

# Inferring Grammars for Mildly Context Sensitive Languages in Polynomial-Time

Tim Oates<sup>1</sup>, Tom Armstrong<sup>1</sup>, Leonor Becerra Bonache<sup>2</sup>, and Mike Atamas<sup>1</sup>

<sup>1</sup> University of Maryland Baltimore County  
Baltimore, MD 21250 USA

{oates, arm1, m39}@umbc.edu

<sup>2</sup> Rovira i Virgili University

Pl. Imperial Tarraco 1, 43005, Tarragona, Spain

leonor.becerra@estudiants.urv.es

**Abstract.** Natural languages contain regular, context-free, and context-sensitive syntactic constructions, yet none of these classes of formal languages can be identified in the limit from positive examples. Mildly context-sensitive languages are able to represent some context-sensitive constructions, those most common in natural languages, such as multiple agreement, crossed agreement, and duplication. These languages are attractive for natural language applications due to their expressiveness, and the fact that they are not fully context-sensitive should lead to computational advantages as well. We realize one such computational advantage by presenting the first polynomial-time algorithm for inferring Simple External Context Grammars, a class of mildly context-sensitive grammars, from positive examples.

## 1 Introduction

Despite the fact that every normal child masters his native language, the learning mechanisms that underly this distinctly human feat are poorly understood. The ease with which children learn language belies the underlying complexity of the task. They face a number of theoretical challenges, including apparently insufficient data from which to learn lexical semantics (Quine's "gavagai" problem [1]) or syntax (Chomsky's "argument from the poverty of the stimulus" [2]). For example, it is known that many classes of formal languages, such as regular and context-free, cannot be learned solely from positive examples, i.e., strings that are in the (regular or context-free) language to be learned [3]. This is problematic because children either do not receive negative examples (i.e., strings that are not in the language to be learned) or pay little attention when such examples are presented [4].

There are a few standard ways of avoiding these theoretical obstacles to learning syntax from positive examples. One is to assume the existence of information in addition to positive examples that comprise the training data. For example, most algorithms for learning context-free grammars from positive examples assume that each example is paired with its unlabeled derivation tree, which is the parse tree for the string from which the non-terminal labels on the interior nodes

have been removed [5,6]. An alternative is to restrict the class of languages from which the language to be learned can be drawn. A variety of subsets of the set of all regular languages are known to be learnable from positive examples [7].

These approaches to learning the syntax of formal languages are problematic when applying the resulting algorithms to learning the syntax of natural languages. For example, the additional information assumed to exist may in fact not be available to children. For example, children clearly do not receive utterances paired with unlabeled derivation trees (though see [8] for a method for inferring these trees from the sensory context). Also, natural languages exhibit some syntactic constructions that are regular, some that are context-free, and some that are context-sensitive. Therefore, restricted classes of regular or context-free languages have insufficient expressiveness.

In this paper, we present a polynomial-time algorithm for learning Simple External Contextual ( $\mathcal{SEC}_p$ ) languages from positive data.  $\mathcal{SEC}_p$  languages are *mildly context sensitive*. That is, they can express the context-sensitive syntactic constructions that are most prevalent in natural languages, such as multiple agreement, crossed agreement, and duplication [9]. In addition, neither the regular languages nor the context-free languages are a proper subset of the  $\mathcal{SEC}_p$  languages. That is, there exist regular and context-free languages for which no corresponding  $\mathcal{SEC}_p$  language exists.

Because expressiveness and computational tractability are typically inversely related, the goal of most mildly context-sensitive languages is to provide sufficient context sensitivity for natural language applications while keeping open the possibility of polynomial-time algorithms for standard tasks such as parsing and grammar induction. Currently, the most efficient algorithm known for learning  $\mathcal{SEC}_p$  languages from positive examples is exponential [10]. This paper presents the first polynomial-time algorithm for learning  $\mathcal{SEC}_p$  languages from positive data, and proves its correctness, thereby realizing the computational advantages of the restricted expressiveness of  $\mathcal{SEC}_p$  languages.

The remainder of this paper is organized as follows. Section 2 describes External Contextual grammars and Simple External Contextual grammars in more detail. Section 3 describes our algorithm for learning  $\mathcal{SEC}_p$  grammars from positive data. Section 4 establishes various theorems about the correctness of the algorithm. Finally, Section 5 concludes and points to future work.

## 2 External Contextual Grammars

This section reviews Mildly Context-Sensitive grammars beginning with External Contextual ( $\mathcal{EC}_p$ ) grammars and then a subset of them called Marcus Simple Many-Dimensional External Contextual ( $\mathcal{SEC}_p$ ) grammars.

An  $\mathcal{EC}_p$  grammar is a three-tuple  $G = (\Sigma, A, N)$ ,  $\Sigma$  is the alphabet of  $G$ ,  $A$  is a finite set of  $p$ -words over  $\Sigma$  called the *axioms* of  $G$ , and  $N$  is a finite set of productions. A production is a pair,  $(S, C)$  where  $S$  is a set of *selectors*, strings over  $\Sigma^*$ , and  $C$  is a  $p$ -context. The parameter  $p \in \mathbb{N}$  specifies the dimensionality of the  $p$ -words and  $p$ -contexts.

A  $p$ -word  $x$  over  $\Sigma$  is a  $p$ -dimensional vector whose components are strings over  $\Sigma$ , i.e.,  $x = (x_1, x_2, \dots, x_p)$ , where  $x_i \in \Sigma^*$ . A  $p$ -context  $c$  over  $\Sigma$  is a  $p$ -dimensional vector whose components are contexts over  $\Sigma$ , i.e.,  $c = [c_1, c_2, \dots, c_p]$  where  $c_i = (u_i, v_i)$ ,  $u_i, v_i \in \Sigma^*$ ,  $1 \leq i \leq p$ . We denote vectors of words with parentheses, and vectors of contexts with square brackets. Strings in the language of the grammar are derived by *applying* contexts to the axioms and subsequent strings (wrapping elements of  $p$ -contexts around corresponding elements of  $p$ -words) if an element of the selector set matches a substring of the derivation (e.g. maximal matching) for the context. For the remainder of the paper, we consider languages with maximal matching selector sets containing  $\Sigma^*$ . The number of contexts applied in a derivation of a string is called the *depth* of the string.

Let  $x = (x_1, x_2, \dots, x_p)$  and  $y = (y_1, y_2, \dots, y_p)$  be two  $p$ -words over  $\Sigma$ . By definition,  $x \Rightarrow_G y$  iff  $y = (u_1x_1v_1, u_2x_2v_2, \dots, u_px_pv_p)$  for some  $p$ -context  $c = [(u_1, v_1), (u_2, v_2), \dots, (u_p, v_p)] \in C$ .

The  $\mathcal{EC}_p$  language family is superfinite (a language class is superfinite if it contains every finite language and at least one infinite language) and therefore is not learnable in the limit from positive data [3]. All finite languages can be represented in  $\mathcal{EC}_p$  by a grammar consisting of a finite set of axioms and an empty set of productions. We restrict the class of languages to Simple External Contextual ( $\mathcal{SEC}_p$ ) grammars where the axiom set contains a single  $p$ -word over  $\Sigma$ .  $\mathcal{SEC}_p$  grammars are further specified by a parameter  $q$ , the degree of the grammar or the number of productions, and can be abbreviated by  $\mathcal{SEC}_p^q$ .

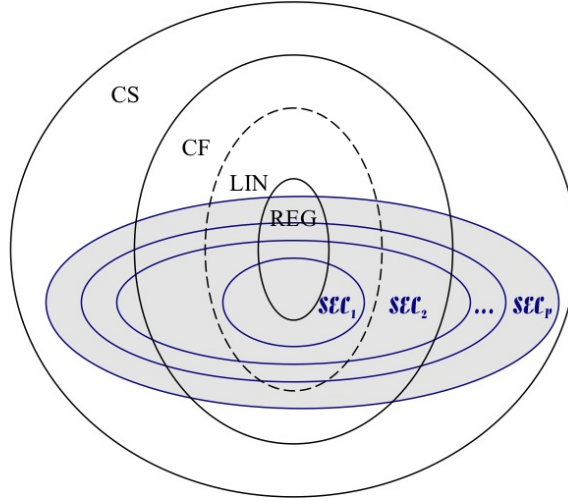
The  $\mathcal{SEC}_p$  family spans multiple language classes in the Chomsky hierarchy (see figure 1). Both are strictly contained in the class of context-sensitive languages and are incomparable with the classes of context-free and regular languages. Yet they can express the context-sensitive syntactic constructions that are most prevalent in natural languages.

Consider the  $\mathcal{SEC}_2^1$  grammar  $G$ , the three-tuple,  $(\{a, b, c\}, \{(\lambda, \lambda)\}, \{(\{w|w \in \Sigma^*\}, [(a, b), (c, \lambda)])\})$ . The derivation of the string  $aabbcc$  is the application of the context twice to the axiom:  $(\lambda, \lambda) \vdash (a\lambda b, c\lambda\lambda) \vdash (aa\lambda bb, cc\lambda\lambda)$ . Finally the internal strings are concatenated resulting in  $aabbcc$ . Note that  $L(G) = \{a^n b^n c^n \mid n \geq 0\}$ .

### 3 Induction Algorithm

The input to our algorithm is a set of strings,  $S$ , in an  $\mathcal{SEC}_p^q$  language where the dimensionality,  $p$ , and degree,  $q$ , are known.  $S$  must contain at least all strings in the language up to a depth  $d$  (the value of  $d$  is discussed later). The shortest string in  $S$  is the concatenation of the elements in the axiom. However, there are multiple possibilities for what the actual axiom is (depending on the dimensionality of the language). From the shortest string, a set of possible axioms,  $A$ , is generated. For example, if  $p=2$  and the shortest string in  $S$  is  $aa$ , then the set of possible axioms is  $\{(\lambda, aa), (a, a), (aa, \lambda)\}$ .

Each subsequent string in  $S$  is derived beginning with the axiom and some number of context applications. For every  $a_j \in A$ , every further element of  $S$



**Fig. 1.** The  $\mathcal{SEC}_p$  family spans multiple language classes in the Chomsky hierarchy

is processed assuming that its derivation is of depth one (i.e. a set of candidate contexts can be generated by inferring contexts that derive a string from the axiom in one step) and the contexts stored in a set  $P$ . Next, all ways of choosing  $q$  elements from  $P$  are paired with the axiom to form grammars. Each candidate grammar,  $G'$ , is used to derive all the strings in  $L(G')$  up to depth  $d$ . If  $L^d(G') \not\subseteq S$  or  $G'$  cannot parse all of the strings in  $S - L^d(G')$ , then  $G'$  is removed from the set of candidate grammars. Additional strings in the input set increase the number of candidate grammars, but will not exclude grammars. Later, we prove that if a grammar,  $G$ , generates only  $L^d(G)$  in derivations up to depth  $d$ , then  $G$  generates  $L(G)$ .

Consider the following example, an input set  $S = \{a, aaa, bab, ababa, baaab, aaaaa, bbabb\}$  for a language generated by an  $\mathcal{SEC}_1^2$  grammar. The input is exhaustive up to depth 2. The shortest string in  $S$  is  $a$  and, because  $p = 1$ ,  $(a)$  is the only possible axiom. Assuming that  $aaa$  was generated by a single application of a context to  $(a)$ , the possible contexts that can generate the input set are  $\{[(\lambda, aa)], [(a, a)], [(aa, \lambda)]\}$ . This is done again for  $bab$ , resulting in  $\{[(b, b)]\}$ , and for the remainder of the strings in the input set. All of the possible pairs of contexts are then put in a set  $G$ . The only  $g_i \in G$  that can generate  $S$  and only generate  $S$  is  $\{[(a, a)], [(b, b)]\}$ ; the rest are discarded. In this example, there was only one possible axiom. If this language had been  $\mathcal{SEC}_2^2$ , then the possible axioms would have been  $\{[(\lambda, a)], [(a, \lambda)]\}$ . Unlike this example, input sets can be generated by several grammars, some of which have distinct axioms.

The running time of the algorithm (see figure 2) depends upon the dimensionality and degree of the grammar, the size of the axiom, and the maximum depth of the input set. Given the shortest string, the axiom, in the input set, there are  $(|axiom| + 1)^{p-1}$  possible axioms (inserting  $p-1$  separations into the shortest string). Given a string  $s_i$ , all of the possible contexts to

```

INFER-GRAMMAR( $d, p, q, inputSet$ )
1   $shortest \leftarrow \text{FIND-SHORTEST}(inputSet)$ 
2   $axioms \leftarrow \text{GENERATE-POSSIBLE-AXIOMS}(p, shortest)$ 
3   $finalGrammars \leftarrow \{\}$ 
4   $result \leftarrow \{\}$ 
5
6  for  $axiom \in axioms$ 
7  do  $possibleContexts \leftarrow \{\}$ 
8      for  $i \in inputSet$ 
9      do  $possibleContexts.push[\text{GENERATE-POSSIBLE-CONTEXTS}(i, axiom)]$ 
10
11      $possibleGrammars \leftarrow \text{GENERATE-ALL-CONTEXT-COMBINATIONS}(possibleContexts)$ 
12     for  $j \in possibleGrammars$ 
13     do if  $\text{CAN-GENERATE-ALL-STRINGS}(j, axiom)$ 
14         then  $finalGrammars.push[\{j, axiom\}]$ 
15
16 for  $k \in finalGrammars$ 
17 do if  $\text{GENERATES-EXACT-LIST}(d, k, inputSet)$ 
18     then  $result.push[k]$ 
19 RETURN}(result)
    
```

**Fig. 2.** Pseudocode of the inference algorithm

derive  $s_i$  with one application is at most  $(|s| + 1)^{2p}$ . Given the set of all contexts from above, we can pick all sets of contexts of size  $q$ , or  $\binom{(|s|+1)^{2p}}{q}$ . Given a candidate grammar, all of the strings that it generates up to a depth of  $d$  can be enumerated and compared to elements in the input set in at most  $(\sum_{i=1}^d q^i) * |inputSet|$  steps. Therefore, the cost of the nested loop is bounded by  $(\sum_{i=1}^d q^i) * |inputSet| * \binom{(|s|+1)^{2p}}{q} * (|axiom| + 1)^{p-1}$ .

What we will next prove is that the size of the input set is finite and small (in some cases depth 3 suffices), and that with a small dimension, degree and axiom, the problem of inferring the grammar with our algorithm becomes tractable.

## 4 Proof of the Existence of a Characteristic Sample

We prove that a finite sample of the language, a characteristic sample, is sufficient to infer a grammar for the language. Let  $G$  and  $G'$  be minimal  $\mathcal{SEC}_p$  grammars as follows:

$$\begin{aligned}
 G &= (p, q, A, C_1, C_2, \dots, C_q) \\
 G' &= (p', q', A', C_1', C_2', \dots, C_q')
 \end{aligned}$$

$A$  is the axiom and  $C_i$  is a context. Assume that  $p = p'$  and  $q = q'$ . A grammar is minimal if there is no other grammar with smaller values of  $p$  or  $q$  with the same language.

For a vector of strings  $X$ , let  $\text{CONCAT}(X)$  equal the string obtained by concatenating all strings in  $X$  in the order in which they occur. We use the symbol  $/$  to denote application of a production. For example,  $C_i/A$  is the  $p$ -vector

obtained by applying context  $C_i$  to axiom  $A$ . Let  $L^d(G)$  be the set of strings in  $L(G)$  that require  $d$  or fewer derivation steps.  $L^0(G) = \{\text{CONCAT}(A)\}$ . We want to prove the following theorem.

**Theorem 1.**  *$L(G) = L(G')$  and  $\text{CONCAT}(C_i/A) \neq \text{CONCAT}(C_j/A)$  for  $i \neq j$  and  $1 \leq i, j \leq q$  if and only if  $L^d(G) = L^d(G')$  for an appropriately small  $d$  and minimal grammars,  $G$  and  $G'$ .*

We now prove a series of lemmas and theorems to this end.

**Lemma 1.** *If  $L(G) = L(G')$  then  $\text{CONCAT}(A) = \text{CONCAT}(A')$ .*

*Proof.* Suppose  $L(G) = L(G')$  but  $\text{CONCAT}(A) \neq \text{CONCAT}(A')$ . The shortest string in  $L(G)$  is  $\text{CONCAT}(A)$  because all other contexts, being non-empty, add characters when applied to the axiom. If  $\text{CONCAT}(A) \neq \text{CONCAT}(A')$  then the shortest string in  $L(G)$  is not equal to the shortest string in  $L(G')$ , which means that  $L(G) \neq L(G')$ . This is a contradiction, so  $\text{CONCAT}(A)$  must equal  $\text{CONCAT}(A')$ .  $\square$

**Lemma 2.** *If  $L(G) = L(G')$  and  $\text{CONCAT}(C_i/A) \neq \text{CONCAT}(C_j/A)$  for  $i \neq j$  and  $1 \leq i, j \leq q$ , then there is a bijective mapping (one-to-one and onto)  $f$  from contexts in  $G$  to contexts in  $G'$  such that  $\text{CONCAT}(C_i/A) = \text{CONCAT}(C'_{f(i)}/A')$ . That is, for each  $C_i$  in  $G$  there exists precisely one  $C'_j$  in  $G'$  such that  $\text{CONCAT}(C_i/A) = \text{CONCAT}(C'_j/A')$ , and for each  $C'_i$  in  $G'$  there exists precisely one  $C_j$  in  $G$  such that  $\text{CONCAT}(C'_i/A') = \text{CONCAT}(C_j/A)$ .*

*Proof.* Suppose that no bijective mapping,  $f$ , from contexts in  $G$  to contexts in  $G'$  such that  $\text{CONCAT}(C_i/A) = \text{CONCAT}(C'_{f(i)}/A')$  exists. Without such a mapping, there is a string in  $L(G)$  and  $L(G')$ ,  $S$ , that is the least criminal, the shortest string violating this mapping. This string must have a derivation of depth one in  $G$ , else the mapping would hold. The derivation of  $S$  in  $G$  is  $\text{CONCAT}(C_i/A)$  and in  $G'$  is  $\text{CONCAT}(C'_l/\dots/C'_s/A')$ . That is, the derivation of  $S$  in  $G$  requires one context whereas the derivation of  $S$  in  $G'$  requires at least two contexts. All contexts used in the derivation of  $S$  in  $G'$  are used to derive shorter strings contained in both  $L(G)$  and  $L(G')$  (e.g.  $\text{CONCAT}(C'_l/A') = \text{CONCAT}(C_j/A)$  for some  $l$  and  $j$ ). These depth one strings, shorter than  $S$ , form a bijective mapping,  $h$ , because they are not the least criminal string. The characters contributed by  $C_i$  are the same as  $C'_l, \dots, C'_s$ . Each of  $C'_l, \dots, C'_s$  map to a context in  $G$  that contribute the same characters, therefore  $C_{h(l)}, \dots, C_{h(s)}$  contribute the same characters as  $C_i$ . The string  $S$  is derived in  $G$  using either  $C_i$  or  $C_{h(l)}, \dots, C_{h(s)}$  thus making  $C_i$  superfluous and  $G$  non-minimal, a contradiction.  $\square$

**Lemma 3.** *If  $L(G) = L(G')$  and  $\text{CONCAT}(C_i/A) \neq \text{CONCAT}(C_j/A)$  for  $i \neq j$  and  $1 \leq i, j \leq q$ , then strings in  $L(G)$  and  $L(G')$  have the same derivations up to renaming of contexts.*

*Said differently, there is a bijective mapping (one-to-one and onto)  $f$  from contexts in  $G$  to contexts in  $G'$  such that if string  $S$  has derivation  $D$  in  $G$  then the derivation of  $S$  in  $G'$  can be obtained by applying the mapping to each context in  $D$ . That is, if the derivation of  $S$  in  $L(G)$  is  $C_i/C_j/\dots/C_k/A$  then the derivation of  $S$  in  $G'$  is  $C'_{f(i)}/C'_{f(j)}/\dots/C'_{f(k)}/A'$*

*Proof.* All derivations begin with the axiom A. Consider one-step derivations of the form  $C_i/A$ , which means context  $C_i$  is applied to axiom A. Let  $S_0 = \text{CONCAT}(A)$  and  $S_1 = \text{CONCAT}(C_i/A)$ . The only difference between  $S_0$  and  $S_1$  is that  $S_1$  contains, in addition to the characters in A, the characters in  $\text{CONCAT}(C_i)$  in the same linear order, though they may not be contiguous. No context other than  $C_i$  can add those characters in one step because  $\text{CONCAT}(C_i/A) \neq \text{CONCAT}(C_j/A)$  for  $i \neq j$  and  $1 \leq i, j \leq q$ .

By lemma 2, there must exist a context  $C'_j$  in  $G'$  such that  $\text{CONCAT}(C_i/A) = \text{CONCAT}(C'_j/A)$ . For the sake of simplicity, we assume that context is  $C'_i$ . Because  $C_i$  is unique among the contexts in  $G$ , so must  $C'_i$  be among the contexts in  $G'$  (by lemma 2). Therefore, the only way to derive  $S_1$  in  $G'$  is  $C'_i/A'$ , and all strings in  $L^1(G)$  have the same derivations in  $L(G')$  up to renaming of contexts.

Suppose, inductively, that this holds for derivations up to depth  $d$ . Let  $S_d$  be any string in  $L(G)$  that requires  $d$  derivations and let  $S_{d+1}$  be any string in  $L(G)$  derivable from  $S_d$  by applying one more context. By the induction assumption, the derivation of  $S_d$  is the same (up to renaming of contexts) in  $G$  and  $G'$ . Whatever context is added in  $G$  to  $S_d$  to obtain  $S_{d+1}$ , its analog in  $G'$  must be applied to derive  $S_{d+1}$  because no other context can add the required characters.  $\square$

**Theorem 2.** *If  $L(G) = L(G')$  and  $\text{CONCAT}(C_i/A) \neq \text{CONCAT}(C_j/A)$  for  $i \neq j$  and  $1 \leq i, j \leq q$ , then  $L^d(G) = L^d(G')$  for any depth  $d$ .*

*Proof.* Suppose  $L(G) = L(G')$  but  $L^d(G) \neq L^d(G')$  for some depth  $d$ . Let  $d$  be the smallest such depth for which this is the case. Then either there exists an  $S$  in  $L^d(G)$  that requires  $d$  derivation steps that is not in  $L^d(G')$ , or vice versa. Without loss of generality, we focus on the former case. Because  $L(G) = L(G')$ , there must be some depth  $d' > d$  such that  $S$  is in  $L^{d'}(G')$ . However, by lemma 3, if there is a depth  $d$  derivation of  $S$  in  $G$ , there is a depth  $d$  derivation of  $S$  in  $G'$ . This is a contradiction, so  $L^d(G) = L^d(G')$  for all depths  $d$ .  $\square$

**Theorem 3.** *There exists a finite  $d$ , such that for all grammars  $G$  and  $G'$ , if  $L^d(G) = L^d(G')$  then  $L(G) = L(G')$ .*

*Proof.* Let the proposition  $P$  be equivalent to  $L^d(G) = L^d(G')$  and let the proposition  $Q$  be equivalent to  $L(G) = L(G')$ . Then the theorem can be stated formally as

$$(\forall G)(\forall G')(\exists d)[P \rightarrow Q] \quad (1)$$

To prove that this is a tautology, it suffices to show that the negation of (1) is a contradiction. The negation can be written

$$(\exists G)(\exists G')(\forall d)[P \wedge \neg Q] \quad (2)$$

$$(\exists G)(\exists G')(\forall d)[(L^d(G) = L^d(G')) \wedge (L(G) \neq L(G'))] \quad (3)$$

The last equation states that there exist grammars  $G$  and  $G'$  such that at any finite depth the strings produced by either grammar are equal but the languages are not. With any finite language, this is clearly a contradiction. In an

infinite language, the set of all possible depths in that language is clearly denumerable. The only restriction on  $d$  is that it belong to  $\mathbb{N}$ , making the set of all  $d$  countably infinite. For both the set of all  $d$  and the set of all possible depths, a bijection from  $\mathbb{N}$  exists, so they must be of the same size. Thus, comparing the languages at all finite  $d$  is equivalent to comparing the whole language, making (3) a contradiction. Since the negation of (1) is a contradiction, (1) must be a tautology. Therefore, there exists a finite  $d$  such that for all grammars  $G$  and  $G'$ , if  $L^d(G) = L^d(G')$  then  $L(G) = L(G')$ .  $\square$

Theorems 2 and 3 collectively prove Theorem 1. Theorem 1 and its proof establish that there exists a finite depth  $d$  such that the strings in  $L^d(G)$  for any  $\mathcal{SEC}_p^q$  grammar are a characteristic sample, i.e., they uniquely identify the language. In the complexity analysis of our learning algorithm,  $d$  is a constant, but as with any constant hidden inside  $O()$  notation, if the constant is large it can make the algorithm practically infeasible. We now show that for  $\mathcal{SEC}_1^1$  languages  $d = 2$  suffices, and for  $\mathcal{SEC}_1^q$  languages  $d = 3$  suffices. We conjecture that  $d = 3$  is sufficient for arbitrary  $p$  and  $q$ , and are working on the proof.

We begin with the following lemma:

**Lemma 4.** *Let  $A, B \in \Sigma^+$ . If  $AB = BA$ , then there exists  $p \in \Sigma^+$  and integers  $x, y > 0$  such that  $A = p^x$  and  $B = p^y$ .*

*Proof.* Let  $n = |A|$  and  $m = |B|$ . Let  $S_i$  denote the  $i^{\text{th}}$  character in string  $S$ , where the index of the first character is 1, and  $S_{i,j}$  denote the sub-string of  $S$  from characters  $i$  through  $j$ . Trivially,  $AB_i = BA_i$  because  $AB = BA$ . That is, the  $i^{\text{th}}$  character in the string  $AB$  must equal the  $i^{\text{th}}$  character in the string  $BA$ . Also,  $AB_i = BA_{(i+m) \bmod (n+m)}$  because regardless of whether  $A$  or  $B$  contributes the  $i^{\text{th}}$  character in  $AB$ , that same character contributed by the same string occurs in position  $(i+m) \bmod (n+m)$  in  $BA$ . From this we can conclude that  $AB_i = BA_i = BA_{(i+km) \bmod (n+m)}$  for positive integers  $k$ . Therefore, for  $g = \text{GCD}(n, m)$  every  $g^{\text{th}}$  character in  $AB$  is the same, and thus for  $p = A_{1,g}$ ,  $x = n/g$ , and  $y = m/g$  it is the case that  $A = p^x$  and  $B = p^y$ .  $\square$

The following two theorems say that the strings in  $L^2(G)$  are a characteristic sample for  $\mathcal{SEC}_1^1$  grammars and that the strings in  $L^3$  are a characteristic sample for  $\mathcal{SEC}_q^1$  grammars, respectively.

**Theorem 4.** *Let  $G$  and  $G'$  be minimal  $\mathcal{SEC}_1^1$  grammars. It is the case that  $L^2(G) = L^2(G')$  iff  $L(G) = L(G')$ .*

*Proof.* Let  $A$  and  $A'$  be the axioms of  $G$  and  $G'$ , and let  $C$  and  $C'$  be the contexts of  $G$  and  $G'$ . Because the grammars are minimal, neither  $C$  nor  $C'$  can be  $(\lambda, \lambda)$ . Therefore, the strings in  $L^2(G)$  in increasing order of length are  $\{A, C/A, C/C/A\}$ , and likewise for  $L^2(G') = \{A', C'/A', C'/C'/A'\}$ . Clearly, if  $L(G) = L(G')$  then  $L^2(G) = L^2(G')$ . It remains to show that if  $L^2(G) = L^2(G')$  then  $L(G) = L(G')$ .

It follows from  $L^2(G) = L^2(G')$  that  $A = A'$  (otherwise the two shortest strings in the languages would differ) and  $\text{CONCAT}(C/A) = \text{CONCAT}(C'/A')$

(otherwise there is no way for  $C/A = C'/A'$ ). Henceforth, we will use  $A$  to refer to the axiom of either grammar.

Because  $\text{CONCAT}(C/A) = \text{CONCAT}(C'/A')$ , it must be the case that for some  $X, Y, Z \in \Sigma^*$ , with at least one in  $\Sigma^+$ , we can write  $C = (XY, Z)$  and  $C' = (X, YZ)$ . If  $C = C'$  then  $Y = \lambda$  and the languages are equal. Consider the case where  $Y \neq \lambda$ . We need to establish that if this change impacts the generated language, this impact will be seen at or before depth 2. Note that any of  $X, Z$ , and  $A$  can be  $\lambda$ , so there are eight possible values for the depth 2 languages shown in table 1.

**Table 1.** Eight possible cases of assigning the empty string for  $L^2$

EMPTY STRINGS	$L^2(G)$	$L^2(G')$
$A, X, Z$	$\{\lambda, Y, YY\}$	$\{\lambda, Y, YY\}$
$A, X$	$\{\lambda, YZ, YYZZ\}$	$\{\lambda, YZ, YZY Z\}$
$A, Z$	$\{\lambda, XY, XYXY\}$	$\{\lambda, XY, XXY Y\}$
$A$	$\{\lambda, XYZ, XYXYZZ\}$	$\{\lambda, XYZ, XXYZY Z\}$
$X, Z$	$\{A, YA, YYA\}$	$\{A, AY, AYY\}$
$X$	$\{A, YAZ, YYAZZ\}$	$\{\lambda, AYZ, AYZY Z\}$
$Z$	$\{A, XYA, XYXYA\}$	$\{A, XAY, X XAY Y\}$
	$\{A, XYAZ, XYXYAZZ\}$	$\{A, XAYZ, X XAYZY Z\}$

It is easy to show that for each of these cases, if  $L^2(G) = L^2(G')$  then  $L(G) = L(G')$ . We demonstrate the line of reasoning with the most difficult case, which is the last in the table above and corresponds to the case in which none of  $A, X$ , or  $Z$  are empty. Note that  $C/A = XYAZ$  and  $C'/A' = XAYZ$ . Because the depth 2 languages are the same,  $XYAZ = XAYZ$ . Stripping away the common prefix and suffix of these strings yields  $YA = AY$  and, by Lemma 4 above, there exists some  $p \in \Sigma^+$  such that  $Y = p^n$  and  $A = p^m$ . Also, because  $C/C/A = XYXYAZZ = X XAYZY Z = C'/C'/A'$ , it is the case that  $YXYAZ = XAYZY$ , which can be rewritten as  $p^n X p^n p^m Z = X p^m p^n Z p^n$ . Clearly, it must be the case that  $X = p^k$  and  $Z = p^l$  for some integers  $k, l > 0$ . Therefore,  $d$  applications of context  $C$  to  $A$  yields the string  $p^{m+d(n+k+l)}$ , as do  $d$  applications of context  $C'$  to  $A$ . Therefore,  $L(G) = L(G')$ . Analogous reasoning yields the same result in the seven other cases in the table above.  $\square$

**Theorem 5.** *Let  $G$  and  $G'$  be minimal  $\text{SEC}_q^1$  grammars such that for contexts in  $G$  it is the case that  $\text{CONCAT}(C_i/A) \neq \text{CONCAT}(C_j/A)$  for  $i \neq j$  and  $1 \leq i, j \leq q$ . It is the case that  $L^3(G) = L^3(G')$  iff  $L(G) = L(G')$ .*

*Proof. (sketch)* This proof is similar to the proof of Lemma 4. Because the concatenation of the strings in the contexts of  $G$  differ, and because the languages of  $G$  and  $G'$  are the same up to depth 3, there is a bijective mapping between contexts in  $G$  and  $G'$ , at least for derivations of strings up to depth 3. Suppose there is a context  $C_i = (XY, Z)$  in  $G$ , which is mapped to context  $C'_i = (X, YZ)$  in  $G'$ . We need to show that if the movement of  $Y$  from  $C$  to  $C'$  will impact the language, such impact will appear in  $L^3$ .

Let  $C'_j = (U, V)$  be some other context in  $G'$ . This context may be unmodified from  $C_j$  in  $G$ , or the characters in  $UV$  may be split differently between the two halves of the context. In the latter case, Theorem 4 tells us that the subsets of  $L(G)$  and  $L(G')$  that involve only applications of  $C_j$  and  $C'_j$  are the same. Therefore, regardless of whether  $C_j = C'_j$  or only  $\text{CONCAT}(C_j/A) = \text{CONCAT}(C'_j/A')$ , we will write the contents of  $C_j$  and  $C'_j$  as  $(U, V)$  because the exact distribution of characters in the contexts is immaterial to our purposes.

Because  $L^3(G) = L^3(G')$ , it is the case that  $C_i/C_j/A = C'_i/C'_j/A'$ , and therefore  $XYUAVZ = XUAVYZ$ . Stripping away the common prefix and suffix, we are left with  $YUAV = UAVY$  and by Lemma 4 for some  $p \in \Sigma^+$  it is the case that  $Y = p^m$  and  $UAV = p^n$ . Because the length of  $p$  cannot be determined, it might be that  $UVA = p$ , with each of  $U$ ,  $V$ , and  $A$  contributing part of  $p$ . However, because  $C_i/C_j/C_j/A = C'_i/C'_j/C'_j/A'$ , we know that  $YUUAUVV = UUAUVVY$ , which means that  $Y, U, A$ , and  $V$  all contain an integer number of occurrences of  $p$ .  $\square$

## 5 Conclusion and Future Work

This paper presented the first polynomial-time algorithm for inferring  $\mathcal{SEC}_p^q$  grammars from positive data. This class of mildly context-sensitive languages is important because it has just enough context sensitivity to express the most common context-sensitive constructions found in natural languages while keeping open the possibility of computational tractability. Theorem 1 established that a finite sample suffices to infer  $\mathcal{SEC}_p^q$  grammars using our algorithm, and theorems 4 and 5 show that this sample is small when the axiom of the grammar to be learned contains a single string (i.e.,  $p = 1$ ).

Future work will proceed by removing some working assumptions such as a priori dimensionality and degree values. We want to extend our algorithm to any number of axioms and arbitrary selectors, thus learning external contextual grammars. As  $\mathcal{EC}_p$  languages are superfinite, additional information is required to learn the grammars. We are interested in leveraging structural information and prior approaches to learning context-free grammars for  $\mathcal{EC}_p$  grammar learning. Also, the ability of this formalism to describe non-context-free structures is particularly alluring for real world data applications (e.g. inference of grammars for natural language, user modeling).

## References

1. Quine, W.V.O.: Word and object. MIT Press (1960)
2. Chomsky, N.: Reflections on Language. Pantheon (1975)
3. Gold, E.M.: Language identification in the limit. *Information and Control* **10** (1967) 447–474
4. Marcus, G.F.: Negative evidence in language acquisition. *Cognition* **46** (1993) 53–85
5. Makinen, E.: On the structural grammatical inference problem for some classes of context-free grammars. *Information Processing Letters* **42** (1992) 1–5

6. Oates, T., Desai, D., Bhat, V.: Learning  $k$ -reversible context-free grammars from positive structural examples. In: Proceedings of the Nineteenth International Conference on Machine Learning. (2002)
7. Makinen, E.: Inferring regular languages by merging nonterminals. TR A-19987-6, Department of Computer Science, University of Tampere (1997)
8. Oates, T., Armstrong, T., Harris, J., Nejman, M.: On the relationship between lexical semantics and syntax for the inference of context-free grammars. In: Proceedings of AAAI. (2004) 431–436
9. Kudlek, M., Martn-Vide, C., Mateescu, A., Mitrana, V.: Contexts and the concept of mild context-sensitivity. In: Linguistics and Philosophy 26. (2002) 703–725
10. Becerra-Bonache, L., Yokomori, T.: Learning mild context-sensitiveness: Toward understanding children’s language learning. In: Proceedings of the 7th International Colloquium on Grammatical Inference. (2004) 53–64