Extending Parikh's Theorem to Weighted and Probabilistic Context-Free Grammars

Vijay Bhattiprolu[1], Spencer Gordon[2(✉)], and Mahesh Viswanathan[2]

[1] Carnegie Mellon University, Pittsburgh, PA 15213, USA
vpb@cs.cmu.edu
[2] University of Illinois at Urbana-Champaign, Urbana 61801, USA
{slgordo2,vmahesh}@illinois.edu

Abstract. We prove an analog of Parikh's theorem for weighted context-free grammars over commutative, idempotent semirings, and exhibit a stochastic context-free grammar with behavior that cannot be realized by any stochastic right-linear context-free grammar. Finally, we show that every unary stochastic context-free grammar with polynomially-bounded ambiguity has an equivalent stochastic right-linear context-free grammar.

1 Introduction

Two words u, v over an alphabet Σ are said to be Parikh equivalent, if for each $a \in \Sigma$, the number of occurrences of a in u and v are the same. The Parikh image of a language L, is the set of Parikh equivalence classes of words in L. One of the most celebrated results in automata theory, Parikh's theorem [27], states that for any context-free language L, there is a regular language L' such that the Parikh images of L and L' are the same. For example, the context-free language $\{a^n b^n \mid n \geq 0\}$ has the same Parikh image as the regular language $(ab)^*$; both the Parikh images only consist of those equivalence classes where the numbers of as is equal to the number of bs. An important and immediate consequence of this result is that every context-free language over the unary alphabet is in fact regular. Parikh's theorem has found many applications—in automata theory to prove non-context-freeness of languages [13], decision problems for membership, universality and inclusions involving context-free languages and semi-linear sets [11,17–19]; in verification of subclasses and extensions of counter machines [8,11,14,15,20,23,33,35,37]; automata and logics over unranked trees with counting [2,34]; PAC-learning [23].

Weighted automata [10,32] are a generalization of classical automata (finite or otherwise) in which each transition has an associated weight from a semiring. Recall that a semiring is an algebra with two operations $\oplus$ and $\otimes$ such that $\oplus$ is a commutative monoid operation, $\otimes$ is a monoid operation, $\otimes$ distributes

V. Bhattiprolu—This work was started while this author was at the University of Illinois, Urbana-Champaign.

M. Viswanathan—Partially suported by NSF CNS 1314485.

N. Bertrand and L. Bortolussi (Eds.): QEST 2017, LNCS 10503, pp. 3–19, 2017.
DOI: 10.1007/978-3-319-66335-7_1

over $\oplus$, and the identity of $\oplus$ is an anhilator for $\otimes$. Unlike classical automata that compute Boolean-valued functions over words, weighted automata compute more general functions over words—the weight of an accepting computation is the product of the weights of its transitions, and the weight of a word is the sum of the weights of all its accepting computations. Since the seminal paper of Schützerberger [32], weighted automata have inspired a wealth of extensions and further research (see [10] for a recent handbook compilation). Weighted automata have found applications in verification [5,6,9,24], reasoning about competitive ratio of online algorithms [1], digital image compression [7,16,21,22], in speech-to-text processing [4,25,26], and data flow analysis [30,31]. A special case of weighted automata are probabilistic automata [28,29] that model randomized algorithms and stochastic uncertainties in the system environment.

In this paper, we investigate whether Parikh's theorem can be generalized to the weighted case. In particular we investigate if for any weighted context-free grammar G there is a weighted right-linear grammar G$'$ such that for any Parikh equivalence class C, the sum of the weights of words in C under G and G$'$ is the same. It is easy to see that if the weight domain is not commutative (i.e., $\otimes$ is not a commutative operation) then Parikh's theorem does not hold. Thus we focus our attention on commutative weight domains.

Our first result concerns weight domains that are additionally idempotent, which means that $\oplus$ is an idempotent operation. A classical example of such a semiring is the min-plus or tropical semiring over the natural numbers where min is the "addition" operation, and + is the "product" operation. We show that Parikh's theorem does indeed hold for weighted automata over commutative, idempotent semirings.

Next, we show that our assumption about idempotence of semirings is necessary. In particular, we give an example of a stochastic context-free grammar G over the unary alphabet such that the function computed by G cannot be realized by any stochastic right linear grammar.

Our last result concerns unary grammars that are polynomially ambiguous. Recall that a grammar is polynomially ambiguous if there is a polynomial p such that on any word of length n in the language, the number of derivation trees for the word is bounded by $p(n)$. We prove that Parikh's theorem extends for such grammars. Specifically, we show that, over the unary alphabet, any probability function realized by a stochastic context-free grammar can also be realized by a right-linear grammar. Though we present this result in the context of stochastic grammars, the proof applies to any polynomially ambiguous weighted context-free grammar over a semiring that is commutative, but not necessarily idempotent.

The rest of the paper is organized as follows. We introduce the basic models and notation in Sect. 2. The Parikh's theorem for weighted automata over commutative, idempotent semirings is presented in Sect. 3. In Sect. 4, we present an example unary stochastic context-free grammar, and show that there is no stochastic right-linear grammar that is equivalent to it. Section 5 contains our proof for Parikh's theorem for polynomially ambiguous grammars. Eventhough this proof is presented in the context of stochastic grammars, it is easy to see

that it extends to any weighted context free grammar over a commutative (but not necessarily idempotent) semiring. Finally, we present our conclusions and directions for future work in Sect. 6.

2 Preliminaries

Strings. Let us fix a finite string/word $w \in \Sigma^*$ over Σ. For a subset $\Gamma \subseteq \Sigma$, $w\restriction_\Gamma$ will denote the string over Γ obtained by removing the symbols not in Γ from w. The ***Parikh map***, or Parikh image, of $w \in \Sigma^*$, denoted by $\mathrm{Pk}(w)$, is a mapping from Σ to $\mathbb{N}$, such that for $a \in \Sigma$, $\mathrm{Pk}(w)(a)$ is the number of occurrences of a in w. The ***Parikh equivalence class*** of w, $[w]_{\mathrm{Pk}} = \{w' \mid \mathrm{Pk}(w') = \mathrm{Pk}(w)\}$, is the set of all words with the same Parikh image as w. We can extend the Parikh map to languages $L \subseteq \Sigma^*$, defining $\mathrm{Pk}(L) \triangleq \{\mathrm{Pk}(w) \mid w \in L\}$.

Context Free Grammars. We will consider context free grammars in ***Greibach Normal Form***. Formally (in this paper) a ***context-free grammar*** is $\mathrm{G} = (\mathrm{V}, \Sigma, \mathrm{P}, S)$, where V and Σ are disjoint sets of variables (or non-terminals) and terminals, respectively; $S \in \mathrm{V}$ is the start symbol; and $\mathrm{P} \subseteq \mathrm{V} \times \Sigma\mathrm{V}^*$ is a finite set of productions where each production is of the form $A \to \mathtt{a}\beta$ with $\mathtt{a} \in \Sigma$ and $\beta \in \mathrm{V}^*$. Without loss of generality, we assume that every production in the grammar is used in some derivation from S to a string in Σ^*. A ***sentence*** is a string in $(\Sigma \cup \mathrm{V})^*$. A ***right-linear grammar*** is a context-free grammar where the productions have at most one non-terminal on the right-hand side, i.e., $\mathrm{P} \subseteq \mathrm{V} \times (\Sigma(\{\epsilon\} \cup \mathrm{V}))$. It is well known that a language is generated by a right-linear grammar if and only if it is regular.

We will find it convenient to partition the variables of a grammar into those that have exactly one derivation tree and those that have more than one. Formally, the set of ***single-derivation variables*** $\mathrm{X} \subseteq \mathrm{V}$ is the smallest set containing all variables A with exactly one production of the form $A \to \mathtt{a}$ (with $\mathtt{a} \in \Sigma$) and having the property that if a variable A has exactly one production of the form $A \to \mathtt{a}\alpha$ where $\mathtt{a} \in \Sigma$ and $\alpha \in \mathrm{X}^*$ then $A \in \mathrm{X}$. The remaining variables, i.e. $\mathrm{Y} = \mathrm{V} \setminus \mathrm{X}$, are ***multiple-derivation variables***.

Prioritized leftmost derivations. In this paper we will consider special derivation sequences of a context-free grammar that expand the leftmost variable while giving priority to single-derivation variables. We call these ***prioritized leftmost (PLM) derivations***, and we define them precisely next.

Definition 1. *Consider a context-free grammar* $\mathrm{G} = (\mathrm{V}, \Sigma, \mathrm{P}, S)$*, where the non-terminals* V *have been partitioned into the set of single-derivation variables* X *and multiple-derivation variables* Y*. We say that* $\alpha A\beta$ *rewrites in a* single prioritized leftmost derivation step *to* $\alpha\gamma\beta$ *(denoted as* $\alpha A\beta \Rightarrow_{\mathrm{plm}} \alpha\gamma\beta$*) iff* $\exists \pi \in \mathrm{P}, \pi = (A \to \gamma)$ *such that either*

1. $A \in \mathrm{X}$*,* $\alpha \in (\Sigma \cup \mathrm{Y})^*$*, and* $\beta \in \mathrm{V}^*$*, or*
2. $A \in \mathrm{Y}$*,* $\alpha \in \Sigma^*$*, and* $\beta \in (\Sigma \cup \mathrm{Y})^*$

In other words, either A is the leftmost single-derivation variable in $\alpha A\beta$, or A is the leftmost multiple-derivation variable and $\alpha A\beta$ has no single-derivation variables. If $\alpha \Rightarrow_{\text{plm}} \beta$ by application of π, we'll write $\alpha \overset{\pi}{\Rightarrow}_{\text{plm}} \beta$. Note that if $\alpha \Rightarrow_{\text{plm}} \beta$ there is always a unique π such that $\alpha \overset{\pi}{\Rightarrow}_{\text{plm}} \beta$.

A prioritized leftmost (PLM) derivation *is a sequence $\psi = \alpha_1, \ldots, \alpha_n$ such that $\alpha_1 \Rightarrow_{\text{plm}} \alpha_2 \Rightarrow_{\text{plm}} \cdots \Rightarrow_{\text{plm}} \alpha_n$. The set of all PLM derivations is denoted* $\text{Der}_{\text{plm}}(\text{G})$.

The ***language generated by*** G is $\text{L(G)} \triangleq \left\{ \alpha \in \Sigma^* \;\middle|\; S \Rightarrow^*_{\text{plm}} \alpha \right\}$ where $\Rightarrow^*_{\text{plm}}$ is the reflexive and transitive closure of $\Rightarrow_{\text{plm}}$. Finally, the ***parse*** of a word $w \in (\Sigma \cup \text{V})^*$, denoted $\text{parse}_{\text{G}}(w)$, is the set of all PLM derivations yielding w:

$$\text{parse}_{\text{G}}(w) \triangleq \{\alpha_1, \ldots, \alpha_n \in \text{Der}_{\text{plm}}(\text{G}) \mid \alpha_1 = S \,\text{and}\, \alpha_n = w\}\,.$$

Example 1. We present a simple example to illustrate the definitions. Consider the grammar $\text{G} = (\{S, B\}, \{\mathtt{a}, \mathtt{b}\}, \text{P}, S)$ where P consists of the following productions: $\pi_1 = S \to \mathtt{a}SB$, $\pi_2 = S \to \mathtt{a}B$, and $\pi_3 = B \to \mathtt{b}$. The set of single-derivation variables is $\{B\}$ and the set of multiple-derivation variables is $\{S\}$. An example of a prioritized leftmost derivation is

$$S \overset{\pi_1}{\Rightarrow}_{\text{plm}} aSB \overset{\pi_3}{\Rightarrow}_{\text{plm}} aSb \overset{\pi_2}{\Rightarrow}_{\text{plm}} aaBb \overset{\pi_3}{\Rightarrow}_{\text{plm}} aabb$$

The language generated by this grammar is $\{\mathtt{a}^n\mathtt{b}^n \mid n \geq 1\}$.

Derivation trees. The set of all derivation trees for G will be denoted as Δ_{G}. For a derivation tree τ, a node n in τ, and a path p from the root in τ, $\ell(\tau)$, $\ell(n)$ and $\ell(p)$ will denote the label of the root, the node n, and the node reached by path p in τ, respectively. For any node n in a tree τ and path p from the root, we denote the subtree rooted at n by $\tau(n)$, and the subtree rooted at the node reached by path p by $\tau(p)$. The ***frontier of a tree*** τ, denoted $\text{Fr}(\tau)$ is the sentence $\ell(n_1)\ell(n_2)\ldots\ell(n_k)$ where $n_1, \ldots, n_k$ are the leaves of τ in left-to-right order.

For any variable $A \in \text{V}$, $\Delta_{\text{G}}(A) \triangleq \{\tau \in \Delta_{\text{G}} \mid \ell(\tau) = A\}$ is the subset of derivation trees rooted at A. A tree τ for which $\text{Fr}(\tau) \in \Sigma^*$ is called a ***complete derivation tree***, and the set of all complete derivation trees rooted at A is $\Delta^{\Sigma}_{\text{G}}(A) \triangleq \{\tau \in \Delta_{\text{G}}(A) \mid \text{Fr}(\tau) \in \Sigma^*\}$. The set of all complete derivation trees is $\Delta^{\Sigma}_{\text{G}} \triangleq \{\tau \in \Delta_{\text{G}} \mid \text{Fr}(\tau) \in \Sigma^*\}$. A tree $\tau \in \Delta_{\text{G}}(A)$ is said to be an A-***pumping tree*** if $\text{Fr}(\tau)\restriction_{\text{V}} = A$. The set of A-pumping trees is $\Delta^{p}_{\text{G}}(A) \triangleq \{\tau \in \Delta_{\text{G}}(A) \mid \text{Fr}(\tau)\restriction_{\text{V}} = A\}$. The set of all pumping trees is given by $\Delta^{p}_{\text{G}} = \{\tau \in \Delta_{\text{G}} \mid \text{Fr}(\tau)\restriction_{\text{V}} \in \text{V}\}$.

Remark 1. In a context-free grammar (where all productions are "useful"), every single-derivation variable is the root of exactly one complete derivation tree, and every multiple-derivation variable is the root of at least two complete derivation trees.

Tree Notation. We will use the following notation on derivation trees. Let $\tau \in \Delta_{\mathrm{G}}$, n be a node in τ, and p be a path from the root in τ. The leftmost child of the node reached by path p, will be the one reached by the path $p \cdot 0$ with the other children corresponding to the paths $p \cdot 1$, $p \cdot 2$, etc. For $\tau_1 \in \Delta_{\mathrm{G}}(\ell(n))$ ($\tau_1 \in \Delta_{\mathrm{G}}(\ell(p))$), $\tau[n \mapsto \tau_1]$ ($\tau[p \mapsto \tau_1]$) denotes the derivation tree obtained by replacing $\tau(n)$ ($\tau(p)$) by the tree τ_1. We denote by $\mathsf{rem}_p(\tau)$ the tree obtained by replacing $\tau(p)$ by the root of $\tau(p)$, i.e., by "removing" all the children of p. Finally, for a rule $A \to \mathtt{a}\alpha$ with $\alpha = A_1 A_2 \cdots A_k$, and trees $\tau_i \in \Delta_{\mathrm{G}}(A_i)$, $A_{\mathtt{a}\alpha}(\tau_1, \tau_2, \ldots \tau_k)$ denotes the tree with root labeled A and children $\mathtt{a}, \tau_1, \ldots \tau_k$ from left-to-right. Thus, $A_{\mathtt{a}}$ denotes the tree with root labeled A and one child labeled $\mathtt{a}$.

Cuts. Observe that, for any string $\alpha \in (\mathrm{V} \cup \Sigma)^*$, there is a bijection between derivation trees τ with $\mathrm{Fr}(\tau) = \alpha$ and PLM derivations in $\mathrm{parse}_{\mathrm{G}}(\alpha)$. A set of nodes separating the root of τ from all of the leaves in τ is a ***cut*** of τ. Now consider the unique PLM derivation Ψ corresponding to τ. Every sentence in Ψ corresponds to a cut $\mathcal{C}$ in τ. We call any such $\mathcal{C}$ a ***prioritized leftmost (PLM) cut*** of τ. For a set of trees T and a variable $A \in \mathrm{V}$, the ***Parikh supremum*** of variable A in T, denoted by $\sup_{\mathrm{Pk}}(A, T)$, is the maximum number of occurrences of A in any PLM cut of any tree $\tau \in T$. Observe that any PLM derivation sequence corresponding to a tree τ in T can have at most $\sup_{\mathrm{Pk}}(A, T)$ occurrences of the variable A in any sentence.

Ambiguity. We will say that a set of trees Γ is ***ambiguous*** if there are two distinct trees τ_1, τ_2 such that $\mathrm{Fr}(\tau_1) = \mathrm{Fr}(\tau_2)$; if Γ is not ambiguous, we say it is ***unambiguous***. The ***ambiguity function*** $\mu_G : \mathbb{N} \to \mathbb{N}$ for a grammar G is a function mapping every natural number n to the maximal number of PLM derivations which a word of length n may have. Formally, $\mu_G(n) = \max_{w \in \mathrm{L}(\mathrm{G}), |w|=n} |\mathrm{parse}_{\mathrm{G}}(w)|$. A grammar is said to have ***exponential ambiguity*** if its ambiguity function is in $2^{\Theta(n)}$, and it is said to have ***polynomially-bounded ambiguity***, or to be polynomially ambiguous, if its ambiguity function is in $O(n^d)$ for some $d \in \mathbb{N}_0$. Any grammar G has either exponential ambiguity or polynomially-bounded ambiguity [36]. The following characterization of polynomial ambiguity was proved in [36].

Theorem 1 [36]. *A context-free grammar* G *has polynomially-bounded ambiguity if and only if* Δ^p_{G} *is unambiguous.*

We conclude the preliminaries by recalling a classical result due to Parikh [27].

Theorem 2 (Parikh's Theorem [27]**).** *For every context-free grammar* G*, there is a right-linear context-free grammar* G′ *such that* $\mathrm{Pk}(\mathrm{L}(\mathrm{G})) = \mathrm{Pk}(\mathrm{L}(\mathrm{G}'))$.

2.1 Weighted and Stochastic Context-Free Grammars

Weighted context-free grammars define a function that associates a value in a semiring with each string. Stochastic context-free grammars are special weighted

context-free grammars that associate probabilities with strings. We recall these classical definitions in this section. We begin by defining a semiring.

Semiring. A ***semiring*** is a structure $\mathbb{D} = (D, \oplus, \otimes, 0_D, 1_D)$ where $(D, \oplus, 0_D)$ is a *commutative monoid* with identity 0_D, $(D \setminus \{0_D\}, \otimes, 1_D)$ is a *monoid* with identity 1_D, $\otimes$ distributes over $\oplus$ (i.e., $(a \oplus b) \otimes c = (a \otimes c) \oplus (b \otimes c)$ and $a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c)$, for every $a, b, c \in D$), and 0_D is an *annhilator* for $\otimes$ (i.e., $a \otimes 0_D = 0_D \otimes a = 0$ for every $a \in D$). We abuse notation and use $\mathbb{D}$ to denote the semiring and the underlying set where the meaning is clear from context. We define $\mathbb{D}_0 = D \setminus \{0_D\}$. When considering an abstract semiring $\mathbb{D}$, we'll write $0_{\mathbb{D}}$ and $1_{\mathbb{D}}$ for 0_D and 1_D respectively. An ***idempotent semiring*** satisfies the additional requirement that for all $a \in \mathbb{D}$, $a \oplus a = a$. A ***commutative semiring*** is one where $\otimes$ is commutative, i.e., $(D \setminus \{0_D\}, \otimes, 1_D)$ is a commutative monoid as well.

Example 2. Classical examples of a semiring are the tropical semiring and the probability semiring. The tropical or min-plus semiring is $(\mathbb{N} \cup \{\infty\}, \min, +, \infty, 0)$, where ∞ is taken to be larger than everything in $\mathbb{N}$. It is commutative and idempotent as $\min(a, a) = a$ for any a. The probability semiring is $([0, 1], +, \times, 0, 1)$, where $[0, 1]$ is the set of reals between 0 and 1. It is commutative as $\times$ is commutative. However, since the addition of two numbers is not idempotent, the probability semiring is not idempotent.

Weighted context-free grammars. A ***weighted context-free grammar*** is a pair (G, W) where $G = (\mathrm{V}, \Sigma, \mathrm{P}, S)$ is a context-free grammar, and $\mathrm{W} : \mathrm{P} \to \mathbb{D}$ assigns a weight from $\mathbb{D}$ to each production in P, for some semiring $\mathbb{D}$. (Note that W may assign $0_{\mathbb{D}}$ to some productions in P.) The ***weight*** of a PLM derivation $\psi = \alpha_1 \overset{\pi_1}{\Longrightarrow}_{\mathrm{plm}} \alpha_2 \overset{\pi_2}{\Longrightarrow}_{\mathrm{plm}} \cdots \overset{\pi_{n-1}}{\Longrightarrow}_{\mathrm{plm}} \alpha_n$ of G, is given by $\mathrm{W}(\psi) \triangleq \otimes_{i=1}^{n-1} \mathrm{W}(\pi_i)$. For $w \in \Sigma^*$, $\mathrm{W}(w) \triangleq \oplus_{\psi \in \mathrm{parse}_{\mathrm{G}}(w)} \mathrm{W}(\psi)$; we assume that if $\mathrm{parse}_{\mathrm{G}}(w) = \emptyset$ (i.e., $w \notin \mathrm{L}(\mathrm{G})$) then $\mathrm{W}(w) = 0_{\mathbb{D}}$. The ***semantics*** of a weighted grammar (G, W), denoted $[\![\mathrm{G}]\!]_{\mathrm{W}} : \Sigma^* \to \mathbb{D}$, is the function mapping each word to its weight in G, i.e., $[\![\mathrm{G}]\!]_{\mathrm{W}}(w) \triangleq \mathrm{W}(w)$.

Example 3. Let G be the grammar described in Example 1. Consider a weight function W that assigns weights from the tropical semiring, with the weight of every production $\pi \in \mathrm{P}$ being equal to 1. Then the semantics of (G, W) is given as $[\![\mathrm{G}]\!]_{\mathrm{W}}(a^n b^n) = 2n$ and $[\![\mathrm{G}]\!]_{\mathrm{W}}(w) = 0$, when $w \notin \mathrm{L}(\mathrm{G})$.

Definition 2. *The* Parikh image *of a weighted context-free grammar* (G, W)*, written as* $\mathrm{Pk}\,[\![\mathrm{G}]\!]_{\mathrm{W}}$ *is function defined as*

$$\mathrm{Pk}\,[\![\mathrm{G}]\!]_{\mathrm{W}}(w) \triangleq \bigoplus_{w' \in [w]_{\mathrm{Pk}}} [\![\mathrm{G}]\!]_{\mathrm{W}}(w')$$

Stochastic Context-free Grammars. A ***stochastic context-free grammar*** is a weighted context-free grammar $(\mathrm{G} = (\mathrm{V}, \Sigma, \mathrm{P}, S), \mathrm{W})$ where the weight domain is the probability semiring $([0,1], +, \times, 0, 1)$, and for any $A \in \mathrm{V}$ and $\mathtt{a} \in \Sigma$, we have

$$\sum_{\alpha \in V^* : (A \to \mathtt{a}\alpha) \in \mathrm{P}} \mathrm{W}(A \to \mathtt{a}\alpha) \in [0,1]\,.$$

3 A Parikh's Theorem for Weighted CFGs

The main result of this section is that for any weighted context-free grammar over an idempotent, commutative semiring (like the tropical semiring), there is a Parikh equivalent weighted right-linear context-free grammar. Thus, this observation extends the classical result to weighted CFGs over idempotent semirings.

Theorem 3 (Weighted Parikh's Theorem). *For every weighted context-free grammar* (G, W) *over an* idempotent, commutative *semiring, there exists a Parikh-equivalent weighted right-linear grammar* $(\mathrm{G}_*, \mathrm{W}_*)$*, that is, we have*

$$\mathrm{Pk}\,[\![\mathrm{G}]\!]_{\mathrm{W}} = \mathrm{Pk}\,[\![\mathrm{G}_*]\!]_{\mathrm{W}_*}.$$

Proof. The full proof can be found in [3]. Here we present the broad ideas.

Let $\mathrm{G} = (\mathrm{V}, \Sigma, \mathrm{P}, S)$ be a context-free grammar and let $W : \mathrm{P} \to \mathbb{D}$ be a weight function over a commutative, idempotent weight domain $\mathbb{D}$. Consider the following homomorphism $h : \mathrm{P}^* \to \Sigma^*$ defined as $h(\pi) = \mathtt{a}$, where $\pi = A \to \mathtt{a}\alpha \in \mathrm{P}$.

We begin by first constructing a weighted context-free grammar $(\mathrm{G}_1, \mathrm{W}_1)$ over the alphabet P, whose image under h gives us G. Formally, $\mathrm{G}_1 = (\mathrm{V}, \mathrm{P}, \mathrm{P}_1, S)$ has as productions $\mathrm{P}_1 = \{A \to \pi\alpha \mid \exists \mathtt{a} \in \Sigma.\ \pi = A \to \mathtt{a}\alpha \in \mathrm{P}\}$. In addition, take W_1 to be $\mathrm{W}_1(A \to \pi\alpha) = \mathrm{W}(\pi)$. It is easy to see that $h(\mathrm{L}(\mathrm{G}_1)) = \mathrm{L}(\mathrm{G})$ by construction. Moreover, given W_1 and W, we can conclude $[\![\mathrm{G}]\!]_{\mathrm{W}}(w) = \bigoplus_{\omega \in \mathrm{P}^* : h(\omega) = w} [\![\mathrm{G}_1]\!]_{\mathrm{W}_1}(\omega)$.

By Parikh's theorem (Theorem 2), there is a right-linear grammar $\mathrm{G}_2 = (\mathrm{V}_2, \mathrm{P}, \mathrm{P}_2, S_2)$ such that $\mathrm{Pk}(\mathrm{L}(\mathrm{G}_2)) = \mathrm{Pk}(\mathrm{L}(\mathrm{G}_1))$. Define the weight function W_2 as $\mathrm{W}_2(A \to \pi B) = \mathrm{W}(\pi)$ to give us the weighted CFG $(\mathrm{G}_2, \mathrm{W}_2)$. Using the fact that $\otimes$ is commutative, and $\oplus$ is idempotent, we can prove that $\mathrm{Pk}\,[\![\mathrm{G}_1]\!]_{\mathrm{W}_1} = \mathrm{Pk}\,[\![\mathrm{G}_2]\!]_{\mathrm{W}_2}$.

Finally, we consider the context free grammar G_3 obtained by "applying the homomorphism h" to G_2. Formally, $\mathrm{G}_3 = (\mathrm{V}_2, \Sigma, \mathrm{P}_3, S_2)$, where $\mathrm{P}_3 = \{A \to h(\pi)B \mid A \to \pi B \in \mathrm{P}_2\}$. The weight function W_3 is defined in such a way that weight of $A \to \mathtt{a}B$ is the sum of the weights of all productions $A \to \pi B$, where $h(\pi) = \mathtt{a}$, i.e.,

$$\mathrm{W}_3(A \to \mathtt{a}B) = \bigoplus_{\pi : h(\pi) = \mathtt{a}} \mathrm{W}_2(A \to \pi B).$$

$(\mathrm{G}_3, \mathrm{W}_3)$ and $(\mathrm{G}_2, \mathrm{W}_2)$ share the same relationship as (G, W) and $(\mathrm{G}_1, \mathrm{W}_1)$. That is, we have $h(\mathrm{L}(\mathrm{G}_2)) = \mathrm{L}(\mathrm{G}_3)$ and $[\![\mathrm{G}_3]\!]_{\mathrm{W}_3}(w) = \bigoplus_{\omega \in \mathrm{P}^* : h(\omega) = w} [\![\mathrm{G}_2]\!]_{\mathrm{W}_2}(\omega)$.

$(\mathrm{G}_3, \mathrm{W}_3)$ is the desired weighted grammar, i.e., $\mathrm{Pk}\,[\![\mathrm{G}_3]\!]_{\mathrm{W}_3} = \mathrm{Pk}\,[\![\mathrm{G}]\!]_{\mathrm{W}}$. □

Corollary 1. *If* (G, W) *is a weighted context-free grammar over an* idempotent, commutative *weight domain and a unary alphabet, then there exists a weighted right-linear context-free grammar* $(\mathrm{G}', \mathrm{W}')$ *such that* $[\![G']\!]_{\mathrm{W}'} = [\![\mathrm{G}]\!]_{\mathrm{W}}$.

Example 4. Starting with the weighted grammar (G, W) from Example 3, the construction used in the proof of Theorem 3 would have P_1 containing the following productions: $S \to \pi_1 SB, \quad S \to \pi_2 B, \quad B \to \pi_3$. The language of this grammar is $\mathrm{L}(\mathrm{G}_1) = \left\{ \pi_1^n \pi_2 \pi_3^{n+1} \mid n \geq 0 \right\}$.

One candidate for G_2 would have as productions $S \to \pi_1 J, \quad S \to \pi_2 K$, $J \to \pi_3 S$, and $K \to \pi_3$. The language is $\mathrm{L}(\mathrm{G}_2) = \{(\pi_1\pi_3)^n \pi_2 \pi_3 \mid n \geq 0\}$.

In that case $(\mathrm{G}_3, \mathrm{W}_3)$ would have productions and weights as follows:

$$\mathrm{W}_3(S \to \mathtt{a}J) = 1 \qquad \mathrm{W}_3(S \to \mathtt{a}K) = 1$$
$$\mathrm{W}_3(J \to \mathtt{b}S) = 1 \qquad \mathrm{W}_3(K \to \mathtt{b}) = 1$$

The language of the underlying grammar would be $\mathrm{L}(\mathrm{G}_3) = \{(\mathtt{ab})^n \mathtt{ab} \mid n \geq 0\}$, and

$$[\![\mathrm{G}_3]\!]_{\mathrm{W}_3}(w) = \begin{cases} 2k & \text{if } w = (\mathtt{ab})^k \text{ for some } k \geq 1 \\ 0 & \text{otherwise} \end{cases}$$

4 A Counterexample to Parikh's Theorem for Stochastic Grammars

Theorem 3 crucially relies on the semiring being idempotent. In this section, we show that Theorem 3 fails to generalize if we drop the requirement of idempotence. We give an example of a stochastic context-free grammar over the unary alphabet that is not equivalent to any stochastic right-linear grammar. Before presenting the example stochastic context-free grammar and proving the inexpressivity result, we recall some classical observations about unary stochastic right linear grammars.

4.1 Properties of Unary Stochastic Right-Linear Grammars

Stochastic right-linear grammars satisfy pumping lemma type properties. Here we recall such an observation for unary stochastic right-linear grammars.

Theorem 4 (Pumping Lemma). *Let* $(\mathrm{G} = (\mathrm{V}, \{\mathtt{a}\}, \mathrm{P}, S), \mathrm{W})$ *be a stochastic right-linear grammar over the unary alphabet. There is a number* k*, and real numbers* $c_0, c_1, \ldots c_k$ *with* $c_0 + c_1 + \cdots + c_k = 1$ *such that for every* $\ell \in \mathbb{N}$

$$[\![\mathrm{G}]\!]_{\mathrm{W}}(\mathtt{a}^{\ell+k+1}) = \sum_{i=0}^{k} c_i \, [\![\mathrm{G}]\!]_{\mathrm{W}}(\mathtt{a}^{\ell+i})$$

Proof. The result is a consequence of the Cayley-Hamilton theorem and the fact that 1 is an eigen value of stochastic matrices. We skip the proof as it is a specialization of Theorem 2.8 in Chapter II.C in [28]. □

Let (G, W) be a unary weighted context-free grammar. The generating function of such a grammar is $P(x) = \sum_{k=0}^{\infty} [\![\mathrm{G}]\!]_{\mathrm{W}}(\mathtt{a}^k)x^k$. We conclude this section by observing that if G is right-linear, then its generating function must be a rational function, i.e., $P(x)$ is an algebraic fraction where both the numerator and denominator are polynomials.

Theorem 5. *Let* (G, W) *be a stochastic right-linear grammar over the unary alphabet. Then the generating function* $P(x) = \sum_{k=0}^{\infty} [\![\mathrm{G}]\!]_{\mathrm{W}}(\mathtt{a}^k)x^k$ *is a rational function.*

Proof. Observe that Theorem 4 says that the sequence $\langle [\![\mathrm{G}]\!]_{\mathrm{W}}(\mathtt{a}^n)\rangle_{n\in\mathbb{N}}$ satisfies a linear homogeneous recurrence with constant coefficients. Thus, its generating function must be rational. □

4.2 The Counterexample

We now present a unary weighted CFG and show that there is no weighted right-linear CFG that is equivalent to it. Consider the grammar $\mathrm{G}_* = (\{S\}, \{\mathtt{a}\}, \{(S \to \mathtt{a}), (S \to \mathtt{a}SS)\}, S)$. Let p be some number in $(0, 1)$. The weight function W_* is defined as follows: $\mathrm{W}_*(S \to \mathtt{a}) = 1 - p$, and $\mathrm{W}_*(S \to \mathtt{a}SS) = p$. Taking c_n to be the nth Catalan number, we can see that $[\![\mathrm{G}_*]\!]_{\mathrm{W}_*}(\mathtt{a}^{2k+1}) = c_k p^k (1-p)^{k+1}$; this is because the probability of any PLM derivation for $\mathtt{a}^{2k+1}$ is $p^k(1-p)^{k+1}$ and there are c_k elements in $\mathrm{parse}_{\mathrm{G}_*}(\mathtt{a}^{2k+1})$. Taking $b_k = [\![\mathrm{G}_*]\!]_{\mathrm{W}_*}(\mathtt{a}^k)$, we have

$$b_k = \begin{cases} c_{(k-1)/2} p^{(k-1)/2}(1-p)^{(k-1)/2+1} & \text{if } k \text{ is odd} \\ 0 & \text{otherwise} \end{cases}$$

Recall that the generating function for the Catalan numbers, $C(z) = \sum_{k\geq 0} c_k z^k$, is given by $C(z) = \frac{1-\sqrt{1-4z}}{2z}$. Based on the above observations, the generating function for the grammar $(\mathrm{G}_*, \mathrm{W}_*)$, $P(z) = \sum_{k\geq 0} b_k z^k$ can be written as follows.

$$\begin{aligned} P(z) &= \sum_{k\geq 0} b_k z^k = \sum_{k\geq 0} b_{2k+1} z^{2k+1} \\ &= z \sum_{k\geq 0} b_{2k+1} \left(z^2\right)^k = z \sum_{k\geq 0} c_k p^k (1-p)^{k+1} \left(z^2\right)^k \\ &= z(1-p) \sum_{k\geq 0} c_k \left(z^2 p(1-p)\right)^k \\ &= z(1-p) C(z^2 p(1-p)) \end{aligned}$$

$$= z(1-p)\frac{1-\sqrt{1-4z^2p(1-p)}}{2z^2p(1-p)}$$
$$= \frac{1}{2zp}\left(1-\sqrt{1-4z^2p(1-p)}\right)$$

Having identified an expression for the generating function for $(\mathrm{G}_*, \mathrm{W}_*)$, we are ready to prove that there is no Parikh equivalent right-linear grammar for $(\mathrm{G}_*, \mathrm{W}_*)$. First notice that if a weighted grammar (G, W) is over the unary alphabet, then $[\![\mathrm{G}]\!]_{\mathrm{W}} = \mathrm{Pk}\,[\![\mathrm{G}]\!]_{\mathrm{W}}$. Therefore, to establish the result of this section, it suffices to prove the statement that there is no right-linear grammar that is equivalent to $(\mathrm{G}_*, \mathrm{W}_*)$; this is the content of the next theorem.

Theorem 6. *There is no stochastic right-linear grammar* (G, W) *such that* $[\![\mathrm{G}]\!]_{\mathrm{W}} = [\![\mathrm{G}_*]\!]_{\mathrm{W}_*}$.

Proof. Given Theorem 5, it suffices to prove that the generating function $P(z)$ for $(\mathrm{G}_*, \mathrm{W}_*)$ is not rational. Taking $Q(z) = \sqrt{1-4z^2p(1-p)}$, we see that $P(z) = \frac{1}{2zp}(1-Q(z))$. Given this relationship, we can conclude that if $P(z)$ is rational function, then so is $Q(z)$. Thus, our goal will be to prove that $Q(z)$ is not a rational function.

Assume for contradiction that $Q(z)$ is rational. Then $Q(z) = f(z)/g(z)$ where f and g are both polynomials with greatest common divisor having degree 0. Then $Q^2(z) = f^2(z)/g^2(z)$ and $Q^2(z)g^2(z) = \left(1-4z^2p(1-p)\right)g^2(z) = f^2(z)$. Thus, $Q^2(z)$ must divide $f^2(z)$. We observe that $Q^2(z) = (1-2z\sqrt{p(1-p)})(1+2z\sqrt{p(1-p)})$ is square-free so $Q^2(z)$ must divide $f(z)$, and $f(z) = Q^2(z)h(z)$ for some polynomial h. Substituting for $f(z)$ in $Q^2(z)g^2(z) = f^2(z)$ and rearranging we obtain $Q^2(z)g^2(z) = (Q^2(z))^2h^2(z) \implies g^2(z) = Q^2(z)h^2(z)$, and by the same argument as above, $Q^2(z)$ divides $g(z)$. Thus, $Q^2(z)$ divides both $f(z)$ and $g(z)$. Since $Q^2(z)$ is not a degree 0 polynomial, we contradict the assumption that the greatest common divisor of f and g has degree 0. □

5 Parikh's Theorem for Unary Polynomially Ambiguous Stochastic Grammars

The weighted stochastic context-free grammar $(\mathrm{G}_*, \mathrm{W}_*)$ in Sect. 4.2 is exponentially ambiguous; the ambiguity function μ_{G_*} is bounded by the Catalan numbers. Exponential ambiguity turns out to be critical to construct such counterexamples. In this section, we prove that any unary stochastic context-free grammar with polynomial ambiguity is equivalent to a unary stochastic right-linear grammar. The proof of this result relies on an observation that in any PLM cut in a complete derivation tree of a unary polynomially ambiguous grammar, the number of occurences of any variable is bounded by a constant dependent on the grammar. The unary alphabet assumption is crucial in obtaining such a bound (Lemma 3). In the next two subsections we present a proof of this observation by first bounding the number of occurrences in cuts of pumping trees

and then using it to bound it in complete derivation trees. In Sect. 5.3, we then present the construction of the right-linear grammar. Though we present this result in the context of stochastic grammars, it applies to any weighted CFG over a commutative (but not necessarily idempotent) semiring.

In the rest of this section, let us fix a unary, polynomially ambiguous, context-free grammar $\mathrm{G} = (\mathrm{V}, \{\mathtt{a}\}, \mathrm{P}, S)$ and a stochastic weight function Pr (for probability). We assume that the set of variables V is partitioned into single-derivation variables X and multiple-derivation variables Y. As we have done throughout this paper, we assume that every production in G is "useful", that is, is used in some complete derivation tree whose root is labeled S. Finally we will assume that m is the maximum length of the right-hand side of any production in P.

5.1 Parikh Suprema in Pumping Trees

In this section we will bound the number of times a variable can appear in any PLM cut of a pumping tree in G. We begin by observing some simple properties about single-derivation variables X and multiple-derivation variables Y. Since every production in the grammar is useful, we can conclude that there is a unique complete derivation tree with root A if $A \in \mathrm{X}$, and that there are at least two complete derivation trees with root A if $A \in \mathrm{Y}$, i.e., $|\Delta^{\Sigma}_{\mathrm{G}}(A)| = 1$ if $A \in \mathrm{X}$, and $|\Delta^{\Sigma}_{\mathrm{G}}(A)| > 1$ if $A \in \mathrm{Y}$. Next, for $A \in \mathrm{X}$, the unique $\tau \in \Delta^{\Sigma}_{\mathrm{G}}(A)$ has the following properties: (a) no node is labeled by a variable in Y; (b) each variable in X labels at most one node along any path in τ. Property (b) holds because if $A \in \mathrm{X}$ has a derivation $A \Rightarrow^{*}_{\mathrm{plm}} \alpha A \beta$, then A cannot have any complete derivation tree and it would be useless. This also means that any pumping tree $\tau \in \Delta^{p}_{\mathrm{G}}$ must have $\ell(\tau) \in \mathrm{Y}$. These properties allow us to bound the size of the unique complete derivation for variable $A \in \mathrm{X}$.

Lemma 1. *For any $A \in \mathrm{X}$, the unique tree $\tau \in \Delta^{\Sigma}_{\mathrm{G}}(A)$ has size at most $m^{|\mathrm{X}|}$.*

Proof. Since only variables in X can appear as labels in τ and no variable appears more than once in any path, the height of τ is $\leq |\mathrm{X}|$. Finally, since any node has at most m children, we get the bound on the size of τ. □

Next we prove that Lemma 1 allows one to bound the number of times any single-derivation variable appears in any PLM cut of a pumping tree.

Lemma 2. *For any $A \in \mathrm{Y}$ and $B \in \mathrm{X}$, $\sup_{\mathrm{Pk}}(B, \Delta^{p}_{\mathrm{G}}(A)) \leq m^{|\mathrm{X}|+1}$.*

Proof. Let $\tau \in \Delta^{p}_{\mathrm{G}}(A)$ be an arbitrary A-pumping tree, where $A \in \mathrm{Y}$. Let $\mathcal{C}$ be an arbitrary PLM cut of τ. We will prove a slightly stronger statement; we will show that the total number of single-derivation variables in $\mathcal{C}$ is $\leq m^{|\mathrm{X}|+1}$. This will bound the Parikh supremum for any single-derivation variable.

Without loss of generality, assume that $\mathcal{C}$ has at least one node with label in X. Amongst all nodes in $\mathcal{C}$ that are labeled by a variable in X, let n be the node that is closest to the root, and if there are multiple such nodes, take n to be the leftmost one. From the definition of PLM cuts, the following property holds for

$\mathcal{C}$ and n: (a) any node to right of n in $\mathcal{C}$ that is labeled by a variable in X must be a sibling of n, and (b) all nodes to the left of n in $\mathcal{C}$ labeled by variables of X must be descendents of some left sibling (say n_1) of n that is also labeled by a variable in X. Thus, the number of nodes to the right of n (including n) in $\mathcal{C}$ labeled by X is at most m, and, by Lemma 1, the number of nodes to the left of n in $\mathcal{C}$ labeled by X is at most $m^{|\mathrm{X}|}$. Putting these together, the total number of nodes in $\mathcal{C}$ labeled by some variable in X is at most $m + m^{|\mathrm{X}|} \leq m^{|\mathrm{X}|+1}$. □

Lemma 3. *For any* $A \in \mathrm{V}$*, and* $B \in \mathrm{Y}$*,* $\sup_{\mathrm{Pk}}(B, \Delta^p_{\mathrm{G}}(A)) \leq 2$.

Proof. Let τ be a A-pumping tree, for some variable A. Note that A must be a multiple-derivation variable because of property (b) before Lemma 1. Let $\mathcal{C}$ be any PLM cut of τ. Since τ is an A-pumping tree it must contain A in its frontier. Then there must be some node n in $\mathcal{C}$ such that the subtree $\tau(n)$ contains A in its frontier. Let $C = \ell(n)$. Observe that C is a multiple-derivation variable because a node labeled $A \in \mathrm{Y}$ is a descendent. Thus, there are two complete derivation trees τ_1^C, τ_2^C with roots labeled C (Remark 1).

We'll first show that there cannot be more than two occurrences of nodes labeled C in $\mathcal{C}$. Assume towards the contrary that there are at least three nodes n_1, n_2, n_3 in $\mathcal{C}$ with $\ell(n_1) = \ell(n_2) = \ell(n_3) = C$. Without loss of generality, assume n_1, n_2, and n_3 are in left-to-right order in τ and $n \in \{n_1, n_2, n_3\}$. Since n_1, n_2, n_3 belong to a cut, they are not related by the ancestor/descendent relationship.

Let τ_1 be the tree $\tau[n_1 \mapsto \tau_1^C, n_2 \mapsto \tau_2^C, n_3 \mapsto \tau(n)]$, and let τ_2 be the tree $\tau[n_1 \mapsto \tau_2^C, n_2 \mapsto \tau_1^C, n_3 \mapsto \tau(n)]$. By construction, τ_1 and τ_2 are both A-pumping trees with $\mathrm{Fr}(\tau_1) = \mathrm{Fr}(\tau_2)$ and $\tau_1 \neq \tau_2$. However, since G is polynomially ambiguous, by Theorem 1, the set of pumping trees is unambiguous, giving us the desired contradiction.

Next, we show that there cannot be more than two nodes labeled $B \in \mathrm{Y}$ in $\mathcal{C}$, where $B \neq C$. Assume that there are at least three nodes n_1, n_2, n_3 in $\mathcal{C}$ with $\ell(n_1) = \ell(n_2) = \ell(n_3) = B$. Again assume n_1, n_2, and n_3 are in left-to-right order in τ. Further, since $B \in \mathrm{Y}$, there are two complete derivation trees τ_1^B and τ_2^B with root labeled B (Remark 1).

Observe that at least two nodes of $\{n_1, n_2, n_3\}$ must lie to one side of n in τ. Without loss of generality we may assume that n_1 and n_2 are those nodes. Let τ_1 be the tree $\tau[n_1 \mapsto \tau_1^B, n_2 \mapsto \tau_2^B]$, and let τ_2 be the tree $\tau[n_1 \mapsto \tau_2^B, n_2 \mapsto \tau_1^B]$. Clearly, τ_1, τ_2 are A-pumping trees with $\mathrm{Fr}(\tau_1) = \mathrm{Fr}(\tau_2)$, and $\tau_1 \neq \tau_2$. □

5.2 Parikh Suprema in Complete Derivation Trees

We will now use the results in Sect. 5.1 to bound the Parikh supremum of any variable in a complete derivation tree of G. The key property we will exploit is the fact that any complete derivation tree can be written as the "composition" of a small number of pumping trees (see Fig. 1) such that any PLM cut is the union of cuts in each of these pumping trees. The bounds on Parikh suprema will then follow from the observations in Sect. 5.1.

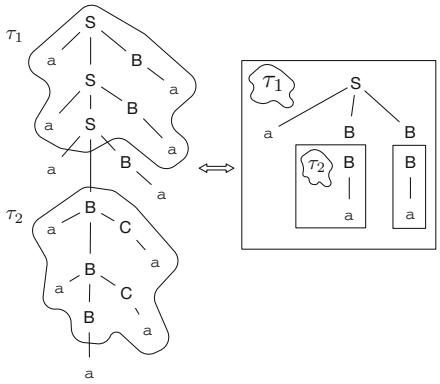

Fig. 1. A complete derivation tree for the grammar in Example 5 on the left and the compressed tree data structure with removed pumping trees on the right.

We begin with some convenient notation. For a $\tau \in \Delta_G$, let longestpath(τ) denote the longest path from the root of τ to a node labeled $\ell(\tau)$. If there are multiple such paths, longestpath(τ) is the lexicographically-first path among them. Note that longestpath(τ) can be ε if the root is the only node with label $\ell(\tau)$ in τ. Let depth(τ) denote the length of the longest path from root to leaf in τ.

We now describe two procedures `compress` and `decompress`. Let us fix a complete derivation tree τ. The procedure `compress` returns a data structure of pumping trees. These pumping trees are small in number and τ is the "composition" of these pumping trees. Let n be the lowest node in τ that has the same label as the root. `compress` identifies the pumping tree obtained by removing the children of n, and recursively compresses the subtrees rooted at the children of n. Note that if n is the same as the root, then the pumping tree identified by `compress` will just be the tree with one node.

`compress`(τ):
  If $\tau = A_{\mathtt{a}}$ for some $A \in \mathrm{V}$, **return** τ
  $p \leftarrow$ longestpath(τ)
  Let A, $\mathtt{a}$ and α be such that $\tau(p) = A_{\mathtt{a}\alpha}(\tau(p \cdot 1), \ldots, \tau(p \cdot k))$
  $\tau_{\mathrm{pump}} \leftarrow \mathsf{rem}_p(\tau)$
  Return $[\tau_{\mathrm{pump}}, A \to \mathtt{a}\alpha, \mathtt{compress}(\tau(p \cdot 1)), \ldots, \mathtt{compress}(\tau(p \cdot k))]$

The tree τ is the "composition" of pumping trees in the data structure returned by `compress`. We describe this "composition operation" itself by an algorithm `decompress`.

`decompress`(τ_c):
  If $\tau_c = A_{\mathtt{a}}$ for some $A \in \mathrm{V}$, **return** τ_c
  Let τ_c be of the form $[\tau_{\mathrm{pump}}, A \to \mathtt{a}\alpha, \tau_c^1, \ldots, \tau_c^k]$
  $\tau' \leftarrow A_{\mathtt{a}\alpha}(\mathtt{decompress}(\tau_c^1), \ldots, \mathtt{decompress}(\tau_c^k))$
  Return $\tau_{\mathrm{pump}}[\text{longestpath}(\tau_{\mathrm{pump}}) \mapsto \tau']$

The following lemma characterizing the relationship between `compress` and `decompress` is easy to see.

Lemma 4. *For any complete derivation tree τ, $\tau = \texttt{decompress}(\texttt{compress}(\tau))$.*

Example 5. Consider a grammar $(\{S, B, C\}, \{\mathtt{a}\}, \mathrm{P}, S)$ with productions

$$S \to \mathtt{a}SB|\mathtt{a}BB|\mathtt{a}B, \quad B \to \mathtt{a}BC|\mathtt{a}, \quad C \to \mathtt{a}.$$

Consider the complete derivation tree shown on the left in Fig. 1. The output of `compress` will be

$$[\tau_1, S \to \mathtt{a}BB, [\tau_2, B \to \mathtt{a}], B_{\mathtt{a}}]$$

We will now show that the data structure returned by `compress` has a constant number of pumping trees. Consider a call of $\texttt{compress}(\tau)$, where p is the longestpath(τ). The key property that we exploit is the fact that the label $\ell(\tau)$ does not appear in the subtrees rooted at the children of p.

Lemma 5. *For any complete derivation tree τ, the number of trees in the data structure returned by* $\texttt{compress}(\tau)$ *is at most $m^{|\mathrm{V}|}$.*

Proof. Let $p = \text{longestpath}(\tau)$, and let $\tau(p{\cdot}0), \tau(p{\cdot}1), \ldots, \tau(p{\cdot}k)$ be the children of $\tau(p)$. As observed before, the label $\ell(\tau)$ does not appear in the subtrees $\tau(p \cdot i)$. Thus, the depth of the recursion in `compress` is bounded by $|\mathrm{V}|$. Finally, observing that $k \leq m$, we get the desired bound. □

We are now ready to prove the main result of this section.

Lemma 6. *For any variable A, $\sup_{\mathrm{Pk}}(A, \Delta_{\mathrm{G}}^{\Sigma}) \leq m^{|\mathrm{X}|+|\mathrm{V}|+1}$*

Proof. By Lemma 5, we know that the number of trees in $\texttt{compress}(\tau)$ is at most $m^{|\mathrm{V}|}$. Consider any PLM cut $\mathcal{C}$ of τ. Any node in $\mathcal{C}$ belongs to at most one tree in $\texttt{compress}(\tau)$. Further for any $\tau_1 \in \texttt{compress}(\tau)$, $\mathcal{C}$ restricted to τ_1 is a PLM cut of τ_1. Thus, $\mathcal{C}$ can be seen as the union of at most $m^{|\mathrm{V}|}$ PLM cuts in pumping trees. By Lemmas 2 and 3, the Parikh supremum of any variable in any of these pumping trees is at most $m^{|\mathrm{X}|+1}$. These observations together establish the bound. □

5.3 Right-Linear SCFG for Polynomially Ambiguous SCFGs

For this section, let us fix $k = m^{|\mathrm{V}|+|\mathrm{X}|+1}$. By Lemma 6, in any PLM derivation of G, any variable appears at most k times at any step. Since k is a constant, the right-linear grammar can simulate every PLM derivation of G by explicitly keeping track of only k copies of any variable. This idea is very similar to the one used in [12]. We now give the formal definition of the right-linear grammar based on this intuition.

For a sentence $\alpha \in (\Sigma \cup \mathrm{V})^*$, we define $\boldsymbol{\ell f}(\alpha) \triangleq \alpha\restriction_{\Sigma}\alpha\restriction_{\mathrm{X}}\alpha\restriction_{\mathrm{Y}}$. The stochastic right-linear grammar $(\mathrm{G}_1 = (\mathrm{V}_1, \{\mathrm{a}\}, \mathrm{P}_1, S_1), \mathrm{Pr}_1)$ is formally defined as follows.

1. $\mathrm{V}_1 = \{\langle\alpha\rangle \mid \alpha \in \mathrm{X}^*\mathrm{Y}^*$such that for each $A \in \mathrm{V}.\ \mathrm{Pk}(\alpha)(A) \leq k\}$. Thus, the variables of G_1 are sequences of single-derivation variables followed by multiple-derivation variables from G in which each variable appears at most k times.
2. $S_1 = \langle S\rangle$
3. For any production $\pi = (A \to \mathtt{a}\beta) \in \mathrm{P}$ and sentence $\alpha \in \mathrm{V}^*$ we define a production
$$\pi^\alpha = (\langle A\alpha\rangle \to \mathtt{a}\,\langle \boldsymbol{\ell f}(\beta\alpha)\rangle)$$
corresponding to applying the production π as a PLM step from the sentence $A\alpha$. The set P_1 is defined as
$$\mathrm{P}_1 = \{\pi^\alpha \mid \pi = (A \to \mathtt{a}\beta) \in \mathrm{P} \wedge \langle A\alpha\rangle, \langle\beta\alpha\rangle \in \mathrm{V}_1\}$$
4. Finally Pr_1 is defined as $\mathrm{Pr}_1(\pi^\alpha) = \mathrm{Pr}(\pi)$ for all $\pi^\alpha \in \mathrm{P}_1$.

We first observe that $(\mathrm{G}_1, \mathrm{Pr}_1)$ is a stochastic CFG. The proof is in [3]

Proposition 1. $(\mathrm{G}_1, \mathrm{Pr}_1)$ *is a stochastic CFG.*

$(\mathrm{G}_1, \mathrm{Pr}_1)$ is equivalent to $(\mathrm{G}, \mathrm{Pr})$. Its proof is in [3].

Theorem 7. *For any unary, stochastic grammar* $(\mathrm{G}, \mathrm{Pr})$ *of polynomial ambiguity, there is a stochastic right-linear grammar* $(\mathrm{G}_1, \mathrm{Pr}_1)$ *such that* $[\![\mathrm{G}]\!]_{\mathrm{Pr}} = [\![\mathrm{G}_1]\!]_{\mathrm{Pr}_1}$.

6 Conclusions

In this paper we investigated whether Parikh's theorem generalizes to weighted automata. We proved that it does indeed when the weighted context-free grammar is over a commutative, idempotent semiring. We showed that idempotence of the weight domain is necessary by demonstrating that Parikh's theorem does not extend to unary, stochastic grammars. However, we proved that if the context-free grammar is polynomially ambiguous, then idempotence of the weight domain is not required for Parikh's theorem to hold.

Our proof for Parikh's theorem for commutative and idempotent semirings extends (as is) to pushdown automata (as opposed to context-free grammars). However, the same does not apply to our result for unary, polynomially ambiguous grammars over non-idempotent rings. Our current proof subtly relies on the "one state" property of context-free grammars. It would be interesting to see how to generalize these ideas to the case of pushdown automata. Finally, stochastic context-free grammars have a (weaker) semantics as language acceptors — the grammar accepts a word if its weight is $> \frac{1}{2}$. Our results imply that every unary language accepted by a polynomially ambiguous, stochastic context-free grammar is also accepted by a probabilistic automata (with probability $> \frac{1}{2}$). It is open if this also holds when the grammar is exponentially ambiguous; our counterexample in this paper only shows that there is no probabilistic automaton that satisfies the stronger requirement that words are accepted with the *same* probability.

References

1. Aminof, B., Kupferman, O., Lampert, R.: Reasoning about online algorithms with weighted automata. ACM Trans. Algorithms **6**(2), 28 (2010)
2. Beeri, C., Milo, T.: Schemas for integration and translation of structured and semi-structured data. In: Beeri, C., Buneman, P. (eds.) ICDT 1999. LNCS, vol. 1540, pp. 296–313. Springer, Heidelberg (1999). doi:10.1007/3-540-49257-7_19
3. Bhattiprolu, V., Gordon, S., Viswanathan, M.: Extending Parikh's theorem to weighted and probabilistic context-free grammars. Technical report, University of Illinois, Urbana-Champaign (2017). http://hdl.handle.net/2142/96262
4. Buchsbaum, A., Giancarlo, R., Westbrook, J.: On the determinization of weighted finite automata. SIAM J. Comput. **30**(5), 1502–1531 (2000)
5. Chatterjee, K., Doyen, L., Henzinger, T.A.: Quantitative languages. In: Kaminski, M., Martini, S. (eds.) CSL 2008. LNCS, vol. 5213, pp. 385–400. Springer, Heidelberg (2008). doi:10.1007/978-3-540-87531-4_28
6. Chatterjee, K., Doyen, L., Henzinger, T.A.: Alternating weighted automata. In: Kutyłowski, M., Charatonik, W., Gębala, M. (eds.) FCT 2009. LNCS, vol. 5699, pp. 3–13. Springer, Heidelberg (2009). doi:10.1007/978-3-642-03409-1_2
7. Culik, K., Kari, J.: Image compression using weighted finite automata. Comput. Graph. **17**, 305–313 (1993)
8. Dang, Z., Ibarra, O.H., Bultan, T., Kemmerer, R.A., Su, J.: Binary reachability analysis of discrete pushdown timed automata. In: Emerson, E.A., Sistla, A.P. (eds.) CAV 2000. LNCS, vol. 1855, pp. 69–84. Springer, Heidelberg (2000). doi:10.1007/10722167_9
9. Droste, M., Gastin, P.: Weighted automata and weighted logics. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 513–525. Springer, Heidelberg (2005). doi:10.1007/11523468_42
10. Droste, M., Kuich, W., Vogler, H. (eds.): Handbook of Weighted Automata. Springer, Heidelberg (2009)
11. Esparza, J.: Petri nets, commutative context-free grammars, and basic parallel processes. Fundam. Inform. **31**(1), 13–25 (1997)
12. Esparza, J., Ganty, P., Kiefer, S., Luttenberger, M.: Parikh's theorem: a simple and direct automaton construction. Inf. Process. Lett. **111**(12), 614–619 (2011)
13. Goldstine, J.: A simplified proof of Parikh's theorem. Discret. Math. **19**, 235–239 (1977)
14. Göller, S., Mayr, R., To, A.: On the computational complexity of verifying one-counter processes. In: Proceedings of the IEEE Symposium on Logic in Computer Science, pp. 235–244 (2009)
15. Gurari, E., Ibarra, O.: The complexity of decision problems for finite-turn multicounter machines. J. Comput. Syst. Sci. **22**, 220–229 (1981)
16. Hafner, U.: Low bit-rate image and video coding with weighted finite automata. Ph.D. thesis, Universität Würzburg (1999)
17. Huynh, T.-D.: The complexity of semilinear sets. In: Bakker, J., Leeuwen, J. (eds.) ICALP 1980. LNCS, vol. 85, pp. 324–337. Springer, Heidelberg (1980). doi:10.1007/3-540-10003-2_81
18. Huynh, D.: Deciding the ineuivalence of context-free grammars with 1-letter terminal alphabet is σ_2^p-complete. Theor. Comput. Sci. **33**, 305–326 (1984)
19. Huynh, D.: Complexity of equivalence problems for commutative grammars. Inf. Control **66**(1), 103–121 (1985)

20. Ibarra, O.: Reversal-bounded multi-counter machines and their decision problems. J. ACM **25**, 116–133 (1978)
21. Jiang, Z., Litow, B., Vel, O.: Similarity enrichment in image compression through weighted finite automata. In: Du, D.-Z.-Z., Eades, P., Estivill-Castro, V., Lin, X., Sharma, A. (eds.) COCOON 2000. LNCS, vol. 1858, pp. 447–456. Springer, Heidelberg (2000). doi:10.1007/3-540-44968-X_44
22. Katritzke, F.: Refinements of data compression using weighted finite automata. Ph.D. thesis, Universität Siegen (2001)
23. Kopczyński, E., To, A.: Parikh images of grammars: complexity and applications. In: Proceedings of the IEEE Symposium on Logic in Computer Science, pp. 80–89 (2010)
24. Kuperberg, D.: Linear temporal logic for regular cost functions. In: Proceedings of the Symposium on Theoretical Aspects of Computer Science, pp. 627–636 (2011)
25. Mohri, M.: Finite-state transducers in language and speech processing. Comput. Linguist. **23**, 269–311 (1997)
26. Mohri, M., Pereira, F., Riley, M.: The design principles of weighted finite-state transducer library. Theor. Comput. Sci. **231**, 17–32 (1997)
27. Parikh, R.J.: On context-free languages. J. ACM **13**(4), 570–581 (1966)
28. Paz, A.: Introduction to Probabilistic Automata. Academic Press, Cambridge (1971)
29. Rabin, M.: Probabilistic automata. Inf. Control **6**(3), 230–245 (1963)
30. Reps, T., Lal, A., Kidd, N.: Program analysis using weighted pushdown systems. In: Arvind, V., Prasad, S. (eds.) FSTTCS 2007. LNCS, vol. 4855, pp. 23–51. Springer, Heidelberg (2007). doi:10.1007/978-3-540-77050-3_4
31. Reps, T., Schwoon, S., Jha, S., Melski, D.: Weighted pushdown systems and their application to interprocedural dataflow analysis. Sci. Comput. Program. **58**(1–2), 206–263 (2005)
32. Schützenberger, M.: On the definition of a family of automata. Inf. Control **4**, 245–270 (1961)
33. Sen, K., Viswanathan, M.: Model checking multithreaded programs with asynchronous atomic methods. In: Ball, T., Jones, R.B. (eds.) CAV 2006. LNCS, vol. 4144, pp. 300–314. Springer, Heidelberg (2006). doi:10.1007/11817963_29
34. Siedl, H., Schwentick, T., Muscholl, A.: Counting in trees. Texts Log. Games **2**, 575–612 (2007)
35. To, A.W., Libkin, L.: Algorithmic metatheorems for decidable LTL model checking over infinite systems. In: Ong, L. (ed.) FoSSaCS 2010. LNCS, vol. 6014, pp. 221–236. Springer, Heidelberg (2010). doi:10.1007/978-3-642-12032-9_16
36. Wich, K.: Exponential ambiguity of context-free grammars. In: Developments in Language Theory, Foundations, Applications, and Perspectives, Aachen, Germany, 6–9 July 1999, pp. 125–138 (1999)
37. Yen, H.: On reachability equivalence for BPP-nets. Theor. Comput. Sci. **179**, 301–317 (1996)