

# On the hardness of learning in the “Principles and Parameters” framework

Jacob Andreas

January 10, 2012

## Abstract

We investigate models for learning the class of context-free and context-sensitive languages (CFLs and CSLs). We begin with a discussion of some early hardness results which show that unrestricted language learning is impossible, and unrestricted CFL learning is computationally infeasible; we then briefly survey the literature on algorithms for learning restricted subclasses of the CFLs. Finally, we introduce a new family of subclasses, the principled parametric context-free grammars (and a corresponding family of principled parametric context-sensitive grammars), which roughly model the “Principles and Parameters” framework in psycholinguistics. We present three novel hardness results: first, that the PPCFGs are not efficiently learnable given equivalence and membership oracles, second, that the PPCFGs are not efficiently learnable from positive presentations unless  $P = NP$ , and third, that the PPCSGs are not efficiently learnable from positive presentations unless integer factorization is in  $P$ .

## 1 Introduction

A great deal modern psycholinguistics has concerned itself with resolving the problem of the so-called “poverty of the stimulus,” the claim that natural languages are unlearnable given only the data available to infants, and thus that some part of syntax must be “native” (i.e. prespecified) rather than learned. Gold’s theorem (described below), which states that there exists a superfinite class of languages which is not learnable in the limit from positive presentations, is often offered as proof of this fact (though the extent to which the theorem is psycholinguistically informative remains a contentious issue). [gor90]

But how is linguistic nativism represented in practice? One mechanism usually offered is the Chomskian “Principles and Parameters” framework [cho93], which suggests that there is a set of universal principles of grammar which are inherent in the structure of the brain. In this framework, the process of language learning simply consists of determining appropriate settings for a finite number of parameters which determine how those principles are applied.

While this problem is assumed to be computationally easier than generalized language learning, we are not aware of any previous work specifically aimed at studying the principles and parameters model in a computational setting. In this paper, we introduce a family of subclasses of the context-free languages which we believe roughly captures the intuition behind the Principles and Parameters model, and explore the difficulty of learning that model in various learning environments.

This paper has two parts: in the first, we present a brief survey of the existing literature on the hardness of language learning; in the second, we present three hardness results, one generic, one complexity-theoretic and one cryptographic, which suggest that the existence of a generalized algorithm for learning in the principles and parameters framework is highly unlikely. While we obviously cannot produce any psychologically definitive results in this setting, we at least hope to challenge the notion that the principles and parameters framework is somehow a *computationally* satisfying explanation of the language learning process.

## 2 Background

### 2.1 Definitions

We assume a basic understanding of automata theory, and in particular the construction and properties of context-free grammars and the context-free languages.

#### 2.1.1 Learnability in the limit

Gold begins by introducing a definition of language learning, as follows: [gol67]

**Definition 1.** Given a class of languages  $\mathcal{L}$  and an algorithm  $A$ , we say  $A$  **identifies  $\mathcal{L}$  in the limit from positive presentations** if  $\forall L, \forall i_1, i_2, \dots \in L$ , there is a time  $t$  such that for all  $u > t$ ,  $h_u = h_t = A(i_1, i_2, \dots, i_t)$ .

#### 2.1.2 Exact identification using queries

Modeling the language learning process as being entirely dependent on positive examples seems like a rather extreme model; it's useful to consider environments in which the learner has access to a much richer representation of the language. Angluin [ang90] describes a model of language learnability from oracle queries, as follows:

**Definition 2.** An **equivalence oracle** for a language  $L$  takes as input the representation of a language  $r(L)$  and outputs “true” if  $L = L^*$ , or some  $w \in L\Delta L^*$  (the symmetric difference of the languages) otherwise. There is an obvious equivalence, first pointed out by Littlestone [lit88], between the equivalence query model and the online mistake bound model.

**Definition 3.** A **membership oracle** for a language  $L$  with start symbol  $S$  takes a string  $w$ , and outputs true if  $S \Rightarrow^* w$  and false otherwise.

**Definition 4.** A **nonterminal membership oracle** for a language  $L$  takes a string  $w$  (not necessarily in  $L$ ) and a nonterminal  $A$ , and outputs whether  $A \Rightarrow^* w$  (i.e. whether the set of possible derivations with  $A$  as a start symbol includes  $w$ ).

**Definition 5.** A class of languages  $\mathcal{L}$  is **learnable from an equivalence oracle** (or analogously from an equivalence oracle and a membership oracle; i.e. from a “minimal adequate teacher”) if there exists a learning algorithm with runtime polynomial in the size of the representation of the class and length of the longest counterexample.

## 2.2 Hardness of language learning

**Theorem 1 (Gold).** *There exists a class of languages not learnable in the limit from positive presentations.*

*Proof sketch.* Construct an infinite sequence of languages  $L_1 \subset L_2 \subset \dots$ , all finite, and let  $L_\infty = \bigcup_i L_i$ . Suppose there existed some algorithm  $A$  that could identify each  $L_i$  from positive. Then there is a positive presentation of  $L_\infty$  that causes  $A$  to make an infinite number of mistakes. First present a set of examples, all in  $L_1$ , that force  $A$  to identify  $L_1$ . Then present a set of examples forcing it to identify  $L_2$ , then  $L_3$ , and so on. An infinite number of mistakes can be forced in this way, so  $L_\infty$  is not learnable in the limit.  $\square$

While space does not permit us to discuss the proof here, we also note the following important result for CFL learning:

**Theorem 2 (Angluin [ang80]).** *There exists a class of context-free languages with “natural” representations which are not learnable from equivalence queries in time polynomial in the size of the representation.*

## 2.3 Learnable subclasses of the CFLs

While this last result rules out the possibility of a general algorithm for learning CFLs, subsets of the CFLs have been shown to be learnable when given slightly more powerful oracles. These include simple deterministic languages [ish90], one-counter languages [ber87] and so-called very simple languages [yok91].

Particularly heartening is Angluin’s result that  $k$ -bounded CFGs can be learned in polynomial time if nonterminal membership queries are permitted. A proof of this fact is provided below.

### 2.3.1 $k$ -bounded CFGs are learnable

A  $k$ -bounded context-free grammar is a CFG where the right hand side of every production has at most  $k$  non-terminal symbols.

**Theorem 3** (Angluin [ang87]). *There is an algorithm that learns a grammar equivalent to any  $k$ -bounded CFG  $G$  using equivalence and nonterminal membership queries, and runs in time polynomial in the size of  $G$  and the length of the longest counterexample.*

Let  $l$  denote the language we are trying to learn. Begin with a set of empty productions  $P'$ . At each iteration, propose  $P'$  as the grammar. If  $P'$  is not correct, there are two cases to consider: either the counterexample is in  $L(G')$  but not  $l$ , or the counterexample is in  $l$  but not  $L(G')$ . In the first case, the parse tree is diagnosed for the erroneous production; in the second, the set of all possible candidate productions is computed and added to  $P'$ . Each of these steps is described in more detail below:

Removing productions: first a DAG describing a correct parse of the counterexample under  $G'$ . The graph is then recursively searched for a nonterminal  $B$  such that the yield of  $B$  in the graph is not possible in  $G$ ; this determination is made using a nonterminal membership query. Eventually some wrong production must be found; this production is eliminated from the set of candidates.

Adding productions: For every  $m \leq k$ , every factorization of the counterexample into  $2m + 1$  strings (which may be thought of as taking the form  $x_0 y_1 x_1 y_2 x_2 \cdots y_m x_m$ ) is computed. Note that there are only polynomially many such factorizations. Then all productions of the form

$$A \rightarrow x_0 A_1 x_1 A_2 \cdots A_m x_m$$

are added to  $P'$ .

Clearly, if the algorithm terminates, it has found a correct grammar; all that remains is to prove termination.

*Proof.* Note that on every adding stage, at least one of the productions added is correct, and so is never removed again. Thus, there will be at most  $|P|$  adding stages. At each adding stage, only a polynomial number of new productions are added. Finally, there can never be more removing stages than adding stages, as there must be some production to remove, so the total number of iterations must be at most  $2p(|P|)$ , where  $p$  is some polynomial.  $\square$

### 3 Principled Parametric Grammars

We now introduce a formal model of the “principles and parameters” framework described in the introduction.

### 3.1 Motivation

Before moving on to the details of the construction, it's useful to consider a few example “principles” and “parameters” suggested by proponents of the model.

**The pro-drop parameter:** does this language allow pronoun dropping? If PNP is a non-terminal symbol designating a pronoun, this parameter determines whether or not a rule of the form  $\text{PNP} \rightarrow \varepsilon$  exists in the language.

**The ergative/nominative parameter:** ergative languages distinguish between transitive and intransitive senses of verb by marking the subject, while nominative languages (like English) marks the object. In the first case, there is a rule of the form  $S \rightarrow \text{NP}_{\text{trans}} \text{VP}$ , and in the second case a rule of the form  $S \rightarrow \text{NP} \text{VP}_{\text{trans}}$ .

In each of these cases, the general idea seems to be that for each possible parameter setting, there is some finite set of context-free productions in the native grammar, from which only one must be selected as the element of the learned grammar. This leads very naturally to the following development of principled parametric context-free grammars as a model of the principles and parameters model.

### 3.2 Construction

**Definition 6.** An **n-principled, k-parametric context-free grammar**  $((n, k)\text{-PPCFG}) \Gamma$  is defined as the 4-tuple  $(V, \Sigma, \Pi, S)$ , where:

1.  $V$  is a finite alphabet of nonterminal symbols
2.  $\Sigma$  is a finite alphabet of terminal symbols
3.  $\Pi$  is a set of  $n$  production groups of the form

$$(A_{i,1} \rightarrow \alpha_{i,1}), \dots, (A_{i,j} \rightarrow \alpha_{i,j}), \dots, (A_{i,k} \rightarrow \alpha_{i,k})$$

where each  $\alpha \in (V \cup \Sigma)^*$ , i.e. is a finite sequence of terminals and nonterminals. Let  $\Pi_{i,j}$  denote the production  $(A_{i,j} \rightarrow \alpha_{i,j})$ .

4.  $S \in V$  is the start symbol.

**Definition 7.** A **parameter setting**  $p = (p_1, p_2, \dots, p_n)$  is a sequence of length  $n$ , with each  $p_i \in 1..k$ . Then define  $\Gamma_p$  to be the ordinary context-free grammar  $(V, \Sigma, R, S)$  with  $R = \{\Pi_{i,p_i} : i \in 1..n\}$ .

As usual, let  $L(G)$  denote the context free language represented by the CFG  $G$ . Then let  $\Lambda(\Gamma) = \{L(G) : \exists p : G = \Gamma_p\}$ .

**Definition 8.** An algorithm  $A$  **learns the PPCFGs from an equivalence oracle** if  $\forall$  PPCFGs  $\Gamma$  and languages  $l \in \Lambda(\Gamma)$ , after a finite number of oracle queries,  $A$  outputs some  $p$  such that  $L(\Gamma_p) = l$ , or concludes that no such  $p$  exists.

**Definition 9.** A **efficiently** learns the PPCFGs from an equivalence oracle if the number of oracle queries it makes is bounded by some polynomial function  $poly(n, k)$ .

**Definition 10.** Finally, a **principled parametric context-sensitive grammar** is defined exactly as above, with corresponding learning definitions, but with context-sensitive productions in each production group.

### 3.3 Equivalence

Some useful facts about the PPCFGs:

**Observation.** A “heterogeneous PPCFG” with a variable number of right hand sides can be transformed into a “homogeneous PPCFG” of the kind described above by “padding” out the shorter principles with duplicate rules (i.e. to insert an unambiguous production  $A \rightarrow \alpha$  into an  $(n, 2)$ -PPCFG, add to  $\Pi$  the production group  $(A \rightarrow \alpha), (A \rightarrow \alpha)$ ).

**Observation.** A  $(n, k)$ -PPCFG can be converted into an  $(n(k-1), 2)$ -PPCFG as follows: replace each principle

$$A \rightarrow (\alpha_1, \alpha_2, \dots, \alpha_k)$$

with a set of principles

$$\begin{aligned} A_1 &\rightarrow (\alpha_1, A_2) \\ A_2 &\rightarrow (\alpha_2, A_3) \\ &\vdots \\ A_{k-1} &\rightarrow (\alpha_{k-1}, \alpha_k) \end{aligned}$$

Thus without loss of generality we may treat every PPCFG as a  $(n, 2)$ -PPCFG. The conversion above results in only a polynomial increase in the number of principles, so any algorithm which is polynomial in  $n$ , and which assumes  $k = 2$ , can be used to solve  $k > 2$  with only a polynomial increase in running time. This also means that we may specify an individual language in a PPCFG by a bit string of length  $n$ .

Finally, note that a  $k$ -PPCFG with  $n$  rules contains at most  $k^n$  languages.

## 4 Generic hardness results for PPCFGs

We will construct a minimal adequate teacher  $T$  consisting of two oracles  $EQ$  (an equivalence oracle) and  $M$  (a membership oracle), such that any algorithm  $A$  requires an exponential number of queries to identify the correct parameter setting  $p$  from a PPCFG  $\Gamma$ .

**Theorem 4.** *Without qualification, there exists no algorithm  $A$  capable of learning the PPCFGs from equivalence queries and membership queries in polynomial time.*

*Proof.* Fix some number  $N$ . Construct the PPCFG  $\Gamma$  with

$$\begin{aligned} V &= X_i : i \in 1..N \\ S &= \text{START} \\ \Sigma &= \{0, 1\} \end{aligned}$$

and  $\Pi$  as defined as follows:

$$\begin{aligned} (\text{START} &\rightarrow X_1 X_2 \cdots X_N) \\ (X_k &\rightarrow 0, X_k \rightarrow 1) && \forall k \in 1..N \end{aligned}$$

Every parameter setting  $p$  in this grammar allows it to derive precisely 1 string: every production is deterministic. Consequently, the  $N$  possible settings of the grammar derive  $2^N$  unique strings. Given some algorithm  $A$  for learning PPCFGs, the procedure specified below describes an adversarial distinguisher for this PPCFG which forces the learner to make a total of  $2^N - 1$  queries.

Initially, there are a total of  $2^N$  grammars consistent with  $\Gamma$ . Whenever  $A$  makes an equivalence query  $\Gamma'$ ,  $A$  outputs the single string in  $L(\Gamma')$ . Whenever  $A$  makes a membership query  $w$ ,  $A$  outputs false.

After each query, the number of grammars still possible given the evidence provided so far decreases by precisely 1 (because each grammar is capable of producing only string), so after  $2^N - 1$  queries of either kind, the oracle must output true.

Thus, only after  $2^N - 1$  queries (superpolynomial in  $|\Gamma|$  and the length of the longest production) can the learner halt, so the grammar is not efficiently learnable from membership and equivalence queries.  $\square$

## 5 Complexity-theoretic hardness results for PPCFGs

We will construct a reduction from 3SAT to PPCFG learning. Let  $X = \{x_i\}$  be a set of variables and  $C = \{c_i\}$  be a set of clauses. Let us write  $x_j \in c_i$  if the  $j$ th  $i$ th clause is satisfied by the  $j$ th variable, and  $\bar{x}_j \in c_i$  if the  $i$ th clause is satisfied by the negation of the  $j$ th variable.

Then construct the PPCFG  $\Gamma$  with  $V = X \cup \{\text{START}\}, \Sigma = C, S = \text{START}$ , and  $\Pi$  with the following production groups:

$$\begin{aligned} (\text{START} &\rightarrow x_1 x_2 \cdots x_n) \\ (x_i &\rightarrow x_{i,T}), (x_i \rightarrow x_{i,F}) && \forall x_i \in X \\ (x_i &\rightarrow \varepsilon) && \forall x_i \in X \\ (x_{j,T} &\rightarrow c_i) && \forall c_i \in C, \forall x_j \in c_i \\ (x_{j,F} &\rightarrow c_i) && \forall c_i \in C, \forall \bar{x}_j \in c_i \end{aligned}$$

Note that only for production groups of the form  $(x_i \rightarrow x_{i,T}), (x_i \rightarrow x_{i,F})$  does the parameter setting change the resulting language. These groups may be thought of as assigning truth values to the variables.

**Lemma 1.** *If there exists some  $l \in \Gamma$  such that  $\forall c_i \in C : c_i \in l$ , then the 3SAT instance is satisfiable.*

*Proof.* Set  $x_i$  true if the rule  $x_i \rightarrow x_{i,T}$  is chosen, and false otherwise. For any  $c_i$  in the language, there is a derivation from  $\text{START} \Rightarrow^* c_i$ ; moreover, that derivation *must* have the following form:

$$\begin{aligned} \text{START} &\Rightarrow x_1 x_2 \cdots x_n \\ &\Rightarrow x_j \\ &\Rightarrow x_{j,a} \\ &\Rightarrow c_i \end{aligned}$$

Then  $x_j$  satisfies  $c_i$ . □

**Lemma 2.** *If the 3SAT instance is satisfiable, there exists some  $l \in \Gamma$  such that  $\forall c_i \in C : c_i \in l$ .*

*Proof.* Choose the rule  $(x_i \rightarrow x_{i,T})$  if  $x_i$  is set true in the satisfying assignment, and  $(x_i \rightarrow x_{i,F})$  if  $x_i$  is set false. These settings determine  $l$ . Then, consider any clause  $c_i$ . There is some  $x_j$  which satisfies the clause; let  $a$  be that truth assignment. Then a derivation of the same form given in Lemma 1 above must exist. □

**Theorem 5.** *If  $P \neq NP$ , no efficient algorithm exists for learning PPCFGs from positive presentations.*

*Proof.* Assume that there exists some algorithm  $A$  which efficiently learns the PPCFGs from positive presentations. We will use  $A$  to construct a SAT solver  $S$  by simulating the oracle. Construct  $\Gamma$  from the SAT instance as described above. Then  $S$ 's interaction with  $A$  takes the following form:

Present  $A$  with the positive presentation  $C$  (i.e. whenever  $A$  asks for an example, give it some  $c \in C$  it hasn't seen yet until there are no more). If  $A$  outputs a failure signal,  $S$  outputs **FALSE** and halts.

Otherwise,  $A$  outputs some  $p$  such that  $\forall c_i \in C, c_i \in \Gamma_p$ . In this case, all clauses are satisfied, and  $S$  outputs **TRUE** and halts.

By assumption, after observing polynomially many positive presentations, and performing polynomially many computations,  $A$  outputs a parameter setting  $p$  which produces every  $c_i \in C$ , or a signal indicating no such assignment exists. From Lemmas 1 and 2, such a  $p$  exists if and only if the SAT instance is satisfiable. Thus  $S$  determines in a polynomial number of steps whether the SAT instance is satisfiable, and the existence of  $A$  implies  $P = NP$ . □

## 6 Cryptographic hardness results for PPCSGs

We will construct another reduction, this time from integer factorization to PPCSG learning. Let  $N$  be a product of two  $(n - 1)$ -digit primes.

Now let  $B$  be a set of non-terminal symbols  $B_0, B_1, B_2, \dots, B_{\lceil \sqrt{N} \rceil}$ . Let  $A, Z$  be a set of non-terminal symbols  $A_0, A_1, \dots, A_{\lceil \lg N \rceil}$  and  $Z_0, Z_1, \dots, Z_{\lceil \lg N \rceil}$  respectively. Then construct the PPCSG  $\Gamma$  with

$$\begin{aligned} V &= A \cup Z \cup B \cup \{\mathbf{s}\} \\ \Sigma &= \{a_k\} & \forall k \in 0..\lceil \lg N \rceil \\ S &= \mathbf{s} \end{aligned}$$

and  $\Pi$  with the following production groups:

$$\begin{aligned} (\mathbf{s} &\rightarrow B_{\lceil \lg \sqrt{N} \rceil}) \\ (B_0 &\rightarrow A_0), (B_0 \rightarrow \varepsilon) \\ (B_j &\rightarrow B_{j-1}), (B_j \rightarrow A_j B_{j-1}) & \forall j \in 1..\lceil \lg \sqrt{N} \rceil \\ (A_k A_k &\rightarrow A_k Z_k) \\ (A_k Z_k &\rightarrow A_{k+1} Z_k) & \forall k \in 0..\lceil \lg N \rceil \\ (A_{k+1} Z_k &\rightarrow A_{k+1}) \\ (A_k &\rightarrow a_k) \end{aligned}$$

where  $A^x$  represents a string of  $x$   $A$ s. Intuitively, the parameter settings in this grammar  $(B_j \rightarrow B_{j-1}), (B_j \rightarrow A_0^{2^k} B_{j-1})$  fix some number  $m$  between 1 and  $\sqrt{N}$ , and require that all outputs be a string whose length is some integer multiple of  $m$ . The last three rules then allow this output to be collapsed into a representation that requires only a logarithmic number of bits to store.

Let  $l$  be the language consisting of the single string  $w$ , where  $w$  is the concatenation of every  $a_i$  such that the  $i$ th digit of the binary representation of  $w$  is 1.

**Lemma 3.** *Given a parameter setting  $s$  for  $\Gamma$ , for each production group  $(B_j \rightarrow B_{j-1}), (B_j \rightarrow A_j B_{j-1})$ , let  $t_i = 0$  if the first setting is chosen and 1 if the second setting is chosen. Let  $T$  be the number whose binary representation is given by the  $t_i$ s. Then, given numbers  $p$  and  $q$ ,  $pq = N$  if and only if  $p = T$  or  $q = T$ .*

*Proof.* As discussed above, any string in the produced by the grammar can be expanded into a string of the form  $(a_0^T)^m$  for some integer  $m$ . If  $p = T$ , then  $m = q$  and vice-versa.  $\square$

**Theorem 6.** *If integer factorization is hard, no efficient algorithm exists for learning random PPCSGs with non-negligible probability from an equivalence oracle.*

*Proof.* Assume that there exists some algorithm  $A$  which, given  $\Gamma$  and the positive presentation of the single string  $w$  as specified above, outputs a parameter setting for  $\Gamma$  such that  $w \in L(\Gamma_w)$  with probability  $p$  after a polynomial number of computations. Then we will construct a factorizer  $F$  that decomposes  $N$  into  $p$  and  $q$ .

$A$  is given the positive presentation of the single string  $w$ . If there is some parameter assignment  $p$  such that  $Tm = N$  as described above,  $S$  outputs  $(T, n)$  and halts. Otherwise,  $A$  outputs a failure signal and halts.

From Lemma 3 above, if a correct  $p$  is found then  $mT = N$ , so if  $A$  can find a parameter setting in polynomial time then this algorithm finds a factorization in polynomial time.  $\square$

This final proof is neither particularly interesting or satisfying: even the task of finding a derivation in a CSG is known to be PSPACE-complete (though it’s easy to see that a polynomial-time parsing algorithm for this particular family of grammars exists). Note that the only context-sensitive production groups employed in this production are used to guarantee a compact encoding of  $w$ ; we suspect that there is an alternative way of constructing this “grammar arithmetic” that requires only context-free rules. We thus close with the following conjecture:

**Conjecture 1.** If integer factorization is hard, then there is no polynomial-time algorithm for learning random PPCFGs with non-negligible probability from positive presentations.

## 7 Conclusion

We have introduced a new model, the principled parametric context-free (also context-sensitive) grammars as a model of the “Principles and Parameters” model in psycholinguistics, and presented three novel hardness-of-learning results for the class of PPCFGs and PPCSGs. While these results certainly do not demonstrate definitively that learning under the Principles and Parameters framework is completely impossible (all that is required for human language learning to be possible is that one PPCFG be efficiently learnable), we have demonstrated that there is likely no generic algorithm for learning a class of PPCFGs given either oracle and membership queries or a positive presentation. In general, these results demonstrate that even radically restricting the class potential grammars does not guarantee a successful outcome when attempting to learn context-free grammars.

## Acknowledgments

We would like to thank Prof. John McWhorter for useful feedback on the linguistic implications (and non-implications) of the proofs presented above.

## References

- [ang80] Angluin, Dana. “Inductive inference of formal languages from positive data.” *Info. Contr.* 1980, Vol. 45, No. 2.
- [ang87] Angluin, Dana. “Learning  $k$ -bounded context-free grammars.” Yale Technical Report RR-557, 1987.
- [ang90] Angluin, Dana. “Negative Results for Equivalence Queries.” In *Machine Learning* 1990, Vol. 2, No. 2.
- [ber87] Berman, Piotr, and Roos, Robert. “Learning one-counter languages in polynomial time.” In *Foundations of Computer Science*, 1987.
- [cho93] Chomsky, Noam and Lasnik, Howard. “Principles and Parameters Theory.” In *Syntax: An International Handbook of Contemporary Research*, Berlin: De Gruyter.
- [gol67] Gold, E. Mark. “Language Identification in the Limit.” In *Information and Control* 1967, Vol. 10, No. 5.
- [gor90] Gordon, Peter. “Learnability and Feedback.” In *Developmental Psychology* 1990, Vol. 26, No. 2.
- [ish90] Ishizawa, Hiroki. “Polynomial Time Learnability of Simple Deterministic Languages.” In *Machine Learning* 1990, Vol. 5, No. 2.
- [lit88] Littlestone, Nick. “Learning Quickly When Irrelevant Attributes Abound.” In *Machine Learning* 1988, Vol. 2, No. 1.
- [yok91] Yokomori, Takashi. “Polynomial Time Learning of Very Simple Grammars from Positive Data.” In proceedings of *COLT*, 1991.