Journal of Machine Learning Research 5 (2004) 473-497 Submitted 10/03; Revised 3/04; Published 5/04

# PAC-learnability of Probabilistic Deterministic Finite State Automata

**Alexander Clark** ASC@ACLARK.DEMON.CO.UK
*ISSCO/TIM, Université de Genève 40*
*Bvd du Pont d'Arve CH-1211*
*Genève 4, Switzerland*

**Franck Thollard** THOLLARD@UNIV-ST-ETIENNE.FR
*EURISE, Université Jean Monnet, 23*
*Rue du Docteur Paul Michelon*
*42023 Saint-Etienne Cédex 2, France*

**Editor:** Dana Ron

## Abstract

We study the learnability of Probabilistic Deterministic Finite State Automata under a modified PAC-learning criterion. We argue that it is necessary to add additional parameters to the sample complexity polynomial, namely a bound on the expected length of strings generated from any state, and a bound on the distinguishability between states. With this, we demonstrate that the class of PDFAs is PAC-learnable using a variant of a standard state-merging algorithm and the Kullback-Leibler divergence as error function.

**Keywords:** Grammatical inference, PAC-learnability, finite state automata, regular languages.

## 1. Introduction

Probabilistic Deterministic Finite State Automata (PDFAs) are widely used in a number of different fields, including Natural Language Processing (NLP), Speech Recognition (Mohri, 1997) and Bio-informatics (Durbin et al., 1999). The deterministic property here means that at any point in producing a string they are in a single state, which allows for very efficient implementations and accounts for much of their appeal. Efficient algorithms for inducing the models from data are thus important. We are interested primarily in algorithms which learn from positive data under stochastic presentation, where the data are drawn from the distribution defined by the target automaton. In addition we are concerned with automata that generate finite strings of unbounded length. This is necessary in many problem classes, particularly in NLP, where the sequences can be words or sentences with particular global structures.

Here we provide a proof that a particular algorithm Probably Approximately Correctly (PAC) learns the class of PDFAs, using the Kullback-Leibler divergence as the error function, when we allow the algorithm to have amounts of data polynomial in three quantities associated with the complexity of the problem: first, the number of states of the target, secondly the distinguishability of the automata — a lower bound on the $L_\infty$ norm between the suffix distributions of any pair of states of the target — and thirdly a bound on the expected length of strings generated from *any* *state* of the target, a quantity that can be used to bound the variance of the length of the strings generated by the target. The algorithm uses polynomial amounts of computation. We will motivate these additional parameters to the sample complexity polynomial with reference to specific counter-examples.

The algorithm we present here is a state-merging algorithm of a fairly standard type (Carrasco and Oncina, 1994; Ron et al., 1995; Thollard et al., 2000; Kermorvant and Dupont, 2002). The convergence properties of this class of algorithm have been studied before, but proofs have been restricted either to the subclass of acyclic automata (Ron et al., 1995, 1998), or using only the identification in the limit with probability one paradigm (de la Higuera et al., 1996; Carrasco and Oncina, 1999; de la Higuera and Thollard, 2000), which is generally considered less interesting as a guide to their practical usefulness. We wish to note here that we shall follow closely the notation and techniques of Ron et al. (1995). Unfortunately, our technique is too different to merely present modifications to their algorithm, but in points we shall adopt wholesale their methods of proof. We shall also use notation as close as possible to the notation used in their paper.

### 1.1 Learning Paradigm

We are interested in learning in the Probably Approximately Correct (PAC) framework (Valiant, 1984), more specifically in cases where the amount of data and the amount of computation required can be bounded by polynomials in the various parameters, including some characterisation of the complexity of the target.

We will use, as is standard, the Kullback-Leibler Divergence (KLD) (Cover and Thomas, 1991) between the target and the hypothesis to measure the error. Intuitively, this is the hardest measure to use since it is unbounded, and bounds other plausible distance measures (quadratic, variational etc.) (Cover and Thomas, 1991; Abe et al., 2001).

The problem class $\mathcal{C}$ is a subset of the set of all distributions over $\Sigma^*$, where $\Sigma$ is a finite alphabet. The algorithm is presented with a sequence $S_m$ of $m$ strings from $\Sigma^*$ that are drawn identically and independently from the target distribution $c$. Such a sequence is called a sample. Given this sample, the algorithm returns a hypothesis $H(S_m)$.

**Definition 1 (KL-PAC)** *Given a class of stochastic languages or distributions $\mathcal{C}$ over $\Sigma^*$, an algorithm $A$* KL-Probably Approximately Correctly (KL-PAC)-learns *$\mathcal{C}$ if there is a polynomial $q$ such that for all $c$ in $\mathcal{C}$, all $\varepsilon > 0$ and $\delta > 0$, $A$ is given a sample $S_m$ and produces a hypothesis $H$, such that $Pr[D(c||H) > \varepsilon] < \delta$ whenever $m > q(1/\varepsilon, 1/\delta, |c|)$, where $|c|$ is some measure of the complexity of the target, with running time bounded by a polynomial in $m$ plus the total length of the strings in $S_m$.*

Note here two points: first, the examples are drawn from the target distribution so this approach is very different from the normal distribution-free approach. Indeed here we are interested in learning the distribution itself, so the approach is closely related to the traditional problems of density estimation, and also to the field of language modelling in speech recognition and NLP. It can also be viewed as a way of learning languages from positive samples under a restricted class of distributions, which is in line with other research which has found that the requirement to learn under all distributions is too stringent. Secondly, we are concerned with distributions over $\Sigma^*$ not over $\Sigma^n$, for some fixed $n$, which form a finite set, which distinguishes us from Ron et al. (1995). Note that though Ron et al. (1995) study distributions over $\Sigma^*$ since they study acyclic automata of bounded size and depth, the distributions are, as a result, limited to distributions over $\Sigma^n$.

## 1.2 Negative Results

For PAC-learning stochastic languages there are some negative results that place quite strict limits on how far we can hope to go. First, let us recall that the stochastic languages generated by PDFAs are a proper subclass of the class of all stochastic regular languages. The results in Abe and Warmuth (1992) establish that robust learning of general (non-deterministic) finite state automata is hard. This strongly suggests that we cannot hope to learn all of this class, though the class of Probabilistic Residual Finite State Automata (Esposito et al., 2002) is a promising intermediate class between stochastic deterministic regular languages and stochastic regular languages. Secondly, Kearns et al. (1994) show that under certain cryptographic assumptions it is impossible to efficiently learn PDFAs defining distributions over two letters. They define a correspondence between noisy parity functions and a certain subclass of automata. Since the complexity results they rely upon are generally considered to be true, this establishes that the class of all PDFAs is not PAC-learnable using polynomial computation. These results apply to distributions over $\Sigma^n$ for some fixed $n$, and thus also to our more general case of distributions over $\Sigma^*$. However, Ron et al. (1995) show that if one restricts one's attention to a class of automata that have a certain distinguishability criterion between the states, that we shall define later, it is possible to PAC-learn acyclic PDFAs. Formally, there are two ways to define this: either one defines a subclass of the problem class where the distinguishability is bounded by some inverse polynomial of the number of states or we allow the sample complexity polynomial to have another parameter. We follow Ron et al. (1995) in using the latter approach.

However the extension from acyclic PDFAs to all PDFAs produces an additional problem. As we shall see, the KLD between the target and the hypothesis can be decomposed into terms based on the contributions of particular states. The contribution of each state is related to the expected number of times the target automaton visits the state and not the probability that the target automaton will visit this state. Therefore there is a potential problem with states that are both rare (have a low probability of being reached) and yet have a high expected number of visits, a combination of properties that can happen, for example, when a state is very rare but has a transition to itself with probability very close to one. If the state is very rare we cannot guarantee that our sample will contain any strings which visit the state, but yet this state can make a non-negligible contribution to the error.

In particular, as we show in Appendix A, it is possible to construct families of automata, with bounded expected length, that will with high probability produce the same samples, but where some targets must have large KLD from any hypothesis. We refer the reader to the appendix for further details. This is only a concern with the use of the KLD as error function; with other distance measures the error on a set of strings with low aggregate probability is also low. In particular with the variation distance, while it is straightforward to prove a similar result with a bound on the overall expected length, we conjecture that it is also possible with no length bound at all – i.e. with a sample complexity that depends only on the number of states, the distinguishability and the alphabet size.

Accordingly, we argue that, in the case of learning with the KLD, it is necessary to accept an upper bound on the expected length of the strings from any state of the target automaton, or a bound on the expected length of strings from the start state and a bound on the variance. It also seems reasonable that in most real-world applications of the algorithm, the expectation and variance of the length will be close to the mean observed length and sample variance.

### 1.3 Positive Results

Carrasco and Oncina (1999) proposed a proof of the identification in the limit with probability one of the structure of the automaton. In de la Higuera and Thollard (2000), the proof of the identification of the probabilities was added achieving the complete identification of the class of the PDFA with rational probabilities. With regard to the more interesting PAC-learnability criterion with respect to the KLD (KL-PAC), Ron et al. proposed an algorithm that can KL-PAC-learn the class of distinguishable acyclic automata. The distinguishability is a guarantee that the distributions generated from any state differ by at least $\mu$ in the $L_\infty$ norm. This is sufficent to immunise the algorithm against the counterexample discussed by Kearns et al. (1994).

Our aim is to extend the work of Ron et al. (1995) to the full class of PDFAs. This requires us to deal with cycles, and with the ensuing problems caused by allowing strings of unbounded length, since with acyclic automata a bound on the number of states is also a bound on the expected length. These are of three types: first, the support of the distribution can be infinite, which rules out the direct application of Hoeffding bounds at a particular point in the proof; secondly, the automaton can be in a particular state more than once in the generation of a particular string, which requires a different decomposition of the KLD, and thirdly deriving the bound on the KLD requires slightly different constraints. Additionally, we simplify the algorithm somewhat by drawing new samples at each step of the algorithm, which avoids the use of "reference classes" in Ron et al. (1995), which are necessary to ensure the independence of different samples.

The article is organized as follow: we first start with the definitions and notations we will use (Section 2). The algorithm is then presented in Section 3 and its correctness is proved in Section 4. We then conclude with a discussion of the relevance of this result. The reader will find on Page 481 a glossary for the notation.

## 2. Preliminaries

We start by defining some basic notation regarding languages, distributions over languages and finite state automata of the type we study in this paper.

### 2.1 Languages

We have a finite alphabet $\Sigma$, and $\Sigma^*$ is the free monoid generated by $\Sigma$, i.e. the set of all words with letters from $\Sigma$, with $\lambda$ the empty string (identity). For $s \in \Sigma^*$ we define $|s|$ to be the length of $s$. The subset of $\Sigma^*$ of strings of length $d$ is denoted by $\Sigma^d$. A distribution or stochastic language $D$ over $\Sigma^*$ is a function $D : \Sigma^* \to [0,1]$ such that $\sum_{s\in\Sigma^*} D(s) = 1$. The *Kullback-Leibler Divergence* (KLD) is denoted by $D_{KL}(D_1||D_2)$ and defined as

$$D_{KL}(D_1||D_2) = \sum_s D_1(s) \log\left(\frac{D_1(s)}{D_2(s)}\right). \tag{1}$$

The $L_\infty$ norm between two distributions is defined as

$$L_\infty(D_1, D_2) = \max_{s\in\Sigma^*} |D_1(s) - D_2(s)|.$$

We will use $\sigma$ for letters and $s$ for strings.

If $S$ is a multiset of strings from $\Sigma^*$ for any $s \in \Sigma^*$ we write $S(s)$ for the multiplicity of $s$ in $S$ and define $|S| = \sum_{s\in\Sigma^*} S(s)$, and for every $\sigma \in \Sigma$ define $S(\sigma) = \sum_{s\in\Sigma^*} S(\sigma s)$. We also write $S(\zeta) = S(\lambda)$. We will write $\hat{S}$ for the empirical distribution of a non-empty multiset $S$ which gives to the string $s$ the probability $S(s)/|S|$. This notation is slightly ambiguous for strings of length one; we will rely on the use of lower-case Greek letters to signify elements of $\Sigma$ to resolve this ambiguity.

### 2.2 PDFA

A probabilistic deterministic finite state automaton is a mathematical object that stochastically generates strings of symbols. It has a finite number of states one of which is a distinguished start state. Parsing or generating starts in the start state, and at any given moment makes a transition with a certain probability to another state and emits a symbol. We have a particular symbol and state which correspond to finishing.

**Definition 2** *A PDFA $A$ is a tuple $(Q, \Sigma, q_0, q_f, \zeta, \tau, \gamma)$ , where*

- *$Q$ is a finite set of states,*
- *$\Sigma$ is the alphabet, a finite set of symbols,*
- *$q_0 \in Q$ is the single initial state,*
- *$q_f \notin Q$ is the final state,*
- *$\zeta \notin \Sigma$ is the final symbol,*
- *$\tau : Q \times \Sigma \cup \{\zeta\} \to Q \cup \{q_f\}$ is the transition function and*
- *$\gamma : Q \times \Sigma \cup \{\zeta\} \to [0,1]$ is the next symbol probability function. $\gamma(q,\sigma) = 0$ when $\tau(q,\sigma)$ is not defined.*

We will sometimes refer to automata by the set of states. All transitions that emit $\zeta$ go to the final state. In the following $\tau$ and $\gamma$ will be extended to strings recursively as follows:

$$\begin{aligned}
\tau(q, \sigma_1\sigma_2 \ldots \sigma_k) &= \tau(\tau(q,\sigma_1), \sigma_2 \ldots \sigma_k), \\
\gamma(q, \sigma_1\sigma_2 \ldots \sigma_k) &= \gamma(q,\sigma_1) \times \gamma(\tau(q,\sigma_1), \sigma_2 \ldots \sigma_k).
\end{aligned}$$

Also we define $\tau(q,\lambda) = q$ and $\gamma(q,\lambda) = 1$. If $\tau(q_0, s) = q$ we say that *$s$ reaches $q$*.

The sum of the output probabilities from each states must be one, so for all $q \in Q$,

$$\sum_{\sigma\in\Sigma\cup\{\zeta\}} \gamma(q,\sigma) = 1.$$

Assuming further that there is a non-zero probability of reaching the final state from each state, i.e.

$$\forall q \in Q\ \exists s \in \Sigma^* : \tau(q, s\zeta) = q_f \wedge \gamma(q, s\zeta) > 0,$$

the PDFA then defines a probability distribution over $\Sigma^*$, where the probability of generating a string $s \in \Sigma^*$ is

$$P^A(s) = \gamma(q_0, s\zeta). \tag{2}$$

We will also use $P_q^A(s) = \gamma(q, s\zeta)$ which we call the suffix distribution of the state $q$. We will omit the automaton symbol when there is no risk of ambiguity. Note that $\gamma(q_0, s)$ where $s \in \Sigma^*$ is the prefix probability of the string $s$, i.e. the probability that the automaton will generate a string that *starts* with $s$.

**Definition 3 (Distinguishability)** *For $\mu > 0$ two states $q_1, q_2 \in Q$ are $\mu$-distinguishable if there exists a string $s$ such that $|\gamma(q_1, s\zeta) - \gamma(q_2, s\zeta)| \geq \mu$. A PDFA $A$ is $\mu$-distinguishable if every pair of states in it is $\mu$-distinguishable.*

Note that any PDFA has an equivalent form in which indistinguishable states have been merged, and thus $\mu > 0$.

## 3. Algorithm

We will now describe the algorithm we use here to learn PDFAs. It is formally described in pseudocode starting at Page 481.

We are given the following parameters

- an alphabet $\Sigma$, or strictly speaking an upper bound on the size of the alphabet,
- an upper bound on the expected length of strings generated from any state of the target $L$,
- an upper bound on the number of states of the target $n$,
- a lower bound $\mu$ for the distinguishability,
- a confidence $\delta$, and
- a precision $\varepsilon$.

We start by computing the following quantities. First, we compute $m_0$, which is a threshold on the size of a multiset. When we have a multiset whose size is larger than $m_0$, which will be a sample drawn from a particular distribution, then with high probability the empirical estimates derived from that sample will be sufficiently close to the true values. Secondly, we compute $N$, which is the size of the sample we draw at each step of the algorithm. Finally, we compute $\gamma_{min}$, which is a small smoothing constant. These are defined as follows, using some intermediate variables to simplify the expressions slightly:

$$\gamma_{min} = \frac{\varepsilon}{4(L+1)(|\Sigma|+1)}, \tag{3}$$

$$\delta' = \frac{\delta}{2(n|\Sigma|+2)}, \tag{4}$$

$$\varepsilon_1 = \frac{\varepsilon^2}{16(|\Sigma|+1)(L+1)^2}, \tag{5}$$

$$m_0 = \max\left(\frac{8}{\mu^2}\log\left(\frac{96n|\Sigma|}{\delta'\mu}\right), \frac{1}{2\varepsilon_1^2}\log\left(\frac{12n|\Sigma||\Sigma+1|}{\delta'}\right)\right), \tag{6}$$

$$\varepsilon_3 = \frac{\varepsilon}{2(n+1)\log(4(L+1)(|\Sigma|+1)/\varepsilon)}, \tag{7}$$

$$N = \frac{4n|\Sigma|L^2(L+1)^3}{\varepsilon_3^2}\max\left(2n|\Sigma|m_0, 4\log\frac{1}{\delta'}\right). \tag{8}$$

The basic data structure of the algorithm represents a digraph $G = (V, E)$ with labelled edges, $V$ being a set of vertices (or nodes) and $E \subseteq V \times \Sigma \times V$ a set of edges. The graph holds our current hypothesis about the structure of the target automaton. We have a particular vertex in the graph, $v_0 \in V$ that corresponds to the initial state of the hypothesis. Each arc in the graph is labelled with a letter from the alphabet, and there is at most one edge labelled with a particular letter leading from any node. Indeed, the graph can be thought of as a (non-probabilistic) deterministic finite automaton, but we will not use this terminology to avoid confusion. We will use $\tau_G(v,\sigma)$ as a transition function in this graph to refer to the node reached by the arc labelled with $\sigma$ that leads from $v$, if such a node exists, and we will extend it to strings as above. At each node $v \in V$ we associate a multiset of strings $S_v$ that represents the suffix distribution of the state that the node represents. At any given moment in the algorithm we wish the graph to be isomorphic to a subgraph of the target automaton. Initially we have a single node that represents the initial state of the target automaton, together with a multiset that is just a sample of strings from the automaton.

At each step we are given a multiset of $N$ data points (i.e. strings from $\Sigma^*$) generated independently by a target PDFA $T$. For each node $u$ in the graph, and each letter $\sigma$ in the alphabet that does not yet label an arc out of $u$, we hypothesize a candidate node. We can refer to this node by the pair $(u,\sigma)$. The first step is to compute the multiset of suffixes of that node. For each string in the sample, we trace its path through the graph, deleting symbols from the front of the string as we proceed, until we arrive at the node $u$, or the string is empty, in which case we discard it. If the string then starts with $\sigma$ we delete this letter and then add the resulting string to the multiset associated with $(u,\sigma)$. Intuitively, this should be a sample from the suffix distribution of the relevant state. More formally, for any input string $s$, if $\tau_G(v_0,s)$ is defined then we discard the string. Otherwise we take the longest prefix $r$ such that there is a node $u$ such that $\tau_G(v_0,r) = u$ and $s = r\sigma t$ and add the string $t$ to the multiset of the candidate node $(u,\sigma)$. If this multiset is sufficiently large (at least $m_0$), then we compare it with each of the nodes in the graph. The comparison operator computes the $L_\infty$-norm between the empirical distributions defined by the multisets. When we first add a node to the graph we keep with it the multiset of strings it has at that step, and it remains with this multiset for the rest of the algorithm.

**Definition 4 (Candidate node)** *A candidate node is a pair $(u,\sigma)$ where $u$ is a node in the graph and $\sigma \in \Sigma$ where $\tau_G(u,\sigma)$ is undefined. It will have an associated multiset $S_{u,\sigma}$. A candidate node $(u,\sigma)$ and a node $v$ in a hypothesis graph $G$ are similar if and only if, for all strings $s \in \Sigma^*$, $|S_{u,\sigma}(s)/|S_{u,\sigma}| - S_v(s)/|S_v|| \leq \mu/2$, i.e. if and only if $L_\infty(\hat{S}_{u,\sigma}, \hat{S}_v) \leq \mu/2$.*

We will later see that the value of $\mu/2$, given that any two states in the target are at least $\mu$ apart in the $L_\infty$-norm, allows us to ensure that the nodes are similar if and only if they are representatives of the same state. We will define this notion of representation more precisely below. If the candidate node $(u,\sigma)$ is similar to one of the nodes in the graph, say $v$ then we add an arc labelled with $\sigma$ from $u$ to $v$. If it is not similar to any node, then we create a new node in the graph, and add an arc labelled with $\sigma$ from $u$ to the new node. We attach the multiset to it at this point. We then delete all of the candidate nodes, sample some more data and continue the process, until we draw a sample where no candidate node has sufficiently large a multiset.

**Completing the Graph** If the graph is incomplete, i.e. if there are strings not accepted by the graph, we add a new node called the ground node which represents all the low frequency states. We then complete the graph by adding all possible arcs from all states leading to the ground node, including from the ground node to itself. Since the hypothesis automaton must accept every string, every state must have an arc leading out of it for each letter in the alphabet. We then define for each node in the graph a state in the automaton. We add a final state $\hat{q}_f$, together with a transition labelled with $\zeta$ from each state to $\hat{q}_f$. The transition function $\tau$ is defined by the structure of this graph.

**Estimating Probabilities** The transition probabilities are then estimated using a simple additive smoothing scheme (identical to that used by Ron et al., 1995). For a state $u$, $\sigma \in \Sigma \cup \{\zeta\}$,

$$\hat{\gamma}(\hat{q},\sigma) = (S_u(\sigma)/|S_u|) \times (1-(|\Sigma|+1)\gamma_{min}) + \gamma_{min}. \tag{9}$$

This is also used for the ground node where of course the multiset is empty.

## 4. Analysis of the Algorithm

We now proceed to a proof that this algorithm will learn the class of PDFAs. We first state our main result.

**Theorem 5** *For every PDFA $A$ with $n$ states, with distinguishability $\mu > 0$, such that the expected length of the string generated from every state is less than $L$, for every $\delta > 0$ and $\varepsilon > 0$, Algorithm LearnPDFA outputs a hypothesis PDFA $\hat{A}$, such that with probability greater than $1-\delta$, $D_{KL}(A,\hat{A}) < \varepsilon$.*

We start by giving a high-level overview of our proof. In order to bound the KLD we will use a decomposition of the KLD between two automata presented in Carrasco (1997). This requires us to define various expectations relating the number of times that the two automata are in various states.

- We define the notion of a good sample – this means that certain quantities are close to their expected values. We show that a sample is likely to be good if it is large enough.
- We show that if all of the samples we draw are good, then at each step the hypothesis graph will be isomorphic to a subgraph of the target.
- We show that when we stop drawing samples, there will be in the hypothesis graph a state representing each frequent state in the target. In addition we show that all frequent transitions will also have a representative edge in the graph.
- We show that the estimates of all the transition probabilities will be close to the target values.
- Using these bounds we derive a bound for the KLD between the target and the hypothesis.

We shall use a number of quantities calculated from the parameters input to the algorithm to bound the various quantities we are estimating. Table 1 summarises these.

| Symbol | Description | Definition |
|---|---|---|
| $\varepsilon$ | Precision | input |
| $\delta$ | Confidence | input |
| $L$ | Bound on the expected length of strings | input |
| $\mu$ | Distinguishability of states | input |
| $n$ | Bound on the number of states of the target automaton | input |
| $\Phi$ | Function from states of the target to nodes in graph | Definition 8 |
| $W_d(q)$ | Probability to reach state $q$ after $d$ letters | Equation 10 |
| $W(q)$ | Expected number of times state $q$ is reached | Equation 12 |
| $W_d(q,\hat{q})$ | Expected number of times a path is in $q$ and $\hat{q}$ | Equation 14 |
| $P(q)$ | Probability that a state $q$ is reached | Equation 4.1 |
| $P(s), P^A(s)$ | Probability that a string is generated | Equation 2 |
| $P_q(s)$ | Probability that a string is generated from $q$ | $\gamma(q, s\zeta)$ |
| $S_u$ | Multiset of node $u$ | Page 477 |
| $\hat{S_u}$ | Empirical distribution of multiset of node $u$ | Page 477 |
| $\|S_u\|$ | Size of multiset of node $u$ | Page 477 |
| $S_u(s)$ | Count of string $s$ in multiset | Page 477 |
| $S_u(\sigma)$ | Count of letter $\sigma$ in multiset | Page 477 |
| $N$ | One step sample size | Equations 8 |
| $m_0$ | Threshold for size of multiset of candidate node | Equation 6 |
| $\gamma_{min}$ | Smoothing constant for transition probabilities | Equation 9 |
| $\varepsilon_1$ | Bound on transition counts | $\varepsilon_4^2/4(\|\Sigma\|+1)$ |
| $\varepsilon_2$ | Threshold for weight of frequent states | $\varepsilon_3/2nL(L+1)$ |
| $\varepsilon_3$ | Bound on difference between weights in target and hypothesis | Equation 7 |
| $\varepsilon_4$ | Bound on smoothed estimates of transition probabilities | $\varepsilon/2(L+1)$ |
| $\varepsilon_5$ | Threshold for frequent transitions | $\varepsilon_3/2\|\Sigma\|L(L+1)$ |
| $\varepsilon_6$ | Bound on exit probability | $\varepsilon_2\varepsilon_5/(L+1)$ |
| $\delta'$ | Fraction of confidence | Equation 4 |

Table 1: Glossary for notation

### 4.1 Weights of States

We will use a particular decomposition of the KLD, that allows us to represent the divergence as a weighted sum of the divergences between the distributions of the transition probabilities between states in the target and states in the hypothesis. This requires us to define a number of quantities which relate to the expected number of times the automaton will be in various states.

We will define below the *weight* of a state. Informally, this is the expected number of times the automaton will reach, or pass through, that state. Here we encounter one of the differences that cyclicity introduces. With acyclic automata, the automaton can reach each state at most once. Thus the expectation is equal to the probability – and thus we can operate with sets of strings, and simply add up the probabilities. With cyclic automata, the automata can repeatedly be in a particular state while generating a single string. We start by defining

$$W_d(q) = \sum_{s\in\Sigma^d:\tau(q_0,s)=q} \gamma(q_0,s). \tag{10}$$

Informally, $W_d(q)$ is the probability that it will generate at least $d$ characters and be in the state $q$ after having generated the first $d$. This is an important difference in notation to that of Ron et al. (1995), who use $W$ to refer to a set of strings.

Since $s$ does not end in $\zeta$, we are not summing over the probability of strings but rather the probability of prefixes. Note that $W_0(q_0)$ is by definition 1 and that we can also calculate this quantity recursively as

$$W_d(q) = \sum_{p\in Q} W_{d-1}(p) \sum_{\sigma:\tau(p,\sigma)=q} \gamma(p,\sigma). \tag{11}$$

We now define the probability that the automaton will generate a string of length at least $d$ as $W_d$:

$$W_d = \sum_{q\in Q} W_d(q).$$

Note that the probability that it will be of length exactly $d$ is $W_d - W_{d+1}$, and that $W_0 = 1$. We also define here the *expected* number of times the automaton will be in the state $q$ as

$$W(q) = \sum_{d=0}^{\infty} W_d(q) = \sum_{s\in\Sigma^*:\ \tau(q_0,s)=q} \gamma(q_0,s). \tag{12}$$

We shall refer to this quantity as the *weight* of the state $q$. Therefore the expectation of the length of strings is

$$E[|s|] = \sum_{d=0}^{\infty} d(W_d - W_{d+1}) = \sum_{d=1}^{\infty} W_d = \left(\sum_{d=0}^{\infty} W_d\right) - W_0 = \left(\sum_{d=0}^{\infty} W_d\right) - 1 = \sum_{s\in\Sigma^*} \gamma(q_0,s) - 1.$$

The expected length of strings generated from any state can be defined likewise. We will be given a bound on the expected length of strings generated from any state, denoted by $L$. Formally, for all $q$ we shall require that

$$\sum_{s\in\Sigma^*} \gamma(q,s) \leq L+1.$$

Using this bound we can establish that for any length $k$,

$$\sum_{d>k}^{\infty} W_d = \sum_{q} W_k(q) \sum_{d>0} \sum_{s\in\Sigma^d} \gamma(q,s) \leq LW_k.$$

We will later need to bound the quantity $\sum_d dW_d(q)$, which we can do in the following way:

$$\sum_{d=0}^{\infty} dW_d(q) \leq \sum_{d=0}^{\infty} dW_d = \sum_{d=0}^{\infty}\sum_{k>d} W_k \leq \sum_{d=0}^{\infty} LW_d \leq L(L+1). \tag{13}$$

In addition to these expected values, we will also need to lower bound the probability of the automaton being in a state at least once in terms of the expected number of times it will be in that state. The probability of it being in a state at some point can be defined in terms of the set of strings that have a prefix that reaches the state,

$$S(q) = \{s \in \Sigma^* : \exists r, t \in \Sigma^* \,\text{s.t.}\, s = rt \wedge \tau(q_0, r) = q\},$$

or in terms of the set of strings that reach $q$ for the first time (i.e. have no proper prefix that also reaches $q$):

$$R(q) = \{s \in \Sigma^* : \tau(q_0, s) = q \wedge (\nexists u, v \in \Sigma^* \,\text{s.t.}\, v \neq \lambda \wedge uv = s \wedge \tau(q_0, u) = q)\}.$$

The probability of it being in the state at least once can therefore be written in two ways

$$P(q) = \sum_{s \in S(q)} \gamma(q_0, s\zeta) = \sum_{s \in R(q)} \gamma(q_0, s).$$

Note that in the absence of a bound on the expected length of strings, a state can have arbitrarily small probability of being visited but have large weight.

**Lemma 6** *For any automaton such that the expected length from any state is less than L, for all states q,* $P(q) \geq W(q)/(L+1)$.

**Proof** Intuitively, after we reach the state $q$, the expected number of times we reach the state $q$ again will be at most the expected number of times we reach any state after this, which is bounded by $L$. Formally,

$$\begin{aligned}
W(q) &= \sum_{s \in S(q):\ \tau(q_0,s)=q} \gamma(q_0, s) \\
&= \sum_{r \in R(q)} \sum_{s \in \Sigma^*:\ \tau(q_0,rs)=q} \gamma(q_0, rs) \\
&= \sum_{r \in R(q)} \sum_{s \in \Sigma^*:\ \tau(q_0,rs)=q} \gamma(q_0, r)\gamma(q, s) \\
&= \sum_{r \in R(q)} \gamma(q_0, r) \sum_{s \in \Sigma^*:\ \tau(q_0,rs)=q} \gamma(q, s) \\
&= P(q) \sum_{s \in \Sigma^*:\ \tau(q_0,rs)=q} \gamma(q, s) \\
&\leq P(q) \sum_{s \in \Sigma^*} \gamma(q, s) \\
&\leq P(q)(L+1).
\end{aligned}$$

■

We also define here a related quantity given two automata. Given two automata $A, \hat{A}$ with sets of states $Q, \hat{Q}$, the joint weight $W(q, \hat{q})$ is defined to be the expected number of times the automata are simultaneously in the states $q \in Q$ and $\hat{q} \in \hat{Q}$, when strings are being generated by $A$, and parsed by $\hat{A}$.

Define

$$W_d(q,\hat{q}) = \sum_{\substack{s:\tau(q_0,s)=q \\ \hat{\tau}(\hat{q_0},s)=\hat{q} \\ |s|=d}} \gamma(q_0,s). \tag{14}$$

We can also define this recursively with $W_0(q_0,\hat{q}_0) = 1$ and

$$W_d(q,\hat{q}) = \sum_{p\in Q}\sum_{\hat{p}\in\hat{Q}} W_{d-1}(p,\hat{p}) \sum_{\substack{\sigma:\tau(p,\sigma)=q \\ \hat{\tau}(\hat{p},\sigma)=\hat{q}}} \gamma(p,\sigma).$$

We now define the expected number of times the first automaton will be in state $q$ and the second in state $\hat{q}$ thus

$$W(q,\hat{q}) = \sum_d W_d(q,\hat{q}).$$

Note also that

$$W(q) = \sum_{\hat{q}\in\hat{Q}} W(q,\hat{q}). \tag{15}$$

Given these quantities we can now use the following decomposition of the KLD (Carrasco, 1997) to bound the error:

$$D_{KL}(T||H) = \sum_{q\in Q_T}\sum_{\hat{q}\in Q_H}\sum_{\sigma\in\Sigma\cup\{\zeta\}} W(q,\hat{q})\gamma(q,\sigma)\log\frac{\gamma(q,\sigma)}{\hat{\gamma}(\hat{q},\sigma)}. \tag{16}$$

### 4.2 Probably Correct

We define a notion of *goodness* of a multiset which is satisfied if certain properties of the multiset are close to their expected values.

**Definition 7 (Good multiset)** *We say that a multiset $S$ is $\mu$-$\varepsilon_1$-*good *for a state $q$ iff $L_\infty(\hat{S},P_q) < \mu/4$ and for every $\sigma \in \Sigma\cup\{\zeta\}$, $|S(\sigma)/|S| - \gamma(q,\sigma)| < \varepsilon_1$.*

The algorithm produces a sequence of graphs $G_0, G_1, \ldots, G_k$, with multisets attached to each node. We will first show that all of these graphs have the right structure, and then we will show that the final graph will have all of the important structure and finally we will bound the KLD.

**Definition 8 (Good hypothesis graph)** *We say that a hypothesis graph $G$ for an automaton $A$ is* good *if there is a bijective function $\Phi$ from a subset of states of $A$ to all the nodes of $G$ such that $\Phi(q_0) = v_0$, and if $\tau_G(u,\sigma) = v$ then $\tau(\Phi^{-1}(u),\sigma) = \Phi^{-1}(v)$, and for every node $u$ in the graph, the multiset attached to $u$ is $\mu$-$\varepsilon_1$-good for the state $\Phi^{-1}(u)$.*

Note that if there is such an isomorphism then it is unique. When $\Phi(q) = u$ we shall call $u$ a representative of the state $q$. When we have a candidate node $(u,\sigma)$ we shall call this also a representative of the state $\tau(q,\sigma)$. We will extend the use of $\Phi$ to a mapping from the states of the target to the states of the final hypothesis automaton. In this case it will no longer be bijective since in general more than one state can be mapped to the ground state.

**Lemma 9** *If a graph $G_i$ is good and we draw a sample of size N such that for every candidate node with multiset S such that $|S| > m_0$ is $\mu$-$\varepsilon_1$-good for the state $\tau(\Phi^{-1}(u),\sigma)$ , and there is at least one such candidate node, then the graph $G_{i+1}$ is also good.*

Consider a candidate node $(u,\sigma)$ and a node $v$ Suppose these are both representatives of the same state $q$ in which case $\Phi^{-1}(v) = \tau(\Phi^{-1}(u),\sigma)$. Then as $L_\infty(\hat{S}_{u,\sigma}, P_q) < \mu/4$ and $L_\infty(\hat{S}_v, P_q) < \mu/4$ (by the goodness of the multisets), the triangle inequality shows that $L_\infty(\hat{S}_{u,\sigma}, \hat{S}_v) < \mu/2$, and therefore the comparison will return true. On the other hand, let us suppose that they are representatives of different states $q$ and $q_v$. We know that $L_\infty(\hat{S}_{u,\sigma}, P_q) < \mu/4$ and $L_\infty(\hat{S}_v, P_{q_v}) < \mu/4$ (by the goodness of the multisets), and $L_\infty(P_q, P_{q_v}) \geq \mu$ (by the $\mu$-distinguishability of the target). By the triangle inequality $L_\infty(P_q, P_{q_v}) \leq L_\infty(\hat{S}_{u,\sigma}, P_q) + L_\infty(\hat{S}_{u,\sigma}, \hat{S}_v) + L_\infty(\hat{S}_v, P_{q_v})$, therefore $L_\infty(\hat{S}_{u,\sigma}, \hat{S}_v) > \mu/2$ and the comparison will return false. If there is a node in the graph that is similar to the candidate node, then this is because there is the relevant transition in the target automaton, which means the new graph will also be good. If there is no such node, then that is because there is no node $v$ such that $\Phi^{-1}(v) = \tau(\Phi^{-1}(u),\sigma)$ in which case we can define a new node $v$ and define $\Phi^{-1}(v) = \tau(\Phi^{-1}(u),\sigma)$, and thereby show that the new graph is good. Additionally since the candidate node multisets are $\mu$-$\varepsilon_1$-good, the multiset of this node will also be good. ■

Since $G_0$ is good if the initial sample is good, using this we can prove by induction on $i$ that the final hypothesis graph will be $\mu$-$\varepsilon_1$-good if all the samples satisfy the criteria defined above.

**Definition 10 (exiting the graph)** *A string exits a graph G if there is no node v such that $\tau_G(v_0,s) = v$. The exit probability of a graph G (with respect to an automaton A) which we write $P_{exit}(G)$ is defined as*

$$P_{exit}(G) = \sum_{s: s \text{ exits } G} \gamma(q_0, s\zeta).$$

**Definition 11 (Good sample)** *We say that a sample of size N is good given a good graph G if for every candidate node $(u,\sigma)$, such that $|S_{u,\sigma}| > m_0$, $S_{u,\sigma}$ is $\mu$-$\varepsilon_1$-good for the state $\tau(\Phi^{-1}(u),\sigma)$ and that if $P_{exit}(G) > \varepsilon_6$ then the number of strings that exit the graph is more than $\frac{1}{2}NP_{exit}(G)$.*

Note that if there are no candidate nodes with multisets larger than $m_0$, then the total number of strings that exited the graph must be less than $n|\Sigma|m_0$ (since there are at most $n$ nodes in a good graph, and therefore at most $n|\Sigma|$ candidate nodes). Therefore in this circumstance, if the samples are good, we can conclude that either $P_{exit}(G) \leq \varepsilon_6$ or $P_{exit}(G) < 2n|\Sigma|m_0/N$.

We then prove that the hypothesis graph will probably contain a node representing each state with non-negligible weight, and an edge representing each transition from a state with non-negligible weight that has a non-negligible transition probability. We define two intermediate quantities: $\varepsilon_2$, a bound on the weight of states, and $\varepsilon_5$, a bound on transition weights. We have

$$\varepsilon_2 = \frac{\varepsilon_3}{2nL(L+1)} \text{ and } \varepsilon_5 = \frac{\varepsilon_3}{2|\Sigma|L(L+1)}. \tag{17}$$

**Lemma 12** *For any state q in the target such that $W(q) > \varepsilon_2$, if all the samples are good, then there will be a node u in the final hypothesis graph such that $\Phi(q) = u$. Furthermore, for such a state for any $\sigma \in \Sigma$, such that $\gamma(q,\sigma) > \varepsilon_5$, then $\tau_G(u,\sigma)$ is defined and is equal to $\Phi(\tau(q,\sigma))$.*

**Proof** Since $W(q) > \varepsilon_2$, we know by Lemma 6 that $P(q) > \varepsilon_2/(L+1)$.

From the definitions of $N$ and $\varepsilon_6$

$$\frac{2n|\Sigma|m_0}{N} < \varepsilon_6 = \frac{\varepsilon_2\varepsilon_5}{(L+1)} < P(q)\varepsilon_5 < P(q).$$

Clearly if there were no representative node $u$ for state $q$, then all strings that reached the state would exit the graph, and thus $P_{exit}(G) \geq P(q)$. Similarly, if there were no edge labelled $\sigma$ from the node, then $P_{exit}(G) \geq P(q)\varepsilon_5$. By the goodness of the sample we know that either $P_{exit}(G) \leq \varepsilon_6$ or $P_{exit}(G) < 2n|\Sigma|m_0/N$, in both cases $P_{exit}(G) < P(q)\varepsilon_5$. Therefore there is a suitable state $u$ and edge in the graph, and since the final hypothesis graph is $\mu$-$\varepsilon_1$-good, $\tau_G$ will have the correct value. ∎

### 4.3 Close Transition Probabilities

Assuming that all the samples are good, we now want to show that the final smoothed transition probabilities will be close to the correct values. More precisely, we want to show that for all transitions, the ratio of the true value to the estimate is not much greater than unity.

For every state $q$ with $W(q) > \varepsilon_2$, with $u = \Phi(q)$ and for every symbol $\sigma \in \Sigma \cup \{\zeta\}$, we will have, by the goodness of the multisets

$$\left|\gamma(q,\sigma) - \frac{S_u(\sigma)}{|S_u|}\right| \leq \varepsilon_1. \tag{18}$$

Define

$$\varepsilon_4 = \frac{\varepsilon}{2(L+1)},$$

$$\gamma_{min} = \frac{\varepsilon_4}{2(|\Sigma|+1)} \quad \text{and} \quad \varepsilon_1 = \frac{\varepsilon_4^2}{4(|\Sigma|+1)}.$$

We use exactly the argument in Lemma 5.3 of Ron et al. (1995): if $\gamma(q,\sigma)$ is small (at most $\gamma_{min} + \varepsilon_1$) then the estimate will anyway be at least $\gamma_{min}$ and thus the ratio will be not much larger than the true value, and if it is large then it will be close to the true value. We can verify that

$$\frac{\gamma(q,\sigma)}{\hat{\gamma}(q,\sigma)} \leq 1 + \varepsilon_4. \tag{19}$$

### 4.4 Close Weights

We now come to the most complex part of the proof. We want to show for every frequent state $q$ in the target, which will have a corresponding state in the hypothesis $\hat{q} = \Phi(q)$, that (again under the assumption that all the samples are good)

$$W(q) - W(q,\hat{q}) \leq \varepsilon_3. \tag{20}$$

As a consequence of this and Equation 15, we will have

$$\sum_{\hat{q}:\Phi(q)\neq\hat{q}} W(q,\hat{q}) \leq \varepsilon_3. \tag{21}$$

So we will do this by showing that for any $d$

$$W_d(q) - W_d(q,\hat{q}) < \frac{\varepsilon_3 d W_d(q)}{L(L+1)}.$$

Then, since we know that $\sum_d dW_d(q) < L(L+1)$ by Equation 13, we can sum with respect to $d$ and derive the result.

**Lemma 13** *For all states $q \in Q$ such that $W(q) > \varepsilon_2$, there is a state $\hat{q}$ in the hypothesis such that $\Phi(q) = \hat{q}$ and for all $d \geq 0$*

$$W_d(q) - W_d(q,\hat{q}) \leq \frac{\varepsilon_3 d W_d(q)}{L(L+1)}.$$

**Proof** We will prove the lemma by induction on $d$. For $d = 0$ this is clearly true. For the inductive step, we assume that it is true for $d-1$.

We can rewrite the joint weight as

$$W_d(q,\hat{q}) = \sum_{p \in Q} \sum_{\hat{p} \in \hat{Q}} W_{d-1}(p,\hat{p}) \sum_{\substack{\sigma:\tau(p,\sigma)=q \\ \hat{\tau}(\hat{p},\sigma)=\hat{q}}} \gamma(p,\sigma). \tag{22}$$

If we consider only the cases where $W(p) > \varepsilon_2$ and $\Phi(p) = \hat{p}$ we can see that

$$W_d(q,\hat{q}) \geq \sum_{p \in Q: W(p) > \varepsilon_2} W_{d-1}(p,\Phi(p)) \sum_{\substack{\sigma:\tau(p,\sigma)=q \\ \hat{\tau}(\Phi(p),\sigma)=\hat{q}}} \gamma(p,\sigma). \tag{23}$$

And then by the inductive hypothesis, for these frequent states

$$W_d(q,\hat{q}) \geq \sum_{p \in Q: W(p) > \varepsilon_2} W_{d-1}(p) \left(1 - \frac{\varepsilon_3(d-1)}{L(L+1)}\right) \sum_{\substack{\sigma:\tau(p,\sigma)=q \\ \hat{\tau}(\Phi(p),\sigma)=\hat{q}}} \gamma(p,\sigma). \tag{24}$$

Using the fact that $W_d \leq 1$ for all $d$, and using the recursive definition of $W_d$ we can write this as

$$\begin{aligned} W_d(q,\hat{q}) &\geq \sum_{p \in Q: W(p) > \varepsilon_2} W_{d-1}(p) \sum_{\substack{\sigma:\tau(p,\sigma)=q \\ \hat{\tau}(\Phi(p),\sigma)=\hat{q}}} \gamma(p,\sigma) - \frac{\varepsilon_3(d-1)W_{d-1}}{L(L+1)} && (25) \\ &\geq \sum_{p \in Q: W(p) > \varepsilon_2} W_{d-1}(p) \sum_{\sigma:\tau(p,\sigma)=q} \gamma(p,\sigma) \\ &\quad - \sum_{p \in Q: W(p) > \varepsilon_2} W_{d-1}(p) \sum_{\substack{\sigma:\tau(p,\sigma)=q \\ \hat{\tau}(\Phi(p),\sigma)\neq\hat{q}}} \gamma(p,\sigma) \\ &\quad - \frac{\varepsilon_3(d-1)}{L(L+1)}. && (26) \end{aligned}$$

So we need to show that most of the weight from $W_{d-1}$ must be from states $p$ with $W(p) > \varepsilon_2$. Then we can change from $W_{d-1}(p)$ to $W_d(q)$.

Using Equation 11 we can see that

$$
\begin{aligned}
W_d(q) &= \sum_{p \in Q : W(p) \le \varepsilon_2} W_{d-1}(p) \sum_{\sigma : \tau(p,\sigma)=q} \gamma(p,\sigma) + \sum_{p \in Q : W(p) > \varepsilon_2} W_{d-1}(p) \sum_{\sigma : \tau(p,\sigma)=q} \gamma(p,\sigma) \\
&\le n\varepsilon_2 + \sum_{p \in Q : W(p) > \varepsilon_2} W_{d-1}(p) \sum_{\sigma : \tau(p,\sigma)=q} \gamma(p,\sigma).
\end{aligned}
$$

Using this to replace the first term on the right hand side of Equation 26 we get

$$
W_d(q,\hat{q}) \ge W_d(q) - n\varepsilon_2 - \frac{\varepsilon_3(d-1)}{L(L+1)} - \sum_{p \in Q : W(p) > \varepsilon_2} W_{d-1}(p) \sum_{\substack{\sigma : \tau(p,\sigma)=q \\ \hat{\tau}(\Phi(p),\sigma) \neq \hat{q}}} \gamma(p,\sigma).
$$

Since by Lemma 12 all of the transitions from frequent states with probability greater than $\varepsilon_5$ must go to the correct states, we know that the values of $\gamma(p,\sigma)$ in the final term must be less than $\varepsilon_5$.

$$
\begin{aligned}
W_d(q,\hat{q}) &\ge W_d(q) - n\varepsilon_2 - \frac{\varepsilon_3(d-1)}{L(L+1)} - \sum_{p \in Q : W(p) > \varepsilon_2} W_{d-1}(p) \sum_{\sigma} \varepsilon_5 \\
&\ge W_d(q) - n\varepsilon_2 - \frac{\varepsilon_3(d-1)}{L(L+1)} - |\Sigma|\varepsilon_5.
\end{aligned}
$$

By our definitions of $\varepsilon_2$ and $\varepsilon_5$

$$
n\varepsilon_2 + |\Sigma|\varepsilon_5 \le \frac{\varepsilon_3}{L(L+1)},
$$

and therefore the lemma will hold. ■

## 5. Proof that the Divergence is Small

We are now in a position to show that the KLD between the target and the hypothesis is small. We use Carrasco's decomposition of the KLD (Carrasco, 1997) to bound the error:

$$
D_{KL}(T||H) = \sum_{q \in Q_T} \sum_{\hat{q} \in Q_H} \sum_{\sigma \in \Sigma \cup \{\zeta\}} W(q,\hat{q}) \gamma(q,\sigma) \log \frac{\gamma(q,\sigma)}{\hat{\gamma}(\hat{q},\sigma)}.
$$

We are going to divide the summands into three parts. We define

$$
D(q,\hat{q}) = W(q,\hat{q}) \sum_{\sigma \in \Sigma \cup \{\zeta\}} \gamma(q,\sigma) \log \frac{\gamma(q,\sigma)}{\hat{\gamma}(\hat{q},\sigma)},
$$

$$
\begin{aligned}
D_1 &= \sum_{q \in Q_T : W(q) > \varepsilon_2} \sum_{\hat{q} \in Q_H : \Phi(q) = \hat{q}} D(q,\hat{q}), \\
D_2 &= \sum_{q \in Q_T : W(q) > \varepsilon_2} \sum_{\hat{q} \in Q_H : \Phi(q) \neq \hat{q}} D(q,\hat{q}), \\
D_3 &= \sum_{q \in Q_T : W(q) \le \varepsilon_2} \sum_{\hat{q} \in Q_H} D(q,\hat{q}).
\end{aligned}
$$

Note that for the frequent states with $W(q) > \epsilon_2$, $\Phi$ will be well-defined.

$$D_{KL}(T||H) = D_1 + D_2 + D_3$$

We will bound these separately. We bound $D_1$ by showing that for these matching pairs of states $q$, $\Phi(q)$, all of the transition probabilities are close to each other. We can bound $D_2$ by showing that $W(q,\hat{q})$ is small when $\hat{q} \neq \Phi(q)$, and $D_3$ since $W(q)$ is small.

Using Equation 19 and recalling that $W(q) \geq W(q,\hat{q})$ we bound $D_1$.

$$D_1 \leq \sum_{q \in Q_T : W(q) > \epsilon_2} W(q) \log(1+\epsilon_4) \leq (L+1)\log(1+\epsilon_4) \leq (L+1)\epsilon_4.$$

Bounding $D_2$ is done using Equation 21, the fact that $\gamma(q,\sigma) \leq 1$ and that $\hat{\gamma}(\hat{q},\sigma) \geq \gamma_{min}$ by the smoothing technique.

$$\begin{aligned} D_2 &\leq \sum_{q \in Q_T, W(q) > \epsilon_2} \sum_{\hat{q} \in Q_H : \Phi(q) \neq \hat{q}} W(q,\hat{q}) \sum_{\sigma \in \Sigma \cup \{\zeta\}} \gamma(q,\sigma) \log \frac{1}{\gamma_{min}} \\ &\leq \sum_{q \in Q_T, W(q) > \epsilon_2} \epsilon_3 \log \frac{1}{\gamma_{min}} \leq n\epsilon_3 \log \frac{1}{\gamma_{min}}. \end{aligned}$$

With regard to $D_3$, using Equation 15 and the bound on $W(q)$ we can see that

$$D_3 \leq \sum_{q \in Q_T, W(q) \leq \epsilon_2} \sum_{\hat{q} \in Q_H} W(q,\hat{q}) \sum_{\sigma \in \Sigma \cup \{\zeta\}} \gamma(q,\sigma) \log \frac{1}{\gamma_{min}} \leq n\epsilon_2 \log \frac{1}{\gamma_{min}}.$$

Substituting in Equation 17, and assuming that $L > 1$,

$$\begin{aligned} D_{KL}(T||H) &< (L+1)\epsilon_4 + \left(n\epsilon_3 + \frac{\epsilon_3}{2nL(L+1)}\right) \log \frac{1}{\gamma_{min}} \\ &< (L+1)\epsilon_4 + (n+1)\epsilon_3 \log \frac{1}{\gamma_{min}}. \end{aligned}$$

Substituting in the values of $\gamma_{min}$, $\epsilon_3$ and $\epsilon_4$ gives us $D_{KL}(T||H) < \epsilon$ as desired. ■

We have thus shown that if all of the samples are good at each step, the resulting hypothesis will be approximately correct. We must now show that all of the samples will be good with high probability, by showing that $N$ and $m_0$ are large enough.

## 6. Bounding the Probability of an Error

We need to show that with high probability a sample of size $N$ will be good for a given good graph $G$. We assume that the graph is good at each step. Each step of the algorithm will increase the number of transitions in the graph by at least 1. There are at most $n|\Sigma|$ transitions in the target; so there are at most $n|\Sigma| + 2$ steps in the algorithm since we need an initial step to get the multiset for $u_0$ and another at the end when we terminate. So we want to show that a sample will be good with probability at least $1 - \delta/(n|\Sigma|+2) = 1 - 2\delta'$.

There are two sorts of errors that can make the sample bad. First, one of the multisets could be bad, and secondly too few strings might exit the graph. There are at most $n|\Sigma|$ candidate nodes, so we ensure that the probability of getting a bad multiset at a particular candidate node is less than $\delta'/n|\Sigma|$, and we will ensure that the probability of the second sort of error is less than $\delta'$.

### 6.1 Good Multisets

First we bound the probability of getting a bad multiset. Recall from Definition 7 that we have two requirements. First, for every string we must have the empirical probability within $\mu/4$ of its true value. Secondly, for each $\sigma \in \Sigma \cup \{\zeta\}$ the empirical probability must be within $\varepsilon_1$ of the true value.

A difficulty here is that in the first case we will be comparing over an infinite number of strings. Clearly only finitely many strings will have non-zero counts, but nonetheless, we need to show that for every string in $\Sigma^*$ the empirical probability is close to the true probability. This will be true if all of the strings with probability less than $\mu/8$ have empirical probability less than $\mu/4$ and all of the strings with probability greater than this have empirical probability within $\mu/4$ of their true values.

Consider the strings of probability greater than $\mu/8$. There are at most $8/\mu$ of these. Therefore for a given string $s$, by Hoeffding bounds (Hoeffding, 1963):

$$Pr\left[\left|\frac{S_u(s)}{|S_u|} - \gamma(q, s\zeta)\right| > \mu/4\right] < 2e^{-m_u\mu^2/8} < 2e^{-m_0\mu^2/8}. \tag{27}$$

The probability of making an error on one of these frequent strings is less than $\frac{16}{\mu}e^{-m_0\mu^2/8}$. We also need to bound the probability of all of the rare strings–those with $\gamma(q, s\zeta) \leq \mu/8$.

Consider all of the strings whose probability is in $[\mu 2^{-(k+1)}, \mu 2^{-k})$.

$$S_k = \{s \in \Sigma^* : \gamma(q, s\zeta) \in [\mu 2^{-(k+1)}, \mu 2^{-k})\}.$$

Define $S_{rare} = \bigcup_{k=3}^{\infty} S_k$. We bound each of the $S_k$ separately, using the binomial Chernoff bound where $n = |S_u|\mu/4 > |S_u|p$ (which is true since $p < \mu/4$):

$$Pr\left[\frac{S_u(s)}{|S_u|} \geq \frac{\mu}{4}\right] \leq \left(\frac{|S_u|p}{n}\right)^n e^{n-|S_u|p}.$$

This bound decreases with $p$, so we can replace this for all strings in $S_k$ with the upper bound for the probability, and we can replace $|S_u|$ with $m_0$. We write $\mu' = \mu/4$ to reduce clutter.

$$Pr\left[\frac{S_u(s)}{|S_u|} \geq \mu'\right] \leq \left(\frac{m_0\mu' 2^{-k}}{m_0\mu'}\right)^{m_0\mu'} e^{m_0\mu' - m_0\mu' 2^{-k}} \tag{28}$$

$$\leq \left(2^{-k}e^{1-2^{-k}}\right)^{m_0\mu'} < 2^{-km_0\mu'}. \tag{29}$$

Assuming that $m_0\mu' > 3$ we can see that

$$Pr\left[\frac{S_u(s)}{|S_u|} \geq \mu'\right] < 2^{-2k}2^{k(2-m_0\mu')} \leq 2^{-2k}2^{2-m_0\mu'} \tag{30}$$

$$Pr\left[\exists s \in S_k : \frac{S_u(s)}{|S_u|} \geq \mu'\right] \leq |S_k|2^{-2k}2^{2-m_0\mu'} \leq \frac{1}{\mu}2^{-k+1}2^{2-m_0\mu'}. \tag{31}$$

The $2^{-k}$ factor allows us to sum over all of the $k$ to calculate the probability that there will be an error with any rare string:

$$Pr\left[\exists s \in S_{rare} : \frac{S_u(s)}{|S_u|} \geq \mu'\right] < \frac{1}{\mu}\sum_{k=3}^{\infty} 2^{-k+1} 2^{2-m_0\mu'} < \frac{2}{\mu} 2^{-m_0\mu'}. \tag{32}$$

The next step is to show that for every $\sigma \in \Sigma \cup \{\zeta\}$,

$$\left|\gamma(q,\sigma) - \frac{S_u(\sigma)}{|S_u|}\right| \leq \varepsilon_1.$$

We can use Chernoff bounds to show that the probability of an error will be less than $e^{-2m_0\varepsilon_1^2}$. Putting these together we can show that the probability of a single multiset being bad can be bounded as in the equation below, and that this is satisfied by the value of $m_0$ defined above in Equation 6:

$$\frac{16}{\mu} e^{-m_0\mu^2/8} + \frac{2}{\mu} 2^{-m_0\mu/4} + (|\Sigma|+1)e^{-2m_0\varepsilon_1^2} < \frac{\delta'}{n|\Sigma|}. \tag{33}$$

### 6.2 Exit Probability Errors

We next need to show that roughly the expected number of strings exit the graph, if the exit probability is greater than $\varepsilon_6 = \varepsilon_2\varepsilon_5/(L+1)$. Again we can use Chernoff bounds to show that the probability of this sort of error will be less than $e^{-NP_{exit}(G)/4} < e^{-N\varepsilon_6/4}$. It is easy to verify that for our chosen value of $N$,

$$e^{-N\varepsilon_6/4} < \delta'. \tag{34}$$

### 6.3 Complexity

Since when the algorithm runs correctly there are at most $n|\Sigma|+2$ steps, the sample complexity will be $N_{total} = N(n|\Sigma|+2)$ which is polynomial by inspection.

As the strings can be of unbounded length the computational complexity has to be bounded in terms of the total lengths of the strings as well. If $S_{total}$ is the multiset representing the overall sample of size at most $N(n|\Sigma|+2)$, then define

$$N_l = \sum_{s \in \Sigma^*} |s| S(s).$$

We denote the maximum observed length $L_{max}$ (clearly $L_{max} \leq N_l$).

Since there are at most $n$ nodes in the graph at any time, and at most $n|\Sigma|$ candidate nodes, the number of comparisons will be at most $n^2|\Sigma|$ at each step and thus $n^2|\Sigma|(n|\Sigma|+2)$ in all. For each multiset we only need to store at most $m_0$ so there will be at most $m_0$ distinct strings in each multiset, so the comparison will need to compare at most $2m_0$ strings, and each comparison will take at most $L_{max}$. The construction of the candidate nodes can be done in less than $N_l$ time. These observations suffice to show that the algorithm is polynomial in $N_l$ and $N_{total}$. We have assumed here that $|\Sigma|$ is sufficiently small that two characters can be compared in constant time.

## 7. Conclusion

We have presented an algorithm and a proof that it KL-PAC-learns the class of PDFAs given certain reasonable additional parameters for the sample complexity. We argue that these additional parameters or some substitutes are necessary given the counter-examples previously studied and presented here. Furthermore the classes we present here cover the whole space of distributions defined by PDFAs. Convergence properties of these sorts of state merging algorithms have been studied before in the identification in the limit paradigm (Carrasco and Oncina, 1999), but not in the KL-PAC framework.

The work presented here can be compared to traditional types of PAC learning. Distribution free learning of regular languages has been shown to be hard under some standard cryptographic assumptions (Kearns and Valiant, 1994), but learnability under limited classes of distributions is still an open question. We can view the algorithm presented here, if the graph completion and smoothing is removed, as learning regular languages from positive samples only, where the distributions are restricted to be generated by PDFAs which define the target language. We can compare the result presented in Parekh and Honavar (2001), where it is shown that DFAs can be PAC-learned from positive and negative samples, using a *simple* distribution. This means a distribution where examples of low Kolmogorov complexity, given a representation of the target, have a high probability. Indeed, Denis (2001) shows that using this approach they can also be probably exactly learned from positive examples only. Since the Solomonoff-Levin distribution dominates every other computable distribution, up to a constant factor, this is in some sense the easiest distribution to learn under. Moreover, the use of simple distributions has a number of flaws: first, there are some very large non-computable constants that appear in the sample complexity polynomial. Secondly, representations of the target will also have low complexity with respect to the target, and thus will almost certainly occur in polynomial samples (what is sometimes called “collusion”), which does allow trivial learning algorithms in the case of the positive and negative samples. This may also be the case with positive samples alone. This is an open question, and beyond the scope of this paper, but certainly if the support of the language is large enough, collusive learning becomes possible. Thirdly, the examples provided will be those that are comparatively predictable – i.e. carry little information; in many realistic situations, the strings are used to carry information and thus generally will have high complexity. Thus we feel that though mathematically quite correct, as a characterisation of the convergence properties of algorithms, these approaches are too abstract.

One further extension we intend to study is to the class of the probabilistic residual automata (Esposito et al., 2002) which should be tractable with matrix perturbation theory.

## Acknowledgments

We are grateful to anonymous reviewers of this paper and an earlier draft, for helpful comments. We would also like to thank Codrin Nichitiu, Colin de la Higuera and Marc Sebban.

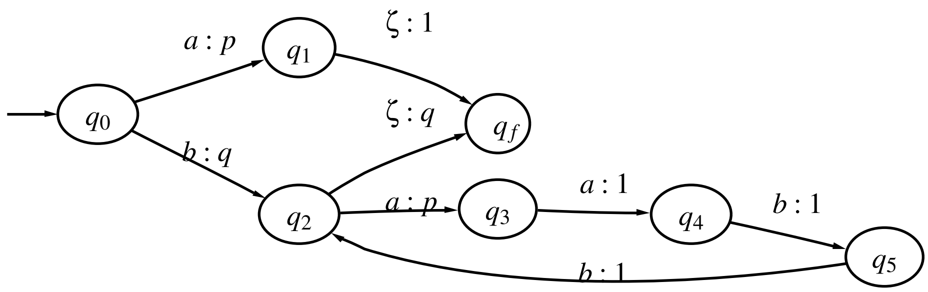


Figure 1: Example automaton with $k=4$ and $s=aabb$. The arcs are labelled with the symbol and then the probability. $p$ will be very close to 1 and $q=1-p$.

## Appendix A.

In this appendix we establish our counter-example that justifies our addition of a bound on the expected length of the strings generated from every state. Our basic argument is as follows: we construct a countable family of automata such that given any sample size N, even exponentially large, all but finitely many automata in the family will give the same sample set with probability greater than 0.5. Given this sample set, the algorithm will produce a particular hypothesis. We show that whatever hypothesis, $H$, a learning algorithm produces must have $D(H||T) > 0.05$ for some of the targets in the subset that will generate this same set. Thus arguing by contradiction, we see that it is not possible to have a PAC-learning algorithm unless we allow it to have a sample complexity polynomial that includes some additional parameter relating in some way to the expected length. We neglect the possibility that the algorithm might be randomised, but it is easy to deal with that eventuality.

It is quite straightforward to construct such a family of automata if we merely want to demonstrate the necessity of a bound on the overall expected length of the strings generated by the automaton. Consider a state of the target that is reached with prob $n^{-1}$, but that generates strings of length $n^2$ through a transition to itself of probability $1-n^{-2}$. For such a state $q$, the weight $W(q)$ will be $n$. Thus any small differences in the transitions can cause unbounded increases in the KLD. Here we want to establish a slightly sharper result, which applies even when we have a bound on the overall expected length.

We define a family of automata $\mathcal{F}_k$ for some $k>0$, over a two letter alphabet $\Sigma=\{a,b\}$, $\mathcal{F}_k = \{A_p^s | s \in \Sigma^k, p \in (0,1)\}$. Each of these automata $A = A_p^s$ defines a distribution as follows:

$$\begin{aligned} P_A(t) &= \quad p \text{ when } t=a; \\ P_A(t) &= (1-p)^2 p^i \text{ when } t = bs^i \text{ for } i \geq 0\ ; \\ P_A(t) &= \quad 0 \text{ otherwise .} \end{aligned}$$

The automata will have $k+2$ states, and the expected length of the strings generated by these automata will be $k+1$. Figure 7 shows a simple example for $k=4$.

Suppose we have an algorithm $H$ that can learn a superset of $\mathcal{F}$. Set $\varepsilon = \frac{1}{8}\log(3/2)$ and $\delta = \frac{1}{2}$. Since we have a polynomial algorithm there must be a polynomial bound on the number of states of the target $q(\varepsilon,\delta,|\Sigma|,k+2,L)$, when there is an upper bound on the length of the strings in the sample, $L$. This additional caveat is necessary because the strings in the sample in general can be of unbounded length. Here we will be considering a sample where all the strings are of length 1. Fixing these values of $\varepsilon,\delta,|\Sigma|$ and $L$ we will have a polynomial in $k$, so we can find a value of $k$ such that $2^k > q(\varepsilon,\delta,|\Sigma|,k+2)$. Denote the smallest such $k$ by $k_0$. Let $N$ be the sample complexity of the algorithm $H$ for these values. We can set $p$ to be so close to 1 that $p^N > 0.5$, which means that with probability greater than 0.5 the sample generated by the target will consist of $N$ repetitions of the string $a$. Let $A_H = (\hat{Q},\hat{\gamma},\hat{\tau})$ be the hypothesis automaton produced by the algorithm $H$ on such a data set. By construction $2^k > |\hat{Q}|$.

It is not enough just to show that for some string $s$ the hypothesis will give a low probability to strings of the form $bs^i$. We must also show that this probability decreases exponentially as $i$ increases. We must therefore show, using a pumping argument, that there is a low probability transition in the cycles in the hypothesis automaton traced by the strings for large values of $i$. We can assume, without loss of generality, that the hypothesis assigns a non-zero probability to all the strings $bs^i$, or the KLD will be infinite. For each string $s \in \Sigma^{k_0}$, consider the states in $\hat{Q}$ defined by $\hat{\tau}(\hat{q}_0, bs^i)$. There must be two distinct values $i < j \leq |\hat{Q}|$ such that $\hat{\tau}(\hat{q}_0,bs^i) = \hat{\tau}(\hat{q}_0,bs^j)$, by the pigeonhole principle. Select the smallest $i$ such that this is true, denote these values of $i$ and $j$ by $s_i$ and $s_j$, and let $q_s$ denote the state $\hat{\tau}(\hat{q}_0,bs^{s_i})$. By construction $0 < s_j - s_i \leq |\hat{Q}|$. The state $q_s$ will therefore be in a suitable cycle since

$$q_s = \hat{\tau}(\hat{q}_0,bs^{s_i}) = \hat{\tau}(\hat{q}_0,bs^{s_j}) = \hat{\tau}(\hat{q}_0,bs^{s_i+k(s_j-s_i)})$$

for all $k \geq 0$.

We now want to show that for some string $s$ there is transition in the cycle with a probability at most $\frac{1}{2}$. Since the number of strings is larger than $|\hat{Q}|$ there must be two distinct strings, $s,s'$, such that $q_s = q_{s'}$.

We can write $s = u\sigma v$ and $s' = u\sigma' v'$, where $u,v,v' \in \Sigma^*, \sigma,\sigma' \in \Sigma$. and $\sigma \neq \sigma'$, $u$ here being the longest common prefix of $s$ and $s'$. Consider the transitions from the state $\hat{\tau}(q_s,u)$. At least one of the two values $\hat{\gamma}(\hat{\tau}(q_s,u),\sigma)$, $\hat{\gamma}(\hat{\tau}(q_s,u),\sigma')$ must be less than or equal to 0.5. Without loss of generality we can say it is $\sigma$, which means that $\hat{\gamma}(q_s,s) \leq 1/2$.

This means that we can say the probability of a string of the form $bs^{k|\hat{Q}|+l}$ for any $k,l \geq 0$ must be less than or equal to $2^{-k}$. For $k=0$ this is trivially true. For $k>0$ define $n = k(|\hat{Q}|+s_i-s_j)+l-s_i \geq 0$ we can write the probability as

$$\begin{aligned}
\hat{\gamma}(\hat{q}_0,bs^{k|\hat{Q}|+l}\zeta) &= \hat{\gamma}(q_0,bs^{s_i})\hat{\gamma}(q_s,s^{s_j-s_i})^k\hat{\gamma}(q_s,s^n)\hat{\gamma}(\hat{\tau}(bs^{k|\hat{Q}|+l}),\zeta) \\
&\leq \hat{\gamma}(q_s,s^{s_j-s_i})^k \\
&\leq \hat{\gamma}(q_s,s)^{(s_j-s_i)k} \\
&\leq 2^{-k}.
\end{aligned}$$

We can now use this upper bound on the probability that the hypothesis gives the strings to lower bound the divergence with respect to the target.

Expanding out the definition of the KLD and the target automaton, we have

$$D_{KL}(A_p^s,A_H) = p\log\frac{p}{\hat{\gamma}(\hat{q}_0,a\zeta)} + \sum_{i=0}^{\infty}\sum_{j=0}^{|\hat{Q}|-1}(1-p)^2p^{i|\hat{Q}|+j}\log\frac{(1-p)^2p^{i|\hat{Q}|+j}}{\hat{\gamma}(\hat{q}_0,bs^{i|\hat{Q}|+j}\zeta)}.$$

Substituting in the bound above, and the fact that $\hat{\gamma}(\hat{q}_0, a\zeta) \leq 1$ yields

$$\begin{aligned}
D_{KL}(A_p^s, A_H) &\geq p\log p + \sum_{i=0}^{\infty}\sum_{j=0}^{|\hat{Q}|-1}(1-p)^2 p^{i|\hat{Q}|+j}\log(1-p)^2 p^{i|\hat{Q}|+j}2^i \\
&\geq p\log p + |\hat{Q}|\sum_{i=0}^{\infty}(1-p)^2 p^{i|\hat{Q}|+|\hat{Q}|}\log(1-p)^2 p^{i|\hat{Q}|+|\hat{Q}|}2^i \\
&\geq p\log p + |\hat{Q}|(1-p)^2 p^{|\hat{Q}|}\left(\log(1-p)^2 p^{|\hat{Q}|}\sum_{i=0}^{\infty}p^{i|\hat{Q}|} + \log 2p^{|\hat{Q}|}\sum_{i=0}^{\infty} i p^{i|\hat{Q}|}\right).
\end{aligned}$$

Recall that $\sum_{i=0}^{\infty} i p^i = p(1-p)^{-2}$ and $\sum_{i=0}^{\infty} p^i = (1-p)^{-1}$, so that

$$D_{KL}(A_p^s, A_H) \geq p\log p + |\hat{Q}|(1-p)^2 p^{|\hat{Q}|}\left(\log(1-p)^2 p^{|\hat{Q}|}(1-p^{|\hat{Q}|})^{-1} + \log 2p^{|\hat{Q}|}p^{|\hat{Q}|}(1-p^{|\hat{Q}|})^{-2}\right).$$

We can take $p$ to be large enough that $p^{|\hat{Q}|} > 3/4$, giving

$$D_{KL}(A_p^s, A_H) \geq p\log p + |\hat{Q}|\frac{(1-p)^2}{(1-p^{|\hat{Q}|})}p^{2|\hat{Q}|}\left(\log(1-p)^2 + (1-p^{|\hat{Q}|})^{-1}\log 3/2\right).$$

Now if we write $p = (1-1/n)$,

$$D_{KL}(A_p^s, A_H) \geq \log p + |\hat{Q}|\frac{(1-p)^2}{(1-p^{|\hat{Q}|})}p^{2|\hat{Q}|}\left(\frac{n}{|\hat{Q}|}\log\frac{3}{2} - 2\log n\right).$$

Now using a simple linear bound on the logarithm it can be shown that for any $\beta > 1$ if $n > 2\beta\log 2\beta$ then $\log n < n/\beta$. If we set $\beta = 4|\hat{Q}|/\log(3/2)$ and $n > 8|\hat{Q}|\log|\hat{Q}|$, and $p^{|\hat{Q}|} > (1-|\hat{Q}|/n)$ and assuming that $p > (2/3)^{1/8}$ we have

$$\begin{aligned}
D_{KL}(A_p^s, A_H) &\geq \log p + |\hat{Q}|\frac{(1-p)^2}{(1-p^{|\hat{Q}|})}p^{2|\hat{Q}|}\frac{n}{2|\hat{Q}|}\log\frac{3}{2} \\
&\geq \log p + \frac{(1-p)}{(1-p^{|\hat{Q}|})}p^{2|\hat{Q}|}\frac{1}{2}\log\frac{3}{2} \\
&\geq \log p + \left(\frac{3}{4}\right)^2\frac{1}{2}\log\frac{3}{2} \geq \frac{1}{8}\log\frac{3}{2}.
\end{aligned}$$

Thus for sufficiently large values of $n$, and thus for values of $p$ sufficiently close to 1, there must be an automaton $A_p^s$ such that the algorithm will with probability at least 0.5 produce a hypothesis with an error of at least $\frac{1}{8}\log\frac{3}{2}$. ■

## References

N. Abe, J. Takeuchi, and M. Warmuth. Polynomial learnability of stochastic rules with respect to the KL-divergence and quadratic distance. *IEICE Transactions on Information and Systems*, E84-D (3), 2001.

N. Abe and M. K. Warmuth. On the computational complexity of approximating distributions by probabilistic automata. *Machine Learning*, 9:205–260, 1992.

P. Adriaans, H. Fernau, and M. van Zaannen, editors. *Grammatical Inference: Algorithms and Applications, ICGI '02*, volume 2484 of *LNAI*, Berlin, Heidelberg, 2002. Springer-Verlag.

R. C. Carrasco. Accurate computation of the relative entropy between stochastic regular grammars. *RAIRO (Theoretical Informatics and Applications)*, 31(5):437–444, 1997.

R. C. Carrasco and J. Oncina. Learning stochastic regular grammars by means of a state merging method. In R. C. Carrasco and J. Oncina, editors, *Grammatical Inference and Applications, ICGI-94*, number 862 in LNAI, pages 139–152, Berlin, Heidelberg, 1994. Springer Verlag.

R. C. Carrasco and J. Oncina. Learning deterministic regular grammars from stochastic samples in polynomial time. *Theoretical Informatics and Applications*, 33(1):1–20, 1999.

T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley Series in Telecommunications. John Wiley & Sons, 1991.

C. de la Higuera, J. Oncina, and E. Vidal. Identification of DFA: data-dependent vs data-independent algorithms. In *Proceedings of 3rd Intl Coll. on Grammatical Inference*, LNAI, pages 313–325. Springer, sept 1996. ISBN 3-540-61778-7.

C. de la Higuera and F. Thollard. Identification in the limit with probability one of stochastic deterministic finite automata. In A. de Oliveira, editor, *Grammatical Inference: Algorithms and Applications, ICGI '00*, volume 1891 of *LNAI*, Berlin, Heidelberg, 2000. Springer-Verlag.

F. Denis. Learning regular languages from simple positive examples. *Machine Learning*, 44(1/2): 37–66, 2001.

R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis: Probabilistic Models of proteins and nucleic acids*. Cambridge University Press, 1999.

Y. Esposito, A. Lemay, F. Denis, and P. Dupont. Learning probabilistic residual finite state automata. In Adriaans et al. (2002), pages 77–91.

W. Hoeffding. Probability inequalities for sums of bounded random variables. *American Statistical Association Journal*, 58:13–30, 1963.

M. Kearns and G. Valiant. Cryptographic limitations on learning boolean formulae and finite automata. *JACM*, 41(1):67–95, January 1994.

M. J. Kearns, Y. Mansour, D. Ron, R. Rubinfeld, R. E. Schapire, and L. Sellie. On the learnability of discrete distributions. In *Proc. of the 25th Annual ACM Symposium on Theory of Computing*, pages 273–282, 1994.

C. Kermorvant and P. Dupont. Stochastic grammatical inference with multinomial tests. In Adriaans et al. (2002), pages 140–160.

M. Mohri. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(4), 1997.

R. Parekh and V. Honavar. Learning DFA from simple examples. *Machine Learning*, 44(1/2):9–35, 2001.

D. Ron, Y. Singer, and N. Tishby. On the learnability and usage of acyclic probabilistic finite automata. In *COLT 1995*, pages 31–40, Santa Cruz CA USA, 1995. ACM.

D. Ron, Y. Singer, and N. Tishby. On the learnability and usage of acyclic probabilistic finite automata. *Journal of Computer and System Sciences (JCSS)*, 56(2):133–152, 1998.

F. Thollard, P. Dupont, and C. de la Higuera. Probabilistic DFA inference using Kullback-Leibler divergence and minimality. In Pat Langley, editor, *Seventh Intl. Conf. on Machine Learning*, San Francisco, June 2000. Morgan Kaufmann.

L. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134 – 1142, 1984.