Strong learning of some Probabilistic Multiple Context-Free Grammars

Alexander Clark
Department of Philosophy, Linguistics and Theory of Science,
University of Gothenburg
alexsclark@gmail.com

Abstract

This paper presents an algorithm for strong learning of probabilistic multiple context free grammars from a positive sample of strings generated by the grammars. The algorithm is shown to be a consistent estimator for a class of well-nested grammars, given by explicit structural conditions on the underlying grammar, and for grammars in this class is guaranteed to converge to a grammar which is isomorphic to the original, not just one that generates the same set of strings.

1 Introduction

A fundamental theoretical goal of linguistics is to characterise a set of possible human languages that is sufficiently large and expressive to describe the attested natural languages while sufficiently restricted that it is learnable given information plausibly available to the child learner in the course of first language acquisition. There is a broad consensus. modulo some concerns about copying (Kobele, 2006), that a class of mildly context-sensitive grammars may be descriptively adequate (Stabler, 2013). Such grammars can generate a rich class of structural descriptions (trees) that serve as latent variables for the syntax/semantics interface, but the acquisition of such richer grammars is harder than the acquisition of simpler formalisms such as context-free grammars (CFGs). Yoshinaka (2009) showed how ideas of distributional learning for CFGs (Clark and Eyraud, 2007) could be extended to these mildly context-sensitive grammar formalism using multiple context free grammars (MCFGs) (Seki et al., 1991). However these algorithms, and their extensions (Yoshinaka, 2010) are only *weak* learning algorithms, that converge only to a grammar that is weakly equivalent to — generates the same set of strings as — the original target grammar. For the purposes of theoretical linguistics this is inadequate (Berwick et al., 2011), as we want a grammar that is strongly equivalent to — generates the same set of trees as — the original grammar. Moreover from a probabilistic perspective weakly equivalent grammars are insufficient (Scicluna and de la Higuera, 2016). Recently Clark and Fijalkow (2020) presented a strong learning algorithm for probabilistic context-free grammars (PCFGs), using only a sample of strings generated by the grammar (Horning, 1969). In this paper we extend that approach to a class of (probabilistic) MCFGs; or rather to several different classes. The approach of Clark and Fijalkow (2020) relies on the existence of what they call, following Stratos et al. (2016), *anchors*, which are terminal symbols that are unambiguously generated by a single nonterminal. On this assumption, the distribution of the terminal will be equal to the distribution of the nonterminal and with some ancillary technical conditions, this suffices to establish identifiability of the class of grammars and thence their learnability.

It is natural to extend this idea to the MCFG case, where some nonterminals generate not strings but tuples of strings; pairs of strings only in this paper. We can therefore define the anchoring condition for nonterminals which generate pairs of strings — of dimension 2 — as saying that there are a pair of terminals a, b which are generated only by a single production from a nonterminal of dimension 2.

However there is a significant technical problem to be overcome: the distribution of this pair may be much larger than the distribution of the nonterminal it anchors since we may have more than one occurrence of each symbol. For example if we have the string *aabb*, then we know that there must be two occurrences of the anchoring production in any derivation of this string but we don't know which pairs of a and b "match". We will state this problem more formally in section 3, but what is necessary is some way of restricting the

distribution of the substring in some way, and we will present a method that works with well-nested MCFGs (Kanazawa et al., 2011), Moreover the additional power of the formalism introduces new sorts of structural indeterminacy of the derivation trees which need to be handled carefully.

This is the first strong learning algorithm for these sorts of grammars and the goal of this paper is not to provide a complete solution to the problem but rather to identify the key difficulties and get an understanding of some of the ways in which these difficulties can be overcome. Nonetheless the class of grammars that can be learned, though descriptively inadequate in several important ways, have some interesting properties which we discuss at the end, and can represent the sorts of cross-serial dependencies that motivated the development of these mildly context-sensitive formalisms (Joshi, 1985).

2 Definitions

We assume some familiarity with MCFGs, or equivalent formalisms, and standard definitions of formal languages, as for example Yoshinaka et al. (2010). We assume a finite alphabet Σ; Σ^* is the set of finite strings of elements of Σ. We use λ for the empty string, since we need ϵ for another purpose. We write concatenation of two strings as uv and occasionally as $u \cdot v$ when we want to emphasize it. We will write $n(u; w)$ for the number of times u occurs in w, where $u, w \in \Sigma^*$, and we write $|w|$ for the length of a string.

We need to deal with various types of string; tuples of strings, like ordered pairs, and strings with gaps in. We will do this by using two additional symbols, $\square$ which represents a gap, and $|$ which represents a boundary. We write $\mathbf{\Sigma}$ for $\Sigma \cup \{\square, |\}$. Define $\mathbb{S}_g^k$ to be the set of strings with $k-1$ boundaries and g gaps.

$$\mathbb{S}_g^k = \{\mathbf{w} \in \mathbf{\Sigma}^* \mid n(\square; w) = g, n(|; w) = k-1\}$$

This represents a k-tuple of strings with g gap symbols occurring somewhere. Notationally we will use $\mathbf{w}$ for elements of $(\Sigma \cup \{\square, |\})^*$ and w for elements of Σ^*. We will occasionally write the boundary symbol as a comma where there is no risk of confusion.

In this paper we consider $k \in \{1, 2\}$ and $g \in \{0, 1, 2\}$. The vast majority of the time we are just going to be using S_0^k and S_k^1 for $k \in \{1, 2\}$. These are k-tuples of strings, and contexts of k-tuples of strings; so things like u,$u|v$, $l\square r$ and $l\square m\square r$.

The gap symbols represent variables, or spaces to be filled and so elements of $\mathbb{S}_g^k$ represent simple functions from $(\Sigma^*)^g \to (\Sigma^*)^k$. In particular a $\mathbb{S}_1^1$ like $l\square r$ represents a function from strings to strings that we might write as $f(x) = lxr$, and an $\mathbb{S}_2^1$ like $l\square m\square r$ represents $f(x, y) = lxmyr$. As functions, $\square$ and $\square|\square$ are the identities on strings and pairs of strings.

More generally and formally we can combine an element of $\mathbb{S}_j^i$ with one of $\mathbb{S}_k^j$ to get one of $\mathbb{S}_k^i$, defined by $\mathbf{u} \odot \mathbf{v}$ in the natural way by replacing all of the gap symbols in u with corresponding tuple elements of v; we lift this to sets in the natural way.

We now define the distribution of a set of k-tuples of strings, $\mathbf{U} \subseteq \mathbb{S}_0^k$, in a language L as

$$\mathbf{U}^{\triangleright} = \{\mathbf{c} \in \mathbb{S}_k^1 \mid \mathbf{c} \odot_k \mathbf{U} \subseteq L\}$$

Conversely for $\mathbf{C} \subseteq \mathbb{S}_k^1$

$$\mathbf{C}^{\triangleleft} = \{\mathbf{u} \in \mathbb{S}_0^k \mid \mathbf{C} \odot_k \mathbf{u} \subseteq L\}$$

For singleton sets $\{w\}$, we will just write $w^{\triangleright}$.

2.1 Grammars

We now define the class of grammars that we use, which are a restricted subclass of MCFGs (Seki et al., 1991) of dimension 2.

Definition 1. *An* MCFG, *a grammar, is a tuple* $\langle \Sigma, V, S, P \rangle$ *where* Σ *is an unranked finite set of terminal symbols,* V *is a finite nonempty ranked set of nonterminal symbols of ranks 1 or 2, which we call the dimension of the nonterminal. We write* $V^{(1)}$ *and* $V^{(2)}$ *for the sets of dimensions 1 and 2 respectively.* $S \in V^{(1)}$ *is a distinguished start symbol and* P *is a set of productions which we define below.*

We only consider non-deleting, non permuting, λ-free productions so we can use a slightly lighter notation than is normal. We have a countably infinite set of variables indexed by a pair of positive integers, $\mathcal{X} = \{x_{i,j} \mid i \in \mathbb{N}_+, j \in \{1, 2\}\}$. The variable $x_{i,j}$ refers to the jth component of the tuple generated by the ith nonterminal on the right hand side of the production. We will abbreviate $x_{1,j}$ as x_j and $x_{2,j}$ as y_j and x_1 as x and y_1 as y. A production, π, is a triple $\langle f, A, \alpha \rangle$ written $A \to f(\alpha)$, where $A \in V, \alpha \in V^*$ and f is a finite string of variables, terminals and boundary symbols. A is the left hand side of the production, α is the right hand side of the production, and f represents a function which constructs a tuple of strings from

a tuple of tuple of strings, via substitution of the variables. If $\alpha = \lambda$ then we omit the brackets and write $A \to f$. In this case f is just a tuple of strings, which represents a leaf node in the derivation. The rank of a production is defined as $\text{rank}(\pi) = |\alpha|$, and $\dim(\pi) = \dim(A)$.

We need to restrict the function f in various ways to make π a well-formed MCFG rule, and further to put it in a restricted normal form. We assume that terminals are introduced by special lexical or bilexical rules. Possible values of f are as follows:

- The rank is 0 and either $\dim(A) = 1$ and $f = a$ for some $a \in \Sigma$, or $\dim(A) = 2$ and $f = a|b$ for some $a, b \in \Sigma$. We call these lexical or bilexical rules, and we can write them as $A \to \mathbf{a}$.
- Alternatively the rank is nonzero, and f contains no terminal symbols. If $\dim(A) = 1$, then f contains only variables and if $\dim(A) = 2$ then it contains one occurrence of the boundary symbol. If there is a boundary symbol then it cannot occur at the beginning or end of the string (the production is λ-free). f contains exactly one occurrence each of variables corresponding to the nonterminals in α; So if $\alpha = B_1 \dots B_k$ then for each i there is exactly one occurrence of the variable $x_{i,j}$ for $1 \leq j \leq \dim(B_i)$, and no other variables. Moreover if $\dim(B_i) = 2$ then $x_{i,1}$ occurs before $x_{i,2}$, (non-permuting) and if $i < j$, then $x_{i,1}$ occurs before $x_{j,1}$.

Given a particular f then, the rank is fixed, as are the dimensions of all nonterminals in the production. We will typically assume that the dimensions of all nonterminals are compatible with f. We say that a production $A \to f(B_1 \dots B_k)$ is non-well-nested if there are two nonterminals on the right hand side B_i, B_j both of dimension 2, such that $f = \dots x_{i,1} \dots x_{j,1} \dots x_{i,2} \dots x_{j,2} \dots$. It is well-nested if it is not non-well-nested. A grammar is well-nested if all of its productions are well-nested.

We will also use a Horn clause notation for productions (Kanazawa, 2009) when we want to discuss particular productions: For example, a CFG production like $A \to BC$ would be written in that notation as $A(xy) \leftarrow B(x)C(y)$, and would be formally the triple $A \to x_{1,1}x_{2,1}(BC)$. If the right hand side is empty then we write the production as $A(a)$ or $A(a, b)$ or $A(\mathbf{b})$. See Clark (2014) for an introduction to this notation and to MCFGs more generally. Other rules are for example the rank 1 production $A(x_1x_2) \leftarrow B(x_1, x_2)$; which takes a dimension 2 production, and concatenates the two components, and $A(x_1y_1, y_2x_2) \leftarrow B(x_1, x_2), C(y_1, y_2)$ which is a well-nested production which combines two dimension 2 nonterminals.

We assume further that S cannot appear on the right hand side of any productions, nor can it appear on the left hand side of any lexical productions. We do allow unary productions with S on the left hand side, $S(x) \to A(x)$; which are not allowed with any other nonterminal on the left hand side. Otherwise the only rank 1 production allowed is of the form $A(x_1x_2) \leftarrow B(x_1, x_2)$.

We define $\mathcal{F}_S$ and $\mathcal{F}$ to be the sets of possible well-nested functions for productions with S on the left hand side, and with some other nonterminal on the left hand side, respectively, that satisfy these conditions. If we restrict the functions to these sets then we define a normal form for the class of well-nested MCFGs of dimension 2; in the sense that for every language generated by a well-nested MCFG of dimension 2, there is a grammar in this class that generates the same language, that uses only productions with functions in these sets, as can be demonstrated using standard techniques (e.g. Kanazawa et al. (2011)), adapted from the standard approaches for converting CFGs into Chomsky normal form. Indeed although we allow in this paper productions of rank greater than 2, restricting productions to those of rank 2 does not change the set of languages generated as these well-nested grammars can be binarised.

2.2 Derivation trees

We will give a tree based semantics for this formalism, since we are interested in learning these trees and defining probability distributions over trees. A derivation tree for a grammar uses the set of productions as a ranked alphabet, where the rank of each production is equal to the number of nonterminals on the right hand side. For each nonterminal A we define an additional symbol $\square_A$ of rank 0. This symbol represents a hole or gap where an A is needed. The *sort* of a tree is the left hand side of the production labeling the root of the tree, or A if the tree is $\square_A$.

These trees must satisfy the condition that if π is a production of rank k, then the sort of the nth child of a node it labels must be equal to the nth nonterminal on the right hand side of π. Obviously

if the production is of rank 0, then the node has no children and the condition is vacuous. Let $\mathbb{T}(G)$ be the set of all these trees. These include degenerate trees which just have a single node labeled with a rank 0 symbol; we won't use a special symbol for these. Given such a tree τ, we can count the number of occurrences of a label in the tree using the notation $n(\pi;\tau)$ or $n(\square_A;\tau)$.

Let $\Omega(G,A)$ be the set of such trees of sort A, and which have no occurrences of any gap symbols. Let $\Xi(G,A)$ be the set of such trees of sort S where there is exactly one occurrence of a gap symbol which is $\square_A$. We can combine an element ξ of $\Xi(G,A)$ with an element τ of $\Omega(G,A)$ to get an element of $\Omega(G,S)$ by replacing the single element of $\square_A$ in ξ with τ. We call the result $\xi \oplus \tau$. We will also lift this in the natural way to sets: for example $\Xi(G,A) \oplus \Omega(G,A)$ is the set of all derivation trees with at least one production of sort A in. Note that since S can only occur on the left hand side of a production, $\Xi(G,S)$ is a set consisting of a single tree with one node labeled $\square_S$.

We can map trees and derivation contexts into strings and string contexts using a string yield function ($\mathbf{sy}$). A tree in $\Omega(G,A)$ has a string yield which is an element of $\mathbb{S}_0^{\dim(A)}$. A derivation context in $\Xi(G,A)$ has a string yield which is in $\mathbb{S}^1_{\dim(A)}$. The function string f represents a function: when we want to evaluate the function we will use the notation $\tilde{f}$; this occurs by substituting the variable $x_{i,j}$ for the jth component of the ith argument and concatenating the results. We can define the string yield function recursively bottom up as follows:

If A is of dimension 1, then $\mathbf{sy}(\square_A) = \square$ and if it is of dimension 2 then $\mathbf{sy}(\square_A) = \square|\square$. Otherwise: if the production is $A \to f(B_1,\dots,B_k)$, then

$$\mathbf{sy}(\pi(\tau_1,\dots,\tau_K)) = \tilde{f}(\mathbf{sy}(\tau_1),\dots,\mathbf{sy}(\tau_K))$$

This also covers the case of a rank 0 production: the string yield of such a production ($A \to f$) is just the constant f, which takes no arguments.

For a set $\mathbf{U} \subseteq \mathbb{S}_0^{\dim(A)}$, we define $\Omega(G,A,\mathbf{U}) = \{\tau \in \Omega(G,A) \mid \mathbf{sy}(\tau) \in \mathbf{U}\}$, and likewise for derivation contexts. Where appropriate we will omit G, and the set brackets so for example, $\Omega(A,w)$ is shorthand for $\{\tau \in \Omega(G,A) \mid \mathbf{sy}(\tau) = w\}$. Given the restrictions on the grammars we have imposed, $\Omega(G,A,w)$ is always finite.

Given this we define

$$\mathcal{L}(G,A) = \{\mathbf{sy}(\tau) \mid \tau \in \Omega(G,A)\}$$

and the set of contexts of a nonterminal as

$$C(G,A) = \{\mathbf{sy}(\xi) \mid \xi \in \Xi(G,A)\}.$$

The language defined by the grammar, $\mathcal{L}(G)$, is then $\mathcal{L}(G,S)$. Note of course that for every context $\mathbf{c} \in C(G,A)$ and every string $\mathbf{w} \in \mathcal{L}(G,A)$, $\mathbf{c} \odot \mathbf{w} \in \mathcal{L}(G)$.

We assume that every production occurs in at least one element of $\Omega(S)$. Under this assumption it's enough to give the list of productions to specify a MCFG, as the nonterminals and terminals can be read off the productions, and the start symbol is the unique nonterminal that occurs only on the left hand side of a production.

2.3 Examples

We give some trivial examples including one with cross-serial dependencies as seen in some natural languages (Huybrechts, 1984; Shieber, 1985)

Example 1. *$V^{(1)} = \{S\}, V^{(2)} = \{A\}$, $\Sigma = \{a, a', b, b'\}$ and we have productions: $\pi_A = A(a,a'), \pi_B = A(b,b')$, and*

$$\pi_S = S(x_1x_2) \leftarrow A(x_1,x_2)$$
$$\pi_X = A(x_1y_1, y_2x_2) \leftarrow A(x_1,x_2), A(y_1,y_2).$$

This defines the language

$$L_1 = \{aa', abbb'b'a', baa'b', \dots\}$$

Note this grammar is ambiguous: $abbb'b'a'$ for example has two parse trees, as shown in fig. 1.

If we change π_X to

$$A(x_1y_1, y_2x_1) \leftarrow A(x_1,x_2), A(y_1,y_2)$$

this gives us a non-context-free language, with a non well-nested MCFG, L_{COPY}, which contains $\{aa', abba'b'b', aba'b', \dots\}$. This is again ambiguous and the two trees above now both have the yield $abba'b'b'$.

Finally we give a well-nested grammar that generates a non context-free language.

Example 2. *$A(a,b), C(c), D(d), E(e), F(f), C(c'), D(d'), E(e'), F(f')$ and*

$$S(x_1x_2) \leftarrow A(x_1,x_2)$$
$$A(x_1y, x_2z) \leftarrow A(x_1,x_2), C(y), E(z)$$
$$A(x_1y, x_2z) \leftarrow A(x_1,x_2), D(y), F(z)$$

Note this grammar is unambiguous, well nested, and generates a non-context-free language, L_{SWISS} related to the Swiss German example of Shieber (1985), which contains strings like $accd'beef$, of the form $aubv$ where u and v must contain matching sequences.

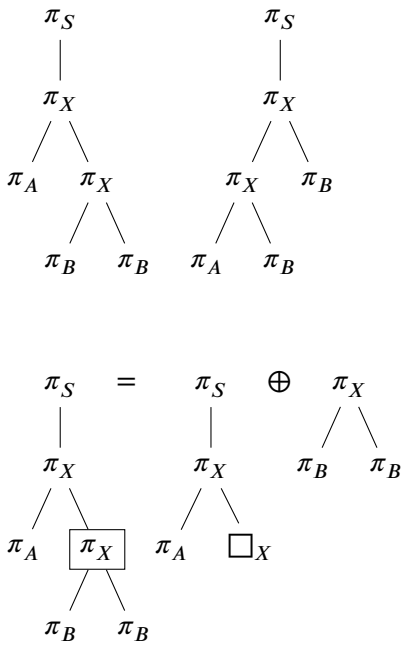


Figure 1: Sample derivation trees from example 1 and a decomposition. On the top we have two derivation trees both of which have yield $abbb'b'a'$. In the bottom we decompose the first of these trees at the boxed node into a context $\xi \in \Xi(G, X)$, where $\mathbf{sy}(\xi) = a\square\square a'$, and a tree $\tau \in \Omega(G, X)$ where $\mathbf{sy}(\tau) = bb|b'b'$. If we combine them we have $\mathbf{sy}(\xi \oplus \tau) = a\square\square a' \odot bb|bb' = abbb'b'a'$.

3 The problem

Clark and Fijalkow (2020) present a learning algorithm for PCFGs using distributional learning based on anchors. In the CFG case an anchor is a terminal which is derived uniquely from a single nonterminal: in the notation of this paper, a is an anchor for $A \in \Sigma^{(1)}$, if $A(a)$ is the unique production involving a in the grammar. In this case of course, the observable distribution of the terminal a is equal to the unobserved distribution of the nonterminal A:

$$a^{\triangleright} = C(G, A)$$

Moreover for any context $\mathbf{c} \in a^{\triangleright}$,

$$\Omega(S, \mathbf{c} \odot a) = \Xi(A, \mathbf{c}) \oplus A(a)$$

These two properties allow one to infer the parameters of productions, indeed the very existence of productions, using distributional analysis.

We can then naturally extend this to define an anchor for a nonterminal of dimension 2, A, as being a pair of distinct terminals (a, a') such that $A(a, a')$ is the only production using a or a'. But now neither of these two properties hold in general. For example, suppose we have the language L_{COPY}. If we let $\mathbf{a} = a, a'$ which is an anchor for A, and consider the string $w = aaa'a'$, then there are not 2 but 4 contexts $\mathbf{c}$ such that $\mathbf{c} \odot \mathbf{a} = w$, namely $\{\square a\square a', \square aa'\square, a\square\square a', a\square a'\square\}$ only two of which are in $C(G, A)$. So $\mathbf{a}^{\triangleright} \supsetneq C(G, A)$. This is because we don't know which pairs of a, a' are being generated in the same production.

We will say that a context $\mathbf{c} \in \mathbf{a}^{\triangleright}$, is an unambiguous context of $\mathbf{a}$, when it is a context of A (i.e. in $C(A)$) and additionally satisfies $\Omega(S, \mathbf{c} \odot \mathbf{a}) = \Xi(A, \mathbf{c}) \oplus A(\mathbf{a})$. For the approach to work, we need some way of restricting the observed contexts of 2-anchors to their unambiguous contexts. Note that all contexts of 1-anchors are unambiguous contexts in this sense.

4 Dyck contexts

Before we describe the key element of our solution, we need to recall the Dyck language (Ginsburg, 1966) and some of its properties. The Dyck language over the two letter alphabet p, q is defined by the unambiguous context-free grammar $S \to \lambda, S \to SpSq$; we write this language as $\mathcal{D}$. We say two occurrences of p and q match in an element of $\mathcal{D}$ if they are derived in the same step. Equivalently we can define the Dyck language as the set of all strings w in $\{p, q\}^*$ such that $n(p; w) = n(q; w)$ and where for every prefix v of w, $n(p; v) \geq n(q; v)$.

Suppose that we have a language L defined by an MCFG where the nonterminal A is anchored by a, a'. Then clearly in every string $w \in L$, we have that $n(a; w) = n(a'; w)$, since every time we introduce an a we also introduce an a'. Moreover, the occurrences of a will always precede the corresponding occurrences of a' since the productions are non permuting, so if u is a prefix of w then $n(a; u) \geq n(a'; u)$. It is therefore clear that in such a language, the occurrences of a, a' will form a subset of the Dyck language if we ignore all the other terminals. Let us formalise this precisely now.

For distinct terminals a, a' define the string homomorphism $h_{a,a'}$ via:

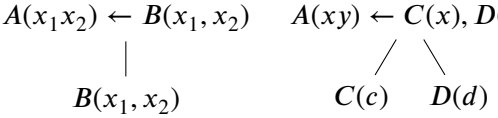
$$h_{a,a'}(b) = \begin{cases} p \text{ if } b = a \\ q \text{ if } b = a' \\ \lambda \text{ otherwise} \end{cases} \tag{1}$$

Definition 2. *A pair of distinct terminals (a, a') is a Dyck pair in a language L if for all $w \in L$, $h_{a,a'}(w) \in \mathcal{D}$.*

Lemma 1. *If A is anchored by a, a' in an* MCFG *defining the language L then a, a' is a Dyck pair.*

Consider L_1: here the nonterminal A is anchored by a, a' which is a Dyck pair; $h_{a,a'}(L_{\text{COPY}}) = \{p^n q^n \mid n \geq 0\} \subset \mathcal{D}$.

The converse is not true: it is not the case that in general the Dyck pairs will correspond to the anchors. There are two issues: first we might have productions like $A(xy) \leftarrow B(x)C(y)$ where B and C only generate b and c respectively, and only occur on the right hand side of this particular rule. A similar situation can occur with any production with two distinct dimension 1 nonterminals on the right hand side. Secondly, we might have the situation where we have two (or more) productions like $A(c, c')$ and $B(c, c')$ and no other productions using c or c'. In this case c, c' will form a Dyck pair but will clearly not be an anchor. This latter case can be handled exactly the way we handle ambiguity for dimension 1 nonterminals, but the former requires some additional assumptions. At its most fundamental the problem is this: suppose we have a language which consists only of the single length two string cd. We can represent this in two obvious ways.



We need to resolve this structural indeterminacy in some way, if we want strong learning to be possible. There are a number of ways to proceed here: here we assume that every nonstart dimension 1 nonterminal must have two anchors. This is unnecessarily strong for this particular case, but it also serves to rule out some other problems later on.

Definition 3. *A grammar G is doubly anchored if for every $A \in V$ there are two terminals a, a' such that if* $\dim(A) = 2$ *then $A(a, a')$ is the only production using either, and if* $\dim(A) = 1$ *then $A(a)$ and $A(a')$ are the only productions using either.*

For instance, example 2 is doubly anchored.

Lemma 2. *Suppose G is a doubly anchored* MCFG. *If a, a' is a Dyck pair then a and a' can occur only in a production of the form $A(a, a')$.*

Proof. Suppose a occurred in a derivation tree τ using a production π of sort B. We can replace a single occurrence of π with π', a anchoring production of B that does not contain a to get a tree τ'. Since the yield of τ' must have matching numbers of a and a', π must be either $B(a, a')$ or $B(a', a)$. Suppose $\pi = B(a', a)$. This can't be the only production for this Dyck pair, or $h_{a,a'}$ would always start with a', so there is some anchor for π. Replace all other occurrences of π with some other anchor for B to obtain a tree τ'. Then $h_{a,a'}(\mathbf{sy}(\tau')) = qp \notin \mathcal{D}$, which is a contradiction. □

Let us suppose now that we have some a, a' which are anchors for a nonterminal A. We can say that an occurrence of a and a' in a string $w = lama'r$ match if there is a derivation of w where they are generated by the same production $A(a, a')$; in other words if their context $l\Box m\Box r \in C(G, A)$. Now we cannot in general determine in such a string the matching pairs as discussed above. But if the grammar is *well-nested* then the matching is always determined, even if the string is ambiguous; indeed they are generated by the same production iff their images match in the Dyck language.

Let $\text{PAIRS}(L)$ be the set of Dyck pairs of the language L and define

$$\mathcal{D}_L = \{\mathbf{w} \in \Sigma^* \mid \forall \mathbf{a} \in \text{PAIRS}(L), h_{\mathbf{a}}(\mathbf{w}) \in \mathcal{D}\}$$

Clearly $L \subseteq \mathcal{D}_L$. We define the following four sets:

$$\begin{aligned}
\mathbb{D}^1_0 &= \mathcal{D}_L \\
\mathbb{D}^2_0 &= \{u|v \in \mathbb{S}^2_0 \mid u \cdot v \in \mathcal{D}_L\} \\
\mathbb{D}^1_1 &= \{l\Box r \in \mathbb{S}^1_1 \mid l \cdot r \in \mathcal{D}_L\} \\
\mathbb{D}^1_2 &= \{l\Box m\Box r \in \mathbb{S}^1_2 \mid m \in \mathcal{D}_L, l \cdot r \in \mathcal{D}_L\}
\end{aligned}$$

Note that by the properties of the Dyck language, $\mathbb{D}^1_2 \odot \mathbb{D}^2_0 \subseteq \mathcal{D}_L$, and $\mathbb{D}^1_1 \odot \mathbb{D}^1_0 \subseteq \mathcal{D}_L$. The crucial observation here is the following lemma, which means that for these grammars it is enough to consider the subsets $\mathbb{D}$ and not the whole sets $\mathbb{S}$.

Lemma 3. *If G is well-nested and double anchored, then for all nonterminals $A \in V^{(k)}$, $\mathcal{L}(G, A) \subseteq \mathbb{D}^k_0$ and $C(G, A) \subseteq \mathbb{D}^1_k$.*

Proof. By induction on $\Omega(G, A)$ in the first case, and in the second case on $\Xi(G, A)$ based on the path from the root of the tree to the gap symbol, $\Box_A$. □

Lemma 4. *If G is well-nested and double anchored, and $\mathbf{a}$ is an anchor for A, then $\mathbf{a}^{\triangleright} \cap \mathbb{D}^1_{\dim(A)} = C(G, A)$, and all of these contexts are unambiguous contexts.*

We now define for $\mathbf{U} \subseteq \mathbb{S}_0^k$

$$\mathbf{U}^{\blacktriangleright} = \mathbf{U}^{\triangleright} \cap \mathbb{D}_k^1$$

and for $\mathbf{C} \subseteq \mathbb{S}_k^1$

$$\mathbf{C}^{\blacktriangleleft} = \mathbf{C}^{\triangleleft} \cap \mathbb{D}_0^k$$

These form a pair of Galois connections between $\mathbb{D}_0^k$ and $\mathbb{D}_k^1$, which obey the usual identities, which we use below. So for example $\mathbf{U}^{\blacktriangleright} = \mathbf{U}^{\blacktriangleright\blacktriangleleft\blacktriangleright}$.

Crucially, these Galois connections also satisfy a fundamental lemma, analogous to the CFG condition that for all sets of strings X, Y $(X \cdot Y)^{\triangleright\triangleleft} = (X^{\triangleright\triangleleft} \cdot Y^{\triangleright\triangleleft})^{\triangleright\triangleleft}$.

Lemma 5. *Suppose f is a well-nested function of rank r, and let L be some language. Then for all $\mathbf{X}_i \in \mathbb{D}_0^{k_i}$ which are sets of tuples of strings of suitable dimension,*

$$f(\mathbf{X}_1, \ldots, \mathbf{X}_r)^{\blacktriangleright\blacktriangleleft} = f(\mathbf{X}_1^{\blacktriangleright\blacktriangleleft}, \ldots, \mathbf{X}_r^{\blacktriangleright\blacktriangleleft})^{\blacktriangleright\blacktriangleleft}$$

Proof. Clearly the left is a subset of the right, since $\mathbf{X}_i \subseteq \mathbf{X}_i^{\blacktriangleright\blacktriangleleft}$. Let d_i be the arity of X_i. Suppose $\mathbf{c} \in f(\mathbf{X}_1, \ldots, \mathbf{X}_r)^{\blacktriangleright}$; if not then it will trivially true.

This means that $\mathbf{c} \odot f(\mathbf{X}_1, \ldots, \mathbf{X}_r) \subseteq L$. So $\mathbf{c} \odot f(\square_{d_1}, \ldots, \mathbf{X}_r) \subseteq \mathbf{X}_1^{\blacktriangleright}$, since f is well-nested. And $\mathbf{X}_1^{\blacktriangleright} = \mathbf{X}_1^{\blacktriangleright\blacktriangleleft\blacktriangleright}$. So $\mathbf{c} \odot f(\square_{d_1}, \ldots, \mathbf{X}_r) \subseteq \mathbf{X}_1^{\blacktriangleright\blacktriangleleft\blacktriangleright}$. So $\mathbf{c} \odot f(\mathbf{X}_1^{\blacktriangleright\blacktriangleleft}, \ldots, \mathbf{X}_r) \subseteq L$. Therefore $\mathbf{c} \in f(\mathbf{X}_1^{\blacktriangleright\blacktriangleleft}, \ldots, \mathbf{X}_r)^{\blacktriangleright}$. Repeating for X_2 etc.:

$$f(\mathbf{X}_1, \ldots, \mathbf{X}_r)^{\blacktriangleright} \subseteq f(\mathbf{X}_1^{\blacktriangleright\blacktriangleleft}, \ldots, \mathbf{X}_r^{\blacktriangleright\blacktriangleleft})^{\blacktriangleright}$$

Therefore

$$f(\mathbf{X}_1, \ldots, \mathbf{X}_r)^{\blacktriangleright\blacktriangleleft} \supseteq f(\mathbf{X}_1^{\blacktriangleright\blacktriangleleft}, \ldots, \mathbf{X}_r^{\blacktriangleright\blacktriangleleft})^{\blacktriangleright\blacktriangleleft}$$

□

This restricted distribution is well-behaved so we can redefine the standard notion of validity to use this. Since S cannot be anchored, and to avoid handling it as a special case, we can stipulate that we have a special terminal symbol s where $s^{\triangleright} = \{\square\}$, which is of course equal to $C(G, S)$, since the start symbol can occur only at the root of a derivation tree.

Definition 4. *Suppose we have a well nested doubly anchored grammar, where $A, B_1, \ldots B_k$ are nonterminals anchored by $\mathbf{a}, \mathbf{b_1}, \ldots \mathbf{b_k}$, respectively. Let $\pi = A \to f(B_1, \ldots, B_k)$ be a production we say that it is valid if $\mathbf{a}^{\blacktriangleright} \subseteq \tilde{f}(\mathbf{b_1}, \ldots, \mathbf{b_k})^{\blacktriangleright}$.*

Note that this is now well-defined (independent of the choice of anchors) by lemma 5. Clearly all productions in the grammar are valid, since $\tilde{f}(\mathbf{b_1}, \ldots, \mathbf{b_k}) \in \mathcal{L}(G, A)$. We can also show something that is approximately the converse.

Lemma 6. *Suppose $G = \langle \Sigma, N, S, P \rangle$ generates the language L; Let $G' = \langle \Sigma, N, S, P' \rangle$ and where $P' \supseteq P$, and all of $\pi \in P'$ are valid with respect to L.*

If $\mathbf{a}$ is an anchor of A, then $\mathcal{L}(G', A) \subseteq \mathbf{a}^{\blacktriangleright\blacktriangleleft}$.

Note that this then implies that $\mathcal{L}(G') = \mathcal{L}(G)$, since $s^{\blacktriangleright\blacktriangleleft} = L$.

Proof. Proof by induction on the derivation trees in $\Omega(G', A)$.

Base case:suppose $\tau = \pi = A(\mathbf{b})$, so $\mathbf{sy}(\tau) = \mathbf{b}$. Then since this production is valid $\mathbf{a}^{\blacktriangleright} \subseteq \mathbf{b}^{\blacktriangleright}$ and so $\mathbf{a}^{\blacktriangleright\blacktriangleleft} \supseteq \mathbf{b}^{\blacktriangleright\blacktriangleleft}$, and since $\mathbf{b} \in \mathbf{b}^{\blacktriangleright\blacktriangleleft}$, $\mathbf{b} \in \mathbf{a}^{\blacktriangleright\blacktriangleleft}$.

Now do inductive step: suppose topmost production is $\pi = A \to f(B_1, \ldots B_k)$ and $\tau = \pi(\tau_1, \ldots, \tau_k)$. By the inductive hypothesis $\mathbf{sy}(\tau_i) \in \mathbf{b}_i^{\blacktriangleright\blacktriangleleft}$. So $\mathbf{sy}(\tau) \in f(\mathbf{b}_1^{\blacktriangleright\blacktriangleleft}, \ldots, \mathbf{b}_k^{\blacktriangleright\blacktriangleleft})$, and so $\mathbf{sy}(\tau) \in f(\mathbf{b}_1^{\blacktriangleright\blacktriangleleft}, \ldots, \mathbf{b}_k^{\blacktriangleright\blacktriangleleft})^{\blacktriangleright\blacktriangleleft}$

By lemma 5, $\mathbf{sy}(\tau) \in f(\mathbf{b}_1, \ldots, \mathbf{b}_k)^{\blacktriangleright\blacktriangleleft}$. So $\mathbf{sy}(\tau)^{\blacktriangleright} \supseteq f(\mathbf{b}_1, \ldots, \mathbf{b}_k)^{\blacktriangleright}$. Validity of π says that $\mathbf{a}^{\blacktriangleright} \subseteq f(\mathbf{b_1}, \ldots, \mathbf{b_k})^{\blacktriangleright}$. Therefore $\mathbf{sy}(\tau)^{\blacktriangleright} \supseteq \mathbf{a}^{\blacktriangleright}$. So $\mathbf{sy}(\tau)^{\blacktriangleright\blacktriangleleft} \subseteq \mathbf{a}^{\blacktriangleright\blacktriangleleft}$, and $\mathbf{sy}(\tau) \in \mathbf{a}^{\blacktriangleright\blacktriangleleft}$. □

5 Other conditions

Beyond the anchoring condition we need two additional conditions which are essentially the same as in Clark (2021a); the first is a bound on the ambiguity, which will allow for identifiability of the parameters.

Definition 5. *A* MCFG *G is locally unambiguous (*LUA*) if for every production π in the grammar of the form $A \to f(B_1 \ldots B_r)$, there is a string w in the language which can be decomposed into a $\mathbf{c} \in \mathbb{D}^1_{\dim(A)}$, and a sequence of tuples $\mathbf{u}_1, \ldots \mathbf{u}_r$, so $\mathbf{c} \odot_k \tilde{f}(\mathbf{u}_1, \ldots \mathbf{u}_r)) = w$, such that:*

$$\Omega(S, w) = \Xi(A, \mathbf{c}) \oplus \pi(\Omega(B_1, \mathbf{u_1}), \ldots, \Omega(B_r, \mathbf{u_r}))$$

Note that this is a generalisation of the definition of an unambiguous context earlier: If $A(\mathbf{a})$ is an anchoring production then the unambiguous contexts are just the ones that satisfy this LUA condition.

The second condition is based on lemma 6; as a result of this lemma, we can see that adding valid productions to a grammar will not increase the set

of strings generated. This means that we can stipulate that the grammar must contain all valid productions. However this may lead to excessive ambiguity, which will then violate the LUA condition. Accordingly we want to eliminate "redundant" productions.

We can combine two productions into another: $\pi_1 = A \to f(\alpha B \gamma)$ and $\pi_2 = B \to g(\beta)$ can be combined to get $\pi_3 = A \to h(\alpha\beta\gamma)$ where h is the suitable function formed from f and g; we will skip the formal definition since it is obvious but fiddly since we need to renumber the variables. Clearly $\text{rank}(\pi_3) = \text{rank}(\pi_2) + \text{rank}(\pi_1) - 1$. So for example $A(x_1x_2) \leftarrow B(x_1, x_2)$ and $B(x_1y, x_2) \leftarrow C(x_1, x_2), D(y)$ can be combined to $A(x_1yx_2) \leftarrow C(x_1, x_2), D(y)$. Intuitively, if we already have the first two, we don't also need the third.

We fix a maximal rank r and define $\mathcal{F}$ and $\mathcal{F}_S$ to be all well-nested productions of rank at most r. We order the productions so that if $\text{rank}(f_1) < \text{rank}(f_2)$ then f_1 precedes f_2, and such that dimension 2 functions precede dimension 1 functions. For the case of $r = 2$, we give the explicit sets:

$$\mathcal{F} = \langle x_1x_2, x|y, x|y_1y_2, xy_1|y_2, x_1|x_2y, x_1|yx_2, x_1x_2|y, x_1y|x_2, x_1|x_2y_1y_2, x_1|y_1y_2x_2, x_1x_2|y_1y_2, x_1y_1|y_2x_2, x_1x_2y_1|y_2, x_1y_1y_2|x_2, xy, x_1x_2y, x_1yx_2, xy_1y_2, x_1x_2y_1y_2, x_1y_1y_2x_2, \rangle$$

and

$$\mathcal{F}_S = \langle x, x_1x_2, xy, xy_1y_2, x_1x_2y, x_1yx_2, x_1x_2y_1y_2, x_1y_1y_2x_2\rangle$$

Note that these are ordered so that if we compose two productions to form a third, the first two will precede the others in these lists. We need to define the notion of a possible production now: given an anchored grammar G with alphabet Σ and nonterminals V, we define $P(G, \mathcal{F}, \mathcal{F}_S)$ to be the union of these sets:

$$\{A(b) \mid A \in V^{(1)} \setminus \{S\}, b \in \Sigma\}$$
$$\{A(b, c) \mid A \in V^{(2)}, b, c \in \Sigma\}$$
$$\{A \to f(B_1, \dots B_k) \mid A, B_i \in V \setminus \{S\}, f \in \mathcal{F}\}$$
$$\{S \to f(B_1, \dots B_k) \mid B_i \in V \setminus \{S\}, f \in \mathcal{F}_S\}$$

where the dimensions of the nonterminals in the last two are restricted to those matching the dimensions of f.

Definition 6. *Given an anchored* MCFG *G, a valid production is redundant wrt $\mathcal{F}, \mathcal{F}_S$ if it is the composition of two other valid productions in $P(G, \mathcal{F}, \mathcal{F}_S)$.*

Definition 7. *(Clark, 2021a) An* MCFG *is complete wrt $\mathcal{F}, \mathcal{F}_S$ if the set of productions is those in $P(G, \mathcal{F}, \mathcal{F}_S)$ that are valid and not redundant.*

This definition then means that the set of productions is determined by the language given the anchored nonterminals. Together these definitions mean that the grammar itself, the nonterminals and the set or productions and of course the terminals are determined by the language, which we prove in the next section.

5.1 Identifiability of Well-Nested Class

We define the class of grammars $\mathfrak{G}_r$ to be all doubly anchored MCFGs where the functions are well nested of rank at most r and which are complete.

Theorem 1. *Suppose G and G' are in $\mathfrak{G}_r$ such that $\mathcal{L}(G) = \mathcal{L}(G')$. Then G is isomorphic to G'.*

Proof. Let $L = \mathcal{L}(G)$. Suppose that $\mathbf{a}$ is a Dyck pair; and $A(\mathbf{a})$ is in the grammar. Let $\mathbf{b}$ be an anchor for A. Then $\mathbf{b}^{\blacktriangleright} \subseteq \mathbf{a}^{\blacktriangleright}$. Therefore consider the set of distributions $\{\mathbf{a}^{\blacktriangleright} \mid \mathbf{a} \text{ is a Dyck pair}\}$. The minimal elements of this ordered by set inclusion will correspond to the dimension 2 nonterminals via $C(G, A) = \mathbf{a}^{\blacktriangleright}$; by completeness we cannot have more than one nonterminal for each of these minimal elements. Consider now the set of bilexical productions: by completeness, these must consist of exactly those productions $A(\mathbf{b})$, where if $\mathbf{a}$ is an anchor for A, $\mathbf{a}^{\blacktriangleright} \subseteq \mathbf{b}^{\blacktriangleright}$. Therefore the set of bilexical productions is defined by L. Consider now the set of all terminals, T, that do not occur in any bilexical productions. Consider the set of distributions $\{a^{\triangleright} \mid a \in T\}$. The minimal elements of this set must correspond to the nonterminals of dimension 1, via the correspondance $a^{\triangleright} = C(A)$. Again the lexical productions must be exactly the set of all valid productions; namely those where $a^{\triangleright} \subseteq b^{\triangleright}$, where a is an anchor for A.

There is therefore a bijection ϕ between the nonterminals of G and G' where $\phi(A) = A'$ if $C(G, A) = C(G', A')$.

It remains to be shown that the set of productions is uniquely determined given these productions. The set of valid productions is clearly determined by the language; the only issue then is to verify that given this set there is a unique set of non-redundant productions. If a production π is formed by the composition of two productions π_1

and π_2 then we can see that functions of π_1 and π_2 will strictly precede the function of π. This ensures uniqueness.

□

6 Weighted MCFGs and Probabilistic MCFGs

The learning paradigm we use is probabilistic so we start by defining the probabilistic grammars that we use. A stochastic language is a probability distribution over Σ^*, wrtten $\mathbb{P}$, and this will be defined by a probabilistic MCFG, (Levy, 2005; Kato et al., 2006; Kallmeyer and Maier, 2013) which we define via a slightly more general formalism called weighted MCFGs. We use two different parameterisations, the top-down which corresponds to the stochastic MCFGs studied by Kato et al. (2006) and the bottom-up introduced in the CFG case by Clark and Fijalkow (2020) which is useful from a learnability perspective. The presentation here follows that in that paper.

Definition 8. *A weighted* MCFG $\langle G; \theta \rangle$ *is a* MCFG *together with a parameter function θ, from productions to positive real numbers.*

$$\theta : P \to \mathbb{R}^+.$$

We define a weight function $w : \mathbb{T}(P) \to \mathbb{R}^+$. For each such tree τ we define:

$$w(\tau) = \prod_{\pi \in P} \theta(\pi)^{n(\pi;\tau)}$$

Note that for $\xi \in \Xi(G, A)$ and $\tau \in \Omega(G, A)$,

$$w(\xi \oplus \tau) = w(\xi) w(\tau)$$

The weight of a set of trees Ω, is $w(\Omega) = \sum_{\tau \in \Omega} w(\tau)$. For a nonterminal A define two quantities which are normalization constants, the sums of the inside and outside probabilities to use the CFG terminology (Eisner, 2016):

$$I(A) = w(\Omega(G, A))$$
$$O(A) = w(\Xi(G, A))$$

The quantity $I(S)$ is called the partition function in the case of PCFGs (Nederhof and Satta, 2008). If $I(S) = 1$ this then defines a probability distribution over $\Omega(G, S)$ and via that a stochastic language whose support is $\mathcal{L}(G)$, by summing over all derivation trees with a given yield as follows:

$$w(u) = \sum_{\tau \in \Omega(G,S) : \mathbf{sy}(\tau) = w} w(\tau) = w(\Omega(G, S, \{w\})) \tag{2}$$

Given such a weighted MCFG, with $I(S) = 1$, for every production π, we can define $\mathbb{E}(\pi)$ the expected number of times that the production π occurs in a tree. This is

$$\mathbb{E}(\pi) = \sum_{\tau \in \Omega(G)} w(\tau) n(\pi; \tau) \tag{3}$$

We can define the expectation of a nonterminal as the sum of expectations of all productions with that nonterminal on the left hand side.

$$\mathbb{E}(A) = \sum_{A \to f(\alpha) \in P} \mathbb{E}(A \to f(\alpha)) \tag{4}$$

We assume throughout that the expected length of strings under these distributions, $\sum_{a \in \Sigma} \mathbb{E}(a)$, is finite, which implies that all these expectations are finite too. We can see that for all nonterminals A,

$$I(A) O(A) = \mathbb{E}(A)$$

Note that $\mathbb{E}(S) = 1$ under these assumptions, as it occurs exactly once in every tree in $\Omega(G, S)$, at the root; indeed clearly $I(S) = O(S) = 1$, since $\Xi(G, A)$ has one element, $\square_S$, which has weight 1.

There is an indeterminacy in these parameters in that one can always pick some nonterminal that isn't S, say A, and scale the parameters of productions with $A \to f(B_1, \ldots, B_r)$ on the left hand side by some $\alpha > 0$, and scale all productions with A on the right hand side $B \to f(C_1, \ldots, C_r)$ by α^{-n} where n is the number of times A occurs on the right hand side, and the distribution over trees will remain unchanged. There are two natural ways of resolving this, two parameterisations: one where we stipulate that for all A, $O(A) = 1$ (and as a result $I(A) = \mathbb{E}(A)$) and one where we stipulate that for all A, $I(A) = 1$, and therefore $O(A) = \mathbb{E}(A)$. The latter gives us the regular probabilistic top down generative process. The former gives us the bottom up parameterisation, which is crucial to the success of these primal distributional learning algorithms.

Note that for any production $\pi = A \to f(B_1, \ldots, B_r)$

$$\mathbb{E}(\pi) = O(A) \theta(\pi) \prod_i I(B_i)$$

Therefore in the bottom up parameterization, where $O(A) = 1$ and $I(A) = \mathbb{E}(A)$ this gives us

$$\mathbb{E}(\pi) = \theta(\pi) \prod_i \mathbb{E}(B_i)$$

and so

$$\theta(\pi) = \frac{\mathbb{E}(\pi)}{\prod_i \mathbb{E}(B_i)} \tag{5}$$

6.1 Anchored grammars with bottom-up parameters

Given a stochastic language $\mathbb{P}$, for a terminal a we define $\mathbb{E}(a) = \sum_{\mathbf{c} \in a^{\triangleright}} \mathbb{P}(\mathbf{c} \odot a)$. This is the expected number of times the terminal a will appear in a random string. Now suppose that $\mathbf{a} = a, a'$ is a Dyck pair; clearly $\mathbb{E}(a) = \mathbb{E}(a')$ so we define $\mathbb{E}(\mathbf{a}) = \mathbb{E}(a)$.

Then if $\mathbf{a}$ is an anchor for A then in the bottom up parameterisation

$$\theta(A(\mathbf{a})) = \mathbb{E}(\mathbf{a})$$

Suppose a context $\mathbf{c} \in \mathbf{a}^{\blacktriangleright}$. Then

$$\Omega(G, S, \mathbf{c} \odot \mathbf{a}) = \Xi(G, A, \mathbf{c}) \oplus A(\mathbf{a})$$

So

$$\mathbb{P}(\mathbf{c} \odot \mathbf{a}) = w(\Xi(G, A, \mathbf{c}))\theta(A(\mathbf{a}))$$

Therefore

$$w(\Xi(G, A, \mathbf{c})) = \frac{\mathbb{P}(\mathbf{c} \odot \mathbf{a})}{\mathbb{E}(\mathbf{a})}$$

Consider some other tuple of terminals $\mathbf{b}$; not necessarily an anchor, and a context $\mathbf{c} \in \mathbf{a}^{\blacktriangleright}$. Then

$$\Omega(G, S, \mathbf{c} \odot \mathbf{b}) \supseteq \Xi(G, A, \mathbf{c}) \oplus A(\mathbf{b})$$

Therefore:

$$\mathbb{P}(\mathbf{c} \odot \mathbf{b}) \geq w(\Xi(G, A, \mathbf{c}))\theta(A(\mathbf{b}))$$

So:

$$\theta(A(\mathbf{b})) \leq \frac{\mathbb{P}(\mathbf{c} \odot \mathbf{b})\mathbb{E}(\mathbf{a})}{\mathbb{P}(\mathbf{c} \odot \mathbf{a})}$$

Since this is true for all contexts $\mathbf{c} \in \mathbf{a}^{\blacktriangleright}$ we have:

$$\theta(A(\mathbf{b})) \leq \inf_{\mathbf{c} \in \mathbf{a}^{\blacktriangleright}} \frac{\mathbb{P}(\mathbf{c} \odot \mathbf{b})\mathbb{E}(\mathbf{a})}{\mathbb{P}(\mathbf{c} \odot \mathbf{a})} \tag{6}$$

More generally, suppose that $\pi = A \to f(B_1, \ldots, B_r)$ is some production with A, B_i anchored by $\mathbf{a}, \mathbf{b_i}$. Then again for any unambiguous context $\mathbf{c}$, writing $u = \mathbf{c} \odot f(\mathbf{b_1}, \ldots, \mathbf{b_r})$

$$\mathbb{P}(u) \geq w(\Xi(G, A, \mathbf{c}))\theta(\pi) \prod_i \theta(B_i(\mathbf{b_i}))$$

Rearranging and minimizing with respect to $\mathbf{c}$:

$$\theta(\pi) \leq \inf_{\mathbf{c} \in \mathbf{a}^{\blacktriangleright}} \frac{\mathbb{P}(\mathbf{c} \odot f(\mathbf{b_1}, \ldots, \mathbf{b_r}))\mathbb{E}(\mathbf{a})}{\mathbb{P}(\mathbf{c} \odot \mathbf{a}) \prod_i \mathbb{E}(\mathbf{b_i}))} \tag{7}$$

Importantly the right hand sides of eqs. (6) and (7) do not depend on the grammar, but only on the distribution over strings. If the grammar is LUA, then in these two equations the minimum will be attained, and we will have an equality.[1]

[1] Proving this is a little involved as we need to verify that the context is still LUA when all of the subtrees are just anchoring productions, which requires verifying some properties of $(\cdot)^{\blacktriangleright}$.

7 Algorithm

Given the identifiability of the class $\mathfrak{G}_r$ from strings, and the parameter identities above it is straightforward to design an algorithm which will learn this class from a sample of strings generated by the target grammar; we define the convergence criterion below in theorem 2.

The algorithm takes as input a sample of strings, and hyperparameters $\mathcal{F}, \mathcal{F}_S$; these implicitly define an upper bound r on the rank of the productions. It outputs a weighted MCFG, which can then be converted into a probabilistic MCFG (Clark and Fijalkow, 2020). The grammar will uses tuples of terminal symbols, i.e. anchors, directly as nonterminals — with this grammar format there is no need to distinguish the terminal and nonterminal symbols; but it's important to remember that we use $\tilde{f}$ to denote the application of a function f. So we might have a production $a \to f(b, c)$ where $f = xy$, and $a, b, c \in \Sigma$ are anchors that represent nonterminals. $\tilde{f}(b, c)$ is then the application of that concatenation function to b, c taken as terminals, namely the string bc. The start nonterminal will be a special distinguished symbol s as discussed.

7.1 Estimators

We start by defining some naive plug in estimators (Clark and Fijalkow, 2020). Assume we have a sample of N strings $w_1, \ldots, w_N$. Let $\hat{\Sigma}$ be the observed terminal symbols. For a string $w \in \hat{\Sigma}^*$, let $N(w)$ be the number of times $w = w_i$, and $\hat{\mathbb{P}}(w) = N(w)/N$. For $a \in \hat{\Sigma}$ define

$$\hat{\mathbb{E}}(a) = \frac{1}{N} \sum_i n(a; w_i)$$

If $\mathbf{a} = a_1, a_2$

$$\hat{\mathbb{E}}(\mathbf{a}) = \hat{\mathbb{E}}(a_1)$$

Lemma 7. *All of these estimators are consistent;* $\hat{\mathbb{P}}(w)$ *is a consistent estimator for* $\mathbb{P}(w)$ *and* $\hat{\mathbb{E}}(a)$ *is a consistent estimator for* $\mathbb{E}(a)$*, and if* $\mathbf{a}$ *is a Dyck pair, then* $\hat{\mathbb{E}}(\mathbf{a})$ *is a consistent estimator for* $\mathbb{E}(\mathbf{a})$.

For a k-tuple $\mathbf{a}$ define the frequent contexts to be

$$F(\mathbf{a}) = \{\mathbf{c} \in \mathbf{a}^{\blacktriangleright} : N(\mathbf{c} \odot \mathbf{a}) > \sqrt{N}\}$$

For a suitable linear function f, and tuples of appropriate arity, $\mathbf{a}, \mathbf{b}_1, \ldots, \mathbf{b}_r$:

$$\hat{\theta}(\mathbf{a} \to f(\mathbf{b_1}, \ldots, \mathbf{b_r})) = \frac{\hat{\mathbb{E}}(\mathbf{a})}{\prod_{i=1}^{r} \hat{\mathbb{E}}(\mathbf{b_i})} \min_{\mathbf{c} \in F(\mathbf{a})} \frac{\hat{\mathbb{P}}(\mathbf{c} \odot \tilde{f}(\mathbf{b_1}, \ldots, \mathbf{b_r}))}{\hat{\mathbb{P}}(\mathbf{c} \odot \mathbf{a})} \tag{8}$$

For tuples of the same dimension $\mathbf{a}, \mathbf{b}$, we will write the estimate for the rank 0 production as a $\hat{\theta}(\mathbf{a} \to \mathbf{b})$.

Lemma 8. *Suppose the strings are generated by a weighted* MCFG, *$G; \theta$ where grammar $G \in \mathfrak{G}_r$; given a production $\pi_* = A \to f(B_1, \ldots, B_r)$, let π be the result of replacing all nonterminals with their anchors.*

If π is not valid, then for any $\delta > 0$, there is an N such that with probability greater than $1 - \delta$, $\hat{\theta}(\pi) = 0$.

If $\pi_* \in P$, then for any $\delta > 0, \epsilon > 0$, there is an N such that with probability greater than $1 - \delta$, $|\hat{\theta}(\pi) - \theta(\pi_*)| < \epsilon$.

Proof. Since it is not valid there is some $\mathbf{c} \in \mathbf{a}^{\blacktriangleright}$, such that $\mathbf{c} \odot \tilde{f}(\mathbf{b_1}, \ldots, \mathbf{b_k}) \notin \mathcal{L}(G)$. Let N be sufficiently large that $\mathbf{c} \in F(\mathbf{a})$ with probability at least $1 - \delta$, and the result follows. □

7.2 Algorithm

Given these estimators we define the following algorithm.

- Let Σ be the observed alphabet and let s be an additional symbol. $P \leftarrow \emptyset, B \leftarrow \emptyset$
- We first compute the set of possible Dyck pairs $D \subseteq \Sigma \times \Sigma$.
- We use Algorithm FindExtremal with D as input to identify a set of 2-anchors, V_2.
- For every pair $\mathbf{b} \in \Sigma \times \Sigma$, and every $\mathbf{a} = (a_1, a_2) \in V_2$: if $\hat{\theta}(\mathbf{a} \to \mathbf{b}) > 0$, then $P \leftarrow P \cup \{\mathbf{a}(\mathbf{b})\}$ with parameter $\hat{\theta}(\mathbf{a} \to \mathbf{b})$ and $B \leftarrow B \cup \{a_1, a_2\}$.
- We use Algorithm FindExtremal with $\Sigma \setminus B$ as input, and initial symbol s to identify a set of 1-anchors, V_1.
- For every $b \in \Sigma$ and $a \in V_1 \setminus \{s\}$, if $\hat{\theta}(a \to b) > 0$, $P \leftarrow P \cup \{a(b)\}$ with parameter $\hat{\theta}(a \to b)$.
- For every $f \in \mathcal{F}$, if r is the rank of f then for every $A, B_1, \ldots B_r$ in V_1 or V_2 of appropriate dimension, let $\pi = \mathbf{a} \to f(\mathbf{b_1}, \ldots, \mathbf{b_r})$. If π is not redundant wrt P then let $\alpha = \hat{\theta}(\pi)$, If $\alpha > 0$ then add this production to P with parameter α,
- Do the same for every $f \in \mathcal{F}_S$ where A is restricted to be s.
- Construct a weighted MCFG with nonterminals of dimension 2, V_2, of dimension 1, $V_1 \cup \{s\}$, start symbol s, set of productions P, with these associated parameters.
- Convert the weighted MCFG to a probabilistic MCFG via one iteration of the expectation-maximisation algorithm.

This algorithm is polynomial in the size of the input data. The only computationally nontrivial part of this is the final conversion step, which is of the same complexity as parsing (Eisner, 2016). In the case of well-nested MCFGs, the grammars can be binarised and complexity is $\mathcal{O}(|w_i|^6)$ (Gómez-Rodríguez et al., 2010). This step can be omitted if all that is needed is the conditional distribution of trees given strings.

Input: Strings $w_1, w_2, \ldots, w_N$, and k-tuples A, optional initial symbol s
Output: A set of k-anchors $A_k \subseteq A$
$A_k \leftarrow \emptyset$ or $\{s\}$;
for $\mathbf{a} \in A$ **do**
  if $\forall \mathbf{b} \in A, \hat{\theta}(\mathbf{a} \to \mathbf{b}) > 0$ *or* $\hat{\theta}(\mathbf{b} \to \mathbf{a}) = 0$ **then**
    if $\forall \mathbf{b} \in A_k, \hat{\theta}(\mathbf{b} \to \mathbf{a}) = 0$ **then**
      $A_k \leftarrow A_k \cup \{\mathbf{a}\}$;
    end
  end
end
return A_k

Algorithm FindExtremal: Find extremal elements of a set of tuples, of dimension 1 or 2.

8 Correctness

We will now prove the *correctness* of the algorithm for a certain class of probabilistic grammars that we will now define as $\mathfrak{P}_r$. For clarity we will gather together the various conditions that we require in one place.

- First, we have the normal form restrictions defined in section 2.1. These restrict the languages generated to those generated by well-nested MCFGs of dimension 2, a well studied class, except for the empty string which is not generated by any of these grammars.
- Double anchoring (definition 3): this requires that for each nonterminal we have some representative structures that are generated only by those nonterminals.

- Local unambiguity (definition 5): a weak condition on the ambiguity of the grammar.
- Completeness: the requirement that all productions that are valid either are in the grammar or can be derived from other productions: definition 7

We define the class $\mathfrak{P}_r$ to be the set of probabilistic grammars $\langle G; \theta \rangle$ where G satisfies the conditions above, and where θ is a top down parameterisation of finite expected length.[2] We claim that the algorithm is a consistent estimator for $\mathfrak{P}_r$, as stated in the following theorem.

Theorem 2. *For any $\langle G_*; \theta_* \rangle \in \mathfrak{P}_r$ for any $\epsilon > 0, \delta > 0$, there is an N such that if the grammar is provided with at least N samples then it outputs a probabilistic* MCFG *$\langle \hat{G}; \hat{\theta} \rangle$ such that, with probability at least $1 - \delta$, $\hat{G}$ is isomorphic to G_* and for all productions π in the original grammar $|\theta_*(\pi) - \hat{\theta}(\hat{\pi})| < \epsilon$, where $\hat{\pi}$ is the production in $\hat{G}$ isomorphic to π.*

Proof. (Sketch)

The backbone of the proof is Theorem 1; here we just need to show that we can derive this probabilistically. The outline of the proof is that we make a series of assumptions that will hold probabilistically. We can then bound the probability of an error in each case by some fraction of δ; using a union bound we can then bound the probability of any of them being false. Let D be the set of observed strings.

- We assume every element of the original alphabet occurs in some string in D.
- We assume that the Dyck pairs of the language are equal to the Dyck pairs of D.
- We assume that for every two Dyck pairs $\mathbf{a}, \mathbf{b}$ $\mathbf{a}^\blacktriangleright \subseteq \mathbf{b}^\blacktriangleright$ iff $\hat{\theta}(\mathbf{a} \to \mathbf{b}) > 0$.
- We assume that if a is a 1-anchor and b is any other terminal, then $a^\blacktriangleright \subseteq b^\blacktriangleright$ iff $\hat{\theta}(a \to b) > 0$, and $b^\blacktriangleright \subseteq a^\blacktriangleright$ iff $\hat{\theta}(b \to a) > 0$.

Under these assumptions, the nonterminals in the hypothesis grammar will correspond to the original nonterminals where $\mathbf{a}$ corresponds to A iff $\mathbf{a}^\blacktriangleright = C(G_*, A)$.

- We assume that for every possible production π; π is valid iff $\hat{\theta}(\pi) > 0$.
- We assume that for every possible production π that does correspond to a production π_* in the original grammar: $|\hat{\theta}(\pi) - \theta(\pi_*)| \leq \epsilon'$.

Under these assumptions, we will construct only the productions that are valid and non redundant, and the convergence of the parameters to the bottom-up parameters follows. We can then show that the conversion to the probabilistic MCFG gives bounded errors using some elementary analysis. □

The conditions stated are merely sufficient conditions for the learnable class: the actual class learned is larger. For example the finite language $\{a\}$ is not problematic, but violates the requirement that each dimension 1 nonterminal has two anchors.

9 Discussion

First, note that the strings themselves are enough to infer the "movement" structure but only when there is a fairly explicit clue in at least some of the strings that exhibit this structure. Secondly, well-nestedness (Kuhlmann and Möhl, 2007; Kanazawa et al., 2011) seems to make things a lot simpler; this tends to support Kanazawa et al. (2011)'s arguments for well-nestedness as a condition on mildly context-sensitive language formalisms to go with the parsing efficiency arguments (Gómez-Rodríguez et al., 2010) and the corpus based arguments of, among others, Kuhlmann and Nivre (2006). Note that for example MIX (Kanazawa and Salvati, 2012) is not in the class of languages defined; the strong learning classes are much more restricted in linguistically significant ways that the weak learning algorithms considered in Hao (2019).

Finally, we don't use the additional MCFG machinery only in those cases where it is strictly required. Rather some context-free languages, and indeed even some finite languages will have nonterminals of dimension 2; in this approach, all long distance dependencies must be handled by the mildly context-sensitive part even if they could be handled by some regular or CFG component.

The plugin estimators are computationally trivial but very slow to converge: but this can be improved using standard NLP techniques as is shown by Clark and Fijalkow (2020). The algorithm is relativised to a set of admissible production types $\mathcal{F}$. It is not the case that increasing the set of types will strictly increase the set of grammars learned: typically there will be some languages/grammars that

[2] For all nonterminals A, $I(A) = 1$ and $O(A)$ is finite.

will no longer be complete in the larger class. This approach then defines a family of different learnable classes.

The approach is quite close to the (weak) primal distributional learning approach for MCFGs developed in Yoshinaka (2010).

Language Acquisition The learning model here is highly idealised: the learner only has access, passively, to a sample of strings generated from the grammar. The child learner of course normally has access to much additional information derived from the situational context via other modalities, and additionally is an active participant, being able to interact and produce utterances of their own. The model is therefore highly pessimistic in the assumptions about the information available to the child; the positive result we are able to obtain in this model is thus correspondingly strengthened. Nonetheless, the classic term, the *primary linguistic data* correctly highlights the importance of this data in language acquisition since it is the only data source that is essential in the sense that when it is unavailable, language acquisition fails, see Clark and Lappin (2011) for discussion. At least in the early phases of language acquisition, this seems a reasonable assumption. However when we consider the acquisition of mildly context-sensitive grammars as we do here, we are considering fairly subtle properties of adult syntax which may only emerge later.

The anchoring condition is clearly unrealistic: natural languages to a certain extent have one dimensional anchors but they simply don't have the two dimensional anchors at least if we consider terminal symbols to be undisambiguated sequences of acoustic features. But the model we use here does not rely on this assumption about the terminal words. In parameter setting models of language acquisition (Yang and Roeper, 2011) it is common to assume that the input is a sequence of more abstract grammatical categories. If we consider the input data then to be sequences of some sort of shallow chunks, or non recursive phrases, then the double anchoring assumption starts to seem more realistic. In summary, from a modeling perspective it is a bit naive to attempt to model all of syntax learning using only the strings as input, even if it is cleaner from the mathematical perspective we adopt in this paper: some additional information derived from semantic constraints is certainly active in language acquisition.

Future work That said, there are no in principle reasons why this approach couldn't be extended to grammars that are only partially or approximately anchored. It's more realistic to remove productions like $A(a, b)$ entirely, and introduce lexical items only via dimension 1 nonterminals.

However a lot of the problems are because MCFGs themselves are inadequate representationally: we need a formalism with a little more structure. Alternatively one can learn some simpler constituent structure, and then subsequently learn a richer movement structure (Rogers, 2003) on top of that as in Clark (2021b). Again this seems to require the restriction to the well-nested class.

Unary rules like $A(x) \leftarrow B(x)$ and the 2-dimensional analogue, seem straightforward to introduce but will add some algorithmic complexity as in Clark (2021a). Moving to well-nested MCFGs of higher dimension seems straightforward, though the corresponding anchoring assumption is still less plausible: the corresponding generalisation of a Dyck language is, in the dimension 3 case that given by $S \rightarrow SpSqSr, \rightarrow \lambda$. In contrast the extension to non well-nested MCFGs has many technical difficulties; it is not clear whether this approach can be extended beyond the well-nested class. Moving from dimension 2 to dimension 3 seems particularly tricky then because the appropriate generalisation of the Dyck language, $D3$, is poorly understood (Moortgat, 2014).

Conclusion This paper has presented a family of algorithms for strong learning of a representative mildly context-sensitive grammar formalism, using only strings as input, with good theoretical guarantees. A realistic learning model, a representationally adequate grammar formalism, and a sufficiently strong convergence criterion: these are all good, but the class of grammars is still too small, and the grammars are too big, as the MCFG formalism itself is not succinct enough.

Acknowledgments

I am grateful to the reviewers for their helpful comments.

References

Robert C. Berwick, Paul Pietroski, Beracah Yankama, and Noam Chomsky. 2011. Poverty of the stimulus revisited. *Cognitive Science*, 35:1207–1242.

Alexander Clark. 2014. An introduction to multiple context free grammars for linguists. Unpublished manuscript.

Alexander Clark. 2021a. Beyond Chomsky normal form: Extending strong learning algorithms for PCFGs. In *Proceedings of the Fifteenth International Conference on Grammatical Inference*, volume 153 of *Proceedings of Machine Learning Research*, pages 4–17. PMLR.

Alexander Clark. 2021b. Strong learning of probabilistic tree adjoining grammars. *Proceedings of the Society for Computation in Linguistics*, 4(48).

Alexander Clark and Rémi Eyraud. 2007. Polynomial identification in the limit of substitutable context-free languages. *Journal of Machine Learning Research*, 8:1725–1745.

Alexander Clark and Nathanaël Fijalkow. 2020. Consistent unsupervised estimators for anchored PCFGs. *Transactions of the Association for Computational Linguistics*, 8:409–422.

Alexander Clark and Shalom Lappin. 2011. *Linguistic Nativism and the Poverty of the Stimulus*. Wiley-Blackwell, Malden, MA.

Jason Eisner. 2016. Inside-outside and forward-backward algorithms are just backprop (tutorial paper). In *Proceedings of the Workshop on Structured Prediction for NLP*, pages 1–17.

Seymour Ginsburg. 1966. *The Mathematical Theory of Context-Free Languages*. McGraw-Hill, Inc., New York, NY, USA.

Carlos Gómez-Rodríguez, Marco Kuhlmann, and Giorgio Satta. 2010. Efficient parsing of well-nested linear context-free rewriting systems. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 276–284.

Yiding Hao. 2019. Learnability and overgeneration in computational syntax. *Proceedings of the Society for Computation in Linguistics*, 2(1):124–134.

James Jay Horning. 1969. *A study of grammatical inference*. Ph.D. thesis, Computer Science Department, Stanford University.

Riny A .C. Huybrechts. 1984. The weak inadequacy of context-free phrase structure grammars. In G. de Haan, M. Trommelen, and W. Zonneveld, editors, *Van Periferie naar Kern*. Foris, Dordrecht, Holland.

Aravind K. Joshi. 1985. Tree adjoining grammars: How much context-sensitivity is required to provide reasonable structural descriptions? In David R. Dowty, Lauri Karttunen, and Arnold M. Zwicky, editors, *Natural Language Parsing: Psychological, Computational, and Theoretical Perspectives*, Studies in Natural Language Processing, page 206–250. Cambridge University Press.

Laura Kallmeyer and Wolfgang Maier. 2013. Data-driven parsing using probabilistic linear context-free rewriting systems. *Computational Linguistics*, 39(1):87–119.

Makoto Kanazawa. 2009. The pumping lemma for well-nested multiple context-free languages. In *Developments in Language Theory*, pages 312–325. Springer.

Makoto Kanazawa, Jens Michaelis, Sylvain Salvati, and Ryo Yoshinaka. 2011. Well-nestedness properly subsumes strict derivational minimalism. In Sylvain Pogodalla and Jean-Philippe Prost, editors, *Logical Aspects of Computational Linguistics*, volume 6736 of *Lecture Notes in Computer Science*, pages 112–128. Springer Berlin Heidelberg.

Makoto Kanazawa and Sylvain Salvati. 2012. MIX is not a tree-adjoining language. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 666–674. Association for Computational Linguistics.

Yuki Kato, Hiroyuki Seki, and Tadao Kasami. 2006. Stochastic multiple context-free grammar for RNA pseudoknot modeling. In *Proceedings of the Eighth International Workshop on Tree Adjoining Grammar and Related Formalisms*, pages 57–64.

Gregory M. Kobele. 2006. *Generating Copies: An investigation into structural identity in language and grammar*. Ph.D. thesis, University of California Los Angeles.

Marco Kuhlmann and Mathias Möhl. 2007. Mildly context-sensitive dependency languages. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 160–167.

Marco Kuhlmann and Joakim Nivre. 2006. Mildly non-projective dependency structures. In *Proceedings of the COLING/ACL 2006 main conference poster sessions*, pages 507–514.

Roger Levy. 2005. *Probabilistic models of word order and syntactic discontinuity*. Ph.D. thesis, Stanford university.

Michael Moortgat. 2014. A note on multidimensional Dyck languages. In *Categories and Types in Logic, Language, and Physics*, pages 279–296. Springer.

Mark-Jan Nederhof and Giorgio Satta. 2008. Computing partition functions of PCFGs. *Research on Language and Computation*, 6(2):139–162.

James Rogers. 2003. Syntactic structures as multi-dimensional trees. *Research on Language and Computation*, 1(3-4):265–305.

James Scicluna and Colin de la Higuera. 2016. Grammatical inference of PCFGs applied to language modelling and unsupervised parsing. *Fundamenta Informaticae*, 146(4):379–402.

Hiroyuki Seki, Takashi Matsumura, Mamoru Fujii, and Tadao Kasami. 1991. On multiple context-free grammars. *Theoretical Computer Science*, 88(2):229.

Stuart M. Shieber. 1985. Evidence against the context-freeness of natural language. *Linguistics and Philosophy*, 8:333–343.

Edward P Stabler. 2013. The epicenter of linguistic behavior. In Montserrat Sanz, Itziar Laka, and Michael K. Tanenhaus, editors, *Language Down the Garden Path: The Cognitive and Biological Basis of Linguistic Structures*, pages 316–323. Oxford University Press.

Karl Stratos, Michael Collins, and Daniel Hsu. 2016. Unsupervised part-of-speech tagging with anchor hidden markov models. *Transactions of the Association for Computational Linguistics*, 4:245–257.

Charles Yang and Tom Roeper. 2011. Minimalism and language acquisition. In Cedric Boeckx, editor, *Oxford Handbook of Linguistic Minimalism*, chapter 24, pages 551–573. Oxford University Press, Oxford.

Ryo Yoshinaka. 2009. Learning mildly context-sensitive languages with multidimensional substitutability from positive data. In *International Conference on Algorithmic Learning Theory*, volume 5809 of *Lecture Notes in Computer Science*, pages 278–292. Springer.

Ryo Yoshinaka. 2010. Polynomial-time identification of multiple context-free languages from positive data and membership queries. In *Proceedings of the International Colloquium on Grammatical Inference*, pages 230–244.

Ryo Yoshinaka, Yuichi Kaji, and Hiroyuki Seki. 2010. Chomsky-Schützenberger-type characterization of multiple context-free languages. In *International Conference on Language and Automata Theory and Applications*, pages 596–607.