Distributional Learning of Context-Free and Multiple Context-Free Grammars

Alexander Clark and Ryo Yoshinaka

Abstract This chapter reviews recent progress in distributional learning in grammatical inference as applied to learning context-free and multiple context-free grammars. We discuss the basic principles of distributional learning, and present two classes of representations, primal and dual, where primal approaches use nonterminals based on strings or sets of strings and dual approaches use nonterminals based on contexts or sets of contexts. We then present learning algorithms based on these two models using a variety of learning paradigms, and then discuss the natural extension to mildly context-sensitive formalisms, using multiple context-free grammars as a representative formalism.

1 Introduction

In this chapter we look at the problem of learning certain classes of phrase structure grammars from information about the language; a classic problem in grammatical inference. In particular we look at techniques using what is broadly called distributional learning, and which have been developed in recent years starting with Clark & Eyraud (2007) [16]. The term *distributional* as we use it has nothing to do with probability distributions or statistical learning, but rather concerns the linguistic notion of distribution: the set of contexts or environments in which strings or words can appear.

This is indeed a classic problem: Gold [23] suggests the following question as worthy of research:

Alexander Clark
Department of Philosophy, King's College London, e-mail: alexander.clark@kcl.ac.uk

Ryo Yoshinaka
Kyoto University e-mail: ry@i.kyoto-u.ac.jp

> However, it would be useful to determine if there are interesting subclasses of context-free languages which can be identified in the limit by either of these approaches (i.e. by statistical approaches like distributional analysis or by approaches sensitive to order).

Distributional learning itself has a long history, which we review briefly in Section 2 to provide some intellectual context. In this chapter we present a tutorial overview of modern approaches to distributional learning as applied to the inference of context-free grammars (CFGs) and multiple context-free grammars (MCFGs) which we take to be a representative mildly context-sensitive formalism. We will focus on the general properties of these algorithms; the representational ideas and the types of algorithms that exploit these representational assumptions. We will try to provide a full bibliography for at least the current resurgence of interest in distributional learning, the key earlier papers, together with pointers into the literature for the rest. We are not trying to provide a complete survey of the inference of CFGs and MCFGs; we restrict ourselves to algorithms that are computationally efficient in some sense (so excluding purely enumerative algorithms [54]), have some theoretical guarantees (as opposed to heuristic algorithms [39, 34]), and take as input only strings, or information about strings (as opposed to algorithms that take trees or partially bracketed strings as input [47]). Within these parameters, the only algorithms that we are aware of are distributional algorithms in the sense that we define below.

2 Distributional Learning: a historical note

Distributional learning has a long history. Beyond the well-known work of the American structuralists, most famously Zellig Harris [24] and Rulon Wells [60], structuralist linguistics had an autonomous history in Russia and Eastern Europe under the name of the Kulagina school, which has its origins in a seminar in mathematical linguistics initiated by Kolmogorov and which takes its name from Olga Kulagina's seminal 1957 paper [37]. While in the USA, structuralist linguistics largely died out after Chomskyan linguistics became the dominant research paradigm, it continued for quite some time elsewhere. The most accessible introduction to this literature is either Marcus's book [42] or the two volume survey [25]. Important early papers are by Sestier [51] and Kunze [38]. None of this work has any real learning results—it merely uses distributional learning as an analytical tool. A lot of computational work also uses distributional learning explicitly or implicitly, [5, 1, 70, 35] but we do not discuss this work here.

Distributional learning is also closely related to the context-free grammar formalism. The word 'context' after all appears in the term context-free and is also a foundational concept in distributional learning—this is not a coincidence. The context-free grammar formalism was originally devised to represent the outputs from distributional learning procedures. Chomsky [7, p.172, fn.15] says:

> The concept of "phrase structure grammar" was explicitly designed to express the richest system that could reasonably be expected to result from the application of Harris-type procedures to a corpus.

Over the years since then many different learning procedures for context-free grammars have been devised based on the intuitions of distributional learning. Typically these algorithms are based on the justification that two strings derived from the same nonterminal will be distributionally similar; therefore one can try to reverse this process by finding clusters of distributionally similar strings and creating a grammar with nonterminals that generate these strings. The naive application of these heuristic approaches has been known for a long time to suffer from some serious problems; Chomsky was perhaps the first to articulate these problems explicitly in his doctoral thesis [6], and indeed much of the technical work that we describe can be seen as an attempt to either answer or avoid those problems, which at the time were taken to be compelling arguments against the very possibility of distributional learning.

3 Languages and Grammars

We start by defining some standard notation. We assume that we have a finite nonempty set called the *alphabet* which we denote by Σ. This set might consist of the words in a language, or the set of phonemes, a set of letters, or even DNA bases or amino acids, depending on the application. We write Σ^* for the set of all finite strings of elements of Σ. We write Σ^+ for the nonempty strings. We write λ for the empty string. A *(formal) language* is just a subset of Σ^*; if L is a language then $L \subseteq \Sigma^*$. In this chapter we consider this very restricted notion of a language: this might be for example the set of grammatical sentences in a language, or the set of phonotactically well formed words in a language or something else. We abstract away from the particular details and consider it just as a set of strings that is defined in some way.

The languages we are interested in are typically infinite, or even if finite are very large, so we need finite representations. In this chapter we look only at the class of multiple context-free grammars (MCFGs). Context-free grammars (CFGs) are a special case of MCFGs; we start by defining the standard class of CFGs, and in Section 8 we define the larger class of MCFGs.

A CFG over an alphabet Σ is a tuple which consists of a nonempty finite set of nonterminal symbols V, together with a set of productions of the form $N \to \alpha$ where $N \in V$ and $\alpha \in (V \cup \Sigma)^*$. We also have a finite set of initial symbols $I \subseteq V$, which in the standard definition consists of just one symbol S. The extension to multiple symbols does not change anything. We denote the standard derivation relation by $\stackrel{*}{\Rightarrow}_G$, and define the set of strings derivable from a nonterminal N to be $\mathscr{L}(G,N) = \{ w \in \Sigma^* \mid N \stackrel{*}{\Rightarrow}_G w \}$. The language defined by the grammar is defined to be $\mathscr{L}(G) = \bigcup_{S \in I} \mathscr{L}(G,S)$.

3.1 Learning models

We are interested in learning: we therefore assume that there is some language that we are trying to learn. We write L_* for the target language: i.e. the language that we are trying to learn. We consider a variety of learning models here, from ones where the information sources available to the learner are very limited to ones where they are quite rich; we always consider only information about the language (the set of strings generated by the grammar), and none about the grammar itself. Additionally we always require polynomial update time – the learner at each step can only use a polynomial amount of computation. This is not enough on its own to be truly restrictive; there is some technical detail which we omit here [45].

- The first model is the oldest: positive only identification in the limit in the Gold style [23]. The learner receives a sequence of examples drawn from L_*, and must converge after a finite but unbounded time to an exactly correct hypothesis. Crucially there are no constraints on the sequence of examples other than the trivial ones that the examples are all in L_* and that every element of L_* must occur at least once somewhere in the sequence. In this model we cannot learn any superfinite classes of language; we obtain learnable classes by considering only languages which satisfy some language theoretic closure properties. This model therefore places some significant restrictions on the classes of languages that can be learned.
  We require the existence of a polynomially bounded characteristic set and polynomial update time. There are some technical issues about the appropriate way of defining this for CFGs, since we can have grammars that define languages that have very long strings, and this needs to be taken into account when defining the appropriate bound on the characteristic set.
- The second model we consider is that of positive data and membership queries (MQs). The same as the previous model, but the learner can also ask MQs and find out whether a particular string is in the target language. This is the easiest model: the easiest model to learn under, but also the model that is easiest to understand and easiest to prove results in, and accordingly we will focus on this model. However, on its own it is not restrictive as it is possible to define vacuous enumerative algorithms that nonetheless can learn using various computational tricks. We do not use these tricks in the algorithms we present.
- Minimally adequate teacher (MAT) model[1] [2, 55, 10, 68]. Here the learner has two sources of information: it can ask MQs and equivalence queries (EQs). Here we allow extended EQs. The learner can construct any CFG and ask whether it is correct or not. The teacher either says yes or provides a counterexample in the symmetric difference of the hypothesis and the target. Note that this is not computable for all CFGs. This is a restrictive learning model, in that it is known that classes such as regular grammars, and CFGs are not learnable in this model [3], whereas deterministic regular grammars, and congruential CFGs are, as we shall see.

[1] See Learning grammars and automata with queries, C. de la Higuera (this volume).

- Finally, there are results that use stochastic data – we assume the learner has access to positive samples drawn independently and identically from a fixed distribution, where the distribution is generated by a probabilistic version of the target grammar [8, 41].

3.2 Contexts and distributions

One of the most basic notions is that of a context which is just a pair of strings. In the original papers this is written as an ordered pair (l,r) where $l,r \in \Sigma^*$. Here we will use a slightly different notation $l\Box r$. This avoids confusion when we move to MCFGs, and makes it clearer that it represents a sentence with a hole. We therefore write the special empty context (λ,λ) as just $\Box$. In linguistics, a context is sometimes called an *environment*.

We can combine a context and a string using the 'wrap' operation, for which we use the symbol $\odot$. This combines a context with a string, by inserting the string into the gap in the context: we define this therefore as $(l\Box r)\odot u = lur$. The empty context thus does not change the string it is wrapped around: $\Box \odot u = u$.

We extend this to sets of strings and contexts in the natural way, so

$$C \odot S = \{ lur \mid l\Box r \in C, u \in S \}.$$

If we fix a language $L \subseteq \Sigma^*$ then we can talk about the relation between contexts and substrings given by $l\Box r \sim_L u$ iff $lur \in L$. If $lur \in L$ then we say that *u occurs in* the context $l\Box r$ in the language L.

We now define the notion of the distribution of a string in the language. Note that this has nothing to do with the notion of a probability distribution.

$$C_L(u) = \{ l\Box r \mid lur \in L \}$$

If we can successfully model this distribution then we will have learned the language since $\Box \in C_L(u)$ iff $u \in L$. We also write this set $C_L(u)$ as $u^{\triangleright}$ when L is understood. Conversely for a context $l\Box r$ we define $(l\Box r)^{\triangleleft}$ to be $\{ u \mid lur \in L \}$. Note that the residual languages $u^{-1}L = \{ v \mid uv \in L \}$ are in this notation $(u\Box)^{\triangleright}$. We extend these to sets of contexts: $C^{\triangleleft} = \{ u \mid \forall l\Box r \in C,\, lur \in L \}$. Alternatively and perhaps more intuitively:

$$C^{\triangleleft} = \{ u \mid C \odot u \subseteq L \}.$$

Given a set of strings S we can also define

$$S^{\triangleright} = \{ l\Box r \mid (l\Box r) \odot S \subseteq L \}.$$

For a set of strings D we define

$$\mathrm{Sub}(D) = \{ u \in \Sigma^* \mid lur \in D \text{ for some } l,r \in \Sigma^* \},$$

$$\mathrm{Con}(D) = \{ l\Box r \mid lur \in D \text{ for some } u \in \Sigma^* \}.$$

Distributional learning techniques are based on modeling the context-substring relation of a language. There are two technical details which we need to pay attention to: one is the case of the empty string, which as always in CFGs needs to be dealt with as a special case; the second is whether the CFG has more than two nonterminals on the right hand side of a rule. While every CFG can be put into Chomsky normal form, these learning algorithms depend on a correspondence between the nonterminals and sets of strings in the grammar that may not be preserved under binarisation.

3.3 Observation tables

A natural way of visualising the relation between strings and contexts is through observation tables (OTs) [2, 1]. We show a simple example in Table 1. We assume a finite set of substrings that we call K; these form the rows of the table; we have a finite set of contexts, F, that we use as rows. In the entry corresponding to the row indexed by u and the column indexed by $l\Box r$ we put a 1 if $lur \in L_*$ and a 0 if $lur \notin L_*$. The table in the example contains a limited amount of information about the language: the language includes $\lambda, ab, aabb$, but does not include the strings $a, b, aa, bb, aaa, aab, abb, bbb, aaab, abbb$ or $aabbb$. The table does not contain any information about other strings, for example $abab$, which may or may not be in L_*. Thus there are a number of different CFLs that are compatible with this information, from the finite language consisting just of those three examples, $\lambda, ab, aabb$ that are certainly in the language to the nonregular language $\{a^n b^n \mid n \geq 0\}$.

Table 1 Example of an OT.

	$\Box$	$a\Box$	$\Box b$	$\Box bb$
λ	1	0	0	0
a	0	0	1	0
b	0	1	0	0
aa	0	0	0	1
ab	1	0	0	0
aab	0	0	1	0

These approaches can be seen to be closely related to the classical techniques for regular inference which are based on modeling the relationship between prefixes and suffixes. In that model (e.g. [2]), an OT has rows indexed by prefixes, and columns indexed by suffixes, where a cell in the table has a 1 if and only if the concatenation of the corresponding prefix and suffix is in the language.

4 Context-Free Algorithms

All of these algorithms correspond to making a representational decision about what sets of strings the nonterminals should generate. We use the notation $[\![x]\!]$ for a nonterminal corresponding to some object x; typically a string or context, or set of strings or contexts. We need to decide what $\mathscr{L}(G, [\![x]\!])$ should be.

4.1 Primal-dual distinction

There is an important conceptual division that we want to discuss in general terms now – which is the division between primal methods and dual methods. In distributional learning, we have contexts and substrings and the relation between them. We have a choice – we can either take the substrings as being the primary objects and consider the contexts as being features, or we can swap the role of the contexts and substrings, and consider the contexts as being the primary objects and the substrings as being features.

Primal algorithms thus define sets of strings using one of the following schemes, where u is a single string and X is a set of strings:

$$\begin{aligned} [u] &= \{ v \mid v^{\triangleright} = u^{\triangleright} \} \\ u^{\triangleright\triangleleft} &= \{ v \mid v^{\triangleright} \supseteq u^{\triangleright} \} \\ X^{\triangleright\triangleleft} &= \{ v \mid v^{\triangleright} \supseteq X^{\triangleright} \} \end{aligned}$$

Dual approaches on the other hand take a context $(l\Box r)$ or a finite set of contexts C and use the sets of strings defined as follows:

$$\begin{aligned} (l\Box r)^{\triangleleft} &= \{ w \mid lwr \in L \} \\ C^{\triangleleft} &= \{ w \mid C \odot w \subseteq L \} \end{aligned}$$

In the case of left regular grammars, these two are essentially similar: we consider only contexts of the form $\Box u$, and switching between primal and dual approaches is just equivalent to reversing the strings of the language. There is therefore no theoretically interesting difference between the primal and dual techniques. In the case of CFGs and MCFGs the two approaches differ radically in the types of languages that can be learned. One immediately obvious difference is that using a dual approach one can always define the language itself, as a set of strings, using the single context $\Box$ since $\Box^{\triangleleft} = L$.

5 Primal Algorithms

We now survey the major primal algorithms for CFG inference, not in chronological order. We will start by considering the most basic and mathematically tractable model; the congruential model. This model is also the closest to the models informally described by the American structuralists. If we consider a clustering model based on the distributions of strings, then the most fundamental model is one where the clusters are sets of distributionally identical sets of strings.

5.1 Congruential languages

The most basic result is the MAT-learner result for what are called *congruential* CFG*s* [10]. The class of languages that can be learned using this algorithm is the class of congruential CFLs. Congruential CFGs are such that for all nonterminals N if $u, v \in \mathscr{L}(G,N)$ then they are *congruent* in the sense that $C_L(u) = C_L(v)$, which we will write $u \equiv_L v$. These are closely related to the *non-terminally separated (*NTS*) languages* [50, 4]. These form a proper subclass of CFLs that nontheless include all regular languages.

We can construct a grammar directly from an OT: we will explain this case in full detail, as this is the simplest model and the basic ideas are reused several times later. Taking the example from Table 1, we construct the grammar on lower part of the same table using the following procedure. Recall that the rows in the table are indexed by substrings of strings that are in the language: for each row in the table, corresponding to a substring u, we create a new nonterminal $[\![u]\!]$. We want this nonterminal to generate the string u and all other strings that are congruent to it. This gives us 6 nonterminals. First of all we note that the two strings λ and ab occur in the empty context $\square$ and are therefore in the language: we accordingly pick the two symbols, $[\![\lambda]\!]$ and $[\![ab]\!]$, as being the start symbols. We now add productions of three types: lexical, branching, and chain (unary) productions. First of all if w is of length 1 or zero, that is to say $w = a$ for some letter $a \in \Sigma$ or is equal to λ, we add a rule of the form $[\![a]\!] \to a$ or $[\![\lambda]\!] \to \lambda$. Note that in this rule $[\![a]\!]$ is a nonterminal symbol, and a is a terminal symbol, which are different in a CFG. Next, for every string w which is of length at least two, we add all possible branching rules of the following form: We split w into two strings u, v each of length at least 1 that occur in the table, such that $w = uv$, and add a production for each of these splits of the form $[\![w]\!] \to [\![u]\!][\![v]\!]$. This is a binary production with two nonterminal symbols on the right hand side of the rule. In the example, we have two strings of length 2, which each have a unique split. We have one string of length 3, aab, which can be split in two different ways. We therefore have two branching productions with the symbol $[\![aab]\!]$ on the left hand side. In general, if we have a string of length n, where $n > 1$, then we will have $n-1$ corresponding branching productions.

These productions on their own are rather trivial—if the grammar consisted only of these productions then a nonterminal $[\![u]\!]$ could generate only the string u and

no other string. We also want each nonterminal to generate other strings that are distributionally identical to u; accordingly we add nonbranching rules between two nonterminals $[\![u]\!]$ and $[\![v]\!]$ whenever it appears that the substrings u and v are distributionally identical. Of course it is impossible in general to tell from a finite amount of information whether $u \equiv_L v$ since this is an infinitary property – the distributions $C_L(u)$ and $C_L(v)$ are often infinite sets, and thus we cannot expect to get an exact answer in a finite amount of time. We can however get an approximate answer: we can use the OT to get a finite approximation of the distribution of the two strings u, v. For example, in Table 1, the two strings a and aab appear to be distributionally identical, or at least in the table there is no evidence suggesting that they are not identical, as the two rows that they label are identical. If we use F to refer to the finite set of contexts that label the columns of the OT, then we are testing whether $C_L(u) \cap F = C_L(v) \cap F$, rather than whether $C_L(u) = C_L(v)$. We therefore add productions of the form $[\![u]\!] \to [\![v]\!]$ whenever u and v have the same rows in the table. Now the grammar is more interesting: the nonterminals like $[\![a]\!]$ generate an infinite set of strings. The generated grammar is shown in Table 2.

Table 2 The generated grammar based on Table 1.

N	$N \in I$?	Lexical rules	Branching rules	Chain rules
$[\![\lambda]\!]$	Y	$[\![\lambda]\!] \to \lambda$		$[\![\lambda]\!] \to [\![ab]\!]$
$[\![a]\!]$		$[\![a]\!] \to a$		$[\![a]\!] \to [\![aab]\!]$
$[\![b]\!]$		$[\![b]\!] \to b$		
$[\![aa]\!]$			$[\![aa]\!] \to [\![a]\!][\![a]\!]$	
$[\![ab]\!]$	Y		$[\![ab]\!] \to [\![a]\!][\![b]\!]$	$[\![ab]\!] \to [\![\lambda]\!]$
$[\![aab]\!]$			$[\![aab]\!] \to [\![a]\!][\![ab]\!]$, $[\![aab]\!] \to [\![aa]\!][\![b]\!]$	$[\![aab]\!] \to [\![a]\!]$

The final grammar then has six nonterminals, two of which are initial, three lexical rules, three branching rules and four unary rules. In this form it is hard to see what is happening, and so it is convenient to convert it into a more readable grammar by merging nonterminals that are linked via unary rules. This gives us a grammar which has four nonterminals, one of which is initial. We can relabel the nonterminals for legibility, with S being the nonterminal corresponding to the two original nonterminals $[\![\lambda]\!], [\![ab]\!]$, A being the nonterminal corresponding to the nonterminals $[\![a]\!]$ and $[\![aab]\!]$, B corresponding to $[\![b]\!]$ and X corresponding to $[\![aa]\!]$. We then have the following grammar:

- Nonterminals are $\{S, A, B, X\}$ with one start symbol S
- Lexical productions $A \to a$, $B \to b$, $S \to \lambda$
- Branching productions $S \to AB$, $A \to AS$, $A \to XB$, $X \to AA$

This grammar generates an infinite nonregular language which is $\{\lambda, ab, aabb, aababb, \dots\}$.

So given an OT, we can write down a set of nonterminals and productions; but this leaves unanswered a very important question: how do we pick the rows and columns of the OT? The construction procedure that we have just defined has two

interesting properties that make it possible to answer this question. These are called the *monotonicity* properties.

Table 3 Example where we have added two more columns to Table 1; $aa\square bb$ and $\square abb$, As a result the generated grammar no longer contains any unary or chain rules and just generates the finite language $\{\lambda, ab, aabb\}$.

	$\square$	$a\square$	$\square b$	$\square bb$	$aa\square bb$	$\square abb$
λ	1	0	0	0	1	0
a	0	0	1	0	0	1
b	0	1	0	0	0	0
aa	0	0	0	1	0	0
ab	1	0	0	0	0	0
aab	0	0	1	0	0	0

Table 4 The generated grammar no longer contains any unary or chain rules and just generates the finite language $\{\lambda, ab, aabb\}$.

N	$N \in I$?	Lexical rules	Branching rules
$[\![\lambda]\!]$	Y	$[\![\lambda]\!] \to \lambda$	
$[\![a]\!]$		$[\![a]\!] \to a$	
$[\![b]\!]$		$[\![b]\!] \to b$	
$[\![aa]\!]$			$[\![aa]\!] \to [\![a]\!][\![a]\!]$
$[\![ab]\!]$	Y		$[\![ab]\!] \to [\![a]\!][\![b]\!]$
$[\![aab]\!]$			$[\![aab]\!] \to [\![a]\!][\![ab]\!]$, $[\![aab]\!] \to [\![aa]\!][\![b]\!]$

First, if we increase the columns in the table, the language defined by the grammar generated from the table will always be smaller than or equal to the grammar generated from the original table. Table 3 shows a table with two more columns; which we have filled in using MQs on some hypothetical language. Now the resulting grammar, shown in Table 4, contains no unary rules, and each nonterminal only generates the single string that it is labeled with. This grammar therefore defines a small finite language which consists of just the two strings $\{\lambda, ab\}$; this is a proper subset of the language defined by the original grammar. It is easy to see why this will in general always be the case: if we add columns, the generated grammar will have the same set of nonterminals, lexical and binary productions, but may have fewer unary productions. Adding additional columns means that the approximate test for congruence becomes more accurate and stringent, and as a result some unary rules will be removed. Thus the resulting grammar will in general generate a language which is a subset of the original, though it may remain the same.

We have a complementary result when we add an additional row or rows to Table 1. Table 5 gives a simple example: we add one more row, labeled with the string *bab*. The resulting grammar shown in Table 6 is now larger: the set of productions and nonterminals include the original productions and nonterminals and as a result

Table 5 Example where we have added one more row (*bab*) to Table 1.

	$\square$	$a\square$	$\square b$	$\square bb$
λ	1	0	0	0
a	0	0	1	0
b	0	1	0	0
aa	0	0	0	1
ab	1	0	0	0
aab	0	0	1	0
bab	0	1	0	0

Table 6 The resulting grammar now has more nonterminals and productions than the original and as a result generates a larger language.

N	$N \in I$?	Lexical rules	Branching rules	Chain rules
$[\![\lambda]\!]$	Y	$[\![\lambda]\!] \to \lambda$		$[\![\lambda]\!] \to [\![ab]\!]$
$[\![a]\!]$		$[\![a]\!] \to a$		$[\![a]\!] \to [\![aab]\!]$
$[\![b]\!]$		$[\![b]\!] \to b$		$[\![b]\!] \to [\![bab]\!]$
$[\![aa]\!]$			$[\![aa]\!] \to [\![a]\!][\![a]\!]$	
$[\![ab]\!]$	Y		$[\![ab]\!] \to [\![a]\!][\![b]\!]$	$[\![ab]\!] \to [\![\lambda]\!]$
$[\![aab]\!]$			$[\![aab]\!] \to [\![a]\!][\![ab]\!]$, $[\![aab]\!] \to [\![aa]\!][\![b]\!]$	$[\![aab]\!] \to [\![a]\!]$
$[\![bab]\!]$			$[\![bab]\!] \to [\![b]\!][\![ab]\!]$	$[\![bab]\!] \to [\![b]\!]$

the language defined is going to be larger. The grammar after merging nonterminals and relabeling is this:

- Nonterminals are $\{S,A,B,X\}$ with one start symbol S
- Lexical productions $A \to a, B \to b, S \to \lambda$
- Branching productions $S \to AB$, $A \to AS, A \to XB$, $X \to AA$, $B \to BS$

This grammar generates a nonregular language that is larger and includes the string *abab*.

The end result of these two monotonicity properties is that it is easy to construct a learning algorithm. We maintain an OT; if the grammar generates too small a language, then we can add some rows to reinforce the grammar, and if, on the other hand, the grammar overgenerates, then we can add columns in order to make the grammar more accurate. There are a number of different ways of doing this: if we observe some string w which is not generated by the current grammar, the most naive approach is simply to add every element of $\mathrm{Sub}(w)$ as a row. This is guaranteed to make the grammar generate w and simplifies the convergence analysis. In this learning model we have an oracle that can be used to make MQs, and so we can fill in all of the spaces in the OT easily. In other learning models, we need to use other approaches.

The fundamental representational assumption though is that nonterminals are indexed by substrings u, and we want each nonterminal $[\![u]\!]$ to generate all strings that are distributionally identical to u. That is to say we want $\mathscr{L}(G,[\![u]\!]) = [u]$. This assumption is what distinguishes these congruential approaches from others that we examine later: this is the simplest primal approach.

The same representational assumption can be used to define an algorithm to learn related classes of languages in a stochastic setting [8, 41, 52]. In these models, we cannot ask queries but only have a randomly generated sequence of examples. In this case we can have an OT that stores counts rather than just a zero/one value. In each cell of the table we store the number of times we have seen the string that corresponds to that cell. Congruence then can be replaced by its stochastic variant. The classes of languages that we can prove we can learn here are quite limited, and the assumptions quite strong and unrealistic; nevertheless, this shows that stochastic variants of these algorithms are possible.

5.2 Substitutable languages

If we want to learn under the more stringent Gold paradigm, where we have neither queries nor any constraints on how the positive samples are being selected, then we need to use a slightly different algorithm that relies on a language theoretic closure property in order to guarantee convergence. We maintain the same representational assumption as in the previous section—the nonterminals will generate congruence classes.

Given two nonempty strings u and v, we say that $u \doteq_L v$ if there is a context $l\Box r$ such that $lur \in L$ and $lvr \in L$. A language L is substitutable if $u \doteq_L v$ implies $u \equiv_L v$. This is a very strong condition; analogous to reversibility in the inference of regular languages [16]. Indeed substitutability implies reversibility; there are however languages which are substitutable but not context-free (see for example the language MIX which we define below). Languages that are substitutable include examples like $\{a^n cb^n \mid n > 0\}$ but is too strong to be of much practical interest: for example even the language $\{a^n b^n \mid n > 0\}$ is not substitutable. There are even finite languages that are not substitutable: $\{a, aa\}$ is a trivial example.

Clark and Eyraud [16] show that this class of languages can be learned from positive data alone using a Gold model; the algorithm is polynomial update time, and has a characteristic set with polynomial number of elements. When the algorithm observes two substrings of the data that occur in a single context then it assumes that they are congruent; the restriction of the class substitutes for the lack of MQs. This has been extended to k,l-substitutable CFGs [62], in a manner analogous to k-reversibility in the inference of regular languages.

Interestingly the criteria of substitutability is quite natural and was already noted in the early days of structuralist linguistics: Myhill in 1950 [43] gives an equivalent definition and suggests calling languages that satisfy that definition 'regular'!

5.3 Finite kernel property

In the case of congruential languages, we considered nonterminals that generate congruence classes. $\mathscr{L}(G, [\![u]\!]) = [u]_{L^*} = \{ v \mid C_L(v) = C_L(u) \}$. A slightly different condition would be to consider the set: $u^{\triangleright\triangleleft} = \{ v \mid C_L(v) \supseteq C_L(u) \}$. Now we allow sets of strings that include strings whose distribution properly includes the distribution of u. These strings will have an asymmetric substitution property: whenever we have a string like lur we can substitute any string $v \in u^{\triangleright\triangleleft}$ to get a string lvr but perhaps not in reverse. There are a number of cases where this could be useful; in natural languages we often have that an ambiguous word has a wider distribution than an unambiguous one. In this case we might want to have a nonterminal that generates not just the unambiguous words but also words that can have other lexical categories as well.

A further generalization of this is to consider the case where rather than considering the nonterminals to be generated by individual strings, we can consider them to be generated by small finite sets of strings. Given a bound k. we can consider sets of strings S, such that $|S| \leq k$, and consider nonterminals that are indexed by these sets. In this case we want the nonterminal $[\![S]\!]$ to generate the set of strings that can occur in all of the contexts that are *shared* by the elements of S. This allows us to have nonterminals that correspond to clusters of strings that are distributionally similar but not identical.

More formally we say a CFG has the *k-Finite kernel property* (k-FKP) if every nonterminal N has a set of strings S_N, $|S_N| \leq k$ such that $\mathscr{L}(G,N)^{\triangleright} = S_N^{\triangleright}$. The class of all CFGs with k-FKP can be learned using examples and MQs [66].

6 Dual Algorithms

In dual algorithms we swap the roles of the substrings and the contexts: we index nonterminals by contexts or sets of contexts, and use substrings to eliminate the incorrect rules. The representational assumption is then quite different. We take a context or more generally a finite set of contexts C, and consider the sets of strings that can occur in all of the contexts: $C^{\triangleleft}$. We then define grammars where the nonterminals correspond to these sets of strings.

6.1 Context deterministic grammars

The first dual learning result is by Shirakawa and Yokomori who, in an important early paper [55], which in our opinion has not received enough attention, define the class of c-deterministic grammars as those grammars G such that whenever $S \stackrel{*}{\Rightarrow}_G lNr$ it is the case that $\mathscr{L}(G,N) = (l\Box r)^{\triangleleft}$. They then provide a MAT learning algorithm for this class. There is a small error in this paper: the paper claims that

the class includes all regular languages, but the grammar construction is slightly too weak for this. A minor modification—allowing rules of the form $N \to Pa$ and $N \to aP$, where N, P are nonterminals and $a \in \Sigma$—is sufficient to correct this.

Having defined the nonterminals as corresponding to sets of strings that occur in a given context $l\Box r$ we then can use the strings to eliminate rules. Suppose we have three nonterminals that correspond to the three contexts $l_1\Box r_1$, $l_2\Box r_2$, $l_3\Box r_3$; reusing the earlier notation we can say the nonterminal symbols are $[\![l_1\Box r_1]\!]$, $[\![l_2\Box r_2]\!]$ and $[\![l_3\Box r_3]\!]$. We can consider the possible production $[\![l_1\Box r_1]\!] \to [\![l_2\Box r_2]\!][\![l_3\Box r_3]\!]$. If this rule is correct, then the result of concatenating any string that can occur in $l_2\Box r_2$ with any string that can occur in $l_3\Box r_3$ will be a string that can occur in $l_1\Box r_1$, or using the notation we defined earlier $(l_1\Box r_1)^\triangleleft \supseteq (l_2\Box r_2)^\triangleleft (l_3\Box r_3)^\triangleleft$. Crucially, if this is *false*, then we can observe some strings u, v that show that it is false: if l_2ur_2 and l_3vr_3 are in L_* but l_1uvr_1 is *not*, then we will know that the production is incorrect in a certain sense. Thus, just as contexts were used to eliminate undesirable unary chain rules in the congruential case earlier, strings are used to eliminate undesirable binary rules in this c-deterministic case.

6.2 Finite context property

One can weaken this condition in two ways. One is by requiring only that there be some context $l\Box r$ such that $\mathscr{L}(G,N) = (l\Box r)^\triangleleft$; this is a weaker condition because the c-deterministic condition requires this to be true for *any* context of the nonterminal N. The second is that we allow more than one context. This leads us to the *k-finite context property* (k-FCP) [66, 40].[2] A CFG has the k-FCP if for every nonterminal we can find a set of contexts C, where $|C| \leq k$, such that $\mathscr{L}(G,N) = C^\triangleleft$. One can also modify this to a slightly weaker form as in [66]. A closely related idea, context-separability is defined in [1] – this is equivalent to the 1-FCP.

7 Combined Primal-Dual Methods

The primal and dual techniques can be combined, to produce an algorithm which can learn classes where the nonterminals can be defined either primally or dually [67]; we can use these techniques to combine for example the congruential learning result and the c-deterministic learning result to get an algorithm which can MAT-learn a larger class. The class that is learnable using these combined methods will be strictly larger than the union of the learnable classes with either primal or dual on its own, as it will include languages where some of the nonterminals can only be defined primally and some can only be defined dually. The classes learned are stratified by three natural numbers: r the maximum number of nonterminals used on

[2] The original paper defining this [12], unfortunately contains some errors.

the right hand side of a production, p the maximum number of strings used to define a nonterminal primally, and q the maximum number of contexts used to define a nonterminal dually; we denote $\mathbb{G}(p,q,r)$ to refer to the class of CFGs that satisfy these bounds. Yoshinaka [67] shows that the class $\mathbb{G}(p,q,r)$ can be learned using positive data and MQs.

It is important to realise that there are CFLs that cannot be generated by any $\mathbb{G}(p,q,r)$ for any values of p,q,r. A simple example is the language $\{ a^n b^m \mid n \neq m \}$. This is clearly a CFL but cannot be represented by any grammar in this class. This is because in order to represent this language we need nonterminals that will generate sets of strings like $\{ a^n b^m \mid n > m \}$ and $\{ a^n b^m \mid n < m \}$. Neither of these sets can be defined by any finite number of strings or contexts. Thus this language, and others like it, are not learnable using any of these distributional techniques. Figure 1 shows the relationship of the various learnable classes to the classes of regular and CFLs. It is possible to get a more integrated view of the representational assumptions of these algorithms by looking at the Syntactic Concept Lattice—the residuated lattice consisting of all distributionally definable sets of strings [9, 61, 14].

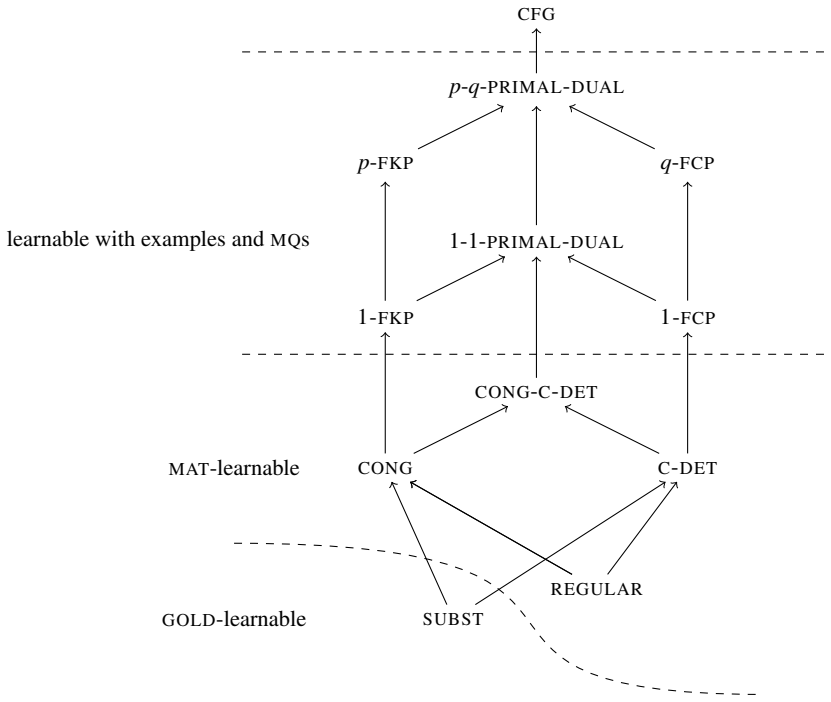

Fig. 1 Diagram showing the various classes of CFGs learnable using these techniques. All inclusions are strict. CONG is the class of congruential CFGs in Section 5.1, SUBST is the class of substitutable CFGs in Section 5.2, and C-DET is the class of context-deterministic grammars in Section 6. The dual techniques are on the right and the primal techniques are on the left; substitutable languages are both primal and dual.

8 Multiple Context-Free Grammars

CFGs are fairly expressive for describing natural languages, yet the literature has found several natural language phenomena that cannot be described by CFGs. The example which definitively established that CFGs were not weakly adequate was the case of cross-serial dependencies in Swiss German [27, 53]. We present here the data in a form very close to the original presentation. In the particular dialect of Swiss German considered by Shieber, the data concerns a sequence of embedded clauses.

Let us abstract this a little bit and consider a formal language for this non-context-free fragment of Swiss German. We consider that we have the following words or word types: n_a, n_d which are respectively accusative and dative noun phrases, v_a, v_d which are verb phrases that require accusative and dative noun phrases respectively, and finally c which is a complementizer which appears at the beginning of the clause. Thus the 'language' we are looking at consists of sequences like cn_av_a and cn_dv_d and $cn_an_an_dv_av_av_d$, but crucially does not contain examples where the sequence of accusative/dative markings on the noun sequence is different from the sequence of requirements on the verbs. So it does not contain cn_dv_a, because the verb requires an accusative and it only has a dative. The sublanguage we are concerned with is the language $L_{\mathrm{SG}} = \{\, cn_a^i n_d^j v_a^i v_d^j \mid i, j \geq 1 \,\}$. This language is defined through intersection of the original language with a suitable regular language and a homomorphism relabeling the strings. Since CFGs are closed under these operations, and L_{SG} is clearly not context-free, this establishes the non-context-freeness of the original language.

Joshi et al. [28] proposed the notion of *mildly context-sensitive (*MCS*) grammars* to pursue a better formalism to describe natural languages. They suggested that an MCS family of languages should

1. include CFLs,
2. allow limited cross-serial dependencies,
3. have the constant-growth property,[3]
4. have polynomial-time parseability.

Here what limited cross-serial dependencies means is unclear, and various definitions of MCS formalisms have been proposed. These different grammar definitions have converged to three different language classes. The smallest class is the class defined by Linear Indexed Grammars, Combinatory Categorial Grammars and Tree Adjoining Grammars (TAGs) [58]. The largest class is defined by the formalism we use here, Multiple Context-Free Grammars (MCFG) [49], which are essentially equivalent to Linear Context-Free Rewriting Systems [59]; these are also equivalent[4] to the more linguistically motivated Minimalist Grammars [57] which came

[3] An infinite language L is said to have the constant-growth property if $\exists k \in \mathbb{N}. \forall u \in L. \exists v \in L.\ 0 < |v| - |u| \leq k$.

[4] It is worth noting that MCFGs may be much larger than the smallest equivalent Minimalist Grammar.

out of an attempt to formalise some ideas in contemporary syntactic theory. There is also an intermediate class of well-nested MCFGs [31], non-duplicating Macro Grammars [22] and Coupled CFGs [26], which form a proper subset of the class of all MCFGs, and a proper superset of the class of TAG-equivalent grammars, which are equivalent to well-nested MCFGs of dimension 2.

MCFGs are a very natural generalisation of CFGs. While nonterminals of a CFG generate strings, those in an MCFG generate tuples of strings; for example pairs of strings. These strings in a tuple need not be adjacent in a complete sentence in the language of the grammar. This allows MCFGs to generate languages which have cross-serial dependencies.

We will start this section by defining the MCFG formalism which may be unfamiliar to the reader. We will then discuss the learnability of these, using distributional methods. We structure this section somewhat differently from the CFG section; given that the reader is now familiar with the primal/dual distinction, we will start with the positive data only learning model, and then move to the query-based learning models.

8.1 Definition

An MCFG is a quadruple $G = \langle \Sigma, V, R, I \rangle$ just like a CFG but each nonterminal symbol $N \in V$ is assigned a positive integer called the *dimension*, which we will denote by $\dim(N)$. By V_d we denote the subset of V whose elements have dimension d. Every start symbol has dimension 1: $I \subseteq V_1$. A nonterminal of dimension d generates d-tuples of strings. We write production rules of an MCFG in Horn clause notation[5], which consists of a single literal on the left and a possible empty sequence of literals on the right, where a nonterminal of dimension d appears as a predicate of d-ary. Terminal symbols can only appear on the left hand side of the rule. A rule $N \to aPQ$ of a CFG, for example, is written in this notation as

$$N(axy) :- P(x), Q(y),$$

where x, y are *variables* and a a terminal symbol, or a constant. This rule is read as follows: if P derives a string x and Q derives y, N derives axy. In an MCFG, nonterminals may have dimension more than 1. For example, the rule

$$N(x_1 y_1, a x_2) :- P(x_1, x_2), Q(y_1)$$

means that if P derives a pair (x_1, x_2) and Q derives y_1, N derives the pair $(x_1 y_1, a x_2)$, where the dimensions of N, P and Q are 2, 2 and 1, respectively.

More formally, production rules have the following form in general:

[5] The notation adopted in this chapter follows Smullyan's elementary formal systems [56] rather than [49].

$$N_0(\alpha_1,\ldots,\alpha_{d_0}) :- N_1(x_{1,1},\ldots,x_{1,d_1}),\ldots,N_k(x_{k,1},\ldots,x_{k,d_k})$$

where $N_0,N_1,\ldots,N_k \in V$ for some $k \geq 0$, $d_i = \dim(N_i)$ for each $i \in \{0,\ldots,k\}$, variables $x_{1,1},\ldots,x_{k,d_k}$ are pairwise distinct, and each $\alpha_1,\ldots,\alpha_{d_0}$ are strings of terminals and variables such that all and only variables $x_{1,1},\ldots,x_{k,d_k}$ occur just once through $\alpha_1,\ldots,\alpha_{d_0}$.[6] If $k=0$ then the right-hand side is empty, and the production is of the form $N_0(\mathbf{v}) :-$ where $\mathbf{v} \in (\Sigma^*)^{d_0}$. If $x_{i,j}$ always occurs left of $x_{i,j+1}$ in $\alpha_1 \ldots \alpha_m$ for $1 \leq i \leq k$ and $1 \leq j < d_i$, the rule is said to be *non-permuting*. An example of a rule that is *not* non-permuting is

$$P(x_1y_2,x_2y_1) :- Q(x_1,x_2),R(y_1,y_2)\,,$$

as y_2 occurs left to y_1 in $P(x_1y_2,x_2y_1)$.

Example 1. We define an MCFG $G_{\mathrm{SG}} = \langle \{c,n_a,n_d,v_a,v_d\}, V_1 \cup V_2, R, \{S\}\rangle$ where $V_1 = \{S\}$ and $V_2 = \{P,Q\}$. R consists of the rules

$$\begin{aligned}
S(cx_1y_1x_2y_2) &:- P(x_1,x_2),Q(y_1,y_2)\,;\\
P(n_ax_1,v_ax_2) &:- P(x_1,x_2)\,;\\
Q(n_dx_1,v_dx_2) &:- Q(x_1,x_2)\,;\\
P(n_a,v_a) &:- \,;\\
Q(n_d,v_d) &:- \,.
\end{aligned}$$

All of the above rules are non-permuting.

The derivation process of an MCFG is formally defined with a substitution. A *substitution* θ is a map from variables to strings, which is extended to the homomorphism $\hat{\theta}$ such that $\hat{\theta}(y) = \theta(y)$ if y is in the domain of θ, and $\hat{\theta}(y) = y$ otherwise. We identify $\hat{\theta}$ and θ if no confusion arises. A substitution θ is often denoted as a suffix operator $[x_1 \mapsto \theta(x_1),\ldots,x_k \mapsto \theta(x_k)]$, or even as $[\theta(x_1),\ldots,\theta(x_k)]$ if the domain of θ is understood and ordered as $x_1,\ldots,x_k$.

We write $\vdash_G N(\mathbf{v})$ if $N(\mathbf{v}) :-$ is a rule in R. If we have $\vdash_G N_i(\mathbf{v}_i)$ for all $i = 1,\ldots,k$ and R has a rule $N_0(\alpha_1,\ldots,\alpha_{d_0}) :- N_1(\mathbf{x_1}),\ldots,N_k(\mathbf{x}_k)$, then we deduce

$$\vdash_G N_0(\theta(\alpha_1),\ldots,\theta(\alpha_{d_0}))$$

where $\theta(\mathbf{x_i}) = \mathbf{v}_i$ for all $i = 1,\ldots,k$. We will abbreviate this substitution θ as $[\mathbf{v}_1,\ldots,\mathbf{v}_k]$. The *language of* N is defined by

$$\mathscr{L}(G,N) = \{\, \mathbf{v} \in (\Sigma^*)^{\dim(N)} \mid \vdash_G N(\mathbf{v}) \,\}\,.$$

The *language of* G is $\mathscr{L}(G) = \bigcup_{S \in I} \mathscr{L}(G,S)$.

Recall Example 1. It is easy to see that we have

[6] We only consider non-deleting productions in this chapter.

$$\vdash_{G_{\mathrm{SG}}} P(n_a, v_a), \qquad \vdash_{G_{\mathrm{SG}}} P(n_a n_a, v_a v_a), \qquad \vdash_{G_{\mathrm{SG}}} P(n_a n_a n_a, v_a v_a v_a),$$
$$\vdash_{G_{\mathrm{SG}}} Q(n_d, v_d), \qquad \vdash_{G_{\mathrm{SG}}} Q(n_d n_d, v_d v_d),$$

and

$$\vdash_{G_{\mathrm{SG}}} S(cn_a n_a n_a n_d n_d v_a v_a v_a v_d v_d),$$

for example. Figure 2 describes this derivation process in a tree form, where boxes emphasise the fact that the pair $\langle n_a n_a, v_a v_a \rangle$ generated by P appears as discontinuous strings in the final product $cn_a n_a n_a n_d n_d v_a v_a v_a v_d v_d$. It is easy to see that $\mathscr{L}(G_{\mathrm{SG}}) = L_{\mathrm{SG}} = \{ cn_a^i n_d^j v_a^i v_d^j \mid i, j \geq 1 \}$.

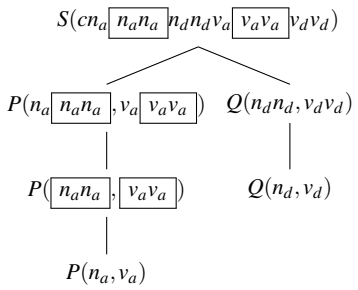

Fig. 2 Derivation tree of an MCFG

By MCFG(p,q) we denote the class of MCFGs such that

- every nonterminal has a dimension at most p,
- every rule has at most q occurrences of nonterminals on the right hand side.

Thus G_{SG} belongs to MCFG$(2,2)$. Then we define MCFL$(p,q) = \{ \mathscr{L}(G) \mid G \in \mathrm{MCFG}(p,q) \}$. We also write MCFG$(p,*) = \bigcup_{q \in \mathbb{N}} \mathrm{MCFG}(p,q)$ and MCFL$(p,*) = \bigcup_{q \in \mathbb{N}} \mathrm{MCFL}(p,q)$. The class of CFGs is identified with MCFG$(1,*)$ and the CFGs in Chomsky normal form are all in MCFG$(1,2)$. Hence MCFL$(1,2) = \mathrm{MCFL}(1,q)$.

Seki et al. [49] and Rambow and Satta [46] have investigated the hierarchy of MCFLs.

Proposition 1 (Seki et al. [49], Rambow and Satta [46]).

For $p \geq 1$, $\mathrm{MCFL}(p,*) \subsetneq \mathrm{MCFL}(p+1,*)$.

For $p \geq 2$, $q \geq 1$, $\mathrm{MCFL}(p,q) \subsetneq \mathrm{MCFL}(p,q+1)$ *except for* $\mathrm{MCFL}(2,2) = \mathrm{MCFL}(2,3)$.

For $p \geq 1$, $q \geq 3$ *and* $1 \leq k \leq q-2$, $\mathrm{MCFL}(p,q) \subseteq \mathrm{MCFL}((k+1)p, q-k)$.

Hereafter we fix p and q to be small natural numbers. An important property of the class MCFG(p,q) is the polynomial-time uniform parsability.

Theorem 1 (Seki et al. [49], Kaji et al. [30]). *Let p and q be fixed. It is decidable in $O(\|G\|^2 |w|^{p(q+1)})$ time whether $w \in \mathscr{L}(G)$ for any* MCFG $G \in \mathrm{MCFG}(p,q)$ *and $w \in \Sigma^*$.*

It is known that every MCFG in $\mathrm{MCFG}(p,q)$ has an equivalent one in $\mathrm{MCFG}(p,q)$ whose rules are all non-permuting [36], so we assume without loss of generality that all MCFGs are non-permuting in this chapter.

8.2 Generalisation of contexts and substrings and observation tables

Recall that classical algorithms for learning regular languages observe the relation between two strings p and s as prefixes and suffixes, respectively. That is, we have an OT whose rows are indexed by prefixes p and columns are by suffixes s and the entries show whether the concatenations ps belong to the target language L_*. The choice of those two types of objects correspond to the fact that a nonterminal of a (right) regular grammar generates suffixes of members of the language: $S \stackrel{*}{\Rightarrow} pN \stackrel{*}{\Rightarrow} ps$. In the distributional learning of CFGs the two sorts of objects we choose are contexts $l\Box r$ and substrings v, which correspond to the fact that a nonterminal of a CFG generates substrings: $S \stackrel{*}{\Rightarrow} lNr \stackrel{*}{\Rightarrow} lvr$. In accordance with the fact that an MCFG generates discontinuous substrings, we now generalise the notion of a context to a *multi-context* and a substring to a *multi-word* and define the corresponding wrap operation in the natural way. We call a pair of strings $\langle u,v\rangle$ *2-word*. A *2-context* contains exactly two occurrences of the hole: thus a 2-context has the form $l\Box m\Box r$. The wrapping operation is accordingly generalised as

$$l\Box m\Box r \odot_2 \langle u,v\rangle = lumvr\,.$$

Similarly we can have 3-contexts, 3-words, and so on, if we target MCFGs with nonterminals of dimension 3 or more. We denote p-word and p-context by bold letters $\mathbf{u},\mathbf{v},\ldots$ and sans-serif $\mathsf{u},\mathsf{v},\ldots$, respectively, and the sets of p-words and p-contexts by $\mathscr{S}_p$ and $\mathscr{C}_p$, respectively. The wrapping operations $\odot_p$ for p-words and p-contexts are also defined accordingly. For a string set D, we define

$$\mathrm{Sub}_p(D) = \{\,\mathbf{v} \in \mathscr{S}_p \mid \mathsf{u} \odot_p \mathbf{v} \in D \text{ for some } \mathsf{u} \in \mathscr{C}_p\,\},$$
$$\mathrm{Con}_p(D) = \{\,\mathsf{u} \in \mathscr{C}_p \mid \mathsf{u} \odot_p \mathbf{v} \in D \text{ for some } \mathbf{v} \in \mathscr{S}_p\,\}.$$

Understanding the concerned language L, the polar maps $\triangleright$ and $\triangleleft$ are also generalised for $\mathbf{S} \subseteq \mathscr{S}_p$ and $\mathsf{C} \subseteq \mathscr{C}_p$:

$$\mathbf{S}^{\triangleright} = \{\,\mathsf{u} \in \mathscr{C}_p \mid \mathsf{u} \odot_p \mathbf{S} \subseteq L\,\},$$
$$\mathsf{C}^{\triangleleft} = \{\,\mathbf{v} \in \mathscr{S}_p \mid \mathsf{C} \odot_p \mathbf{v} \subseteq L\,\}.$$

For two p-words $\mathbf{u}$ and $\mathbf{v}$, we define

$$\mathbf{u} \equiv_L \mathbf{v} \text{ if and only if } \{\mathbf{u}\}^{\triangleright} = \{\mathbf{v}\}^{\triangleright}\,.$$

Distributional learning algorithms for CFGs use an OT to observe which combination of a context and a substring together forms a sentence in the concerned language. Since an MCFG may have nonterminals of different dimensions, we now have an OT for each dimension.[7]

Table 7 Examples of OTs for $L_{\mathrm{SG}} = \{ cn_a^i n_d^j v_a^i v_d^j \mid i, j \geq 1 \}$.

1D	$\square$	$c\square$	$c\square v_a v_d$
λ	0	0	0
$cn_a v_a$	1	0	0
$n_a v_a$	0	1	0
$n_a n_d$	0	0	1
$n_a n_a$	0	0	0

2D	$cn_a\square n_d v_a\square v_d$	$c\square n_a n_d\square v_a v_d$	$cn_a\square n_d\square$
$\langle n_a, v_a \rangle$	1	1	0
$\langle n_d, v_d \rangle$	1	0	0
$\langle n_a n_d, v_a v_d \rangle$	1	0	0
$\langle n_a, n_d v_a v_d \rangle$	0	0	0
$\langle n_a, v_a v_a v_d \rangle$	0	0	1

Once we have obtained those generalisations, almost every technique in the distributional learning of CFGs can be translated for the learning of MCFGs straightforwardly as we will see in the following subsections.

8.3 Substitutability

The learnability result of substitutable CFGs presented in Section 5.2 can be translated to the MCFG learning [65]. For two 2-words $\langle u_1, u_2 \rangle, \langle v_1, v_2 \rangle \in \mathscr{S}_2$, let us write $\langle u_1, u_2 \rangle \doteq_L \langle v_1, v_2 \rangle$ if there is a 2-context $l\square m\square r$ such that $lu_1mu_2r \in L$ and $lv_1mv_2r \in L$. We say that a language L is *2*D*-substitutable*[8] if $\langle u_1, u_2 \rangle \doteq_L \langle v_1, v_2 \rangle$ implies $\langle u_1, u_2 \rangle \equiv_L \langle v_1, v_2 \rangle$.

The learning algorithm for pD-substitutable MCFGs in $\mathrm{MCFG}(p,q)$ is essentially the same as the one for substitutable CFGs and it runs in polynomial-time under the assumption that p and q are known to the learner. We note that the degree of the polynomial linearly depends on pq. The only difference is in the construction of the learner's conjecture grammar from a positive data set D. The nonterminal set is $V = \bigcup_{1 \leq i \leq p} V_i$, where

$$V_i = \{ [\![\mathbf{v}]\!] \mid \mathbf{v} \in \mathrm{Sub}_i(D) \}$$

is the set of nonterminals of dimension i. What we would like $[\![\mathbf{v}]\!]$ to generate is $\{ \mathbf{u} \mid \mathbf{u} \equiv_{L_*} \mathbf{v} \}$ where L_* is the learning target. The initial symbols are

$$I = \{ [\![v]\!] \in V_1 \mid v \in D \}.$$

[7] Technically speaking, the OT for the highest dimension subsumes the other ones for lower dimensions, since m-contexts and m-words can be seen as special cases of n-contexts and n-words, respectively, for $m < n$: e.g., $l\square r\square \odot_2 \langle u, \lambda \rangle = l\square r \odot u$. However, there are cases where it is more reasonable to exclude the empty string from consideration.

[8] The original definition [65] has a slightly weaker form, where strings m, u_1, u_2, v_1, v_2 are restricted to be non-empty strings. ab^*cd^*e is 2D-substitutable according to the original definition, but it is not the case in our simplified definition.

MCFGs do not have a simple and well-established normal form like Chomsky normal form in CFGs. One may introduce a normal form for MCFG(p,q) but it should involve many different types of rules differently from the case of CFGs, where branching rules and lexical rules suffice. In stead we introduce rules of the conjecture grammar in a general form. We have a decomposition rule of the form

$$[\![\mathbf{v}]\!](\alpha) :- [\![\mathbf{v}_1]\!](\mathbf{x}_1), \ldots, [\![\mathbf{v}_k]\!](\mathbf{x}_k)$$

if

- it is eligible for a rule of an MCFG in MCFG(p,q),
- $\alpha[\mathbf{v}_1, \ldots, \mathbf{v}_k] = \mathbf{v}$.

Decomposition rules can be seen as a generalisation of branching and terminating rules in distributional learning of CFGs. We also have chain rules between two 'substitutable' k-words:

$$[\![\mathbf{v}_1]\!](\mathbf{x}) :- [\![\mathbf{v}_2]\!](\mathbf{x})$$

if there is u such that $\mathsf{u} \odot_k \mathbf{v}_1, \mathsf{u} \odot_k \mathbf{v}_2 \in D$.

For example, when $p = q = 2$, from $D = \{abcde, aabccdee\}$, we construct rules

$$\begin{aligned}
[\![aabccdee]\!](x_{1,1}x_{2,1}, x_{1,2}x_{2,2}) &:- [\![\langle a, cde\rangle]\!](x_{1,1}, x_{1,2}), [\![\langle abc, e\rangle]\!](x_{2,1}, x_{2,2})\,; \\
[\![\langle aa, ccdee\rangle]\!](x_{1,1}a, cx_{1,2}e) &:- [\![\langle a, cde\rangle]\!](x_{1,1}, x_{1,2})\,; \\
[\![\langle a, cde\rangle]\!](a, cde) &:- \,; \\
[\![\langle a, cde\rangle]\!](x_{1,1}, x_{1,2}) &:- [\![\langle aa, ccdee\rangle]\!](x_{1,1}, x_{1,2})
\end{aligned}$$

among others. The first three rules are decomposition rules, whereas the last one is a chain rule, which is induced from the fact $\Box b \Box \odot_2 \langle a, cde\rangle, \Box b \Box \odot_2 \langle aa, ccdee\rangle \in D$.

However, the property of pD-substitutability for $p \geq 2$ is far too strong a requirement to be useful. The flexibility of the decomposition of a sentence into 2-contexts and 2-words often makes many 2-words to be weakly substitutable, and thus the 2D-substitutability assumption causes too much generalisation. For example, some singleton language, say $\{aaabaaa\}$, is not 2D-substitutable. We have $a\Box a\Box a \odot_2 \langle aab, a\rangle = a\Box a\Box a \odot_2 \langle a, baa\rangle = aaabaaa$, which means $\langle aab, a\rangle \doteq_{\{aaabaaa\}} \langle a, baa\rangle$, but actually $\langle aab, a\rangle \not\doteq_{\{aaabaaa\}} \langle a, baa\rangle$ since $a\Box aa\Box \odot_2 \langle aab, a\rangle = aaabaaa$ and $a\Box aa\Box \odot_2 \langle a, baa\rangle = aaaabaa$. This argument also implies that $\{\, a^n b a^n \mid n \geq 1 \,\}$ is not 2D-substitutable, which is still (1D-)substitutable.

An interesting MCFL which is 2D-substitutable is MIX, the Bach language:

$$\mathsf{MIX} = \{\, w \in \{a,b,c\}^* \mid |w|_a = |w|_b = |w|_c \,\},$$

where $|w|_a$ denotes the number of occurrences of a in w. Salvati [48] showed that MIX is in MCFL$(2,2)$, while Kanazawa and Salvati [32] showed that it is not a tree-adjoining language. It is easy to see that MIX is indeed 2D-substitutable, since $\mathbf{u} \doteq_{\mathsf{MIX}} \mathbf{v}$ iff $\mathbf{u} \equiv_{\mathsf{MIX}} \mathbf{v}$ iff $|\mathbf{u}|_a - |\mathbf{u}|_b = |\mathbf{v}|_a - |\mathbf{v}|_b$ and $|\mathbf{u}|_b - |\mathbf{u}|_c = |\mathbf{v}|_b - |\mathbf{v}|_c$. Joshi [29] suggested that MIX should be excluded from a family of MCS languages for its complete free word order of letters cannot be considered as 'limited' cross-

serial dependencies. Yet MIX seems to be quite a simple language from the learnability point of view.

8.4 MAT result

The other learning algorithms presented in Sections 5 and 6 can also be translated in the same way as the case of substitutable languages and we obtain diverse classes of learnable MCFGs. While the pD-substitutability with $p \geq 2$ is stronger than the original 1D-substitutability and accordingly the obtained learnable MCFGs are very much restricted, the subclasses of MCFGs defined by those properties are indeed larger than the corresponding classes of CFGs.

8.4.1 Congruential MCFGs

The MAT learnability result of congruential CFGs presented in Section 5.1 can be translated as follows [68]. We say that an MCFG G is *congruential* if for every nonterminal N and any elements $\mathbf{u}, \mathbf{v} \in \mathscr{L}(G,N)$, we have $\mathbf{u} \equiv_{\mathscr{L}(G)} \mathbf{v}$. By definition, every congruential CFG is a congruential MCFG.

The grammar G_{SG} of Example 1 is congruential. The languages of respective nonterminals are

$$\begin{aligned}
\mathscr{L}(G_{\mathrm{SG}},S) &= \{\, cn_a^i n_d^j v_a^i v_d^j \mid i,j \geq 1 \,\},\\
\mathscr{L}(G_{\mathrm{SG}},P) &= \{\, (n_a^i, v_a^i) \mid i \geq 1 \,\},\\
\mathscr{L}(G_{\mathrm{SG}},Q) &= \{\, (n_d^j, v_d^j) \mid j \geq 1 \,\},
\end{aligned}$$

and for all $cn_a^i n_d^j v_a^i v_d^j \in \mathscr{L}(G_{\mathrm{SG}},S)$, $(n_a^i, v_a^i) \in \mathscr{L}(G_{\mathrm{SG}},P)$ and $(n_d^j, v_d^j) \in \mathscr{L}(G_{\mathrm{SG}},Q)$, their context sets are

$$\begin{aligned}
(cn_a^i n_d^j v_a^i v_d^j)^{\triangleright} &= \{\Box\},\\
(n_a^i, v_a^i)^{\triangleright} &= \{\, c\Box n_d^j \Box v_d^j \mid j \geq 1 \,\},\\
(n_d^j, v_d^j)^{\triangleright} &= \{\, cn_a^i \Box v_a^i \Box \mid i \geq 1 \,\}.
\end{aligned}$$

The MAT learner for congruential MCFGs constructs a grammar in a way similar to the one for pD-substitutable MCFGs, except the condition for chain rules. Now we have an membership oracle and all the entries of the OTs are fulfilled. Let K_d and F_d be the finite sets of d-words and d-contexts, which label the rows and columns of the OT for dimension d, respectively. The set of nonterminals of dimension d is $V_d = \{\, [\![\mathbf{v}]\!] \mid \mathbf{v} \in K_d \,\}$, where what we would like $[\![\mathbf{v}]\!]$ to generate are again $\mathbf{u}$ such that $\mathbf{u} \equiv_{L_*} \mathbf{v}$. Decomposition rules are constructed in exactly the same manner as before: we have $[\![\mathbf{v}]\!](\alpha) :- [\![\mathbf{v}_1]\!](\mathbf{x}_1), \ldots, [\![\mathbf{v}_k]\!](\mathbf{x}_k)$ if it is eligible for a rule of an MCFG in $\mathrm{MCFG}(p,q)$, and $\alpha[\mathbf{v}_1, \ldots, \mathbf{v}_k] = \mathbf{v}$. The criterion for a chain rule is

different. For $\mathbf{u},\mathbf{v} \in K_d$, we have the chain rule of the form $[\![\mathbf{u}]\!](\mathbf{x}) :- [\![\mathbf{v}]\!](\mathbf{x})$ if and only if $\mathbf{u}^{\triangleright} \cap F_d = \mathbf{v}^{\triangleright} \cap F_d$.

We again have the monotonicity properties. If we increase the rows in the table, no existing rules will be removed but new nonterminals and decomposition rules will be added. On the other hand, if we increase the columns in the table, no new rules will be added but possibly some chain rules will be removed.

For example, according to the OT in Table 7, we have a chain rule

$$[\![\langle n_d, v_d\rangle]\!](x_1,x_2) :- [\![\langle n_a n_d, v_a v_d\rangle]\!](x_1,x_2)$$

since $\langle n_d, v_d\rangle^{\triangleright} \cap F_2 = \langle n_a n_d, v_a v_d\rangle^{\triangleright} \cap F_2 = \{cn_a\Box n_d v_a\Box v_d\}$. If, however, $cn_a n_d\Box v_a v_d\Box$ is added to F_2, this chain rule should be discarded since $cn_a n_d\Box v_a v_d\Box \odot_2 \langle n_d, v_d\rangle \in L_{\mathrm{SG}}$ but $cn_a n_d\Box v_a v_d\Box \odot_2 \langle n_a n_d, v_a v_d\rangle \notin L_{\mathrm{SG}}$.

On the other hand if we add $\langle n_d n_d, v_d v_d\rangle$ to K_2, we obtain a new chain rule

$$[\![\langle n_d, v_d\rangle]\!](x_1,x_2) :- [\![\langle n_d n_d, v_d v_d\rangle]\!](x_1,x_2),$$

which will never been removed. See Table 8.

Table 8 Expansion of the OT for dimension 2 in Table 7

2D	$cn_a\Box n_d v_a\Box v_d$	$c\Box n_a n_d\Box v_a v_d$	$cn_a\Box n_d\Box$	$cn_a n_d\Box v_a v_d\Box$
$\langle n_a, v_a\rangle$	1	1	0	0
$\langle n_d, v_d\rangle$	1	0	0	1
$\langle n_a n_d, v_a v_d\rangle$	1	0	0	0
$\langle n_a, n_d v_a v_d\rangle$	0	0	0	0
$\langle n_a, v_a v_a v_d\rangle$	0	0	1	0
$\langle n_d n_d, v_d v_d\rangle$	1	0	0	1

8.4.2 C-deterministic MCFGs

One can easily define c-deterministic MCFGs as well according to the translation framework argued in Section 8.2, though no preceding work has done it explicitly. A context of a nonterminal N of a CFG is defined through the top-down rewriting derivation process: $l\Box r$ is a context of N if $S \overset{*}{\Rightarrow} lNr$. In the case of an MCFG, we say that $\mathsf{u} \in \mathscr{C}_d$ is a *context* of a nonterminal $N \in V_d$ if it is generated from an initial symbol using a special rule $N(\underbrace{\Box,\dots,\Box}_{d \text{ times}}) :-$ just once [63]. We then say that an MCFG G is *c-deterministic* if for every nonterminal N of G, every context u of N characterises $\mathscr{L}(G,N)$, i.e., $\{\mathsf{u}\}^{\triangleleft} = \mathscr{L}(G,N)$ (or weakly $\{\mathsf{u}\}^{\triangleleft} \equiv_{\mathscr{L}(G)} \mathscr{L}(G,N)$). By generalising Shirakawa and Yokomori's algorithm for c-deterministic CFGs, one can obtain an analogous MAT algorithm for c-deterministic MCFGs in $\mathrm{MCFG}(p,q)$. The grammar G_{SG} itself is not c-deterministic, but a slight modification satisfies the

definition:

$$\begin{aligned} S(cx_1n_an_dy_1x_2v_av_dy_2) &:\!- P(x_1,x_2),Q(y_1,y_2)\,; \\ P(n_ax_1,v_ax_2) &:\!- P(x_1,x_2)\,; \\ Q(n_dx_1,v_dx_2) &:\!- Q(x_1,x_2)\,; \\ P(\lambda,\lambda) &:\!-\,; \\ Q(\lambda,\lambda) &:\!-\,. \end{aligned}$$

8.5 Finite kernel property and finite context property

Yoshinaka [64] has given the MCFG counterpart of the learning of CFGs with the 1-FKP [17] and Clark and Yoshinaka's [18] result have established the learning of MCFGs with the k-FCP. The definitions of the FKP and FCP for MCFGs are now obvious. We say that an MCFG G has the k-FKP if every nonterminal N of dimension d admits a finite d-word set $\mathbf{S}_N \subseteq \mathscr{S}_d$ such that

- $|\mathbf{S}_N| \leq k$,
- $\mathbf{S}_N^{\triangleright} = \mathscr{L}(G,N)^{\triangleright}$.

We say that an MCFG G has the k-FCP if every nonterminal N of dimension d admits a finite d-context set $\mathsf{C}_N \subseteq \mathscr{C}_d$ such that

- $|\mathsf{C}_N| \leq k$,
- $\mathsf{C}_N^{\triangleleft} = \mathscr{L}(G,N)$.

Learners for MCFGs with the k-FKP in MCFG(p,q) and for those with the k-FCP are designed in a way similar to the ones for CFGs with the k-FKP and with the k-FCP, respectively, with the same straightforward translation technique presented in the previous subsections.

One can combine those primal and dual approaches, of course.

9 Discussion

9.1 Distributional learning beyond MCFGs and CFGs

Distributional learning has been applied recently to learning problems beyond the CFGs and MCFGs that we consider in this chapter; we briefly review some of these approaches here.

We also have the extension to Parallel Multiple Context Free Grammars (PMCFGs) [18, 19]; these grammars include a copying operation which allows them to represent some phenomena like reduplication. Two other extensions using nonstandard

formalisms have also been proposed, Distributional Lattice Grammars [11] and Binary Feature Grammars [17] These two formalisms use a limited form of conjunction; it thus seems possible to combine these results with the PMCFG formalism to have a formalism that include copying and conjunction.

Synchronous CFGs that are used for modeling transduction can easily be modeled by MCFGs, but if we assume the transduction is a function this can simplify the learning problem. There is an extension to learning string transductions along the lines of the well-known OSTIA [44] algorithm using a very limited class of synchronous CFGs called Inversion Transduction Grammars [13]; these learn from input output pairs.

All of these approaches have studied string languages: the objects being modeled are strings. These approaches have also been extended beyond string languages to learning languages over other types of objects such as trees [33], and it also has been applied to graphs with an intended application in computer vision [21], and to sentence/meaning pairs using Abstract Categorial Grammars [69].

Finally, all of these learning algorithms use only a weak notion of convergence: the learner must converge to a hypothesis that generates the right language considered as a set of strings. A stronger notion of convergence requires that the hypothesis be isomorphic to the target grammar: in other words that the learner learns a grammar that generates not just the right set of strings but the right set of structures. Such an algorithm is presented in [15].

9.2 Conclusion

We have reviewed a wide spectrum of algorithms using distributional learning techniques: it is clear that the methods we have studied here do not exhaust the range of application of this approach. One important point is that from a learnability point of view, CFGs are just a special case of MCFGs. While there is a significant difference between regular inference and CFG inference, there seem to be no theoretically interesting differences between CFGs and MCFGs. Every learning result for CFGs can be converted into a corresponding result for MCFGs.

References

1. Adriaans, P.: Learning shallow context-free languages under simple distributions. Tech. Rep. ILLC Report PP-1999-13, Institute for Logic, Language and Computation, Amsterdam (1999)
2. Angluin, D.: Learning regular sets from queries and counterexamples. Information and Computation **75**(2), 87–106 (1987)
3. Angluin, D., Kharitonov, M.: When won't membership queries help? J. Comput. Syst. Sci. **50**, 336–355 (1995)
4. Boasson, L., Sénizergues, S.: NTS languages are deterministic and congruential. J. Comput. Syst. Sci. **31**(3), 332–342 (1985)

5. Brill, E., Magermann, D., Marcus, M., Santorini, B.: Deducing linguistic structure from the statistics of large corpora. In: Proceedings of the Third DARPA Workshop on Speech and Natural Language, pp. 275–282 (1990)
6. Chomsky, N.: The logical structure of linguistic theory. Ph.D. thesis, MIT (1955)
7. Chomsky, N.: Language and mind, 3rd edn. Cambridge University Press (2006)
8. Clark, A.: PAC-learning unambiguous NTS languages. In: Y. Sakakibara, S. Kobayashi, K. Sato, T. Nishino, E. Tomita (eds.) Grammatical Inference: Algorithms and Applications, *Lecture Notes in Computer Science*, vol. 4201, pp. 59–71. Springer Berlin Heidelberg (2006)
9. Clark, A.: A learnable representation for syntax using residuated lattices. In: Proceedings of the 14th Conference on Formal Grammar. Bordeaux, France (2009). URL papers/alexcFG2009.pdf
10. Clark, A.: Distributional learning of some context-free languages with a minimally adequate teacher. In: J. Sempere, P. García (eds.) Proceedings of ICGI, no. 6339 in LNCS, pp. 24–37. Springer (2010)
11. Clark, A.: Efficient, correct, unsupervised learning of context-sensitive languages. In: Proceedings of the Fourteenth Conference on Computational Natural Language Learning, pp. 28–37. Association for Computational Linguistics, Uppsala, Sweden (2010)
12. Clark, A.: Learning context free grammars with the syntactic concept lattice. In: J. Sempere, P. García (eds.) Grammatical Inference: Theoretical Results and Applications. Proceedings of the International Colloquium on Grammatical Inference, pp. 38–51. Springer (2010)
13. Clark, A.: Inference of inversion transduction grammars. In: Proceedings of ICML. Bellevue, Washington (2011)
14. Clark, A.: The syntactic concept lattice: Another algebraic theory of the context-free languages? Journal of Logic and Computation (2013). DOI 10.1093/logcom/ext037
15. Clark, A.: Learning trees from strings: A strong learning algorithm for some context-free grammars. Journal of Machine Learning Research **14**, 3537–3559 (2014)
16. Clark, A., Eyraud, R.: Polynomial identification in the limit of substitutable context-free languages. Journal of Machine Learning Research **8**, 1725–1745 (2007)
17. Clark, A., Eyraud, R., Habrard, A.: Using contextual representations to efficiently learn context-free languages. Journal of Machine Learning Research **11**, 2707–2744 (2010)
18. Clark, A., Yoshinaka, R.: Beyond semilinearity: Distributional learning of parallel multiple context-free grammars. In: J. Heinz, C. de la Higuera, T. Oates (eds.) Proceedings of the Eleventh International Conference on Grammatical Inference, *JMLR Workshop and Conference Proceedings*, vol. 21, pp. 84–96 (2012)
19. Clark, A., Yoshinaka, R.: Distributional learning of parallel multiple context-free grammars. Machine Learning pp. 1–27 (2013). DOI 10.1007/s10994-013-5403-2. URL http://dx.doi.org/10.1007/s10994-013-5403-2
20. Dediu, A.H., Martín-Vide, C. (eds.): Language and Automata Theory and Applications - 6th International Conference, LATA 2012, A Coruña, Spain, March 5-9, 2012. Proceedings, *Lecture Notes in Computer Science*, vol. 7183. Springer (2012)
21. Eyraud, R., Janodet, J., Oates, T.: Learning substitutable binary plane graph grammars. In: Proceedings of ICGI, vol. 21, pp. 114–128 (2012)
22. Fisher, M.J.: Grammars with macro-like productions. Ph.D. thesis, Harvard University (1968)
23. Gold, E.M.: Language identification in the limit. Information and Computation **10**(5), 447–474 (1967)
24. Harris, Z.: Distributional structure. Word **10**(2-3), 146–62 (1954)
25. van Helden, W.: Case and gender: Concept formation between morphology and syntax (II volumes). Studies in Slavic and General Linguistics. Rodopi, Amsterdam-Atlanta (1993)
26. Hotz, G., Pitsch, G.: On parsing coupled-context-free languages. Theoretical Computer Science **161**(1&2), 205–233 (1996)
27. Huybrechts, R.A.C.: The weak inadequacy of context-free phrase structure grammars. In: G. de Haan, M. Trommelen, W. Zonneveld (eds.) Van Periferie naar Kern. Foris, Dordrecht, Holland (1984)

28. Joshi, A.K.: Tree adjoining grammars: how much context-sensitivity is required to provide reasonable structural descriptions? In: D.R. Dowty, L. Karttunen, A. Zwicky (eds.) Natural Language Parsing, pp. 206–250. Cambridge University Press, Cambridge, MA (1985)
29. Joshi, A.K., Vijay-Shanker, K., Weir, D.J.: The convergence of mildly context-sensitive grammar formalisms. In: P. Sells, S.M. Shieber, T. Wasow (eds.) Foundational Issues in Natural Language Processing, pp. 31–81. MIT Press, Cambridge, MA (1991)
30. Kaji, Y., Nakanishi, R., Seki, H., Kasami, T.: The universal recognition problems for parallel multiple context-free grammars and for their subclasses. IEICE Transaction on Information and Systems **E75-D**(7), 499–508 (1992)
31. Kanazawa, M., Salvati, S.: The copying power of well-nested multiple context-free grammars. In: Language and Automata Theory and Applications, pp. 344–355. Springer (2010)
32. Kanazawa, M., Salvati, S.: Mix is not a tree-adjoining language. In: ACL (1), pp. 666–674. The Association for Computer Linguistics (2012)
33. Kasprzik, A., Yoshinaka, R.: Distributional learning of simple context-free tree grammars. In: J. Kivinen, C. Szepesvári, E. Ukkonen, T. Zeugmann (eds.) Algorithmic Learning Theory, *Lecture Notes in Computer Science*, vol. 6925, pp. 398–412. Springer (2011)
34. Keller, B., Lutz, R.: Evolutionary induction of stochastic context free grammars. Pattern Recognition **38**(9), 1393–1406 (2005)
35. Klein, D., Manning, C.D.: A generative constituent-context model for improved grammar induction. In: Proceedings of the 40th Annual Meeting of the ACL (2002)
36. Kracht, M.: The Mathematics of Language, *Studies in Generative Grammar*, vol. 63, pp. 408–409. Mouton de Gruyter (2003)
37. Kulagina, O.S.: One method of defining grammatical concepts on the basis of set theory. Problemy Kiberneticy **1**, 203–214 (1958). (in Russian)
38. Kunze, J.: Versuch eines objektivierten Grammatikmodells I, II. Z. Zeitschriff Phonetik Sprachwiss. Kommunikat **20-21** (1967–1968)
39. Langley, P., Stromsten, S.: Learning context-free grammars with a simplicity bias. In: R. López de Mántaras, E. Plaza (eds.) Machine Learning: ECML 2000, *Lecture Notes in Computer Science*, vol. 1810, pp. 220–228. Springer Berlin Heidelberg (2000)
40. Leiss, H.: Learning cfgs with the finite context property: A note on A.Clark's algorithm (2012). Manuscript
41. Luque, F.M., Infante-Lopez, G.: PAC-learning unambiguous k,l-NTS$^{\leq}$ languages. In: J.M. Sempere, P. García (eds.) Grammatical Inference: Theoretical Results and Applications, *Lecture Notes in Computer Science*, vol. 6339, pp. 122–134. Springer Berlin Heidelberg (2010)
42. Marcus, S.: Algebraic Linguistics; Analytical Models. Academic Press, N. Y. (1967)
43. Myhill, J.: Review of *On Syntactical Categories* by Yehoshua Bar-Hillel. The Journal of Symbolic Logic **15**(3), 220 (1950)
44. Oncina, J., García, P., Vidal, E.: Learning subsequential transducers for pattern recognition interpretation tasks. IEEE Transactions on Pattern Analysis and Machine Intelligence **15**, 448–458 (1993)
45. Pitt, L.: Inductive inference, DFAs, and computational complexity. In: Proceedings of 2nd Workshop on Analogical and Inductive Inference, *Lecture Notes in Computer Science*, vol. 397, pp. 18–44 (1989)
46. Rambow, O., Satta, G.: Independent parallelism in finite copying parallel rewriting systems. Theor. Comput. Sci. **223**(1-2), 87–120 (1999)
47. Sakakibara, Y.: Learning context-free grammars from structural data in polynomial time. Theoretical Computer Science **76**(2-3), 223–242 (1990)
48. Salvati, S.: MIX is a 2-MCFL and the word problem in $\mathbb{Z}^2$ is solved by a third-order collapsible pushdown automaton. Tech. Rep. Inria-00564552, version 1, INRIA (2011). URL http://hal.inria.fr/inria-00564552
49. Seki, H., Matsumura, T., Fujii, M., Kasami, T.: On multiple context-free grammars. Theoretical Computer Science **88**(2), 191–229 (1991)
50. Sénizergues, G.: The equivalence and inclusion problems for NTS languages. Journal of Computer and System Sciences **31**(3), 303–331 (1985)

51. Sestier, A.: Contribution à une théorie ensembliste des classifications linguistiques. In: Premier Congrès de l'Association Française de Calcul, pp. 293–305. Grenoble (1960)
52. Shibata, C., Yoshinaka, R.: PAC learning of some subclasses of context-free grammars with basic distributional properties from positive data. In: S. Jain, R. Munos, F. Stephan, T. Zeugmann (eds.) ALT, *Lecture Notes in Computer Science*, vol. 8139, pp. 143–157. Springer (2013)
53. Shieber, S.M.: Evidence against the context-freeness of natural language. Linguistics and Philosophy **8**, 333–343 (1985)
54. Shinohara, T.: Rich classes inferrable from positive data – length-bounded elementary formal systems. Information and computation **108**(2), 175–186 (1994)
55. Shirakawa, H., Yokomori, T.: Polynomial-time MAT learning of c-deterministic context-free grammars. Transactions of the information processing society of Japan **34**, 380–390 (1993)
56. Smullyan, R.: Theory of Formal Systems. Princeton University Press (1961)
57. Stabler, E.: Derivational minimalism. In: C. Retoré (ed.) Logical aspects of computational linguistics (LACL 1996), pp. 68–95. Springer (1997)
58. Vijay-Shanker, K., Weir, D.J.: The equivalence of four extensions of context-free grammars. Mathematical Systems Theory **27**(6), 511–546 (1994)
59. Vijay-Shanker, K., Weir, D.J., Joshi, A.K.: Characterizing structural descriptions produced by various grammatical formalisms. In: Proceedings of the 25th annual meeting of Association for Computational Linguistics, pp. 104–111. Stanford (1987)
60. Wells, R.S.: Immediate constituents. Language **23**(2), 81–117 (1947)
61. Wurm, C.: Completeness of full Lambek calculus for syntactic concept lattices. In: Proceedings of the 17th conference on Formal Grammar 2012 (FG) (2012)
62. Yoshinaka, R.: Identification in the limit of k,l-substitutable context-free languages. In: A. Clark, F. Coste, L. Miclet (eds.) ICGI, *Lecture Notes in Computer Science*, vol. 5278, pp. 266–279. Springer (2008)
63. Yoshinaka, R.: Learning mildly context-sensitive languages with multidimensional substitutability from positive data. In: R. Gavaldà, G. Lugosi, T. Zeugmann, S. Zilles (eds.) ALT, *Lecture Notes in Computer Science*, vol. 5809, pp. 278–292. Springer (2009)
64. Yoshinaka, R.: Polynomial-time identification of multiple context-free languages from positive data and membership queries. In: J.M. Sempere, P. García (eds.) ICGI, pp. 230–244. Springer (2010)
65. Yoshinaka, R.: Efficient learning of multiple context-free languages with multidimensional substitutability from positive data. Theoretical Computer Science **412**(19), 1821–1831 (2011)
66. Yoshinaka, R.: Towards dual approaches for learning context-free grammars based on syntactic concept lattices. In: G. Mauri, A. Leporati (eds.) Developments in Language Theory, *Lecture Notes in Computer Science*, vol. 6795, pp. 429–440. Springer (2011)
67. Yoshinaka, R.: Integration of the dual approaches in the distributional learning of context-free grammars. In: Dediu and Martín-Vide [20], pp. 538–550
68. Yoshinaka, R., Clark, A.: Polynomial time learning of some multiple context-free languages with a minimally adequate teacher. In: P. de Groote, M.J. Nederhof (eds.) Formal Grammar: 15th and 16th International Conference on Formal Grammar, pp. 192–206. Springer (2012)
69. Yoshinaka, R., Kanazawa, M.: Distributional learning of abstract categorial grammars. In: S. Pogodalla, J.P. Prost (eds.) LACL, *Lecture Notes in Computer Science*, vol. 6736, pp. 251–266. Springer (2011)
70. van Zaanen, M.: ABL: Alignment-based learning. In: COLING 2000 - Proceedings of the 18th International Conference on Computational Linguistics, pp. 961–967 (2000)