A Polynomial Algorithm for the Inference of Context Free Languages

Alexander Clark[1], Rémi Eyraud[2] and Amaury Habrard[2]

[1] Department of Computer Science,
Royal Holloway, University of London alexc@cs.rhul.ac.uk
[2] Laboratoire d'Informatique Fondamentale,
University of Aix-Marseille, CNRS,
{remi.eyraud,amaury.habrard}@lif.univ-mrs.fr

Abstract. We present a polynomial algorithm for the inductive inference of a large class of context free languages, that includes all regular languages. The algorithm uses a representation which we call Binary Feature Grammars based on a set of features, capable of representing richly structured context free languages as well as some context sensitive languages. More precisely, we focus on a particular case of this representation where the features correspond to contexts appearing in the language. Using the paradigm of positive data and a membership oracle, we can establish that all context free languages that satisfy two constraints on the context distributions can be identified in the limit by this approach. The polynomial time algorithm we propose is based on a generalisation of distributional learning and uses the lattice of context occurrences. The formalism and the algorithm seem well suited to natural language and in particular to the modelling of first language acquisition.

1 Introduction

For dealing with natural languages, there is a tension between using highly expressive formalisms and using formalisms that can be learned. For example, Tree Adjoining Grammars or other mildly context sensitive formalisms are very powerful but are difficult to handle from a machine learning standpoint. Many learnability results have been obtained for regular languages or for small subclasses of context free languages [1–3], but these results are still much too limited from a language theoretic point of view. In this paper, we propose to bridge for the first time the gap between theoretically well founded grammatical inference methods and the sorts of representations required for modelling natural languages. We present a family of representations for highly structured context free languages and show how they can be learned using a generalisation of distributional learning.

The contributions of this paper are as follows: We present in Section 3 a rich grammatical formalism, which we call *Binary Feature Grammars* (BFG). The class of languages defined by BFGs contains all context free languages (CFL) and some non context free languages. This makes the formalism a good candidate

for representing natural languages. We will then show how the features can be defined in terms of the contexts of strings (Section 4), and how these grammars using context features can be learned directly from samples. We then prove in Section 5 our main result: that there is an algorithm that can efficiently identify in the limit the class of CFLs with certain properties, the finite context property and the finite kernel property, from positive data and a membership oracle.

2 Basic Definitions

We consider a finite alphabet Σ, and Σ^* the free monoid generated by Σ. λ is the empty string, and a language is a subset of Σ^*. We will write the concatenation of u and v as uv, and similarly for sets of strings. $u \in \Sigma^*$ is a substring of $v \in \Sigma^*$ if there are strings $l, r \in \Sigma^*$ such that $v = lur$. Define $Sub(u)$ to be the set of nonempty substrings of u. For a set of strings S define $Sub(S) = \bigcup_{u \in S} Sub(u)$.

A context is an element of $\Sigma^* \times \Sigma^*$. For a string u and a context $f = (l, r)$ we write $f \odot u = lur$; the insertion or wrapping operation. We extend this to sets of strings and contexts in the natural way. Define $Con(w) = \{(l, r) | \exists u \in \Sigma^+ : lur = w\}$; i.e. the set of all contexts of a word; similarly for a set of strings we define: $Con(S) = \bigcup_{w \in S} Con(w)$.

The set of contexts, or context distribution, of a string u of a language L is, $C_L(u) = \{(l, r) \in \Sigma^* \times \Sigma^* | lur \in L\}$. We will often drop the subscript where there is no ambiguity. We define the syntactic congruence as $u \equiv_L v$ iff $C_L(u) = C_L(v)$. The equivalence classes under this relation are the *congruence* classes of the language.

We now recall the definition of a context free grammar.

Definition 1. *A context free grammar (CFG) is a quadruple $G = (\Sigma, V, P, S)$. Σ is a finite alphabet of terminal symbols, V is a set of non terminals s.t. $\Sigma \cap V = \emptyset$, $P \subseteq V \times (V \cup \Sigma)^+$ is a finite set of productions, $S \in V$ is the start symbol.*

We denote a production of P: $N \to \alpha$ with $N \in V$ and $\alpha \in (V \cup \Sigma)^+$. We will write $uNv \Rightarrow_G u\alpha v$ if there is a production $N \to \alpha$ in G. $\overset{*}{\Rightarrow}_G$ denotes the reflexive transitive closure of $\Rightarrow_G$.

The language defined by a CFG G is $L(G) = \{w \in \Sigma^* | S \overset{*}{\Rightarrow}_G w\}$. In general we will assume that λ is not a member of any language.

3 Binary Feature Grammars

Before the presentation of our formalism, we give some results about contexts that will help give an intuition about the representation. A standard lemma is:

Lemma 1. *For any language L and for any strings u, u', v, v' if $C(u) = C(u')$ and $C(v) = C(v')$, then $C(uv) = C(u'v')$.*

This establishes that the syntactic monoid $\Sigma^* / \equiv_L$ is well-defined; from a learnability point of view this means that if we want to compute the contexts of

a string w we can look for a split into two strings uv where u is congruent to u' and v is congruent to v'; if we can do this and we know how u' and v' combine, then we know that the contexts of uv will be exactly the contexts of $u'v'$. There is also a slightly stronger result:

Lemma 2. *For any language L and for any strings u, u', v, v' if $C(u) \subseteq C(u')$ and $C(v) \subseteq C(v')$, then $C(uv) \subseteq C(u'v')$.*

Proof. We write out the proof completely as the ideas will be used later on: suppose we have u, v, u', v' that satisfy the conditions. Suppose $(l, r) \in C(uv)$; then $(l, vr) \in C(u)$ and therefore $(l, vr) \in C(u')$. So $(lu', r) \in C(v)$, and therefore $(lu', r) \in C(v')$, so $(l, r) \in C(u'v')$. □

Looking at Lemma 2 we can also say that, if we have some finite set of strings K, where we know the contexts, then:

Corollary 1.

$$C(w) \supseteq \bigcup_{\substack{u',v':\\ u'v'=w}} \bigcup_{\substack{u\in K:\\ C(u)\subseteq C(u')}} \bigcup_{\substack{v\in K:\\ C(v)\subseteq C(v')}} C(uv)$$

This is the basis of our representation: a word w is characterised by its set of contexts. We can compute the representation of w, from the representation of its parts u', v', by looking at all of the other matching strings u and v where we understand how they combine (with subset inclusion). Rather than representing just the congruence classes, we will represent the lattice structure of the set of contexts using subset inclusion; sometimes called Dobrušin-domination [4].

To express this basic idea of inference, we will define an appropriate formalism: *binary feature grammars*. Initially, we will define it with no reference to learnability. Note the resemblance between this formalism and GPSG [5], and most importantly the class of Range Concatenation Grammars (RCG) [6].

Definition 2. *We define a Binary Feature Grammar (BFG) G as a tuple $\langle F, f_s, P, P_L, \Sigma\rangle$. F is a finite set (of features), where we write $C = 2^F$ for the power set of F defining the categories of the grammar, $P \subseteq C \times C \times C$ is a finite set of productions that we write $x \to yz$ where $x, y, z \in C$ and $P_L \subseteq C \times \Sigma$ is a set of lexical rules, written $x \to a$ and $f_s \in F$ is the sentence feature.*

Normally P_L will contain exactly one production for each letter in the alphabet.

A BFG G defines recursively a map f_G from $\Sigma^* \to C$ as follows:

$$f_G(\lambda) = \emptyset \tag{1}$$

$$f_G(w) = \bigcup_{(c\to w)\in P_L} c \qquad \text{iff } |w| = 1 \tag{2}$$

$$f_G(w) = \bigcup_{u,v:uv=w} \bigcup_{\substack{x\to yz\in P:\\ y\subseteq f_G(u)\wedge\\ z\subseteq f_G(v)}} x \qquad \text{iff } |w| > 1. \tag{3}$$

Note the relation between the third clause above and Corollary 1. Note also that in general we will apply more than one production at each step of the analysis. Given a BFG G and a string w it is possible to compute $f_G(w)$ in time $\mathcal{O}(|F||P||w|^3)$ using standard dynamic programming techniques.

Definition 3. *The language defined by a BFG G is the set of all strings that are assigned the sentence feature: $L(G) = \{u | f_s \in f_G(u)\}$.*

While this formalism has some relationship to a context free grammar, and some to a semi-Thue system (also known as a string rewriting system), it is not formally identical to either of these. The only exact equivalence is to a restricted subset of Range Concatenation Grammars; a very powerful formalism [6]. We include the following relationship, but suggest that the reader unfamiliar with RCGs proceeds to the discussion of the relationship with the more familiar class of context free grammars.

Lemma 3. *For every BFG G, there is a non-erasing positive range concatenation grammar of arity one, in 2-var form that defines the same language.*

Proof. Suppose $G = \langle F, f_s, P, P_L \rangle$. Define a RCG with a set of predicates equal to F and the following clauses, and the two variables U, V. For each production $x \to yz$ in P, for each $f \in x$, where $y = \{g_1, \dots g_i\}$, $z = \{h_1, \dots h_j\}$ add clauses

$$f(UV) \to g_1(U), \dots g_i(U), h_1(V), \dots h_j(V).$$

For each lexical production $\{f_1 \dots f_k\} \to a$ add clauses

$$f_i(a) \to \epsilon.$$

It is straightforward to verify that $f(w) \vdash \epsilon$ iff $f \in f_G(w)$. □

3.1 BFGs and CFGs

Let $G = \langle V, S, P, \Sigma \rangle$ be a CFG in Chomsky Normal Form (CNF). We will now construct an equivalent BFG grammar $G_{BFG} = \langle F, f_s, \hat{P}, P_L, \Sigma \rangle$ where $F = V$, $f_s = S$ and $\hat{P} = \{\{X\} \to \{Y\}\{Z\} | X \to YZ \in P\}$, and $P_L = \{\{X\} \to a | X \to a \in P\}$. We claim that the BFG G_{BFG} defines the same language as G.

Lemma 4. *Let $G = \langle V, S, P, \Sigma \rangle$ be a CFG in CNF. For every string w, $N \stackrel{*}{\Rightarrow} w$ if and only if $N \in f_{G_{BFG}}(w)$.*

Note that for this construction, computing the feature map f_G is exactly equivalent to computing the CKY parse table.

Lemma 5. *Every CF language can be represented by a BFG.*

Proof. Let G a CFG and G_{BFG} the BFG described above. By Lemma 4, if $S \Rightarrow^* w$ then $S = f_S \in f_{G_{BFG}}(w)$ and vice versa; thus $L(G) = L(G_{BFG})$. □

BFG and Non context free languages BFGs are more powerful than CFGs in two respects. First, BFGs can compactly represent languages like the finite language of all $n!$ permutations of an n-letter alphabet, that have no concise representation as a CFG [7]. Secondly, BFGs can represent some non-context free languages. Let $L = \{a^n b^n c^n d | n > 0\}$, which is clearly not context free. However we can construct a BFG that recognises this language. Let $G = \langle F, S, P, P_L, \Sigma \rangle$ a BFG s.t.:

- $F = \{A, A', B, C, C', AB, AB', AAB, BC, BC', BBC, D, S\}$,
- $P = \{\{S\} \to \{AB', BC'\}\{D\}, \{AB'\} \to \{AB\}\{C'\},$
  $\{AB\} \to \{AAB\}\{B\}, \{AB\} \to \{A\}\{B\},$
  $\{AAB\} \to \{A\}\{AB\}, \{BC'\} \to \{A'\}\{BC\},$
  $\{BC\} \to \{BBC\}\{C\}, \{BC\} \to \{B\}\{C\}, \{BBC\} \to \{B\}\{BC\},$
  $\{A'\} \to \{A'\}\{A\}, \{C'\} \to \{C'\}\{C\}\}$,
- $P_L = \{\{A, A'\} \to a, \{B\} \to b, \{C, C'\} \to c, \{D\} \to d\}$.

To give an intuition on the construction of the grammar, we describe the contribution of some features:

- AB defines the language $\{a^n b^n | n > 0\}$
- AB' defines the language $\{a^n b^n c^m | m > 0, n > 0\}$,
- AAB defines the language $\{aa^n b^n | n > 0\}$,
- BC' defines the language $\{a^m b^n c^n | m > 0, n > 0\}$,
- BC defines the language $\{b^n c^n | n > 0\}$,
- BBC defines the language $\{bb^n c^n | n > 0\}$.

3.2 Contextual Binary Feature Grammars

The class of BFGs is a powerful formalism: we are interested in a special case where the features are contexts. Here we define this in the most straightforward way though there are a number of obvious extensions.

Definition 4. *A Contextual Binary Feature Grammar is a BFG where the feature set is a finite set of contexts (i.e. $F \subset \Sigma^* \times \Sigma^*$) and the sentential feature is (λ, λ).*

By itself this is not a constraint but we are interested in cases where there is a correspondence between the language theoretic interpretation of a context, and the occurrence of that context as a feature in the grammar: in this case the features will be observable which will lead to learnability. Clearly from an inference point of view, at the minimum we want the sentence features to be correct: if we are learning a target language L, then if $(\lambda, \lambda) \in f_G(u)$ iff $u \in L$ then $L(G) = L$ which is what we want. But ideally we also want f_G to be correct for all features.

Definition 5. *Given a finite set of contexts $F = \{(l_1, r_1), \ldots, (l_n, r_n)\}$ and a language L we can define the context feature map $F_L : \Sigma^* \to 2^F$ which is just the map $u \mapsto \{(l, r) \in F | lur \in L\} = C_L(u) \cap F$.*

Using this definition, we now need a correspondence between the language theoretic context feature map F_L and the representation in our CBFG, f_G.

Definition 6. *A CBFG G is* exact *if for all $u \in \Sigma^*$, $f_G(u) = F_{L(G)}(u)$.*

Example. Let $L = \{a^n b^n | n > 0\}$. Let $\langle F, (\lambda, \lambda), P, P_L, \Sigma\rangle$ a CBFG s.t. $F = \{(\lambda, \lambda), (a, \lambda), (aab, \lambda), (\lambda, b), (\lambda, abb)\}$. The lexical productions in P_L are: $\{(\lambda, b), (\lambda, abb)\} \to a$ and $\{(a, \lambda), (aab, \lambda)\} \to b$. Then, the productions in P are defined by the set: $\{(\lambda, \lambda)\} \to \{(\lambda, b)\}\{(aab, \lambda)\}$, $\{(\lambda, \lambda)\} \to \{(\lambda, abb)\}\{(a, \lambda)\}$, $\{(\lambda, b)\} \to \{(\lambda, abb)\}\{(\lambda, \lambda)\}$, $\{(a, \lambda)\} \to \{(\lambda, \lambda)\}\{(aab, \lambda)\}$.

This defines an exact CBFG for L.

Clearly every exact CBFG is a BFG, but we conjecture that the class of languages with an exact CBFG is strictly smaller than the class of languages defined by general BFGs.

4 Inference

We have carefully defined the representation so that the inference algorithm will be almost trivial. Given a set of strings, and a set of contexts, we can simply write down a CBFG that will approximate a particular language.

Definition 7. *Let F be the set of contexts, $(\lambda, \lambda) \in F$, K a finite set of strings, $P_L = \{F_L(u) \to u | u \in K \wedge |u| = 1\}$ and $P = \{F_L(uv) \to F_L(u)F_L(v) | u, v, uv \in K\}$. We define $G_0(K, L, F)$ as the CBFG $\langle F, (\lambda, \lambda), P, P_L, \Sigma\rangle$.*

We will call K here the basis for the language. The set of productions is defined merely by observation: we take the set of all productions that we observe as the concatenation of elements of the small set K. Often K will be closed under substrings: i.e. $Sub(K) = K$. This grammar is a CBFG but in general it will not be exact. For example, the language it defines might be empty, in which case $F_L(u) = \emptyset$ for all u, and yet it could define some features on the grammar.

Clearly the language defined depends on two factors: the set of strings K and the set of features F. We now establish two important lemmas: first, that as K increases the language defined by $G_0(K, L, F)$ will increase, and secondly that as F increases the language will decrease.

Lemma 6. *Suppose we have two CBFGs defined by $G = G_0(K, L, F)$ and $G' = G_0(K, L, F')$ where $F \subseteq F'$. Then for all u, $f_G(u) \supseteq f_{G'}(u) \cap F$.*

Proof. Let G' have a set of productions P', P'_L, and G have a set of productions P, P_L. Clearly if $x \to yz \in P'$ then $x \cap F \to (y \cap F)(z \cap F)$ is in P by the definition of G_0, and likewise for P_L, P'_L. By induction on $|u|$ we can show that any feature in $f_{G'}(u) \cap F$ will be in $f_G(u)$. The base case is trivial since $F'_L(a) \cap F = F_L(a)$; if it is true for all strings up to length k, then if $f \in f_{G'}(u) \cap F$; there must be a production in F' with f on the head. By the inductive hypothesis, the right hand sides of the corresponding production in P will be triggered, and so f must be in $f_G(u)$. □

Corollary 2. *Suppose we have two CBFGs defined by* $G = G_0(K, L, F)$ *and* $G' = G_0(K, L, F')$ *where* $F \subseteq F'$*; then* $L(G) \supseteq L(G')$*.*

Conversely, we can show that as we increase K, the language and the map f_G will increase. This is addressed by the next lemma.

Lemma 7. *Suppose we have two CBFGs defined by* $G = G_0(K, L, F)$ *and* $G' = G_0(K', L, F)$ *where* $K \subseteq K'$*. Then for all* u*,* $f_{G_0(K,L,F)}(u) \subseteq f_{G_0(K',L,F)}(u)$*.*

Proof. Clearly the sets of productions of $G_0(K, L, F)$ will be a subset of the set of productions of $G_0(K', L, F)$, and so anything that can be derived by the first can be derived by the second, again by induction on the length of the string. □

To establish learnability, we need to prove that for a target language L, if we have a sufficiently large F then $L(G_0(K, L, F))$ will be contained within L and that if we have a sufficiently large K, then $L(G_0(K, L, F))$ will contain L.

4.1 Fiducial Feature Sets and Finite Context Property

We need to be able to prove that for any K if we have enough features then the language defined will be included within the target language L. We formalise the idea of having enough features in the following way:

Definition 8. *For a language* L *and a string* u*, a set of features* F *is fiducial on* u *if for all* $v \in \Sigma^*$*,* $F_L(u) \subseteq F_L(v)$ *implies* $C_L(u) \subseteq C_L(v)$*.*

Note that if F is fiducial on u and $F \subset F'$ then F' is fiducial on u. Therefore we can naturally extend this to sets of strings.

Definition 9. *For a set of strings* K*, a set of features* F *is fiducial if for all* $u \in K$*,* F *is fiducial on* u*.*

Note the asymmetry between u and v in these definitions. If u and v are both in K then having the same features means they are syntactically congruent. However if two strings, neither of which are in K, have the same features this does not mean they are necessarily congruent (for instance if $F_L(v) = F_L(v') = \emptyset$). For non finite state languages, the set of congruence classes will be infinite, and thus we cannot have a finite fiducial set for the set of all strings in $Sub(L)$, but we can have a feature set that is correct for a finite subset of strings, or more generally for an infinite set of strings, if they fall into a finite number of congruence classes.

We now define the finite context property.

Definition 10. *A language* L *has the Finite Context Property (FCP) if every string has a finite fiducial feature set.*

Clearly if L has the FCP, then any finite set of substrings, K, has a finite fiducial feature set which will be the union of the finite fiducial feature sets for each element of K. If $u \notin Sub(L)$ then any set of features is fiducial since $C_L(u) = \emptyset$.

Not all CFL have the FCP: for instance $L = \{a^n b | n > 0\} \cup \{a^n c^m | n > m > 0\}$, does not have the FCP, since there is no finite fiducial feature set for the string b; for any such set there will be some N such that c^N will have all of those features, but $C_L(c^N)$ is not a superset of $C_L(b)$.

However, all regular languages have the FCP since they have a finite number of syntactic congruence classes.

We can now state the most important lemma: this lemma links up the definition of the feature map in a BFG, with the fiducial set of features to show that only correct features will be assigned to substrings by the grammar. It states that the features assigned by the grammar will correspond to the language theoretic interpretation of them as contexts.

Lemma 8. *For any language L, given a set of strings K and a set of features F, let $G = G_0(K, L, F)$. If F is fiducial on K, then for all $w \in \Sigma^*$ $f_G(w) \subseteq F_L(w)$.*

Proof. We proceed by induction on length of the string. *Base case:* strings of length 1. $f_G(w)$ will be the set of observed contexts of w, and since we have observed these contexts, they must be in the language. *Inductive step:* let w a string of length k. Take a feature f on $f_G(w)$; by definition this must come from some production $x \to yz$ and a split u, v of w. The production must be from some elements of K, u', v' and $u'v'$ such that $y = F_L(u'), z = F_L(v')$ and $x = F_L(u'v')$. If the production applies this means that $F_L(u') = y \subseteq f_G(u) \subseteq F_L(u)$ (by inductive hypothesis), and similarly $F_L(v') \subseteq F_L(v)$. By fiduciality of F this means that $C(u') \subseteq C(u)$ and $C(v') \subseteq C(v)$. So by Lemma 2 $C(u'v') \subseteq C(uv)$. Since $f \in C(u'v')$ then $f \in C(uv) = C(w)$. Therefore, since $f \in F$ and $C(w) \cap F = F_L(w)$, $f \in F_L(w)$, and therefore $f_G(w) \subseteq F_L(w)$. □

Corollary 3. *If F is fiducial on K, and $(\lambda, \lambda) \in F$ then $L(G_0(K, F, L)) \subseteq L$.*

Therefore for any finite set K from an FCP language, we can find a set of features so that the language defined by those features on K is not too big.

4.2 Kernel and Finite Kernel Property

We will now show a complementary result, namely that for a sufficiently large K the language defined by G_0 will include the target language.

Definition 11. *A finite set $K \subseteq \Sigma^*$ is a kernel for a language L, if for any set of features F, $L(G_0(K, F, L)) \supseteq L$.*

To prove that a set is a kernel, it suffices to show that a fiducial set of features will define the language; any smaller set of features define then a larger language. In fact we can take the infinite set of all contexts and define productions based on

the congruence classes. If F is the set of all contexts then we have $F_L(u) = C_L(u)$, thus the productions will be exactly of the form $C(uv) \to C(u)C(v)$.

This is a slight abuse of notation since feature sets are normally finite.

Lemma 9. *Let $F = \Sigma^* \times \Sigma^*$; if $L(G_0(K, L, F)) \supseteq L$ then K is a kernel.*

Proof. By monotonicity of F: any finite feature set will be a subset of F. □

Not all context free languages will have a finite kernel. For example $L = \{a^+\} \cup \{a^n b^m | n < m\}$ does not have a finite kernel, but is clearly CF. Indeed, assume that the a set K contains all strings of length less than or equal to k. Assume w.l.o.g. that the fiducial set of features for K includes all features (λ, b^i), where $i \leq k+1$. Consider the rules of the form $F_L(a^k) \to F_L(a^j)F_L(a^{k-j})$; it is easy to see that no matter how large k is, the derived CBFG will undergenerate as a^k is not congruent to a^{k-1}.

Definition 12. *A context free grammar $G_T = \langle V, S, P, \Sigma \rangle$ has the Finite Kernel Property (FKP) iff for every non-terminal $N \in V$ there is a finite set of strings $K(N)$ such that for all $k \in K(N), N \stackrel{*}{\Rightarrow} k$ and where for every string $w \in \Sigma^*$ such that $N \stackrel{*}{\Rightarrow} w$ there is a string $k \in K(N)$ such that $C(k) \subseteq C(w)$. A CFL L has the FKP, if there is a grammar in CNF for it with the FKP. We also assume that $a \in K(N)$ if $a \in \Sigma$ and $N \to a \in P$.*

Notice that all regular languages have the FKP since they have a finite number of congruence classes.

Lemma 10. *Any context free language with the FKP has a finite kernel.*

Proof. Let $G_T = \langle V, S, P, \Sigma \rangle$ be such a CNF CFG with the FKP. Define

$$K(G_T) = \bigcup_{N \in V} \left(K(N) \cup \bigcup_{X \to MN \in P} K(M)K(N) \right). \tag{4}$$

We claim that $K(G_T)$ is a kernel. Assume that $F = \Sigma^* \times \Sigma^*$. Let $G = G_0(K(G_T), L(G_T), F) = \langle F, (\lambda, \lambda), P, P_L, \Sigma \rangle$.

We will show, by induction on the length of derivation of w in G_T, that for all N, w if $N \stackrel{*}{\Rightarrow} w$ then there is a k in $K(N)$ such that $f_G(w) \supseteq C(k)$. If length of derivation is 1, then this is true since $|w| = 1$ and thus $w \in K(N)$: therefore $C(w) \to w \in P_L$. Suppose it is true for all derivations of length less than j. Take a derivation of length j; say $N \stackrel{*}{\Rightarrow} w$. There must be a production in G_T of the form $N \to PQ$, where $P \Rightarrow^* u$ and $Q \Rightarrow^* v$, and $w = uv$. By inductive hypothesis; we have $f_G(u) \supseteq C(k_u)$ and $f_G(v) \supseteq C(k_v)$. By construction $k_u k_v \in K(G_T)$ and then there will be a rule $C(k_u k_v) \to C(k_u)C(k_v)$ in P. Therefore $f_G(uv) \supseteq C(k_u k_v)$. Since $N \stackrel{*}{\Rightarrow} k_u k_v$ there must be some $k_{uv} \in K(N)$ such that $C(k_{uv}) \subseteq C(k_u k_v)$. Therefore $f_G(w) \supseteq C(k_u k_v) \supseteq C(k_{uv})$. □

Now we can see that if $w \in L$, then $S \stackrel{*}{\Rightarrow} w$, then there is a $k \in K(S)$ such that $f_G(w) \supseteq C(k)$ and $S \stackrel{*}{\Rightarrow} k$, therefore $(\lambda, \lambda) \in f_G(w)$ since $(\lambda, \lambda) \in C(k)$, thus $w \in L(G)$ and therefore K is a kernel.

5 Algorithm

Before we present the algorithm, we will discuss the learning model that we use. The class of languages that we will learn is suprafinite and thus we cannot get a straight identification in the limit (IIL) result [8]. Ultimately we are interested in a more realistic probabilistic learning paradigm, but for mathematical convenience it is appropriate to establish the basic results in a symbolic paradigm. The ultimate goal is to model natural languages, where negative data, or equivalence queries are generally not available or are computationally impossible. Accordingly, we have decided to use the model of positive data together with membership queries: an oracle can tell the learner whether a string is in the language or not [9]. The presented algorithm runs in time polynomial in the size of the sample S: since the strings are of variable length, this size must be the sum of the lengths of the strings in S, $\sum_{w \in S} |w|$.

We should note that this is not a strong enough result: [10] showed that any algorithm can be made polynomial, by only processing a small prefix of the data.

It is hard to tighten the model sufficiently: the suggestion in [11] for a polynomial characteristic set is inapplicable for representations, such as the ones in this paper, that are powerful enough to define languages whose shortest strings are exponentially long. We note that the situation is unsatisfactory, but we do not intend to propose a solution in this paper. We merely point out that the algorithm is genuinely polynomial, processes all of the data in the sample without delaying tricks, is conservative and always produces a hypothesis that is compatible with the observed data and answers from the oracle.

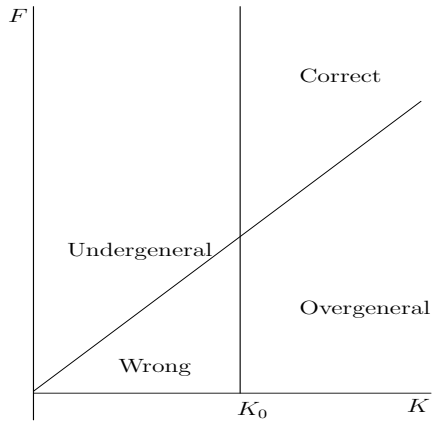

Fig. 1. The relationship between K and F: The diagonal line is the line of fiduciality: above this line means that F is fiducial on K. K_0 is the (a) kernel for the language.

Before we present the algorithm we hope that it is intuitively obvious how the approach will work. Figure 1 shows the relationship between K and F. When

we have a large enough K, we will be to the right of the vertical line; when we have enough features for that K we will be above the diagonal line. Thus the basis of the algorithm is to move to the right, until we have enough data, and then to move up vertically, increasing the feature set until we have a fiducial set.

We can now define our learning algorithm in Algorithm 1. Informally, D is the list of all strings that have been seen so far: the algorithm examines the set of strings $T = Con(D) \odot Sub(D)$. If the current hypothesis generates some element of this set that is not in the language, then it is overgeneralising: we need to add features. If on the other hand we undergeneralise, then we add all of the substrings of D to K, and all possible contexts to F. In Algorithm 1, $G_0(K, \mathcal{O}, F)$ denotes the same construction as $G_0(K, L, F)$, except that we use membership queries with the oracle $\mathcal{O}$ to compute F_L for each element in K.

Algorithm 1: BFG learning algorithm IIL

Data: A sequence of strings $S = \{w_1, w_2 \dots, \}$, membership oracle $\mathcal{O}$
Result: A sequence of CBFGs $G_1, G_2, \dots$
$K \leftarrow \emptyset$; $D \leftarrow \emptyset$; $F \leftarrow \{(\lambda, \lambda)\}$; $G_0 = G_0(K, \mathcal{O}, F)$;
for w_i **do**
  $D \leftarrow D \cup \{w_i\}$; $T \leftarrow Con(D) \odot Sub(D)$;
  if $\exists w \in T$ *such that* $w \in L(G_{i-1}) \setminus L$ **then** $F \leftarrow Con(D)$;
  if $\exists w \in T$ *such that* $w \in L \setminus L(G_{i-1})$ **then** $K \leftarrow Sub(D)$; $F \leftarrow Con(D)$;
  Output $G_i = G_0(K, \mathcal{O}, F)$;
end

Theorem 1. *Algorithm 1 runs in polynomial time in the size of the sample, and makes a polynomial number of calls to the membership oracle.*

Proof. The value of D will just be the set of observed strings; $Sub(D)$ and $Con(D)$ are both polynomially bounded by the size of the sample, so the number of calls to the oracle is clearly polynomial. Computing the feature map is polynomial in the length of the strings, and computing G_0 is also polynomial, since K and F will also be polynomially bounded. □

In the following, we consider the class of context free languages having the FCP and the FKP, represented by CBFG. K_n denotes the value of K at the n^{th} loop, and similarly for F, D and T, which is the test set; called by [12] the "explosion".

Clearly, if the grammar undergeneralises, when it encounters a string in $L(G) \setminus L$ it will increase the kernel. The corresponding fact for overgeneralisation is stated in this lemma:

Lemma 11. *For a given positive presentation of a CFL with the FCP and the FKP, if there is an m such that $L \subset L(G_m)$ and $L \neq L(G_m)$, then there is a string $w \in L(G_m) \setminus L$ such that there exists an $n \geq m$ such that $w \in T_n$.*

Proof. Let w be a shortest string such that there is a feature $f \in f_{G_m}(w) \setminus F_L(w)$. We know that this set is non empty as there is some string with $f = (\lambda, \lambda)$.
If $w \in Sub(L)$, then let f' be some feature in $C_L(w)$, et n be the smallest number such that $f' \odot w \in D_n$. T_n must contain $f \odot w$, which satisfies the lemma.
Alternatively suppose $w \notin Sub(L)$, then the feature f must come from some rule acting upon $uv = w$, where $u, v \in Sub(L)$ (since w is a shortest string, all rules will have non empty features by construction of G). Let $(u'v') \to (u')(v')$ be one of the triples of strings in K_m that produced this rule. Since $C(u'v') \not\subset C(uv)$ by Lemma 2 we have $C(u') \not\subset C(u)$, or $C(v') \not\subset C(v)$. Suppose w.l.o.g. that it is u, and consider a feature from $f'' \in C(u') \setminus C(u)$. Clearly $f'' \in Con(L)$, $u \in Sub(L)$ and $f'' \odot u$ is in $L(G_m) \setminus L$. Let n be the smallest index where we have $u \in Sub(D_n)$ and $f'' \in Con(D_n)$; and then $f'' \odot u$ satisfies the lemma. □

Lemma 12. *For every positive presentation of a CFL L with the FCP and the FKP, there is some n such that either $L(G_n) = L$ or K_n is a kernel, and $\forall N > n$ $K_N = K_n$.*

Proof. First of all if K_n is a kernel, then $L \setminus L(G_n) = \emptyset$, and so the the algorithm will not add any more strings to the kernel. Let m be the smallest number such that $Sub(D_m)$ is a kernel. Recall that any superset of a kernel is a kernel, and that all CFL with the FKP have a finite kernel (Lemma 10) so such an m must exist. Consider the grammar G_m; there are three possibilities:
1. $L(G_m) = L$; in which case the grammar has converged.
2. $L \setminus L(G_m) \neq \emptyset$, in which case, let $w \in L \setminus L(G_m)$, and let n be the first index of the presentation such that $w_n = w$; then K_n must contain $Sub(D_m)$, since at some point the kernel must have been expanded, and therefore K_n is a kernel.
3. $L(G_m) \setminus L(G) \neq \emptyset$, but $L \subset L(G_m)$. Let F be a finite fiducial feature set for K_m, (we assume w.l.o.g. that all such F are in $Con(K_m)$), and consider the smallest m' such that $(F \odot K_m) \cap L \subset D_{m'}$. Consider $G_{m'}$, either it is correct or undergeneralises, in which case we apply the same argument we just used. Otherwise, it strictly overgeneralises, in which case by Lemma 11, we will expand the feature set to a set which by construction will be fiducial. At this point we will either have enlarged K, in which case it will be a kernel, or we have not, in which case we will either undergeneralise or be exact; if it undergeneralises, then when we observe some incorrect string, the basis will be enlarged to a kernel. □

Lemma 13. *For every positive presentation of a language L with the FCP and the FKP, let n be the smallest number such that K_n is a kernel, if there is such an n. There is some $n_2 \geq n$ such that for all $N > n_2$, $F_N = F_{n_2}$ and the $L(G_0(K_N, L, F_N)) = L$.*

Proof. Let F be a finite fiducial feature set for K_n. Let n_1 the smallest number s.t. $D_{n_1} \supseteq (F \odot K_n) \cap L$. Either G_{n_1} is correct, in which case it is done, or it overgeneralises in which case by Lemma 11 there will be an n_2 which triggers the enlargement of the feature set. Since $F \subseteq Con(D_{n_1})$, the feature set will be fiducial. □

Theorem 2. *Algorithm 1 identifies in the limit the class of context free languages with the finite context property and the finite kernel property.*

Proof. Immediate by the preceding two lemmas. □

6 Discussion

First of all, we should establish how large the class of languages with the FCP and the FKP is: it includes all finite languages and all regular languages, since the set of congruence classes is finite for finite state languages. It similarly includes the context-free substitutable languages, [3], since every string in a substitutable language belongs to only one syntactic congruence class.

As already stated it does not include all CFLs since not all CFLs have the FCP and/or the FKP. However it does include languages like the Dyck languages of arbitrary order, Lukacevic language and most other classic simple examples.

As a special case consider the equivalence relation between contexts $f \cong_L f'$ iff $\forall u$ we have that $f \odot u \in L$ iff $f' \odot u \in L$. The class of CFLs where the context distribution of every string is a finite union of equivalence classes of contexts clearly has both the FKP and the FCP.

Our approach to context free grammatical inference is based on a generalisation of distributional learning, following the work of [3]. The current state of the art in context free inductive inference from flat examples only has been rather limited. When learning from stochastic data or using a membership oracle, it is possible to have powerful results, if we allow exponential computation (see for example [13]). The main contribution of this paper is to show that efficient learning is possible, with an appropriate representation. We currently rely on using a membership oracle, but under suitable assumptions about distributions, it should be possible to get a PAC-learning result for this class along the lines of [14], placing some bounds on the number of features required.

Linguistics. The field of grammatical inference has close relations to the study of language acquisition. Attempts to model natural languages with context free grammars require additional machinery: natural language categories such as noun phrases contain many overlapping subclasses with features such as case, number, gender and similarly for verbal categories. Modelling this requires either an exponential explosion of the number of non-terminals employed or a switch to a richer set of features. In our formalism, motivated by normal CF inference, we get this additional power for free. While we have implemented the algorithm described here, and verified that it works in accordance with theory on small artificial examples, there are a number of modifications that would need to be made before it can be applied to real grammar induction on natural language. First, the algorithm is very naive; in practice a more refined algorithm could select both the kernel and the feature set in a more sophisticated way. Secondly, considering features that correspond to individual contexts may be too narrow a definition for natural language given the well known problems of data sparseness and it will be necessary to switch to features corresponding to sets of contexts,

which may overlap. Thus for example one might have features that correspond to sets of contexts of the form $F(u, v) = \{(lu, vr)|l, r \in \Sigma^*\}$. This would take this approach closer to methods that have been shown to be effective in unsupervised learning in NLP[15] where typically $|u| = |v| = 1$. In any event, we think such modifications will be necessary for the acquisition of non context free languages. Finally, at the moment the algorithm has polynomial update time, but in the worst case, there are deterministic finite state automata such that the size of the smallest kernel will be exponential in the number of states. There are, however, natural algorithms for generalising the productions by removing features from the right hand sides of the rules; this would have the effect of accelerating the convergence of the algorithm, and removing the requirement for the FKP.

Acknowledgements

Part of the work described in this paper was carried out while Alex Clark was a *Professeur Invité* at the University of Marseille.

References

1. Higuera, C.D.L., Oncina, J.: Inferring deterministic linear languages. In: COLT '02: 15th Annual Conference, Springer-Verlag (2002) 185–200
2. Yokomori, T.: Polynomial-time identification of very simple grammars from positive data. Theoretical Computer Science **298**(1) (2003) 179–206
3. Clark, A., Eyraud, R.: Polynomial identification in the limit of substitutable context-free languages. Journal of Machine Learning Research **8** (2007) 1725–1745
4. Marcus, S.: Algebraic Linguistics; Analytical Models. Academic Press, N. Y. (1967)
5. Gazdar, G., Klein, E., Pullum, G., Sag, I.: Generalised Phrase Structure Grammar. Basil Blackwell (1985)
6. Boullier, P.: A Cubic Time Extension of Context-Free Grammars. Grammars **3** (2000) 111–131
7. Asveld, P.: Generating all permutations by context-free grammars in Chomsky normal form. Theoretical Computer Science **354**(1) (2006) 118–130
8. Gold, E.M.: Language identification in the limit. Information and Control **10** (1967) 447–474
9. Angluin, D.: Queries and concept learning. Mach. Learn. **2**(4) (1988) 319–342
10. Pitt, L.: Inductive inference, dfa's, and computational complexity. In: Lecture Notes in Artificial Intelligence. Springer-Verlag (1989) 8–14
11. de la Higuera, C.: Characteristic sets for polynomial grammatical inference. Machine Learning **27**(2) (1997) 125–138
12. Adriaans, P.: Learning shallow context-free languages under simple distributions. Algebras, Diagrams and Decisions in Language, Logic and Computation **127** (2002)
13. Horning, J.J.: A Study of Grammatical Inference. PhD thesis, Stanford University, Computer Science Department, California (1969)
14. Clark, A.: PAC-learning unambiguous NTS languages. In: Proceedings of the 8th International Colloquium on Grammatical Inference (ICGI). (2006) 59–71
15. Klein, D., Manning, C.: Corpus-based induction of syntactic structure: models of dependency and constituency. In: Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics. (2004) 478–485