Learning of Context-Free Languages: A Survey of the Literature [*]

Lillian Lee
Harvard University
Center for Research in Computing Technology
Cambridge, MA 01238

Abstract

We survey methods for learning context-free languages (CFL's) in the theoretical computer science literature. We first present some important negative results. Then, we consider five types of methods: those that take text as input, those that take structural information as input, those that rely on CFL formalisms that are not based on context-free grammars, those which learn subclasses of CFL's, and stochastic methods. A description of the subclasses of CFL's considered is provided, as is an extensive bibliography.

1 Introduction

> One may arrive at a grammar by intuition, guess-work, all sorts of partial methodological hints, reliance on past experience, etc. It is no doubt possible to give an organized account of many useful procedures of analysis, but it is questionable whether these can be formulated rigorously, exhaustively and simply enough to qualify as a practical and mechanical discovery algorithm [for grammars]. [Cho57]

The problem of grammatical inference is, in its broadest sense, the problem of learning a description of a language from data drawn from (but

[*]This work was supported by an NSF Graduate Fellowship and a grant from the AT&T Bell Labs Graduate Research Program for Women. We thank Ron Rivest and Mona Singh for helpful comments.

not necessarily *in*) the language. The problem of inferring *context-free* languages (CFL's) is worth considering for both practical and theoretical reasons. Practical applications include pattern recognition and speech recognition: one approach to pattern recognition is to infer a context-free grammar (CFG) that produces a set of patterns [Fu74]; also, the ability to infer (approximate[1]) CFGs for natural languages would enable a speech recognizer to modify its internal grammar on the fly, thus allowing it to adjust to individual speakers [Hor69]. From a theoretical standpoint, the problem poses an interesting challenge in that although CFL's are well-understood, there are serious limitations to learning them. Finally, it can be argued that successful language learning algorithms model how humans acquire language. Indeed, this last point, although questionable, was one of the principal motivations for the early work in grammar inference.

Of course, the above description of the problem is far too informal for us to work with; we need to define what "learning a language" means. In the grammatical inference literature, most attention has been focused on *identification in the limit*, an on-line performance measure [Gol67]. In this framework, at each time t the learner receives an information unit i_t about a language and outputs a hypothesis $H(i_1, \ldots, i_t)$. The learning algorithm is successful if after a finite amount of time, all its guesses are the same and are all a correct description of the language in question. Another popular learning criterion is *exact identification using queries in polynomial time* [Ang88]. In this framework, the learner has access to oracles that can answer questions about a language, and must halt in polynomial time with a correct description of the language. Note that we have not specified what type of information units are presented or what kind of oracles the learner has access to. The omission is intentional: if the information provided is not "powerful" enough, then neither type of grammar inference is possible.

We hope in this paper to "give an organized account of many useful procedures of analysis" for the problem of inferring CFL's. In the interest of readability, we will present results informally whenever possible. We first discuss in section 2 the negative results alluded to above. Sections 3 and 4 review methods for inferring CFG's from "powerful" sources of information. In section 5 we briefly discuss learning results for alternative formulations of CFL's, that is, formulations that do not make reference to CFG's. Section 6 surveys papers on learning subclasses of CFL's. Learning algorithms that have a Bayesian flavor are described in section 7. Finally, we give formal

[1]There is evidence that natural languages are not context-free [Cul85], [Shieb85].

definitions for all the grammars described (section 8), and provide a bibliography which we believe to be mostly complete with respect to the theoretical computer science literature. We hope that the papers reviewed demonstrate that Chomsky was only partially right: for certain scenarios, there indeed are no "practical and mechanical discovery algorithms"; on the other hand, studying other learning situations has yielded interesting results.

2 Feasibility of Grammar Inference

> Too much faith should not be put in the powers of induction, even when aided by intelligent heuristics, to discover the right grammar. After all, stupid people learn to talk, but even the brightest apes do not. [Cho63]

Sadly, there are very strong negative results for the learnability of context-free grammars. The main theorems are that it is impossible to identify any of the four classes of languages in the Chomsky hierarchy in the limit if the data consists only of strings in the language being inferred [Gol67], and that even the ability to ask equivalence queries does not guarantee exact identification in polynomial time [Ang88]. Thus, the literature on inferring context-free grammars has a rather strange flavor, as researchers have resorted to strenuous (one might almost say "desperate") measures to get around the restrictions imposed by these two negative results.

Let us first consider Gold's theorem. For any recursively enumerable (r.e.) language L, a *positive presentation* of L is an infinite sequence containing all and only strings in L, with repetitions allowed. A *super-finite* class of languages is one that contains all finite languages and at least one infinite language.

Definition 1 *Let $\mathcal{L}$ be a class of r.e. languages. An algorithm A* identifies $\mathcal{L}$ in the limit *from positive presentations iff for any $L \in \mathcal{L}$ and any positive presentation $I = i_1, i_2, \ldots$ of L, at time t A receives i_t and outputs hypothesis $h_t = H(i_1, i_2, \ldots, i_t)$, where H is effectively computable, such that there is a time m for which $h_m = h_{m+1} = \ldots$ and h_m is a correct description for L. A* ineffectively identifies $\mathcal{L}$ in the limit *if H is not effectively computable.*

Theorem 1 *There exists a super-finite class of languages that is neither ineffectively learnable nor effectively learnable in the limit from positive presentations.*

Proof Sketch. Let $L_1 \subset L_2 \subset \ldots$ be a sequence of finite languages, and set $L_\infty = \cup L_i$. Suppose there were a learning algorithm A which either ineffectively or effectively identifies $\{L : L \,\text{finite}\} \cup \{L_\infty\}$ in the limit. Then A must identify any finite language in a finite amount of time. So, we can give A a positive presentation that forces it to make an infinite number of mistakes: first present A with enough examples (with repeats) from L_1 to make it guess L_1. Then, give it enough examples (with repeats) from L_2 to force it to guess L_2, and so on. Clearly, all our examples will be in L_∞. ∎

We remark that the proof's reliance on allowing arbitrary repetition seems a bit unsatisfactory, and perhaps a more pleasing diagonalization argument is possible. Also, we note that Angluin has given a necessary and sufficient condition for a class of languages to be identifiable in the limit from positive data, involving each language containing a "telltale" finite subset which prevents the overgeneralization that occurs in the proof of theorem 1 [Ang80].

Gold's theorem tells us that not even the class of regular languages is identifiable in the limit when only positive data is allowed. This is a very disappointing result, especially for computational linguists: the obvious approach to finding a grammar for natural language is to feed large texts to a learning algorithm. We will not speculate on the implications for human language learning here.

It is not actually that surprising that positive data alone is insufficient, since once one hypothesizes a language that is a superset of the target, no example causes one to alter one's guess. One might expect that a more powerful source of information would remedy the situation, but Angluin shows that even equivalence queries do not insure exact identification in polynomial time [Ang90]. Let R be a representation of a class of languages $\mathcal{L}$, and let $r(L)$ be any representation for a language $L \in \mathcal{L}$. An *equivalence oracle* for $L_* \in \mathcal{L}$ takes as input a $r(L) \in R$; it returns "yes" if $L \equiv L_*$, and returns a string in the symmetric difference of L and L_* otherwise.

Definition 2 *A deterministic algorithm A* exactly identifies R using equivalence queries in polynomial time *iff there is a polynomial $p(\cdot,\cdot)$ such that for all L_*, when A is run with access to an equivalence oracle for L_*, at any point in its computation A has used an amount of time bounded by $p(l,m)$, where m is the maximum length of any counterexample returned so far and l is the length of the shortest representation of L.*

Suppose we can exhibit a target set $T \subseteq \mathcal{L}$ such that for every $L \in \mathcal{L}$, there is a string w_L such that only a superpolynomially small fraction of languages in T classify w_L the same way L does. Then, giving w_L as a counterexample to L eliminates very few languages from consideration.

Definition 3 *A represention R has the* approximate fingerprint property *iff there are polynomials p_1 and p_2 such that for all polynomials q, there is a sequence of language classes $T_1, T_2, \ldots$ such that for n sufficiently large,*

i. *there are at least two languages in T_n,*

ii. *every $L' \in T_n$ has a representation in R of length at most $p_1(n)$, and*

iii. *for $r(L) \in R$ such that $r(L)$ has length at most $q(n)$, there is a string w_L of length at most $p_2(n)$ such that $|\{L' \in T_n : \chi_{L'}(w_L) = \chi_L(w_L)\}| < |T_n|/q(n)$.*

All polynomials mentioned above are positive and nondecreasing. χ_L is the characteristic function for L.

Clearly, if R is a representation with the approximate fingerprint property, then there can be no algorithm for exact identification of R using only equivalence queries in polynomial time. Angluin goes on to give a rather involved proof that "natural" representations of CFG's have the approximate fingerprint property.

Finally, we mention in passing that Pitt and Warmuth have shown that, modulo a polynomial time transformation, CFG's are as hard to predict as certain cryptographic predicates are to compute; see [Pit88] for more details.

These negative results are rather formidable, and yet a quick glance at the bibliography shows that people did not give up on the problem. The approaches taken have been to provide learning algorithms with more helpful information, such as negative examples or structural information; to formulate alternative representations of CFG's (notice the dependence of Angluin's result on the type of representation used); to restrict attention to subclasses of CFL's that *don't* contain all finite languages; and to use Bayesian methods to search for "good" grammars. We review each of these approaches below.

3 Learning from text

In this section we examine CFL inference algorithms that take *texts* as input, where a text is a sequence of strings over the same alphabet as the CFL's being learned (so a text may contain strings not in the target language unless otherwise specified). In general, this branch of investigation has not been very productive, since, as Gold's theorem points out, texts are not a very helpful source of information.

One of the earliest methods for learning CFG's is that of [Sol59]. Since Solomonoff's proposed algorithm uses only positive data, we know that his method is not complete (in fact, Fu observes that it cannot discover languages with sequential embedding (e.g., $A \rightarrow aAb|cAd$) [Fu75]), but we discuss his paper here because it has influenced many other approaches. The learning scenario is that the learner is given a positive sample S^+ from a language L and has access to a *membership oracle* for L which, for any string, tells the learner whether or not the string is in L. Solomonoff's strategy is to find repeating patterns in strings: for every string $w \in S^+$, delete substrings from w to create a new string w' and ask the oracle if w' is in the language. If it is, then insert repetitions of the deleted substrings into the string and ask if the new string is also in the language. If so, then we infer that we must have a recursive rule. That is, if there are lots of strings of the form $a^n b^n$ in the language, then $A \rightarrow aAb$ is probably in the grammar.

Clearly, this method is extremely inefficient, depends heavily on what strings in L are included in S^+, and, as mentioned beforehand, cannot learn all CFL's. Nevertheless, the idea of having patterns in strings correspond to nonterminals in a grammar shows up in many papers on CFL inference.

Two other papers studying grammar learnability from texts are [Kno76] and [Tan87]. Knobe and Knobe consider the situation wherein the learner is given positive data as input and has access to a membership oracle. The algorithm they give is really only a collection of obvious heuristics, has enormous time complexity, and is heavily dependent on the order in which the samples are presented. Tanatsugu gives an algorithm that learns CFG's from positive and negative examples of strings. The technique presented is to remove self-embedding structures from a finite sample, infer a linear grammar from the sample, and compose the inferred linear grammars to create a CFG. The method is shown to be complete for all CFL's, but no time analysis is given.

4 Learning from structural data

An alternative to learning from texts is to learn from structural data. After all, we are often interested not just in the strings that a grammar derives, but also in the derivation tree(s) a grammar assigns to strings; that is, we generally want a CFG that makes structural sense.

One way structural data can be represented is by strings generated by a *parenthesis grammar* [McNu67]. For any context-free grammar G, the corresponding parenthesis grammar (G) is formed from G by replacing every production $A \to \alpha$ by $A \to (\alpha)$ (we assume that parentheses are not among the symbols in Σ). Crespi-Reghizzi presents an incremental algorithm for identifying *operator precedence grammars* in the limit from positive parenthesized data (i.e., parenthesizations of strings in the target language) [Cre71a]; he also gives a method for learning *k-distinct and homogeneous grammars* from positive parenthesized and negative (possible parenthesized) data [Cre74].

Structural data can also be represented by *skeletons*, which are derivation trees with the nonterminal labels removed [LJ78]. The key property of skeletons is that they are exactly the set of trees accepted by *skeletal tree automata* (SA's), a variation of finite automata that take skeletons as input. Informally, an SA A, when given a tree T as input, first assigns states to the leaves of T, and then moves up the tree, assigning a state to each node solely on the basis of the states of that node's children. A accepts T iff it assigns a final state to the root of T.

This automata characterization of the skeletons of derivation trees means that the problem of learning a CFG can be reduced to the problem of learning an SA. In particular, Sakakibara extends Angluin's method for learning finite automata (FA's) to an algorithm identifying SA's in polynomial time, although this requires being able to ask *structural membership* and *structural equivalence* queries [Sak92a]. He also shows how to infer *reversible CFG's* in the limit from positive structural data alone by adapting Angluin's technique for reversible automata [Ang82]. Fass gives a similar method, although her algorithm depends on the samples being suitably selected [Fas83].

5 Alternative conceptions of CFL's

A few researchers have attacked the problem by using representations of CFL's that are not based on grammars. Yokomori gives an algorithm that learns *context-free expressions* [Yok88a], and Arikawa *et al.* present a method for inferring *regular elementary formal systems* [ASY92]. Both these papers bring solutions from other domains to bear on the problem of CFL-inference; however, their techniques have not yielded polynomial-time algorithms.

Let L be a CFL. Define a-substitution $(f_a(L))$ as the set of strings generated by taking every string $w \in L$ and replacing all occurrences of a in w by strings in L (the string that replaces one a and the string that replaces another a don't have to be the same). Define a-iteration (L^a) as the set of strings in $L \cup f_a(L) \cup f_a^2(L) \cup \cdots$ that don't contain any occurrence of a. Let Σ be an alphabet, and set $\Sigma' \triangleq \{\sigma' : \sigma \in \Sigma\}$ (that is, Σ' is a copy of Σ, except that each symbol $\sigma \in \Sigma$ is replaced by the new symbol σ').

Definition 4 *The* context-free expressions (CFE's) over the alphabet Σ *are strings over the alphabet* $\Sigma \cup \Sigma' \cup \{\Phi, +, (,)\}$ *defined as follows:*

1. *Φ, $a \in \Sigma$, and e are CFE's.*

2. *if E_1 and E_2 are CFE's and $a \in \Sigma$, then (E_1+E_2), (E_1E_2), and $(E_1)a'$ are CFE's.*

The mapping Lang *from CFE's to languages is as follows:*

1. $\text{Lang}(\Phi) = \phi$

2. $\text{Lang}(a) = \{a\}$

3. $\text{Lang}((E_1+E_2)) = \text{Lang}(E_1) \cup \text{Lang}(E_2), \text{Lang}(E_1E_2) = \text{Lang}(E_1)\text{Lang}(E_2)$, *and* $\text{Lang}(Ea') = (\text{Lang } E)^a$

For example, the CFE $(acb + ab)c'$ represents the set $\{a^nb^n : n \geq 1\}$.

Yokomori simply extends an algorithm that identifies regular expressions in the limit (using positive and negative data) to one that applies to CFE's. However, he states that the "time complexity is ... not less than NP-complete ... due to the essential difficulty of learning from example strings" [Yok89]

Arikawa *et al.* also "reduce" the problem of CFL inference to one that is already solvable. They come up with a formulation of CFL's as sets of symbols, predicate symbols, and definite clauses, and then employ Shapiro's Model Inference System. Once again, though, their algorithm is not efficient.

6 Learning Subclasses of CFL's

The most common tactic researchers have used to "get around" Gold's theorem is to consider classes of CFL's that are bounded in some fashion and thus do not contain all finite languages. We look at some of the results for restricted classes that have relied on rather powerful oracles (since these seem like good candidates for improvement) and at a polynomial-time reduction of learning linear languages to learning regular languages.

Angluin shows that *k-bounded* CFG's are identifiable in polynomial time using equivalence queries and *nonterminal membership* queries [Ang87]. Nonterminal membership queries propose a string w and a nonterminal A; the answer is "yes" if w is derivable from A and "no" otherwise. In effect, the learner is allowed to ask about the structure of the target grammar.

Ishizaka applies Angluin's algorithm to *simple deterministic languages* (SDL's) in such a way that nonterminal membership queries are no longer needed [Ish90]. Instead, the algorithm is allowed *extended equivalence queries*, which propose a grammar G, where G does *not* have to be a grammar for a SDL; the answer is "yes" if the target grammar is equivalent to G. Extended equivalence queries are not as strong as nonterminal membership queries in the sense that they don't (directly) convey structural information. However, equivalence of CFG's is undecidable, so the oracle answering the query must be quite powerful.

Yokomori gives a polynomial algorithm for learning SDL's that only conjectures SDL's, but his algorithm requires *prefix membership oracles* and *derivative membership oracles* [Yok88b]. Prefix membership queries ask whether a string is a prefix of any string in the target language L_*. Derivative queries propose two pairs of strings (u, v) and (u', v'); the answer is "yes" if and only if $\{w : uwv \in L_*\} = \{w : u'wv' \in L_*\}$.

The problem of learning *even linear languages* (ELL's) can be reduced to the problem of learning regular languages, as Takada has shown [Tak88]. The basic idea is to consider the productions of an even linear grammar (ELG) G to be labelled with symbols from a finite alphabet Γ. Let $\gamma = \gamma_1\gamma_2\cdots\gamma_k \in \Gamma^*$. Then, we write $S \Rightarrow_G^\gamma w$ if starting from the start symbol of G and applying the productions in G labelled γ_1, γ_2, $\ldots$, γ_k in order, we derive the string w. We define the *language generated by G with control set C* to be the set $L_C(G) \triangleq \{w \in \Sigma^* : S \Rightarrow_G^\gamma w, \gamma \in C\}$; the control set thus describes the possible derivations of strings in $L(G)$. Takada proves that for any alphabet, there is a *universal* even linear grammar G_0 that generates any ELL over that alphabet with a control set that is regular. Therefore, in order to learn an ELL, it suffices to learn the corresponding regular control set. Furthermore, we can "translate" between control strings and strings in the language and between control sets and even linear grammars in polynomial time; this means that equivalence and membership queries posed by a regular language learner can be rephrased in terms of the corresponding even linear language. This result is extended to linear languages in [Tak87]; a similar result is given in [Mäk90].

Other subclasses of CFL's that have been shown to be learnable are *structurally reversible* languages [Bur94], *one-counter* languages (languages accepted by deterministic one-counter automata) [BR87], *pivot* languages [FGHR69], and *very simple* languages [Yok91].

7 Bayesian Approaches

Finally, we discuss Bayesian methods. These all attempt to search the space of all stochastic CFG's, either exhaustively, i.e., by *enumeration*, or by some sort of heuristic search; in both cases the search is guided by some "goodness" function taking into account the intrinsic complexity or probability of a grammar and the match (or mismatch) between a grammar and the sample data (Feldman gives a formal definition of grammatical complexity, analogous to Blum's formulation of complexity, in [Fel72]).

Solomonoff's method uses an (incomprehensible) encoding scheme to convert grammars into strings, the advantage being that he could then assign *a priori* probabilities to both strings and grammars in a uniform manner. He gives no precise algorithm, but does suggest altering candidate grammars to try to find "nearby" grammars with a higher "goodness" value.

In his PhD thesis, Horning gives an enumerative algorithm that identifies SCFG's in the limit with probability one from stochastic data (data generated by an SCFG). Notice that this means his algorithm works on positive presentations (probability one saves the day!). He also presents the interesting idea of assigning *a priori* probabilities to grammars by means of a stochastic grammar for grammars. Wharton [Wha77] and Van der Mude and Walker [VW78] present other enumerative methods, which search *all* grammars with complexity less than a certain threshold. Needless to say, all three of these algorithms are quite inefficient.

The last paper we look at is that of Cook *et al.*, which discusses a hill-climbing algorithm [CRA76]. They give a complexity measure for strings which has the following property: among all strings of the same length that contain the same number of distinct symbols, the string with the greatest complexity is the one where all the symbols have the same frequency. Thus, the more "random" a string appears, the higher its complexity.

8 A Bestiary of Context-Free Grammars

Let $G = (N, \Sigma, P, S)$, where N is the set of nonterminals, Σ is the alphabet, P is the set of productions, and $S \in N$ is the start symbol. Capital letters denote nonterminals, lowercase letters in the beginning of the alphabet denote terminals, lowercase letters at the end of the alphabet denote strings in Σ^*, and Greek letters represent strings in $(N \cup \Sigma)^*$.

- G is an *operator* grammar iff no production has adjacent nonterminals in its right-hand side, i.e., there are no productions of the form $A \rightarrow \alpha BC\beta$. It is an *operator precedence* grammar if additionally there are three binary relations E, Y, and T on Σ such that

  i. The relation E (equal in precedence) holds between all terminals that are adjacent or separated by a nonterminal in a derivation string.

  ii. The relation Y (yield precedence) holds between the terminal preceding a handle and the leftmost terminal of the handle.

  iii. The relation T (take precedence) holds between the rightmost terminal of a handle and the terminal immediately following the handle.

  iv. E, Y, and T are pairwise disjoint.

  A *handle* of a sentential form is a substring α of the sentential form with the property that there exists a production $A \rightarrow \alpha$ and replacing α by A would produce the previous sentential form in a rightmost derivation [ASU88].

- G is *k-distinct and homogeneous* iff

  i. For all A and B which are not the start symbol,
  $P_k(A) \cap P_k(B) \neq \phi \Rightarrow A = B$, where

  $$L_k(\alpha) = \{u : \alpha \Rightarrow^* y \in \Sigma^* \text{ and } u = y \text{ if } |y| < k \text{ or } y = uv, u \in \Sigma^k, \text{ o.w.}\}$$

  is the *left terminal set of order k*, $R_k(\alpha)$ is defined similarly (reverse the order of u and v), and the *k-profile* $P_k(\alpha) \triangleq [L_k(\alpha), R_k(\alpha)]$. This is the *k-distinct* condition.

  ii. Let $A \neq S$. $A \rightarrow \alpha$ and $B \rightarrow \beta$ in $P \Rightarrow P_k(\alpha) = P_k(\beta)$. This is the *homogeneous* condition.

iii. S is never in the right-hand side of a production.

- G is *reversible* iff

  i. $A \to \alpha$ and $B \to \alpha$ in $P \Rightarrow A = B$.

  ii. $A \to \alpha B\beta$ and $A \to \alpha C\beta$ in $P \Rightarrow B = C$.

- G is *k-bounded* iff the right-hand side of every production contains at most k nonterminals.

- G is a *simple deterministic* grammar iff all productions are of the form $A \to a\alpha$ where $\alpha \in N^*$ and $|\alpha| \leq 2$, and also $A \to a\alpha$ and $A \to a\beta$ in $P \Rightarrow \alpha = \beta$. *Simple deterministic languages* are languages accepted by a 1-state deterministic pushdown automata by empty stack.

- G is *linear* iff all productions are of the form $A \to uBv$ or $A \to u$. G is *even linear* if all productions are of the form $A \to uBv$, where $|u| = |v|$.

- G is a *pivot* grammar (PG) iff Σ can be partitioned into Σ_o and Σ_p such that all productions are of the form $A \to BaC$, $A \to Bb$, $A \to bB$, and $A \to b$, where $a \in \Sigma_p$ and $b \in \Sigma_o$. LG's $\subset$ PG's $\subset$ CFG's.

- G is *very simple* iff it is in Greibach normal form and for every terminal a, there is exactly one rule of the form $A \to a\alpha$ in P. Very simple languages are a proper subset of SDL's.

- G is *stochastic* iff there is an assignment of weights, each weight being between zero and one inclusive, to the right-hand sides of productions such that for every nonterminal A, the weights of the right-hand sides of all the productions with A on their left-hand sides sum to 1.

References

[ASU88] Alfred V. Aho, Ravi Sethi, and Jeffrey D. Ullman, *Compilers: Principles, Techniques, and Tools.* Reading: Addison-Wesley (1988).

[Ang80] Dana Angluin, "Inductive inference of formal languages from positive data," *Info. Contr.* **45**(2), 117–135 (1980).

[Ang82] Dana Angluin, "Inference of reversible languages," *JACM* **2**(3), 741–765 (1982) .

[Ang87] Dana Angluin, “Learning k-bounded context-free grammars,” Yale Tech. Rpt RR-557 (1987).

[Ang88] Dana Angluin, “Queries and concept learning,” *Machine Learning* **2**(4), 319–342 (1988).

[Ang90] Dana Angluin, “Negative results for equivalence queries,” *Mach. Learning* **2**(2), 121–150 (1990) .

[AS83] Dana Angluin and Carl Smith, “Inductive inference: theory and methods,” *ACM Comput. Surveys* **2**(3), 237–269 (1983) .

[ASY92] S. Arikawa, T. Shinohara, and A. Yamamoto, “Learning elementary formal systems,” *Theoret. Comput. Sci* **2**(1), 91–113 (1992) ,

[Bie71] A. W. Biermann, “A grammatical inference program for linear languages,” *Proc.* 4^{rth} *Hawaii Int. Conf. System Sciences.* North Hollywood: Western Periodicals, 121–123 (1971).

[BF71] A. W. Biermann and J. A. Feldman, “A survey of results in grammatical inference,” *Int. Conf. Frontiers of Pattern Recognition*, Honolulu (1971).

[Bur94] Andrey Burago, “Learning structurally reversible context-free grammars from queries and counterexamples in polynomial time,” 7^{th} *COLT*, 140–146 (1994).

[BR87] Piotr Berman and Robert Roos, “Learning one-counter languages in polynomial time,” 28^{th} *FOCS*, 61–67 (1987).

[CL82] John Case and Christopher Lynes, “Machine inductive inference and language identification,” 9^{th} *ICALP*, Lecture Notes in Computer Science 140. Berlin: Springer-Verlag (1982).

[Cho57] Noam Chomsky, *Syntactic Structures.* The Hague: Moutin and Co. (1957).

[Cho63] Noam Chomsky, “Formal properties of grammars,” *Handbook of Mathematical Psychology*, Vol. II. New York: John Wiley and Sons, Inc. (1963).

[Cre71a] Stefano Crespi-Reghizzi, “An effective model for grammar inference,” *Information Processing 71*, 524–529 (1971).

[Cre71b] Stefano Crespi-Reghizzi, “Reduction of enumeration in grammar acquistion,” *Proc. 2^{nd} Int. Conf. Art. Intell.*, 546–552 (1971).

[Cre74] Stefano Crespi-Reghizzi, “Non-counting languages and learning,” *Proc. NATO Advanced Study Inst. on Computer Learning Processes*, 171–190 (1974).

[CRA76] Craig M. Cook, Azriel Rosenfeld, and Alan R. Aronson, “Grammatical inference by hill-climbing,” *Inf. Sciences* **2**(1), 59–80 (1976) .

[Cul85] Christopher Culy, “The complexity of the vocabulary of Bambara,” *Linguistics and Philosophy* **8**, 345–351 (1985).

[Fas80] Leona F. Fass, “A skeletal automata approach to the inference problem for context-free languages,” PhD dissert., Univ. Penn. (1980).

[Fas83] Leona F. Fass, “Learning context-free languages from their structured sentences”, *SIGACT News* **2**(3), 24–35 (1983) .

[Fel72] J. Feldman, “Grammatical inference and complexity,” *Inf. Contr* **2**(3), 244–262 (1972) .

[FGHR69] J. Feldman, J. Gips, J. J. Horning and S. Reder, “Grammatical inference and complexity,” Tech. Rpt. CS-125, Comput. Sci. Dept., Stanford (1969).

[Fu74] King-Sun Fu, *Syntactic Methods in Pattern Recognition.* New York: Academic Press (1974).

[Fu75] King-Sun Fu and Taylor R. Booth, “Grammatical Inference: introduction and survey”, Parts I and II,” *IEEE Trans. Systems, Man, and Cybernetics*, **SMC-5**(1) and (4), 95–111 and 409–423 (1975).

[Gol67] E. Mark Gold, “Language identification in the limit,” *Inf. Contr.* **10**(5), 447–474 (1967).

[Hor69] James Jay Horning, “A study of grammatical inference,” Tech. Rpt. CS 139, Computer Science Department, Stanford (1969).

[Hor71] James Jay Horning, “A procedure for grammatical inference,” *Information Processing 71*, 519–523 (1972).

[Ish90] Hiroki Ishizaka, “Polynomial time learnability of sample deterministic languages,” *Mach. Learning* **2**(2), 151–164 (1990) .

[Kno73] Bruce Knobe and Kathleen Knobe, "An algorithm for inferring context-free grammars," CS Dept. TR 73-7, Hebrew University (1973).

[Kno76] Bruce Knobe and Kathleen Knobe, "A method for inferring context-free grammars," *Inf. Contr.* **2**(2), 129–146 (1976) .

[Lam61] Sydney M. Lamb, "On the mechanization of syntactic analysis," *1961 Conferenceon Machine Translation of Languages and Applied Language Analysis* (National Physical Lab. Symp. 13), Vol. II., 674–685 (1961).

[LJ78] Leon S. Levy and Aravind K. Joshi, "Skeletal structural descriptions," *Inf. Contr.* **2**(2), 192–211 (1978) .

[McNu67] Robert McNaughton, "Parenthesis grammars," *JACM* **2**(3), 490–500 (1967) .

[Mäk90] Erkki Mäkinen, "The grammatical inference problem for the Szilard languages of linear grammars," *IPL* **2**(4), 203–206 (1990) .

[Pao69] Tsyh-Wen Lee Pao, "A solution of the syntactical induction-inference problem for a non-trivial subset of context-free languages," Moore School of Elec. Eng., Univ. Penn. Interim Tech Rpt 69-19 (1969).

[Pit88] Leonard M. Pitt and Manfred K. Warmuth, "Reductions among prediction problems: on the difficulty of predicting automata," 3^{rd} *Structure in Complexity Theory*, 60–69 (1988).

[RN88] V. Radhakrishnan and G. Nagaraja, "Inference of even linear grammars and its application to picture description languages," *Pattern Recognition* **2**(1), 55–62 (1988) .

[Sak92a] Yasumbumi Sakakibara, "Learning context-free grammars from structural data in polynomial time," *Theoret. Comp. Sci* **76**, 223–242 (1992).

[Sak92b] Yasubumi Sakakibara, "Efficient learning of context-free grammars from positive structural examples," *Inf. Comput* **2**(1), 23–60 (1992) .

[Sha83] E. Y. Shapiro, *Algorithmic Program Debugging.* Cambridge: MIT Press (1983).

[Shieb85] Stuart Shieber, "Evidence against the context-freeness of natural language," *Linguistics and Philosophy* **8**, 333–343 (1985).

[Shi90] Takeshi Shinohara, "Inductive inference from positive data is powerful," 3^{rd} *COLT*, 91–110 (1990).

[Smi79] C. H. Smith, "An inductive inference bibliography," Purdue Computer Sci. Dept. TR 323 (1979).

[Sol59] R. J. Solomonoff, "A new method for discovering the grammars of phrase structure languages," *Information Processing.* New York: UNESCO (1959).

[Sol64] R. J. Solomonoff, "A formal theory of inductive inference," *Inf. Contr* **7**, 1–22 and 224–254 (1964).

[Tak87] Yuji Takada, "A constructive method for grammatical inference of linear languages based on control sets," Research Report 78, Int. Inst. for Advanced Study in the Soc. Info Sci. (1987).

[Tak88] Yuji Takada, "Grammatical inference for even linear languages," *IPL* **2**(4), 193–199 (1988) .

[Tan84] Keisuke Tanatsugu, "A grammatical inference for harmonical linear languages," *Int. Journal of Computer and Information Sciences* **2**(5), 413–423 (1984) .

[Tan87] Keisuke Tanatsugu, "A grammatical inference for context-free languages based on self-embedding", *Bull. Informatics and Cybernetics* **2**(3-4), 149–163 (1987) .

[VW78] A. Van der Mude and Adrian Walker, "On the inference of stochastic regular grammars," *Inf. Contr* **2**(5), 310–329 (1978) .

[Wha77] R. M. Wharton, "Grammar enumeration and inference," *Inf. Contr* **2**(3), 253–272 (1977) .

[Yok88a] Takashi Yokomori, "Inductive inference of context-free langauges based on context-free expressions," *Int. J. Computer Math.* **24**, 115–140 (1988).

[Yok88b] Takashi Yokomori, "Learning simple languages in polynomial time," Tech. Rep., SIG-FAI, Japanese Society for AI (1988).

[Yok89] Takashi Yokomori, "Learning context-free languages efficiently – a report on recent results in Japan," in J. P. Jantke, ed., *Analogical and*

Inductive Inference, Lecture Notes in Artificial Intelligence 397. Berlin: Springer-Verlag, 104–123 (1989).

[Yok91] Takashi Yokomori, “Polynomial-time learning of very simple grammars from positive data,” 4^{rth} *COLT*, 213–227 (1991).