)$ denote the equivalence class of all such induced functions $R_u(z)$ where $u \in (\alpha|\beta)^n\$$, and define $T_n(z)$ similarly. We will abuse notation and refer to the elements of these classes as $R_n(z)$ or $T_n(z)$, respectively. The grammar equations become

$$S(z) = T_0 = 2T_1 + R_0^2 = R_0^2 + 2(2T_2 + R_1^2) = R_0^2 + 2R_1^2 + 4R_2^2 + \dots$$

where we can push the $T_n(z)$ to infinity as per the proof of Proposition 2.4. Therefore

$$S(z) = \sum_{n \geq 0} 2^n R_n^2(z) = \sum_{n \geq 0} 2^n z^{2n} = \frac{1}{1 - 2z^2}.$$

4. FURTHER EXAMPLES RELATED TO NUMBER THEORY

In addition to our example from [14] we have the following.

Example 4.1. Define $\mathcal{L}_{\text{div}} = \{a^n (b^n)^* : n > 0\}$ which is generated by the unambiguous balanced grammar³

$$\begin{aligned} \mathbf{S} &\rightarrow \mathbf{T}_{\S} & \mathbf{T} &\rightarrow \mathbf{T}_f|\mathbf{A}_f\mathbf{R}_f & \mathbf{R} &\rightarrow \mathbf{B}\mathbf{R}|\varepsilon \\ \mathbf{A}_f &\rightarrow a\mathbf{A} & \mathbf{A}_{\S} &\rightarrow \varepsilon & \mathbf{B}_f &\rightarrow b\mathbf{B} & \mathbf{B}_{\S} &\rightarrow \varepsilon. \end{aligned}$$

We see some familiar shortcuts: $\mathbf{A}_{f^n\S} \rightarrow a^n$ and $\mathbf{B}_{f^n\S} \rightarrow b^n$. In terms of functions this means $A_n(z) = z^n = B_n(z)$ and furthermore $R_n = B_n R_n + 1$ implies $R_n(z) = \frac{1}{1-z^n}$. Thus our main derivation becomes

$$S(z) = T_0 = T_1 + A_1 R_1 = T_2 + A_1 R_1 + A_2 R_2 = \dots = \sum_{n \geq 1} A_n R_n = \sum_{n \geq 1} \frac{z^n}{1 - z^n}.$$

³As written, this grammar is not reduced because the rule $\mathbf{T} \rightarrow \mathbf{A}_f\mathbf{R}_f$ loads two indices simultaneously. However, by replacing that production by the pair $\mathbf{T} \rightarrow \mathbf{U}_f \mathbf{U}_f \rightarrow \mathbf{A}\mathbf{R}$ we obtain an equivalent grammar in reduced form. We use the former rule for brevity.

and now each row has closed form $\frac{z^n}{(1-z^n)^2}$. Our goal is to modify the grammar of the previous example to obtain $\sum \frac{z^n}{(1-z^n)^2}$. At first glance it would seem that we only need replace the grammar rule $\mathbf{T} \rightarrow \mathbf{T}_f | \mathbf{A}_f \mathbf{R}_f$ with $\mathbf{T} \rightarrow \mathbf{T}_f | \mathbf{A}_f \mathbf{R}_f \mathbf{R}_f$ which replaces $S(z) = \sum A_n R_n$ with $\sum A_n R_n R_n$. However this creates ambiguity because we can produce ab in two ways: $S \xrightarrow{*} A_{f\$} R_{f\$} R_{f\$} \xrightarrow{*} ab\varepsilon$ and $S \xrightarrow{*} A_{f\$} R_{f\$} R_{f\$} \xrightarrow{*} a\varepsilon b$. An unambiguous solution is to *create copies* \mathbf{U} and \mathbf{C} of the original grammar variables \mathbf{B} and \mathbf{R} , respectively, so that $\mathbf{T} \rightarrow \mathbf{T}_f | \mathbf{A}_f \mathbf{R}_f \mathbf{U}_f$ is the replacement and we add new rules $\mathbf{U} \rightarrow \mathbf{UC} | \varepsilon$, $\mathbf{C}_f \rightarrow c\mathbf{C}$, and $\mathbf{C}_\$ \rightarrow \varepsilon$. The details are left to the reader, including how to re-write these changes in reduced form.

We conclude this section with the example of Bridson and Gilman [3] alluded to in our introduction. They derive words that encode the *cutting sequence* of the line segment from the origin to each integer lattice point in the Euclidean plane as the segment crosses the horizontal (h) and vertical (v) lines that join lattice points. Such sequences are made unique by declaring that as a segment passes through a lattice point, the corresponding cutting sequence adds hv . For instance, the cutting sequence for the segment ending at $(2, 4)$ is the word $hhvhhv$.

Example 4.3. The grammar is given by

$$\begin{aligned} \mathbf{S} &\rightarrow \mathbf{T}_\$ & \mathbf{T} &\rightarrow \mathbf{T}_q | \mathbf{T}_r | \mathbf{U}_q & \mathbf{U} &\rightarrow \mathbf{VU} | \mathbf{V} & \mathbf{V}_q &\rightarrow \mathbf{HV} & \mathbf{V}_r &\rightarrow \mathbf{V} \\ & & \mathbf{V}_\$ &\rightarrow v & \mathbf{H}_q &\rightarrow \mathbf{H} & \mathbf{H}_r &\rightarrow \mathbf{VH} & \mathbf{H}_\$ &\rightarrow h. \end{aligned}$$

Attempting to solve the grammar equations, one immediately runs into difficulty. The valid sentential forms $\mathbf{V}_{qqrq\$}$ and $\mathbf{V}_{qqqr\$}$ produce words of length eight and seven respectively, which prevents of simplification *via* Parikh vectors. Indeed, a brute-force numerical attempt using the DSV method has exponential time complexity in the length of index strings.

We circumvent this problem by introducing two commuting formal variables x, y . Define $L(x, y) = \sum_{i,j>0} L_{i,j} x^i y^j$ where $L_{i,j}$ counts the number of cutting sequence words that have i many occurrences of v and j many h 's. The coefficients $L_{i,j}$ comprise a frequency distribution on the first quadrant of the integer lattice. This distribution contains more information than the one dimensional generating function $S(z)$. On the other hand, we can recover $S(z) = \sum_{n>1} v_n z^n$ by the formula $v_n = \sum_{i+j=n} L_{i,j}$, or equivalently, $S(z) = L(z, z)$.

To compute the $L_{i,j}$, let us simplify the grammar by ignoring the context-free copying productions $\mathbf{U} \rightarrow \mathbf{VU} | \mathbf{V}$, change the loading productions to $\mathbf{T} \rightarrow \mathbf{T}_q | \mathbf{T}_r | \mathbf{V}_q$, and begin by unloading sentential forms $\mathbf{V}_{q\sigma}$, where $\sigma \in (q|r)^* \$$. As per the proof of Lemma 2.5 we define a *step* as the application of the leftmost stack symbol to all non-terminals in a sentential form. If we start with an index string attached to \mathbf{V} of length l , then after n steps each non-terminal will have the same index string of length $l - n$. For instance if we start with $\mathbf{V}_{qqr\sigma}$ then the first step is $\mathbf{V}_{qqr\sigma} \rightarrow \mathbf{H}_{qr\sigma} \mathbf{V}_{qr\sigma}$. The second step comprises two productions and ends

with $\mathbf{H}_{r\sigma}\mathbf{H}_{r\sigma}\mathbf{V}_{r\sigma}$ while the third step unloads the r from each index and results in $\mathbf{V}_{\sigma}\mathbf{H}_{\sigma}\mathbf{V}_{\sigma}\mathbf{H}_{\sigma}\mathbf{V}_{\sigma}$.

Let x_i be the number of \mathbf{V} non-terminals after performing step i , and let y_i be the number of \mathbf{H} non-terminals after performing step i . For a step unloading q we observe the recursions $y_i = y_{i-1} + x_{i-1}$ and $x_i = x_{i-1}$. Likewise for r we see recursions $y_i = y_{i-1}$ and $x_i = y_{i-1} + x_{i-1}$. Our simplified grammar always begins the unloading stage with $\mathbf{V}_{q\sigma}$ and thus we obtain the initial condition $x_1 = 1 = y_1$ regardless of σ . This condition, along with our recursions above imply that for each i , the pair of integers (x_i, y_i) are relatively prime.

Suppose that n is the last step needed to produce a cutting sequence word from $\mathbf{V}_{q\sigma}$. Identify each pair (x_n, y_n) with the corresponding point in the integer lattice, so x_n is the total number of vertical lines crossed in the cutting sequence and y_n is the number of horizontal lines crossed. (Note that x_n and y_n depend on σ as well as n).

We saw above that $\mathbf{V}_{qqr\sigma} \xrightarrow{*} \mathbf{V}_{\sigma}\mathbf{H}_{\sigma}\mathbf{V}_{\sigma}\mathbf{H}_{\sigma}\mathbf{V}_{\sigma} \xrightarrow{*} qvvhv$ which is represented by $(3, 2)$. The pair $(2, 3)$ is realized from $\mathbf{V}_{qrq\sigma} \xrightarrow{*} hvhvv$. Indeed, the symmetry of the grammar implies that every generated pair (x_n, y_n) has a generated mirror image (y_n, x_n) obtained by transposing each q and r in the index substring σ attached to the initial unloading symbol $\mathbf{V}_{q\sigma}$.

We claim that every relatively prime pair of positive integers is realized as (x_n, y_n) for some cutting sequence word generated by our simplified grammar. We show this by running in reverse the algorithm that generates cutting sequences. Let $(i, j) \neq (1, 1)$ denote a coprime pair and suppose by induction that all other relatively prime pairs (k, l) are the result of unique cutting sequence words whenever $(k < i$ and $l \leq j)$ or $(k \leq i$ and $l < j)$, *i.e.* whenever the point (k, l) is strictly below or left of the point (i, j) . If $i < j$ then the letter q was applied at the most recent step with the previous pair being defined by $(i, j - i)$. On the other hand, if $i > j$ then the rightmost letter is r and define the previous pair as $(i - j, j)$. In either case the new pair of coordinates remain coprime and lie in the induction hypothesis zone. Note that this is just the Euclidean algorithm for greatest common divisor and always terminates at $(1, 1)$ when the starting pair (i, j) are coprime. Consequently the relatively prime pair (i, j) is uniquely realized as (x_n, y_n) from some cutting sequence word w generated by our simplified grammar. Furthermore $\mathbf{V}_{q\sigma} \xrightarrow{*} w$ satisfies $|q\sigma| = n$ which is the correct number of steps taken.

We apply our argument above to compute the two dimensional generating function $L(x, y)$. For our simplified grammar we have $L_{i,j} = 1$ if the pair (i, j) is relatively prime and $L_{i,j}$ vanishes otherwise. Equivalently, $L_{i,j} = 1$ if and only if i and $i + j$ are coprime. To recover $S(z) = \sum_{n \geq 2} v_n z^n$ from $L(x, y)$ we set $v_n = \sum_{i+j=n} L_{i,j}$, *i.e.* we sum along slope -1 diagonal lines in quadrant one. Thus $v_n = \varphi(n)$ where φ is Euler's totient function that counts the number of integers $1 \leq i < n$ that are coprime to n . Summary: we have successfully circumvented the exponential time complexity of computing $S(z) = \sum_{\sigma \in (q|r)^*\sigma} V_{q\sigma}(z)$ and found that $S(z) = \sum_{n \geq 2} \varphi(n)z^n$.

Recovering the original grammar by restoring the **U** productions allows for the construction of repeated cutting sequences w^t , w being the word associated to a coprime lattice point (i, j) . This serves to add the lattice points (ti, tj) . Here we may assume i and j are relatively prime and $t \geq 2$ which makes these additions unique. (In fact, if $(ti, tj) = (sk, sl)$ for another coprime pair (k, l) , then both s and t are the greatest common divisor of the ordered pair and hence $s = t$). The full grammar is in bijective correspondence with the integer lattice strictly inside quadrant one. Words represent geodesics in the taxicab metric. Simple observation shows that the full growth series is represented by the rational function

$$\sum_{n \geq 2} (n - 1)z^n = \frac{z^2}{(1 - z)^2}$$

and as a byproduct we have re-derived Euler’s identity

$$n - 1 = \sum_{d|n, d > 1} \varphi(d).$$

5. AMBIGUITY

We begin with the first published example of an inherently ambiguous context-free language [16, 22]. It has an unambiguous indexed grammar.

Example 5.1. Define $\mathcal{L}_{\text{amb}} = \{a^i b^j a^k b^l : i, j, k, l \geq 1; i = k \text{ or } j = l\}$. The idea is to divide the language into a disjoint union of the three sub-languages

$$\mathcal{L}_X = \{a^i b^j a^i b^l : j < l\} \quad \mathcal{L}_Y = \{a^i b^j a^i b^l : l < j\} \quad \mathcal{L}_Z = \{a^i b^j a^k b^j\}$$

with no restrictions on i, k other than that all exponents are at least one. An indexed grammar is

$$\mathbf{S} \rightarrow \mathbf{T}_g \mathbf{\$} \quad \mathbf{T} \rightarrow \mathbf{T}_g | \mathbf{U}_f | \mathbf{Z} \quad \mathbf{U} \rightarrow \mathbf{U}_f | \mathbf{X} | \mathbf{Y}$$

$$\mathbf{X} \rightarrow \mathbf{A} \mathbf{B} \mathbf{A} \mathbf{C} \quad \mathbf{Y} \rightarrow \mathbf{A} \mathbf{C} \mathbf{A} \mathbf{B} \quad \mathbf{Z} \rightarrow \mathbf{D} \mathbf{B} \mathbf{E} \mathbf{B}$$

$$\mathbf{A}_f \rightarrow a \mathbf{A} \quad \mathbf{A}_g \rightarrow \varepsilon \quad \mathbf{B}_f \rightarrow \mathbf{B} \quad \mathbf{B}_g \rightarrow b \mathbf{B} \quad \mathbf{B}_\$ \rightarrow \varepsilon$$

$$\mathbf{C}_f \rightarrow \mathbf{C} \quad \mathbf{C}_g \rightarrow b \mathbf{C} \quad \mathbf{C}_\$ \rightarrow b \mathbf{C}_\$ | b \quad \mathbf{D} \rightarrow a \mathbf{D} | a \quad \mathbf{E} \rightarrow a \mathbf{E} | a$$

The reader is invited to draw the typical parse tree and verify that the growth series of this language is $\frac{z^4(1+3z)}{(1-z)^3(1+z)^2}$.

Several other inherently ambiguous context-free language can be generated unambiguously by an indexed grammar.

Exercise 5.2. Another early example of an inherently ambiguous context-free language is featured in [5]: $\mathcal{L} = \{a^n b^m c^p : m = n > 0 \text{ or } m = p > 0\}$. We can write an unambiguous indexed grammar for it. Again, split the language into the disjoint union of three types of words and build a grammar for each. The types are $a^n b^n c^p$ with $0 \leq p < n$, $a^n b^n c^p$ with $0 < n < p$, and $a^n b^p c^p$ with $p > 0$.

Our examples beg the question: are there inherently ambiguous indexed languages? Consider Crestin’s language of palindrome pairs defined by $\mathcal{L}_{\text{Crestin}} = \{vw : v, w \in (a|b)^*, v = v^R, w = w^R\}$. It is a “worst case” example of an inherently ambiguous context-free language (see [8] and its references). We conjecture that $\mathcal{L}_{\text{Crestin}}$ remains inherently ambiguous as an indexed language. What about inherently ambiguous languages that are not context-free?

Consider the composite numbers written in unary as per Example 4.2. What would an unambiguous grammar for $\mathcal{L}_{\text{comp}}$ look like? We would need a unique factorization for each composite c . Since the arithmetic that indexed grammars can simulate on unary output is discrete math (like addition and multiplication, no division or roots, etc), we need the Fundamental Theorem of Arithmetic. In fact, suppose there is a different unique factorization scheme for the composites, that doesn’t involve a certain prime p . Then composite $c_2 = p^2$ has only the factorization $1 \cdot c_2$, and similarly $c_3 = p^3$ has unique factorization $1 \cdot c_3$ since $p \cdot c_2$ is disallowed. But then $p^6 = c_2 \cdot c_2 \cdot c_2 = c_3 \cdot c_3$ has no unique factorization. Therefore all primes p are needed for any unique factorization of the set of composites. Adding any other building blocks to the set of primes ruins unique factorization.

Suppose we have an unambiguous indexed grammar for $\mathcal{L}_{\text{comp}}$. It would be able to generate a^{p^k} for any prime p and all $k > 1$. This requires a copying mechanism (in the manner of \mathbf{R} in Examples 3.1 and 4.2) and an encoding of p into an index string (recall our slogan “encode first, then copy” from Sect. 3.1). In other words, our supposed grammar for $\mathcal{L}_{\text{comp}}$ must be able to first produce its complement $\mathcal{L}_{\text{prime}}$ and *encode these primes into index strings*. However, [21] show that the set of index strings associated to a non-terminal in an indexed grammar is necessarily a regular language. On the other hand [2] shows that the set of primes expressed in any base $m \geq 1$ does not form a regular language. We find it highly unlikely that an indexed grammar can decode all the primes from a regular set of index strings. We conjecture that $\mathcal{L}_{\text{comp}} = \{a^c : c \text{ is composite}\}$ is inherently ambiguous as an indexed language.

Recall that a word is *primitive* if it is not a power of another word. In the copious literature on the subject it is customary to let \mathcal{Q} denote the language of primitive words over a two letter alphabet. It is known that \mathcal{Q} is not unambiguously context-free (see [23, 24], who exploits the original Chomsky–Schützenberger theorem listed in Section 2 above). It is a widely believed conjecture that \mathcal{Q} is not context-free at all (see [7]).

$\mathcal{L}' = \{w^k : w \in (a|b)^*, k > 1\}$ defines the complement of \mathcal{Q} with respect to the free monoid $(a|b)^*$. It is not difficult to construct an ambiguous balanced grammar for \mathcal{L}' (a simple modification of Example 3.7 will suffice). What about an unambiguous grammar? Recall from [20] that $w_1^n = w_2^m$ implies that each w_i

is a power of a common word v . Thus to avoid ambiguity, each building block w used to construct \mathcal{L}' needs to be primitive. This means we must not only be able to *recreate* \mathcal{Q} in order to generate \mathcal{L}' unambiguously, we must be able to *encode* each word $w \in \mathcal{Q}$ as a string of index symbols, as per the language of composites. We refer again to Section 3.1. We find this highly unlikely and we conjecture that $\mathcal{L}' = \{w^k : w \in (a|b)^*, k > 1\}$ is inherently ambiguous as an indexed language.

6. OPEN QUESTIONS

We observed that in many cases the generating function $S(z)$ of an indexed language is an infinite sum (or multiple sums) of a family of functions related by a finite depth recursion (or products/sums of the same). As we mentioned earlier, [17] give explicit sufficient conditions for growth series of indexed languages on a unary alphabet. They also show that such growth series are defined in terms of the recursions we mentioned above.

Into what class do the generating functions of indexed languages fit? Can we characterize the types of productions that lead to an infinite number of singularities in the generating function? Ultimately, this was a common property of many of the examples, and possibly one could generalize the conditions that lead to an infinite number of singularities in Example 2.2 to a general rule on production types in the grammar. It seems that the foundation laid by Fratani and Senizergues, in their work on catenative grammars is a very natural starting point for such a study.

Can we characterize the expressive power of the grammars which satisfy the hypotheses of Proposition 2.7? The alternate characterizations of level 2 sequences in [26] might be useful. Can we show that $\{a^{p(n)} : p(n) \text{ is the } n\text{th prime}\}$ is a level 3 (or higher?) language? Perhaps this should precede a search for an indexed grammar.

Is Crestin’s language inherently ambiguous as an indexed language? What about the composite numbers in unary or the complement of the primitive words?

In step with much modern automatic combinatorics, we would like to build automated tools to handle these, and other questions related to indexed grammars. Towards this goal the first author has written a parser/generator that inputs an indexed grammar and outputs words in the language. It is licensed under GPLv3 and is available from the authors. Is there a way to automate the process of pushing a set of terminal variables off to infinity?

Finally, we end where we began with the non-context-free language $\{a^n b^n c^n : n > 0\}$. It has a context-sensitive grammar

$$\mathbf{S} \rightarrow abc|a\mathbf{B}Sc \quad \mathbf{B}a \rightarrow a\mathbf{B} \quad b\mathbf{B} \rightarrow bb$$

for which the original DSV method works perfectly. The method fails for several other grammars generating this same language. What are the necessary and sufficient conditions to extend the method to generic context-sensitive grammars? To what extent can we see these conditions on the system of catenative recurrent relations?

Acknowledgements. Daniel Jorgensen contributed ideas to the first two authors. The third author offers thanks to Mike Zabrocki and the participants of the Algebraic Combinatorics seminar at the Fields Institute for Research in Mathematical Science (Toronto, Canada) for early discussions on this theme. We are also grateful to the referees for pointing us to several references.

REFERENCES

- [1] A.V. Aho, Indexed grammars—an extension of context-free grammars. *J. Assoc. Comput. Mach.* **15** (1968) 647–671.
- [2] D. Allen, Jr, On a Characterization of the Nonregular Set of Primes. *J. Comput. System Sci.* **2** (1968) 464–467.
- [3] M.R. Bridson and R.H. Gilman, Formal language theory and the geometry of 3-manifolds. *Comment. Math. Helv.* **71** (1996) 525–555.
- [4] M.R. Bridson and R.H. Gilman, Context-free languages of sub-exponential growth. *J. Comput. System Sci.* **64** (2002) 308–310.
- [5] N. Chomsky and M.P. Schützenberger, The algebraic theory of context-free languages, in *Computer programming and formal systems*. North-Holland, Amsterdam (1963) 118–161.
- [6] M. Delest, Algebraic languages: a bridge between combinatorics and computer science, in *Formal power series and algebraic combinatorics (New Brunswick, NJ, 1994)*, DIMACS Ser. Discrete Math. Theoret. Comput. Sci., vol. 24. Amer. Math. Soc. Providence, RI (1996) 7187.
- [7] P. Dömösi, S. Horváth, M. Ito, L. Kászonyi and M. Katsura, Some combinatorial properties of words, and the Chomsky-hierarchy, in *Words, languages and combinatorics, II (Kyoto, 1992)* World Sci. Publ., River Edge, NJ (1994) 105–123.
- [8] Ph. Flajolet, Analytic models and ambiguity of context-free languages. *Theoret. Comput. Sci.* **49** (1987) 283–309. Twelfth international colloquium on automata, languages and programming (Nafplion, 1985).
- [9] Ph. Flajolet and R. Sedgewick, *Analytic combinatorics*. Cambridge University Press, Cambridge (2009).
- [10] S. Fratani and G. Sénizergues, Iterated Pushdown Automata and Sequences of Rational Numbers. *Ann. Pure Appl. Logic* **141** (2005) 363–411.
- [11] E.M. Freden and J. Schofield, The growth series for Higman 3. *J. Group Theory* **11** (2008) 277–298.
- [12] R.H. Gilman, A shrinking lemma for indexed languages. *Theoret. Comput. Sci.* **163** (1996) 277–281.
- [13] R.H. Gilman, Formal languages and their application to combinatorial group theory, in *Groups, languages, algorithms*, vol. 378. *Contemp. Math.* Amer. Math. Soc. Providence, RI (2005) 1–36.
- [14] R.I. Grigorchuk and A. Machì, An example of an indexed language of intermediate growth. *Theoret. Comput. Sci.* **215** (1999) 325–327.
- [15] D.F. Holt and C.E. Röver, Groups with indexed co-word problem. *Internat. J. Algebra Comput.* **16** (2006) 985–1014.
- [16] J.E. Hopcroft and J.D. Ullman, *Introduction to automata theory, languages, and computation*. Addison-Wesley Publishing Co., Reading, Mass. Addison-Wesley Series in Computer Science (1979).
- [17] L.P. Lisovik and T.A. Karnaukh, A class of functions computable by index grammars. *Cybernetics and Systems Analysis* **39** (2003) 91–96.
- [18] R. Incitti, The growth function of context-free languages. *Theoret. Comput. Sci.* **255** (2001) 601–605.
- [19] L. Lipschitz and L.A. Rubel, A gap theorem for power series solutions of algebraic differential equations. *Amer. J. Math.* **108** (1986) 1193–1213.

- [20] R.C. Lyndon and M.P. Schützenberger, The equation $a^M = b^N c^P$ in a free group. *Michigan Math. J.* **9** (1962) 289–298.
- [21] R. Parchmann and J. Duske, The structure of index sets and reduced indexed grammars. *RAIRO: ITA* **24** (1990) 89–104.
- [22] R.J. Parikh, On context-free languages. *J. ACM*, **13** (1966) 570–581.
- [23] H. Petersen, The ambiguity of primitive words, *STACS 94*, in vol. 775 of *Lecture Notes Comput. Sci.* Springer, Berlin (1994) 679–690.
- [24] H. Petersen, On the language of primitive words. *Theoret. Comput. Sci.* **161** (1996) 141–156.
- [25] S. Rees, A language theoretic analysis of combings, in *Groups, languages and geometry (South Hadley, MA, 1998)*, in vol. 250 of *Contemp. Math. Amer. Math. Soc.* Providence, RI (1999) 117–136.
- [26] G. Sénizergues, Sequences of Level 1, 2, 3, . . . , k, In *Computer Science Theory and Applications*, vol. 4649 of *Lect. Notes Comput. Sci.* Springer, Berlin (2007) 24–32.

Communicated by P. Weil.

Received August 15, 2012. Accepted August 9, 2013.