

A Polynomial Time Algorithm for Inferring Grammars for Mildly Context Sensitive Languages

Tim Oates, Tom Armstrong, Mike Atamas

University of Maryland Baltimore County

CSEE Department

1000 Hilltop Circle

Baltimore, MD 21250

{ oates, arm1, m39 } @umbc.edu

Leonor Becerra-Bonache

Rovira i Virgili University

Pl. Imperial Tarraco 1, 43005

Tarragona, Spain

leonor.becerra@estudiants.urv.es

Abstract

Natural languages are largely context-sensitive, yet context-sensitive grammars cannot be identified in the limit from positive examples [Gold, 1967]. Mildly context-sensitive languages are able to express the most common context-sensitive structures found in natural language. We restrict our view to a class of mildly context-sensitive languages which can be described by simple external contextual grammars. We present the first polynomial-time algorithm for inferring these grammars from positive data.

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 [Quine, 1960]) or syntax (Chomsky’s “argument from the poverty of the stimulus” [Chomsky, 1975]). For example, it is known many classes or 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 [Gold, 1967]. 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 [Marcus, 1993].

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 [Makinen, 1992; Oates *et al.*, 2002]. 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 [Makinen, 1997].

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 [Oates *et al.*, 2004] 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 (*SEC*) languages from positive data. *SEC* 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 [Kudlek *et al.*, 2002]. In addition, neither the regular languages nor the context-free languages are a proper subset of the *SEC* languages. That is, there exist regular and context-free languages for which no corresponding *SEC* 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 *SEC* languages from positive examples is exponential [Becerra-Bonache and Yokomori, 2004]. This paper presents the first polynomial time algorithm for learning *SEC* languages from positive data, and proves its correctness, thereby realizing the computational advantages of the restricted expressiveness of *SEC* 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 *SEC* 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. We study External Contextual (\mathcal{EC}) grammars and a subset of them called Simple External Contextual (\mathcal{SEC}) grammars.

An \mathcal{EC} grammar is a three tuple $G = (\Sigma, B, C)$, where Σ is the alphabet of G , B is a finite set of p -words over Σ called the *base* of G , and C is a finite set of p -contexts over Σ . C is called the set of *contexts* of G .

A p -word x over Σ is a p -dimensional vector whose components are words over Σ , i.e., $x = (x_1, x_2, \dots, x_p)$, where $x_i \in \Sigma^*$, $1 \leq i \leq p$. A p -context c over Σ is a p -dimensional vector whose components are contexts over Σ , 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 are derived by *applying* contexts to the base of the grammar (wrapping elements of p -contexts around corresponding elements of p -words). 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 Σ . 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 language generated by G , denoted $L(G)$, is defined as:

$$L(G) = \{y \in \Sigma^* \mid \text{there exists } (x_1, x_2, \dots, x_p) \in B \text{ such that } (x_1, x_2, \dots, x_p) \Rightarrow_G^* (y_1, y_2, \dots, y_p) \text{ and } y = y_1y_2\dots y_p\}.$$

The \mathcal{EC} language family is superfinite (the base of G can be any finite set of p -words) and therefore not learnable in the limit from positive data [Gold, 1967]. We restrict the class of languages to Simple External Contextual (\mathcal{SEC}) grammars where the base, B , is a singleton of p -words over Σ .

\mathcal{SEC} grammars can further be specified by two parameters, p and q , and can be written as $\mathcal{SEC}_{p,q}^d$.

p (dimension) - the number of elements in the base, separated by commas)

q (degree) - the number of contexts.

The \mathcal{EC} and \mathcal{SEC} families occupy eccentric positions 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. They can express the context-sensitive syntactic constructions that are most prevalent in natural languages.

Most state-of-the-art grammatical inference algorithms apply to regular and context-free language, but few apply to mildly context-sensitive languages.

2.1 Example

Consider the following \mathcal{SEC}_1^2 grammar:

Let $G = (\{a, b, c\}, B, C)$, $B = \{(\lambda, \lambda)\}$, $C = \{c_1 = [(a, b), (c, \lambda)]\}$

The derivation of the string $aabbcc$ is the application of context c_1 two times to the base: $(\lambda, \lambda) \vdash (a\lambda b, c\lambda\lambda) \vdash (aa\lambda bb, cc\lambda\lambda\lambda)$. Finally the internal strings are concatenated resulting in $aabbcc$.

Note that $L(G) = \{a^n b^n c^n \mid n \geq 0\}$.

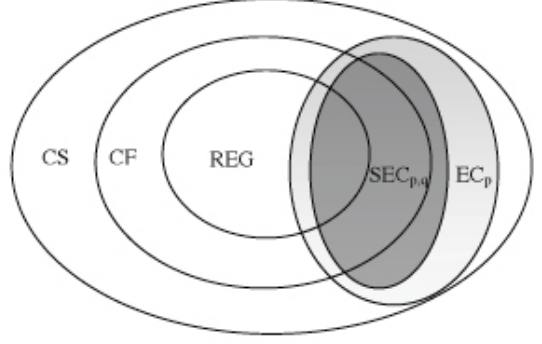


Figure 1: The \mathcal{SEC} family occupies an eccentric position in the Chomsky hierarchy.

3 Induction Algorithm

Our algorithm to infer simple external contextual grammars works given a set of strings, S , that were generated by an $\mathcal{SEC}_{p,q}^d$ grammar (where p and q are known), and a depth d . The list of strings must be exhaustive up to depth d . The smallest string in S is the concatenation of the elements in the base. However, there are multiple possibilities for what the actual base is (depending on the value of p). From the smallest string, a set of possible bases, B , is generated. For example, if $p=2$ and the smallest string in S is aa , then the set of possible bases is $\{(\lambda, aa), (a, a), (aa, \lambda)\}$.

The rest of the algorithm is repeated for every $b_j \in B$. The strings in S are processed for b_j . Each s_i is assumed to have been generated by a single application of a context to the base. The possible contexts that could have generated s_i after one application to b_j are stored in a set P . When all the strings in S have been processed, the correct grammar must be found. The elements of P are grouped into grammars with q contexts in all possible ways and stored in a set G . Any element of G that cannot generate every string in S is discarded, leaving only those grammars in G that generate S . Next, every element $g \in G$ is used to generate all possible strings up to depth d . If any of the generated strings are not in the input set S then g is discarded.

For example, consider an input set $S = \{a, aaa, bab, ababa, baaab\}$ 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 therefore it is the string representing the base. Since $p = 1$, a is the only possible base. 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 $\{[(b, b)]\}$. All of the possible contexts are then put in a set G . The only $g_i \in G$ that can generate the language is $\{[(a, a)], [(b, b)]\}$; the rest are discarded. All possible strings up to depth 2 have derivations in this grammar. Since it is equal to the input set, this grammar is not discarded and is output by the algorithm. In this case, there was only one possible base. If

this grammar had been \mathcal{SEC}_2^2 , then the possible bases would have been $\{[(\lambda, a)], [(a, \lambda)]\}$. For each of the possible bases, the set G would have been constructed and every element of that set tested. Unlike the example chosen here, many input sets can be generated by several grammars, some of which have distinct bases.

The factors that influence the running time of the algorithm (see figure 2) are dimension, degree, the length of the base and maximum depth in the input set. We assume that p, q and the length of the base are all small constants. Obviously, the dominating section is the nested loop which generates all possible contexts given all possible bases and checks the strings generated by these candidate grammars against the inputSet.

Given the smallest string, the base, in the inputSet, there are $(|base| + 1)^{p-1}$ possible bases (inserting $p-1$ commas into the smallest string). Given a string s_i , all of the possible contexts to derive s_i with one application is at most $(|s_i| + 1)^{2p}$. Given the set of all contexts from above, we can pick all sets of contexts of size q , or $\binom{(|s_i|+1)^{2p}}{q}$. 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 inputSet in at most $(\sum_{i=1}^d q^i) * |inputSet|$ steps.

Therefore, the cost of the nested loop is bounded by $(\sum_{i=1}^d q^i) * |inputSet| * \binom{(|s_i|+1)^{2p}}{q} * (|base| + 1)^{p-1}$.

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 base, 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} grammars as follows:

$$G = (p, q, B, C_1, C_2, \dots, C_q)$$

$$G' = (p', q', B', C_1', C_2', \dots, C_{q'}')$$

B is the base 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 the remainder of this paper, we restrict the base in all grammars to λ .

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/B is the p -vector obtained by applying context C_i to base B .

Let $L^d(G)$ be the set of strings in $L(G)$ that require d or fewer derivation steps. $L_0(G) = \{\text{CONCAT}(B)\}$.

We want to prove the following theorem.

Theorem 1 $L(G) = L(G')$ and $\text{CONCAT}(C_i) \neq \text{CONCAT}(C_j)$ for $i \neq j$ and $1 \leq j \leq q$ 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 $\text{CONCAT}(B) = \text{CONCAT}(B')$.

Proof. Suppose $L(G) = L(G')$ but $\text{CONCAT}(B) \neq \text{CONCAT}(B')$. Because G and G' are minimal, there are no contexts containing nothing but empty strings, because they would be superfluous. The shortest string in $L(G)$ is $\text{CONCAT}(B)$ because all other contexts, being non-empty, add characters when applied to the base. Note that all other strings in $L(G)$ are longer than $\text{CONCAT}(B)$. Likewise for the shortest string in $L(G')$. If $\text{CONCAT}(B) \neq \text{CONCAT}(B')$ 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 $\text{CONCAT}(B)$ must equal $\text{CONCAT}(B')$. ■

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

Proof. If a mapping f between contexts in G and contexts in G' such that $\text{CONCAT}(C_i) = \text{CONCAT}(C_{f(i)'})$ exists, then it must be bijective. It must be one-to-one because $\text{CONCAT}(C_i) \neq \text{CONCAT}(C_j)$ for $i \neq j$ and $1 \leq j \leq q$. A many-to-one mapping would require that $\text{CONCAT}(C_i) = \text{CONCAT}(C_j)$ for some $i \neq j$. A one-to-many mapping imposes the same requirement. If $\text{CONCAT}(C_i) = \text{CONCAT}(C_j')$ and $\text{CONCAT}(C_i) = \text{CONCAT}(C_k')$ for $j \neq k$, then because each context in G must be mapped, some context other than C_i , call it C_h , will map to either C_j' or C_k' , which means that $\text{CONCAT}(C_i) = \text{CONCAT}(C_h)$. This establishes that if a mapping exists it must be one-to-one. Finally, because $q = q'$, the mapping must be onto as well. Therefore, if a mapping exists, it must be bijective.

The only way for the theorem to not hold, then, is if for there to be some C_i such that there does not exist a C_j' such that $\text{CONCAT}(C_i) = \text{CONCAT}(C_j')$. That is, there is no mapping with the desired property.

If we assume that there exists a C_i such that no C_j' exists, that means that there is a string, S , in $L(G)$ that can be derived by one application of C_i that need several applications of contexts in $L(G')$ because there is no C_j' such that $\text{CONCAT}(C_i) = \text{CONCAT}(C_j)$. Lets assume that $L(G')$ needs C_1' and C_2' to produce one application of C_i . Then $L(G)$ needs to contain C_1'/B' and C_2'/B' because both of those can be produced by $L(G')$. However, if it contains those then it can also produce C_i by applying those contexts. This makes C_i unnecessary and $L(G)$ is no longer minimal. ■

Lemma 3 If $L(G) = L(G')$ and $\text{CONCAT}(C_i) \neq \text{CONCAT}(C_j)$ for $i \neq j$ and $1 \leq j \leq q$, 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

$$C_i/C_j/\dots/C_k/B$$

then the derivation of S in G' is

```

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

```

Figure 2: Pseudocode of the inference algorithm.

$C_{f(i)}/C_{f(j)}/\dots/C_{f(k)}/B'$

Proof. All derivations begin with the base B . Consider one-step derivations of the form C_i/B , which means context C_i is applied to base B . Let $S_0 = \text{CONCAT}(B)$ and $S_1 = \text{CONCAT}(C_i/B)$. The only difference between S_0 and S_1 is that S_1 contains, in addition to the characters in B , 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 $\text{CONCAT}(C_i) \neq \text{CONCAT}(C_j)$ for $i \neq j$ and $1 \leq j \leq q$.

By lemma 2, there must exist a context C_j' in G' such that $\text{CONCAT}(C_i) = \text{CONCAT}(C_j')$. For the sake of simplicity, we assume that context is C_i' . Because $\text{CONCAT}(C_i)$ is unique among the contexts in G , so must be $\text{CONCAT}(C_i')$ be among the contexts in G' (by lemma 2). Therefore, the only way to derive S_1 in G' is C_i'/B' , 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. ■

Theorem 2 *If $L(G) = L(G')$ and $\text{CONCAT}(C_i) \neq \text{CONCAT}(C_j)$ for $i \neq j$ and $1 \leq j \leq q$, 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 . ■

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 stated formally as

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

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

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

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

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. Because for both the set of all d and for the set of all possible depths, a bijection from \mathbb{N} exists, 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 $L(G) = L(G')$. ■

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}_q^p 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}_q^1 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. (sketch) Let $n = |A|$ and $m = |B|$. Without loss of generality, assume $n = m + k$ for $k \geq 0$. If $k = 0$ then $n = m$ and $A = B$ (because $AB = BA$ and the strings are of equal length) so $p = A = B$ and $x = y = 1$. Now suppose $k > 0$. Let S_i denote the i^{th} character in string S , where the index of the first character is 1. Let $S_{i,j}$ denote the sub-string of S from characters i through j .

Because $AB = BA$ it must be the case that $AB_i = BA_i$ for $1 \leq i \leq m + n$. Note that $AB_{m+1, m+k}$ corresponds to the last k characters of A , and that $BA_{m+1, m+k}$ corresponds to the first k characters of A . Therefore, these sub-strings must be equal. Because the first k characters of A in AB are mapped to the first k characters of B in BA , both A and B must share a length k prefix. Now note that this prefix of B in AB maps to the second k characters in A in BA . Due to the difference in length of A and B , this matching constraint is propagated through both strings, yielding the desired result. ■

The following two theorems says 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. Proof sketches are omitted due to space restrictions.

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 $L(G) = L(G')$.*

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

5 Conclusion and Future Work

We have presented a polynomial time algorithm for inferring \mathcal{SEC}_q^p grammars from positive data. Theorem 1 established that the size of the sample needed to infer an \mathcal{SEC}_q^p is finite. Theorems 4 and 5 are toward demonstrating that the maximum depth required in the characteristic sample is only 3. This is the first such efficient algorithm as the previous attempt was exponential.

Future work will attempt to remove some of our working assumptions. Eventually, we want to extend our algorithm to any number of bases, thus learning external contextual grammars. As \mathcal{EC} languages are superfinite, additional information

is required to learn the grammars. We are interested in looking at ways in which the context-free structural approaches might be applied to \mathcal{EC} languages. The ability of this formalism to describe non-context-free structures is particularly alluring for real world data applications. We are interested in applications for natural language processing.

References

- [Becerra-Bonache and Yokomori, 2004] Leonor Becerra-Bonache and Takashi Yokomori. Learning mild context-sensitiveness: Toward understanding children’s language learning. In *Proceedings of the 7th International Colloquium on Grammatical Inference*, pages 53–64, 2004.
- [Chomsky, 1975] Noam Chomsky. *Reflections on Language*. Pantheon, 1975.
- [Gold, 1967] E. Mark Gold. Language identification in the limit. *Information and Control*, 10:447–474, 1967.
- [Kudlek et al., 2002] M. Kudlek, C. Martn-Vide, A. Ma-teescu, and V. Mitran. Contexts and the concept of mild context-sensitivity. In *Linguistics and Philosophy* 26, pages 703–725, 2002.
- [Makinen, 1992] Erkki Makinen. On the structural grammatical inference problem for some classes of context-free grammars. *Information Processing Letters*, 42:1–5, 1992.
- [Makinen, 1997] Erkki Makinen. Inferring regular languages by merging nonterminals. Technical Report A-19987-6, Department of Computer Science, University of Tampere, 1997.
- [Marcus, 1993] Gary F. Marcus. Negative evidence in language acquisition. *Cognition*, 46(1):53–85, 1993.
- [Oates et al., 2002] Tim Oates, Devina Desai, and Vinay Bhat. Learning k-reversible context-free grammars from positive structural examples. In *Proceedings of the Nineteenth International Conference on Machine Learning*, 2002.
- [Oates et al., 2004] Tim Oates, Tom Armstrong, Justin Harris, and Mark Nejman. On the relationship between lexical semantics and syntax for the inference of context-free grammars. In *Proceedings of AAAI*, pages 431–436, 2004.
- [Quine, 1960] W. V. O. Quine. *Word and object*. MIT Press, 1960.