Machine Learning, 44, 9–35, 2001
© 2001 Kluwer Academic Publishers. Manufactured in The Netherlands.

# Learning DFA from Simple Examples

RAJESH PAREKH rparekh@bluemartini.com
*Blue Martini Software, 2600 Campus Drive, San Mateo, CA 94403, USA*

VASANT HONAVAR honavar@cs.iastate.edu
*Department of Computer Science, Iowa State University, Ames, IA 50011, USA*

**Editor:** Colin de la Higuera

**Abstract.** Efficient learning of DFA is a challenging research problem in *grammatical inference*. It is known that both exact and approximate (in the PAC sense) identifiability of DFA is hard. Pitt has posed the following open research problem: "*Are DFA PAC-identifiable if examples are drawn from the uniform distribution, or some other known simple distribution?*" (Pitt, in *Lecture Notes in Artificial Intelligence*, 397, pp. 18–44, Springer-Verlag, 1989). We demonstrate that the class of DFA whose canonical representations have logarithmic Kolmogorov complexity is efficiently PAC learnable under the Solomonoff Levin universal distribution (**m**). We prove that the class of DFA is efficiently learnable under the PACS (PAC learning with *simple* examples) model (Denis, D'Halluin & Gilleron, *STACS'96—Proceedings of the 13th Annual Symposium on the Theoretical Aspects of Computer Science*, pp. 231–242, 1996) wherein positive and negative examples are sampled according to the universal distribution conditional on a description of the target concept. Further, we show that any concept that is learnable under Gold's model of learning from characteristic samples, Goldman and Mathias' polynomial teachability model, and the model of learning from example based queries is also learnable under the PACS model.

**Keywords:** DFA inference, exact identification, characteristic sets, PAC learning, collusion

## 1. Introduction

The problem of learning a minimum state DFA that is consistent with a given sample has been actively studied for over two decades. DFAs are recognizers for *regular* languages which constitute the simplest class in the Chomsky hierarchy of formal languages (Chomsky, 1956; Hopcroft & Ullman, 1979). An understanding of the issues and problems encountered in learning regular languages (or equivalently, identification of the corresponding DFA) are therefore likely to provide insights into the problem of learning more general classes of languages.

Exact learning of the target DFA from an arbitrary presentation of labeled examples is a hard problem (Gold, 1978). Gold showed that the problem of identifying the minimum state DFA consistent with a presentation $S$ comprising of a finite non-empty set of positive examples $S^+$ and possibly a finite non-empty set of negative examples $S^-$ is $NP$-hard. Under the standard *complexity theoretic* assumption $P \neq NP$, Pitt and Warmuth (1989) showed that no polynomial time algorithm can be guaranteed to produce a DFA with at most $N^{(1-\epsilon)\log\log(N)}$ states from a set of labeled examples corresponding to a DFA with $N$ states.

Efficient algorithms for identification of DFAs assume that additional information is provided to the learner. Trakhtenbrot and Barzdin (1973) described a polynomial time algorithm for constructing the smallest DFA consistent with a *complete labeled sample* i.e., a sample that includes all strings up to a particular length and the corresponding label that states whether the string is accepted by the target DFA or not. Angluin (1981) showed that given a *live-complete* set of examples (that contains a representative string for each live state of the target DFA) and a knowledgeable teacher to answer *membership queries* (queries of the form "*Does the string y belong to the language of the target DFA*?") it is possible to exactly learn the target DFA. In a later paper, Angluin (1987) relaxed the requirement of a live-complete set and has described a polynomial time algorithm ($L^*$) for learning the target DFA using both *membership* and *equivalence* queries (queries of the form "*Is the current hypothesis equivalent to the target DFA*?"). The $L^*$ algorithm tacitly assumes that the learner has the capacity to *reset* the DFA to the start state before posing each membership query. This assumption might not be realistic in situations wherein it is not feasible to remember the start state or the path taken from the start state to reach the current state while evaluating a membership query. To overcome this limitation of the unknown start state, Rivest and Schapire (1993) have proposed a learning method based on *homing sequences* that runs $N$ copies of $L^*$ in parallel, one for each of the $N$ states of the target DFA. The *regular positive and negative inference* (RPNI) algorithm is a framework for identifying in polynomial time, a DFA consistent with a given sample $S$ (Oncina & García, 1992). Further, if $S$ is a superset of a characteristic set (see Section 2.1) for the target DFA then the DFA output by the RPNI algorithm is guaranteed to be equivalent to the target (Oncina & García, 1992; Dupont, 1996).

Pitt surveyed several approaches for *approximate* identification of DFA (Pitt, 1989). Valiant's distribution-independent model of learning, also called the *probably approximately correct* (PAC) learning model (Valiant, 1984), is a widely used framework for approximate learning of concept classes. When adapted to the problem of learning DFA, the goal of a PAC learning algorithm is to obtain in polynomial time, with high probability, a DFA that is a good approximation of the target DFA. We define PAC learning of DFA more formally in Section 2. Angluin's $L^*$ algorithm (Angluin, 1987) that learns DFA in polynomial time using *membership* and *equivalence* queries can be recast under the PAC framework to learn by posing membership queries alone. Pitt and Warmuth (1988) showed that the problem of polynomially approximate predictability of the class of DFA is hard. They used *prediction preserving reductions* to show that if DFAs are polynomially approximately predictable then so are other known hard to learn concept classes such as *boolean formulas*. Further, Kearns and Valiant (1989) showed that an efficient algorithm for learning DFA would entail efficient algorithms for solving problems such as breaking the *RSA* cryptosystem, factoring *Blum* integers, and detecting *quadratic residues*. Under the standard *cryptographic assumptions* these problems are known to be hard to solve. Thus, they argued that learning DFA from any randomly drawn set of examples is a hard problem.

The PAC model's requirement of learnability under all conceivable distributions is often considered too stringent for practical learning scenarios. Pitt's paper (1989) identified the following open research problem: "*Are DFA's PAC-identifiable if examples are drawn from* *the uniform distribution, or some other known simple distribution?*". Using a variant of Trakhtenbrot and Barzdin's algorithm, Lang (1992) empirically demonstrated that random DFAs are approximately learnable from a sparse uniform sample. However, exact identification of the target DFA was not possible even in the average case with a randomly drawn training sample. Several efforts have been made to study the learnability of concept classes under restricted classes of distributions. Li and Vitányi (1991) proposed a model for PAC learning with *simple* examples called the *simple PAC* model wherein the class of distributions is restricted to *simple* distributions (see Section 4). Denis et al. (1996) proposed a model of learning where examples are drawn at random according to the universal distribution conditional on the knowledge of the target concept. This model is known as the PACS learning model. In this paper, we present a method for efficient PAC learning of DFA from simple examples. We will prove that the class of logarithmic Kolmogorov complexity DFA (see Section 4) is learnable under the simple PAC model and the entire class of DFA is learnable under the PACS model. Further, we demonstrate how the model of learning from simple examples naturally extends the model of *learning concepts from representative examples* (Gold, 1978), the *polynomial teachability* model (Goldman & Mathias, 1993), and the model of *learning from example based queries* (Angluin, 1988) to a probabilistic framework.

This paper is organized as follows: Section 2 briefly introduces some concepts used in the results described in this paper. This includes a discussion of the PAC learning model, Kolmogorov complexity, and the universal distribution. Section 3 reviews the RPNI algorithm for learning DFA. Section 4 discusses the PAC learnability of the class of logarithmic Kolmogorov complexity DFA under the simple PAC learning model. Section 5 demonstrates the PAC learnability of the entire class of DFA under the PACS learning model. Section 6 analyzes the PACS model in relation with other models for concept learning. Section 7 addresses the issue of collusion that arises because a helpful teacher can potentially encode the target DFA as a labeled training example. Section 8 concludes with a summary of the main contributions of this paper and some directions for future research.

## 2. Preliminaries

Let $\Sigma$ be a finite set of symbols called the *alphabet*; $\Sigma^*$ be the set of strings over $\Sigma$; $\alpha, \beta, \gamma$ be strings in $\Sigma^*$; and $|\alpha|$ be the length of the string $\alpha$. $\lambda$ is a special string called the *null* string and has length 0. Given a string $\alpha = \beta\gamma$, $\beta$ is the *prefix* of $\alpha$ and $\gamma$ is the *suffix* of $\alpha$. Let $Pref(\alpha)$ denote the set of all prefixes of $\alpha$. A *language* $L$ is a subset of $\Sigma^*$. The set $Pref(L) = \{\alpha \mid \alpha\beta \in L\}$ is the set of *prefixes* of the language and the set $L_\alpha = \{\beta \mid \alpha\beta \in L\}$ is the set of *tails* of $\alpha$ in $L$. The *standard order* of strings of the alphabet $\Sigma$ is denoted by $<$. The standard enumeration of strings over $\Sigma = \{a, b\}$ is $\lambda, a, b, aa, ab, ba, bb, aaa, \ldots$ The set of *short prefixes* $S_p(L)$ of a language $L$ is defined as $S_p(L) = \{\alpha \in Pref(L) \mid \not\exists \beta \in \Sigma^*$ such that $L_\alpha = L_\beta$ and $\beta < \alpha\}$. The *kernel* $N(L)$ of a language $L$ is defined as $N(L) = \{\lambda\} \cup \{\alpha a \mid \alpha \in S_p(L), a \in \Sigma, \alpha a \in Pref(L)\}$. Given two sets $S_1$ and $S_2$, let $S_1 \backslash S_2$ and $S_1 \oplus S_2$ denote the *set difference* and the *symmetric difference* respectively. Let ln and lg denote the log to the bases $e$ and 2 respectively.

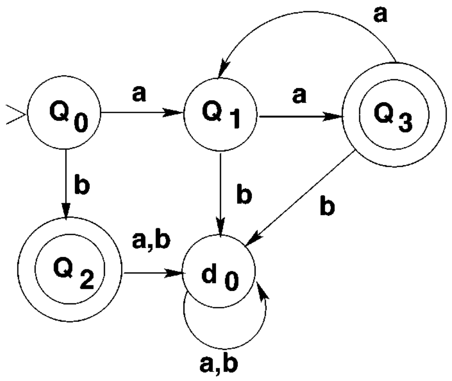


*Figure 1.* Deterministic finite state automaton.

### 2.1. Finite automata

A *deterministic* finite state automaton (DFA) is a quintuple $A = (Q, \delta, \Sigma, q_0, F)$ where, $Q$ is a finite set of states, $\Sigma$ is a finite alphabet, $q_0 \in Q$ is the start state, $F \subseteq Q$ is the set of accepting states, and $\delta$ is the transition function: $Q \times \Sigma \rightarrow Q$. A state $d_0 \in Q$ such that $\forall a \in \Sigma,\ \delta(d_0, a) = d_0$ is called a *dead* state. The extension of $\delta$ to handle input strings is standard and is denoted by $\delta^*$. The set of all strings accepted by $A$ is its language, $L(A)$. The language accepted by a DFA is called a *regular language*. Figure 1 shows the state transition diagram for a sample DFA. A *non-deterministic* finite automaton (NFA) is defined just like the DFA except that the transition function $\delta$ defines a mapping from $Q \times \Sigma \rightarrow 2^Q$. In general, a finite state automaton (FSA) refers to either a DFA or a NFA.

Given any FSA $A'$, there exists a minimum state DFA (also called the *canonical DFA*, $A$) such that $L(A) = L(A')$. Without loss of generality, we will assume that the target DFA being learned is a canonical DFA. Let $N$ denote the number of states of $A$. It can be shown that any canonical DFA has at most one dead state (Hopcroft & Ullman, 1979). One can define a standard encoding of DFA as binary strings such that any DFA with $N$ states is encoded as a binary string of length $O(N \lg N)$. A labeled example $(\alpha, c(\alpha))$ for $A$ is such that $\alpha \in \Sigma^*$ and $c(\alpha) = +$ if $\alpha \in L(A)$ (i.e., $\alpha$ is a positive example) or $c(\alpha) = -$ if $\alpha \notin L(A)$ (i.e., $\alpha$ is a negative example). Let $S^+$ and $S^-$ denote the set of *positive* and *negative* examples of $A$ respectively. $A$ is consistent with a *sample* $S = S^+ \cup S^-$ if it accepts all positive examples and rejects all negative examples.

A set $S^+$ is said to be *structurally complete* with respect to a DFA $A$ if $S^+$ covers each transition of $A$ (except the transitions associated with the dead state $d_0$) and uses every element of the set of final states of $A$ as an accepting state (Pao & Carr, 1978; Parekh & Honavar, 1993; Dupont et al., 1994). It can be verified that the set $S^+ = \{b, aa, aaaa\}$ is structurally complete with respect to the DFA in figure 1. Given a set $S^+$, let *PTA*$(S^+)$ denote the *prefix tree acceptor* for $S^+$. *PTA*$(S^+)$ is a DFA that contains a path from the start state to an accepting state for each string in $S^+$ modulo common prefixes. Clearly, $L(PTA(S^+)) = S^+$. Learning algorithms such as the RPNI (see Section 3) require the states of the PTA to be numbered in standard order. If we consider the set *Pref*$(S^+)$ of prefixes of

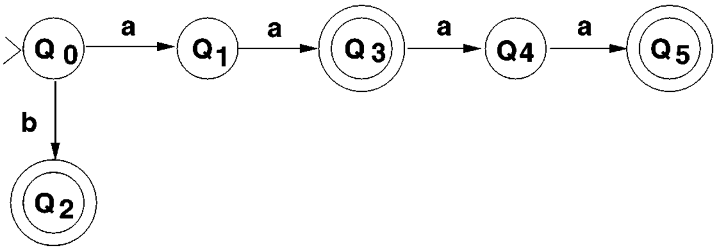


*Figure 2.* Prefix tree automaton.

the set $S^+$ then each state of the PTA corresponds to a unique element in the set *Pref*$(S^+)$ i.e., for each state $q_i$ of the PTA there exists exactly one string $\alpha_i$ in the set *Pref*$(S^+)$ such that $\delta^*(q_0, \alpha_i) = q_i$ and viceversa. The strings of *Pref*$(S^+)$ are sorted in standard order $<$ and each state $q_i$ is numbered by the position of its corresponding string $\alpha_i$ in the sorted list. The *PTA* for the set $S^+ = \{b, aa, aaaa\}$ is shown in figure 2. Note that its states are numbered in standard order.

Given a FSA $A$ and a partition $\pi$ on the set of states $Q$ of $A$ (ignoring the dead state $d_0$ and its associated transitions), we define the *quotient automaton* $A_\pi = (Q_\pi, \delta_\pi, \Sigma, B(q_0, \pi), F_\pi)$ obtained by merging the states of $A$ that belong to the same block of the partition $\pi$ as follows: $Q_\pi = \{B(q, \pi) \mid q \in Q\}$ is the set of states with each state represented uniquely by the block $B(q, \pi)$ of the partition $\pi$ that contains the state $q$, $F_\pi = \{B(q, \pi) \mid q \in F\}$ is the set of accepting states, and $\delta_\pi : Q_\pi \times \Sigma \to 2^{Q_\pi}$ is the transition function such that $\forall B(q_i, \pi), B(q_j, \pi) \in Q_\pi, \forall a \in \Sigma, B(q_j, \pi) = \delta_\pi(B(q_i, \pi), a)$ *iff* $q_i, q_j \in Q$ and $q_j = \delta(q_i, a)$. Note that a quotient automaton of a DFA might be a NFA and viceversa. For example, the quotient automaton corresponding to the partition $\pi = \{\{Q_0, Q_1\}, \{Q_2\}, \{Q_3\}\}$ of the set of states of the DFA in figure 1 is shown in figure 3.

The set of all quotient automata obtained by systematically merging the states of a DFA $A$ represents a *lattice* of FSA (Pao & Carr, 1978). This lattice is ordered by the *grammar cover* relation $\preceq$. Given two partitions $\pi_i = \{B_1, B_2, \ldots, B_r\}$ and $\pi_j = \{B_1, B_2, \ldots, B_k\}$ of the states of $A$, we say that $\pi_i$ covers $\pi_j$ (written $\pi_j \preceq \pi_i$) if $r = k - 1$ and for some

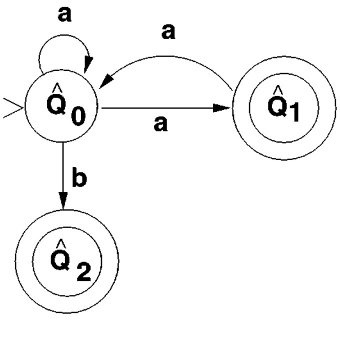


*Figure 3.* Quotient automaton.

$1 \leq l, m \leq k$, $\pi_i = \{\pi_j \backslash \{B_l, B_m\} \cup \{B_l \cup B_m\}\}$. The *transitive closure* of $\preceq$ is denoted by $\ll$. We say that $A_{\pi_j} \ll A_{\pi_i}$ *iff* $L(A_{\pi_j}) \subseteq L(A_{\pi_i})$. Given a canonical DFA $A$ and a set $S^+$ that is structurally complete with respect to $A$, the lattice $\Omega(S^+)$ derived from $PTA(S^+)$ is guaranteed to contain $A$ (Pao & Carr, 1978; Parekh & Honavar, 1993; Dupont et al., 1994).

A sample $S = S^+ \cup S^-$ is said to be *characteristic* with respect to a regular language $L$ (with a canonical acceptor $A$) if it satisfies the following two conditions (Oncina & García, 1992):

- $\forall \alpha \in N(L)$, if $\alpha \in L$ then $\alpha \in S^+$ else $\exists \beta \in \Sigma^*$ such that $\alpha\beta \in S^+$.
- $\forall \alpha \in S_p(L), \forall \beta \in N(L)$, if $L_\alpha \neq L_\beta$ then $\exists \gamma \in \Sigma^*$ such that $(\alpha\gamma \in S^+$ and $\beta\gamma \in S^-)$ or $(\beta\gamma \in S^+$ and $\alpha\gamma \in S^-)$.

Intuitively, $S_p(L)$, the set of short prefixes of $L$ is a live complete set with respect to $A$ in that for each live state $q \in Q$, there is a string $\alpha \in S_p(L)$ such that $\delta^*(q_0, \alpha) = q$. The kernel $N(L)$ includes the set of short prefixes as a subset. Thus, $N(L)$ is also a live complete set with respect to $A$. Further, $N(L)$ covers every transition between each pair of live states of $A$. i.e., for all live states $q_i, q_j \in Q$, for all $a \in \Sigma$, if $\delta(q_i, a) = q_j$ then there exists a string $\gamma \in N(L)$ such that $\gamma = \alpha a$ and $\delta^*(q_0, \alpha) = q_i$. Thus, condition 1 above which identifies a suitably defined suffix $\beta \in \Sigma^*$ for each string $\alpha \in N(L)$ such that the concatenated string $\alpha\beta \in L$ implies structural completeness with respect to $A$. Condition 2 implies that for any two distinct states of $A$ there is a suffix $\gamma$ that would correctly distinguish them. In other words, for any $q_i, q_j \in Q$ where $q_i \not\equiv q_j$, $\exists \gamma \in \Sigma^*$ such that $\delta^*(q_i, \gamma) \in F$ and $\delta^*(q_j, \gamma) \notin F$ or viceversa. Given the language $L$ corresponding to the DFA $A$ in figure 1, the set of short prefixes is $S_p(L) = \{\lambda, a, b, aa\}$ and the kernel is $N(L) = \{\lambda, a, b, aa, aaa\}$. It can be easily verified that the set $S = S^+ \cup S^-$ where $S^+ = \{b, aa, aaaa\}$ and $S^- = \{\lambda, a, aaa, baa\}$ is a characteristic sample for $L$.

### *2.2. PAC learning of DFA*

Let $\mathcal{X}$ denote the *sample space* defined as the set of all strings $\Sigma^*$. Let $x \subseteq \mathcal{X}$ denote a *concept*. For our purpose, $x$ is a *regular language*. We identify the concept with the corresponding DFA and denote the class of all DFA as the *concept class* $\mathcal{C}$. The *representation* $\mathcal{R}$ that assigns a name to each DFA in $\mathcal{C}$ is defined as a function $\mathcal{R} : \mathcal{C} \rightarrow \{0, 1\}^*$. $\mathcal{R}$ is the set of standard encodings of the DFAs in $\mathcal{C}$. Assume that there is an unknown and arbitrary but fixed distribution $\mathcal{D}$ according to which the examples of the target concept are drawn.

*Definition 1* (due to Pitt (1989)). DFAs are PAC-identifiable *iff* there exists a (possibly randomized) algorithm $\mathcal{A}$ such that on input of any parameters $\epsilon$ and $\delta$, for any DFA $M$ of size $N$, for any number $m$, and for any probability distribution $\mathcal{D}$ on strings of $\Sigma^*$ of length at most $m$, if $\mathcal{A}$ obtains labeled examples of $M$ generated according to the distribution $\mathcal{D}$, then $\mathcal{A}$ produces a DFA $M'$ such that with probability at least $1 - \delta$, the probability (with respect to distribution $\mathcal{D}$) of the set $\{\alpha \mid \alpha \in L(M) \oplus L(M')\}$ is at most $\epsilon$. The run time of $\mathcal{A}$ (and hence the number of randomly generated examples obtained by $\mathcal{A}$) is required to be polynomial in $N$, $m$, $1/\epsilon$, $1/\delta$, and $|\Sigma|$.

In several typical learning tasks all example strings for a given target concept are of the same length. Example strings of DFAs can be of different lengths. Thus, in the context of DFA learning, $\mathcal{D}$ is restricted to a probability distribution on strings of $\Sigma^*$ of length at most $m$ in order to prevent inordinately long strings from being drawn. Note that for any distribution $\mathcal{D}$ over $\Sigma^*$ and for an arbitrarily small $\gamma > 0$, there exists a length $m$ such that the probability of any string of length greater than $m$ is at most $\gamma$. Thus, there is a distribution $\hat{\mathcal{D}}$ assigning zero probability to all strings of length greater than $m$ such that $\hat{\mathcal{D}}$ *approximates* $\mathcal{D}$ within $\gamma$ (Pitt, 1989). In the $L^*$ algorithm for learning DFA (Angluin, 1987), the longest example seen by the learner is typically the longest counter-example presented by the teacher. If we assume a scenario wherein the teacher always presents the learner with the shortest possible counter-example then it can be shown that counter-examples of length no more than $2N - 1$ are needed to correctly learn the target DFA of $N$ states. Thus, in practice we will assume that $m \geq 2N - 1$.

*Definition 2.* DFAs are *probably exactly learnable iff* there exists a (possibly randomized) algorithm $\mathcal{A}$ such that for any given value of the input parameter $\delta$, for any DFA $M$ of size $N$, for any number $m$, and for any probability distribution $\mathcal{D}$ on strings of $\Sigma^*$ of length at most $m$, if $\mathcal{A}$ obtains labeled examples of $M$ generated according to the distribution $\mathcal{D}$, then $\mathcal{A}$ produces a DFA $M'$ such that with probability at least $1 - \delta$, $M'$ is equivalent to $M$ i.e., $\Pr_{\mathcal{D}}(\{\alpha \mid \alpha \in L(M) \oplus L(M')\} = 0)$. The run time of $\mathcal{A}$ (and hence the number of randomly generated examples obtained by $\mathcal{A}$) is required to be polynomial in $N$, $m$, $1/\delta$, and $|\Sigma|$.

### 2.3. Kolmogorov complexity

Note that the definition of PAC learning requires that the concept class (in this case the class of DFA) must be learnable under any arbitrary (but fixed) probability distribution. This requirement is often considered too stringent in practical learning scenarios where it is not unreasonable to assume that a learner is first provided with *simple* and *representative* examples of the target concept. Intuitively, when we teach a child the rules of *multiplication* we are more likely to first give simple examples like 3 × 4 than examples like 1377 × 428. A *representative set* of examples is one that would enable the learner to identify the target concept exactly. For example, the characteristic set of a DFA would constitute a suitable representative set. The question now is whether we can formalize what simple examples mean. *Kolmogorov complexity* provides a machine independent notion of *simplicity* of objects. Intuitively, the Kolmogorov complexity of an object represented by a binary string $\alpha$ is the length of the shortest binary program that computes $\alpha$. Objects that have regularity in their structure (i.e., objects that can be easily compressed) have low Kolmogorov complexity. For example, consider the string $s_1 = 010101\ldots01 = (01)^{500}$. On a particular machine $M$, a program to compute this string would be "*Print 01 500 times*". On the other hand consider a totally random string $s_2 = 110011010\ldots 00111$. Unlike $s_1$, it is not possible to compress the string $s_2$ which means that a program to compute $s_2$ on $M$ would be "*Print 1100111010000 . . . 00111*", i.e., the program would have to explicitly specify the string $s_2$. The length of the program that computes $s_1$ is shorter than that of the program that computes

$s_2$. Thus, we could argue that $s_1$ has lower Kolmogorov complexity than $s_2$ with respect to the machine $M$.

We will consider the *prefix* version of the Kolmogorov complexity that is measured with respect to prefix Turing machines and denoted by $K$. A Turing machine $M$ is a *prefix Turing machine* if the set of inputs $P$ for which $M$ halts is a *prefix code*, i.e., no element of $P$ is a prefix of any other element of $P$. Consider a prefix Turing machine that implements the partial recursive function $\phi : \{0,1\}^* \xrightarrow{partial} \{0,1\}^*$. For any string $\alpha \in \{0,1\}^*$, the Kolmogorov complexity of $\alpha$ relative to $\phi$ is defined as $K_\phi(\alpha) = \min\{|\pi| \mid \phi(\pi) = \alpha\}$ where $\pi \in \{0,1\}^*$ is a program input to the Turing machine. Prefix Turing machines can be effectively enumerated and there exists a *Universal Turing Machine* ($U$) capable of simulating every prefix Turing machine. Assume that the universal Turing machine implements the partial function $\psi$. The *Optimality Theorem* for Kolmogorov complexity guarantees that for any prefix Turing machine $\phi$ there exists a constant $c_\phi$ such that for any string $\alpha$, $K_\psi(\alpha) \leq K_\phi(\alpha) + c_\phi$. Note that we use the name of the Turing Machine (say $M$) and the partial function it implements (say $\phi$) interchangeably i.e., $K_\phi(\alpha) = K_M(\alpha)$. Further, by the *Invariance Theorem* it can be shown that for any two universal machines $\psi_1$ and $\psi_2$ there is a constant $\eta \in \mathcal{N}$ (where $\mathcal{N}$ is the set of natural numbers) such that for all strings $\alpha$, $|K_{\psi_1}(\alpha) - K_{\psi_2}(\alpha)| \leq \eta$. Thus, we can fix a single universal Turing machine $U$ and denote $K(\alpha) = K_U(\alpha)$. Note that there exists a Turing machine that computes the identity function $\chi : \{0,1\}^* \rightarrow \{0,1\}^*$ where $\forall\alpha,\ \chi(\alpha) = \alpha$. Thus, it can be shown that the Kolmogorov complexity of an object is bounded by its length, i.e., $K(\alpha) \leq |\alpha| + K(|\alpha|) + \eta$ where $\eta$ is a constant independent of $\alpha$.

Suppose that some additional information in the form of a string $\beta$ is available to the Turing machine $\phi$. The conditional Kolmogorov complexity of any object $\alpha$ given $\beta$ is defined as $K_\phi(\alpha \mid \beta) = \min\{|\pi| \mid \phi(\langle\pi, \beta\rangle) = \alpha\}$ where $\pi \in \{0,1\}^*$ is a program and $\langle x, y\rangle$ is a standard pairing function[1]. Note that the definition of conditional Kolmogorov complexity does not charge for the extra information $\beta$ that is available to $\phi$ along with the program $\pi$. Fixing a single universal Turing machine $U$ we denote the conditional Kolmogorov complexity of $\alpha$ by $K(\alpha \mid \beta) = K_U(\alpha \mid \beta)$. It can be shown that $K(\alpha \mid \beta) \leq K(\alpha) + \eta$ where $\eta$ is a constant independent of $\alpha$.

### 2.4. Universal distribution

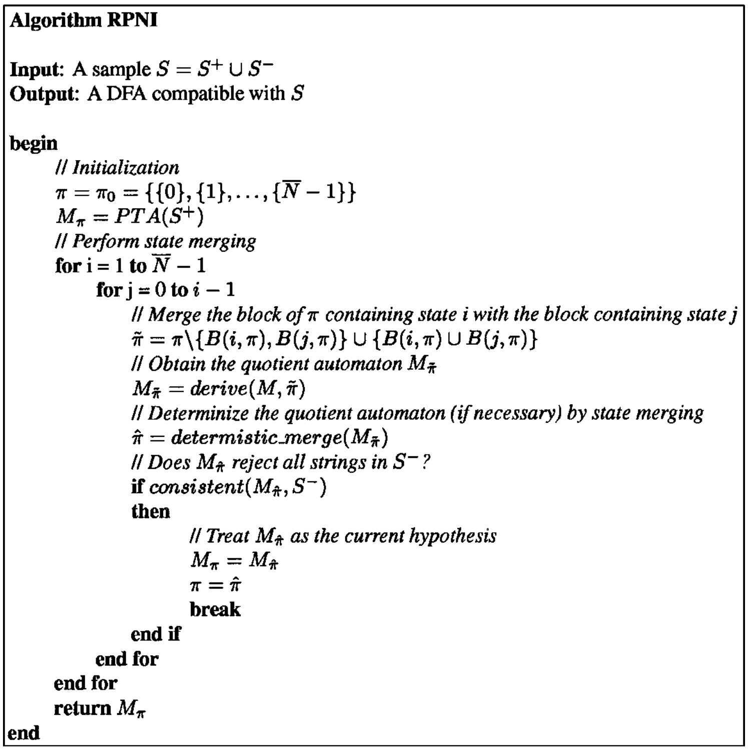
The Solomonoff Levin universal distribution $\mathbf{m}$ is a *universal enumerable probability distribution* in that it multiplicatively dominates all enumerable probability distributions. Formally, $\forall i \in \mathcal{N}^+\ \exists c > 0\ \forall x \in \mathcal{N}\ [c\mathbf{m}(x) \geq P_i(x)]$ where $P_1, P_2, \ldots$ is an enumeration of all enumerable probability distributions and $\mathcal{N}$ is the set of natural numbers. It can be shown that $\mathbf{m}(x) = 2^{-K(x)+O(1)}$. Thus, under $\mathbf{m}$, simple objects (or objects with low Kolmogorov complexity) have a high probability, and complex or random objects have a low probability. Given a string $r \in \Sigma^*$, the universal distribution conditional on the knowledge of $r$, $\mathbf{m}_r$, is defined as $\mathbf{m}_r(\alpha) = 2^{-K(\alpha \mid r)+O(1)}$ (Denis et al., 1996). Further, $\forall r \in \Sigma^*\ \sum_\alpha \mathbf{m}_r(\alpha) < 1$.

The interested reader is referred to Li and Vitányi (1997) for a thorough treatment of Kolmogorov complexity, universal distribution, and related topics.

## 3. The RPNI algorithm

The *regular positive and negative inference* (RPNI) algorithm (Oncina & García, 1992) is a polynomial time algorithm for identification of a DFA consistent with a given set $S = S^+ \cup S^-$. If the sample is a characteristic set for the target DFA then the algorithm is guaranteed to return a canonical representation of the target DFA. Our description of the RPNI algorithm is based on the explanation given in Dupont (1996).

A labeled sample $S = S^+ \cup S^-$ is provided as input to the algorithm. It constructs a prefix tree automaton $PTA(S^+)$ and numbers its states in the standard order. Then it performs an ordered search in the space of partitions of the set of states of $PTA\ (S^+)$ under the control of the set of negative examples $S^-$. The partition, $\pi_0$, corresponding to the automaton $PTA\ (S^+)$ itself is $\{\{0\}, \{1\}, \ldots, \{\bar{N} - 1\}\}$ where $\bar{N}$ is the number of states of the PTA. $M_{\pi_0} = PTA(S^+)$ is consistent with all the training examples and is treated as the initial hypothesis. The current hypothesis is denoted by $M_\pi$ and the corresponding partition is denoted by $\pi$. The algorithm is outlined in figure 4. The nested *for* loop refines the partition $\pi$ by merging the states of $PTA\ (S^+)$ in order. At each step, a partition $\tilde{\pi}$ is obtained from the partition $\pi$ by merging the two blocks that contain the states $i$ and $j$ respectively.



```
Algorithm RPNI

Input: A sample $S = S^+ \cup S^-$
Output: A DFA compatible with $S$

begin
    // Initialization
    $\pi = \pi_0 = \{\{0\}, \{1\}, \ldots, \{\overline{N} - 1\}\}$
    $M_\pi = PTA(S^+)$
    // Perform state merging
    for i = 1 to $\overline{N} - 1$
        for j = 0 to $i - 1$
            // Merge the block of $\pi$ containing state i with the block containing state j
            $\tilde{\pi} = \pi \backslash \{B(i,\pi), B(j,\pi)\} \cup \{B(i,\pi) \cup B(j,\pi)\}$
            // Obtain the quotient automaton $M_{\tilde{\pi}}$
            $M_{\tilde{\pi}} = derive(M, \tilde{\pi})$
            // Determinize the quotient automaton (if necessary) by state merging
            $\hat{\pi} = determistic\_merge(M_{\tilde{\pi}})$
            // Does $M_{\hat{\pi}}$ reject all strings in $S^-$ ?
            if $consistent(M_{\hat{\pi}}, S^-)$
            then
                // Treat $M_{\hat{\pi}}$ as the current hypothesis
                $M_\pi = M_{\hat{\pi}}$
                $\pi = \hat{\pi}$
                break
            end if
        end for
    end for
    return $M_\pi$
end
```

*Figure 4.* The RPNI algorithm.

The function *derive* obtains the quotient automaton $M_{\tilde{\pi}}$, corresponding to the partition $\tilde{\pi}$. $M_{\tilde{\pi}}$ might be a NFA in which case the function *deterministic_merge* converts the NFA to a DFA by recursively merging the states that cause non-determinism. For example, if $q_i$, $q_j$, and $q_k$ are states of $M_{\tilde{\pi}}$ such that for some $a \in \Sigma$, $\delta(q_i, a) = \{q_j, q_k\}$ then the states $q_j$ and $q_k$ are merged together. This recursive merging of states can go on for at most $N - 1$ steps and the resulting automaton $M_{\hat{\pi}}$ is guaranteed to be a DFA (Dupont, 1996). Note that if $M_{\tilde{\pi}}$ is a NFA then the resulting DFA obtained $M_{\hat{\pi}}$ obtained by invoking *deterministic_merge* is not necessarily equivalent to $M_{\tilde{\pi}}$. However, since $\tilde{\pi} \ll \hat{\pi}$ we know by the grammar covers relation that $L(M_{\tilde{\pi}}) \subseteq L(M_{\hat{\pi}})$ and thus, if $M_{\tilde{\pi}}$ accepts a negative example in $S^-$ then so would $M_{\hat{\pi}}$. The function, *consistent*$(M_{\hat{\pi}}, S^-)$ returns *True* if $M_{\hat{\pi}}$ is consistent with all examples in $S^-$ and *False* otherwise. If a partition $\hat{\pi}$ is found such that the corresponding DFA $M_{\hat{\pi}}$ is consistent with $S^-$ then $M_{\hat{\pi}}$ replaces $M_\pi$ as the current hypothesis.

Let $\|S^+\|$ and $\|S^-\|$ denote the sums of the lengths of examples in $S^+$ and $S^-$ respectively. *PTA* $(S^+)$ has $O(\|S^+\|)$ states. The nested *for* loop of the algorithm performs $O(\|S^+\|^2)$ state merges. Further, each time two blocks of the partition $\pi$ are merged, the routine *deterministic_merge* in the worst case would cause $O(\|S^+\|)$ state mergings and the function *consistent* that checks for the consistency of the derived DFA with the negative examples would incur a cost of $O(\|S^-\|)$. Hence the time complexity of the RPNI algorithm is $O((\|S^+\| + \|S^-\|) \cdot \|S^+\|^2)$.

*Example.* We demonstrate the execution of the RPNI algorithm on the task of learning the DFA in figure 1. Note that for convenience we have shown the target DFA in figure 5 without the dead state $d_0$ and its associated transitions. Assume that a sample $S = S^+ \cup S^-$ where $S^+ = \{b, aa, aaaa\}$ and $S^- = \{\lambda, a, aaa, baa\}$. It can be easily verified that $S$ is a characteristic sample for the target DFA. The DFA $M = PTA(S^+)$ is depicted in figure 2 where the states are numbered in the standard order. The initial partition is $\pi = \pi_0 = \{\{0\}, \{1\}, \{2\}, \{3\}, \{4\}, \{5\}\}$.

The algorithm attempts to merge the blocks containing states 1 and 0 of the partition $\pi$. The quotient FSA $M_{\tilde{\pi}}$ and the DFA $M_{\hat{\pi}}$ obtained after invoking *deterministic_merge* are

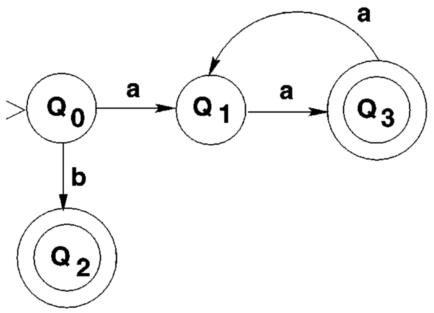


*Figure 5.* Target DFA *A*.

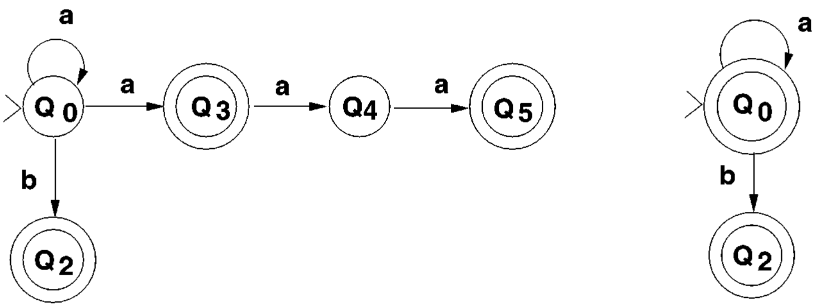


*Figure 6.* $M_{\tilde{\pi}}$ obtained by fusing blocks containing the states 1 and 0 of $\pi$ and the corresponding $M_{\hat{\pi}}$.

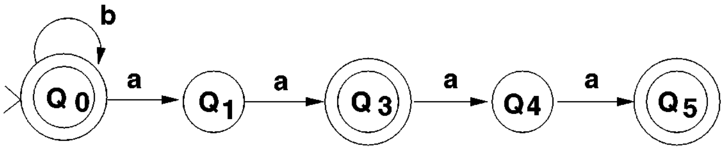


*Figure 7.* $M_{\tilde{\pi}}$ (same as $M_{\hat{\pi}}$) obtained by fusing blocks containing the states 2 and 0 of $\pi$.

shown in figure 6. The DFA $M_{\hat{\pi}}$ accepts the negative example $\lambda \in S^-$. Thus, the current partition $\pi$ remains unchanged.

Next the algorithm merges the blocks containing states 2 and 0 of the partition $\pi$. The quotient FSA $M_{\tilde{\pi}}$ is depicted in figure 7. Since $M_{\tilde{\pi}}$ is a DFA, the procedure *deterministic_merge* returns the same automaton i.e., $M_{\hat{\pi}} = M_{\tilde{\pi}}$. $M_{\hat{\pi}}$ accepts the negative example $\lambda \in S^-$ and hence the partition $\pi$ remains unchanged.

Table 1 lists the different partitions $\tilde{\pi}$ obtained by fusing the blocks of $\pi_0$, the partitions $\hat{\pi}$ obtained by *deterministic_merge* of $\tilde{\pi}$, and the negative example (belonging to $S^-$), if any, that is accepted by the quotient FSA $M_{\hat{\pi}}$. The partitions marked $*$ denote the partition $\pi$ for which $M_{\pi}$ is consistent with all examples in $S^-$ and hence is the current hypothesis. It is easy to see that the DFA corresponding to the partition $\pi = \{\{0\}, \{1, 4\}, \{2\}, \{3, 5\}\}$ is exactly the target DFA we are trying to learn (figure 1).

## 4. Learning logarithmic Kolmogorov complexity DFA under the simple PAC model

Li and Vitányi (1991) have proposed a simple PAC learning model where the class of probability distributions is restricted to *simple* distributions. A distribution is simple if it is multiplicatively dominated by some enumerable distribution. Simple distributions properly include all computable distributions. Distributions that we commonly use in statistics such as the *uniform distribution*, *normal distribution*, *geometric distribution*, and *Poisson distribution* are simple if restricted to finite precision parameters. Further, the *simple distribution independent learning theorem* says that a concept class is learnable under the universal distribution **m** *iff* it is learnable under the entire class of *simple distributions* provided the examples are drawn according to the universal distribution (Li & Vitányi, 1991). Thus, the simple PAC learning model is sufficiently general. Concept classes such as log *n-term DNF*

*Table 1.* Execution of the RPNI algorithm.

| Partition $\tilde{\pi}$ | Partition $\hat{\pi}$ | Negative example |
|---|---|---|
| {{0, 1}, {2}, {3}, {4}, {5}} | {{0, 1, 3, 4, 5}, {2}} | *a* |
| {{0, 2}, {1}, {3}, {4}, {5}} | {{0, 2}, {1}, {3}, {4}, {5}} | λ |
| {{0}, {1, 2}, {3}, {4}, {5}} | {{0}, {1, 2}, {3}, {4}, {5}} | *a* |
| {{0, 3}, {1}, {2}, {4}, {5}} | {{0, 3}, {1, 4}, {2}, {5}} | λ |
| {{0}, {1, 3}, {2}, {4}, {5}} | {{0}, {1, 3, 4, 5}, {2}} | *a* |
| {{0}, {1}, {2, 3}, {4}, {5}} | {{0}, {1}, {2, 3}, {4}, {5}} | *baa* |
| {{0, 4}, {1}, {2}, {3}, {5}} | {{0, 4}, {1, 5}, {2}, {3}} | *a* |
| {{0}, {1, 4}, {2}, {3}, {5}} | {{0}, {1, 4}, {2}, {3, 5}}* | — |
| {{0, 3, 5}, {1, 4}, {2}} | {{0, 3, 5}, {1, 4}, {2}} | λ |
| {{0}, {1, 3, 4, 5}, {2}} | {{0}, {1, 3, 4, 5}, {2}} | *a* |
| {{0}, {1, 4}, {2, 3, 5}} | {{0}, {1, 4}, {2, 3, 5}} | *baa* |
| {{0}, {1, 4}, {2}, {3, 5}} | {{0}, {1, 4}, {2}, {3, 5}}* | — |
| {{0}, {1, 3, 4, 5}, {2}} | {{0}, {1, 3, 4, 5}, {2}} | *a* |

and *simple k-reversible DFA* are learnable under the simple PAC model whereas their PAC learnability in the standard sense is unknown (Li & Vitányi, 1991). We show that the class of DFA whose Kolmogorov complexity is $O(\lg N)$ are polynomially learnable under the simple PAC learning model[2]. We saw in Section 2.3 that a natural learning scenario would typically involve learning from a *simple* and *representative* set of examples for the target concept. We adopt Kolmogorov complexity as a measure of simplicity and define simple examples as those with low Kolmogorov complexity, i.e., with Kolmogorov complexity $O(\lg N)$. Further, a characteristic set for the DFA $A$ can be treated as its representative set.

We demonstrate that for every DFA with Kolmogorov complexity $O(\lg N)$ there exists a characteristic set of simple examples $S_c$.

**Lemma 1.** *For any $N$ state DFA with Kolmogorov complexity $O(\lg N)$ there exists a characteristic set of simple examples $S_c$ such that the length of each string in this set is at most $2N - 1$.*

**Proof:** Consider the following enumeration of a characteristic set of examples for a DFA $A = (Q, \delta, \Sigma, q_0, F)$ with $N$ states[3].

1. Fix an enumeration of the shortest paths (in standard order) from the state $q_0$ to each state in $Q$ except the dead state. This is the set of short prefixes of $A$. There are at most $N$ such paths and each path is of length at most $N - 1$.
2. Fix an enumeration of paths that includes each path identified above and its extension by each letter of the alphabet $\Sigma$. From the paths just enumerated retain only those that do not terminate in the dead state of $A$. This represents the kernel of $A$. There are at most $N(|\Sigma| + 1)$ such paths and each path is of length at most $N$.

3. Let the characteristic set be denoted by $S_c = S_c^+ \cup S_c^-$.

(A) For each string $\alpha$ identified in step 2 above, determine the first suffix $\beta$ in the standard enumeration of strings such that $\alpha\beta \in L(A)$. Since $|\alpha| \leq N$, and $\beta$ is the shortest suffix in the standard order it is clear that $|\alpha\beta| \leq 2N - 1$. Each such $\alpha\beta$ is a member of $S_c^+$.

(B) For each pair of strings $(\alpha, \beta)$ in order where $\alpha$ is a string identified in step 1, $\beta$ is a string identified in step 2, and $\alpha$ and $\beta$ lead to different states of $A$ determine the first suffix $\gamma$ in the standard enumeration of strings such that $\alpha\gamma \in L(A)$ and $\beta\gamma \notin L(A)$ or viceversa. Since $|\alpha| \leq N - 1$, $|\beta| \leq N$, and $\gamma$ is the shortest distinguishing suffix for the states represented by $\alpha$ and $\beta$ it is clear that $|\alpha\gamma|, |\beta\gamma| \leq 2N - 1$. The accepted string from among $\alpha\gamma$ and $\beta\gamma$ is a member of $S_c^+$ and the rejected string is a member of $S_c^-$.

Trivial upper bounds on the sizes of $S_c^+$ and $S_c^-$ are $|S_c^+| \leq N^2(|\Sigma| + 1) + N(|\Sigma|)$, $|S_c^-| \leq N^2(|\Sigma| + 1) - N$. Thus, $|S_c| = O(N^2)$. Further, the length of each string in $S_c$ is less than $2N - 1$.

The strings in $S_c$ can be ordered such that individual strings can be identified by an index of length $O(\lg N)$ bits. There exists a Turing machine $M$ that implements the above algorithm for constructing the set $S_c$. $M$ takes as input an encoding of the DFA of length $O(\lg N)$ bits and an index of length $O(\lg N)$ bits and outputs the corresponding string $\alpha$ belonging to $S_c$. Thus, $\forall\alpha \in S_c$, $K(\alpha) = O(\lg N)$. This proves the lemma. □

**Lemma 2.** *Suppose a sample $S$ is drawn according to* $\mathbf{m}$. *For* $0 < \delta \leq 1$, *there exist constants* $k_1 > 0$ *and* $k_2 > 0$ *such that if* $|S| \geq N^{k_1}(\ln(\frac{1}{\delta}) + \ln(k_2) + \ln(N^2))$ *then with probability greater than* $1 - \delta$, $S_c \subseteq S$.

**Proof:** From Lemma 1 we know that $\forall\alpha \in S_c,\ K(\alpha) = O(\lg N)$. Further, $|S_c| = O(N^2)$. By definition, $\mathbf{m}(\alpha) \geq 2^{-K(\alpha)}$. Thus, $\mathbf{m}(\alpha) \geq 2^{-k_1 \lg N}$ or equivalently $\mathbf{m}(\alpha) \geq N^{-k_1}$ where $k_1$ is a positive constant.

$$\begin{aligned}
\Pr(\alpha \in S_c \text{ is not sampled in one random draw}) &\leq (1 - N^{-k_1}) \\
\Pr(\alpha \in S_c \text{ is not sampled in } |S| \text{ random draws}) &\leq (1 - N^{-k_1})^{|S|} \\
\Pr(\text{ some } \alpha \in S_c \text{ is not sampled in } |S| \text{ random draws}) &\leq |S_c|(1 - N^{-k_1})^{|S|} \\
&\leq k_2 N^2 (1 - N^{-k_1})^{|S|} \\
&\qquad \text{since } |S_c| = O(N^2) \\
\Pr(S_c \not\subseteq S) &\leq k_2 N^2 (1 - N^{-k_1})^{|S|}
\end{aligned}$$

We want this probability to be less than $\delta$.

$$\begin{aligned}
k_2 N^2 (1 - N^{-k_1})^{|S|} &\leq \delta \\
k_2 N^2 (e^{-N^{-k_1}})^{|S|} &\leq \delta \text{ since } 1 - x \leq e^{-x} \text{ if } x \geq 0
\end{aligned}$$

$$\ln(k_2) + \ln(N^2) - N^{-k_1}|S| \leq \ln(\delta)$$
$$|S| \geq N^{k_1}\left(\ln\left(\frac{1}{\delta}\right) + \ln(k_2) + \ln(N^2)\right)$$

Thus, $\Pr(S_c \subseteq S) \geq 1 - \delta$. □

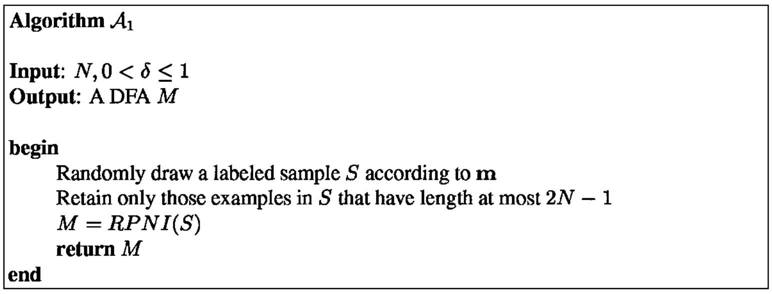
We now prove that the class of DFA with $O(\lg N)$ complexity is polynomially learnable under **m**.

**Theorem 1.** *For all $N$, the class $\mathcal{C}^{\leq N}$ of DFA having at most $N$ states and Kolmogorov complexity $O(\lg N)$ is probably exactly learnable under the simple PAC model.*

**Proof:** Let $A$ be a DFA with at most $N$ states and $K(A) = O(\lg N)$. Let $S_c$ be a characteristic sample of $A$ enumerated as described in Lemma 1 above. Recall that the examples in $S_c$ are simple (i.e., each example has Kolmogorov complexity $O(\lg N)$). Now consider the algorithm $\mathcal{A}_1$ in figure 8 that draws a sample $S$ with the following properties.

1. $S = S^+ \cup S^-$ is a set of positive and negative examples corresponding to the target DFA $A$.
2. The examples in $S$ are drawn at random according to the distribution **m**.
3. $|S|$ is at least $N^{k_1}(\ln(\frac{1}{\delta}) + \ln(k_2) + \ln(N^2))$ where $k_1$ and $k_2$ are positive constants.

Lemma 1 showed that for every DFA $A$ with $K(A) = O(\lg N)$ there exists a characteristic set of simple examples $S_c$ where each example is of length at most $2N-1$. Lemma 2 showed that if a labeled sample $S$ of size at least $N^{k_1}(\ln(\frac{1}{\delta}) + \ln(k_2) + \ln(N^2))$ (where $k_1$ and $k_2$ are positive constants) is randomly drawn according to **m** then with probability greater than $1-\delta$, $S_c \subseteq S$. The RPNI algorithm is guaranteed to return a canonical representation of the target DFA $A$ if the set of examples $S$ provided is a superset of a characteristic set $S_c$. Since the size of $S$ is polynomial in $N$ and $1/\delta$ and the length of each string in $S$ is restricted to $2N-1$, the RPNI algorithm, and thus the algorithm $\mathcal{A}_1$ can be implemented to run in time polynomial in $N$ and $1/\delta$. Thus, with probability greater than $1-\delta$, $\mathcal{A}_1$ is guaranteed to return a canonical representation of the target DFA $A$. This proves the result. □



```
Algorithm $\mathcal{A}_1$

Input: $N, 0 < \delta \leq 1$
Output: A DFA $M$

begin
    Randomly draw a labeled sample $S$ according to m
    Retain only those examples in $S$ that have length at most $2N - 1$
    $M = RPNI(S)$
    return $M$
end
```

*Figure 8.* A probably exact algorithm for learning simple DFA.

## 5. Learning DFA under the PACS model

In Section 4 we proved that the class of $O(\lg N)$ Kolmogorov complexity DFA is learnable under the simple PAC model where the underlying distribution is restricted to the universal distribution **m**. Denis et al. (1996) proposed a model of learning (called the PACS model) where examples are drawn at random according to the universal distribution conditional on the knowledge of the target concept. Under this model, examples with low conditional Kolmogorov complexity given a representation $r$ of the target concept are called simple examples. Specifically, for a concept with representation $r$, the set $S^r_{sim} = \{\alpha \mid K(\alpha \mid r) \leq \mu lg(|r|)\}$ (where $\mu$ is a constant) is the set of simple examples for that concept. Further, $S^r_{sim,rep}$ is used to denote a set of simple and representative examples of $r$. The PACS model restricts the underlying distribution to $\mathbf{m}_r$ (where $\mathbf{m}_r(\alpha) = 2^{-K(\alpha|r)+O(1)}$). Representative examples for the target concept are those that enable the learner to exactly learn the target. As explained earlier, the characteristic set corresponding to a DFA can be treated as a representative set for the DFA. The Occam's Razor theorem proved by Denis et al. (1996) states that if there exists a representative set of simple examples for each concept in a concept class then the concept class is PACS learnable.

We now demonstrate that the class of DFA is efficiently learnable under the PACS model[4]. Lemma 3 proves that for any DFA $A$ with standard encoding $r$ there exists a characteristic set of simple examples $S^r_{sim,rep}$.

**Lemma 3.** *For any $N$ state DFA with standard encoding $r$ ($|r| = O(N \lg N)$), there exists a characteristic set of simple examples (denoted by $S^r_{sim,rep}$) such that each string of this set is of length at most $2N - 1$.*

**Proof:** Given a DFA $A = (Q, \delta, \Sigma, q_0, F)$, it is possible to enumerate a characteristic set of examples $S_c$ for $A$ as described in lemma 1 such that $|S_c| = O(N^2)$ and each example of $S_c$ is of length at most $2N - 1$. Individual strings in $S_c$ can be identified by specifying an index of length $O(\lg N)$. There exists a Turing machine $M$ that implements the above algorithm for constructing the set $S_c$. Given the knowledge of the target concept $r$, $M$ can take as input an index of length $O(\lg N)$ bits and output the corresponding string belonging to $S_c$. Thus, $\forall \alpha \in S_c, K(\alpha \mid r) = O(\lg N) \leq \mu \lg(|r|)$ where $\mu$ is a constant . We define the set $S_c$ to be the characteristic set of simple examples $S^r_{sim,rep}$ for the DFA $A$. This proves the lemma. □

**Lemma 4** (Due to Denis et al. (1996))**.** *Suppose that a sample $S$ is drawn according to $\mathbf{m}_r$. For an integer $l \geq |r|$, and $0 < \delta \leq 1$, if $|S| \geq l^{\mu} (\ln(2) + \ln(l^{\mu}) + \ln(1/\delta))$ then with probability greater than $1 - \delta$, $S^r_{sim} \subseteq S$.*

**Proof:**

*Claim 4.1*: $\forall \alpha \in S^r_{sim},\ \mathbf{m}_r(\alpha) \geq l^{-\mu}$

$$\begin{aligned}\mathbf{m}_r(\alpha) &\geq 2^{-K(\alpha|r)} \\ &\geq 2^{-\mu \lg |r|}\end{aligned}$$

$$
\begin{aligned}
&\geq |r|^{-\mu} \\
&\geq l^{-\mu}
\end{aligned}
$$

*Claim 4.2*: $|S^r_{sim}| \leq 2l^{\mu}$

$$
\begin{aligned}
|S^r_{sim}| &\leq |\{\alpha \in \{0,1\}^* \mid K(\alpha \mid r) \leq \mu \mathrm{lg}(|r|)\}| \\
&\leq |\{\alpha \in \{0,1\}^* \mid K(\alpha \mid r) \leq \mu \mathrm{lg}(l)\}| \\
&\leq |\{\beta \in \{0,1\}^* \mid |\beta| \leq \mu \mathrm{lg}(l)\}| \\
&\leq 2^{\mu \mathrm{lg}(l)+1} \\
&\leq 2l^{\mu}
\end{aligned}
$$

*Claim 4.3*: $|S| \geq l^{\mu}\ (\ln(2) + \ln(l^{\mu}) + \ln(1/\delta))$ then $\Pr(S^r_{sim} \subseteq S) \geq 1 - \delta$

$$
\begin{aligned}
\Pr\left(\alpha \in S^r_{sim} \text{ is not sampled in one random draw}\right) &\leq (1 - l^{-\mu}) \\
&\qquad\qquad\qquad \text{(claim 4.1)} \\
\Pr\left(\alpha \in S^r_{sim} \text{ is not sampled in } |S| \text{ random draws}\right) &\leq (1 - l^{-\mu})^{|S|} \\
\Pr\left(\text{some } \alpha \in S^r_{sim} \text{ is not sampled in } |S| \text{ random draws}\right) &\leq 2l^{\mu}(1 - l^{-\mu})^{|S|} \\
&\qquad\qquad\qquad \text{(claim 4.2)} \\
\Pr\left(S^r_{sim} \not\subseteq S\right) &\leq 2l^{\mu}(1 - l^{-\mu})^{|S|}
\end{aligned}
$$

We would like this probability to be less than $\delta$.

$$
\begin{aligned}
2l^{\mu}(1 - l^{-\mu})^{|S|} &\leq \delta \\
2l^{\mu}(e^{-l^{-\mu}})^{|S|} &\leq \delta, \quad \text{since } 1 - x \leq e^{-x} \text{ if } x \geq 0 \\
\ln(2) + \ln(l^{\mu}) - |S| l^{-\mu} &\leq \ln(\delta) \\
|S| &\geq l^{\mu}\ (\ln(2) + \ln(l^{\mu}) + \ln(1/\delta))
\end{aligned}
$$

Thus, $\Pr(S^r_{sim} \subseteq S) \geq 1 - \delta$ □

**Corollary 1.** *Suppose that a sample $S$ is drawn according to $\mathbf{m}_r$. For an integer $l \geq |r|$, and $0 < \delta \leq 1$, if $|S| \geq l^{\mu}\ (\ln(2) + \ln(l^{\mu}) + \ln(1/\delta))$ then with probability greater than $1 - \delta$, $S^r_{sim,rep} \subseteq S$.*

**Proof:** Follows immediately from Lemma 4 since $S^r_{sim,rep} \subseteq S^r_{sim}$. □

We now prove that the class of DFA is polynomially learnable under $\mathbf{m}_r$.

**Theorem 2.** *For all $N$, the class $\mathcal{C}^{\leq N}$ of DFA whose canonical representations have at most $N$ states is probably exactly learnable under the PACS model.*

**Proof:** Let $A$ be a canonical DFA with at most $N$ states and $r$ be its standard encoding. We define the simple representative sample $S^r_{sim,rep}$ to be the characteristic sample of $A$

```
Algorithm $\mathcal{A}_2$

Input: $N, 0 < \delta \leq 1$
Output: A DFA $M$

begin
    Randomly draw a labeled sample $S$ according to $\mathbf{m_r}$
    Retain only those examples in $S$ that have length at most $2N - 1$
    $M = RPNI(S)$
    return($M$)
end
```

*Figure 9.* A probably exact algorithm for learning DFA.

enumerated as described in Lemma 3. Recall that the length of each example in $S^r_{sim,rep}$ is at most $2N - 1$. Now consider the algorithm $\mathcal{A}_2$ (see figure 9) that draws a sample $S$ with the following properties.

1. $S = S^+ \cup S^-$ is a set of positive and negative examples corresponding to the target DFA $A$.
2. The examples in $S$ are drawn at random according to the distribution $\mathbf{m}_r$.
3. $|S|$ is at least $l^\mu\ (\ln(2) + \ln(l^\mu) + \ln(1/\delta))$.

Lemma 3 showed that for every DFA $A$ there exists a characteristic set of simple examples $S^r_{sim,rep}$. Corollary 1 showed that if a labeled sample $S$ of size $|S|$ is at least $l^\mu\ (\ln(2) + \ln(l^\mu) + \ln(1/\delta))$ is randomly drawn according to $\mathbf{m}_r$ then with probability greater than $1 - \delta$, $S^r_{sim,rep} \subseteq S$. The RPNI algorithm is guaranteed to return a canonical representation of the target DFA $A$ if the set of examples $S$ is a superset of a characteristic set for $A$. Since the size of $S$ is polynomial in $N$ and $1/\delta$ and the length of each string in $S$ is restricted to $2N - 1$, the RPNI algorithm, and thus the algorithm $\mathcal{A}_2$ can be implemented to run in time polynomial in $N$ and $1/\delta$. Thus, with probability greater than $1 - \delta$, $\mathcal{A}_2$ is guaranteed to return a canonical DFA equivalent to the target $A$. This proves that the class $\mathcal{C}^{\leq N}$ of DFA whose canonical representations have at most $N$ states is exactly learnable with probability greater than $1 - \delta$. □

Since the number of states of the target DFA ($N$) might not be known in advance we present a PAC learning algorithm $\mathcal{A}_3$ that iterates over successively larger guesses of $N$. At each step the algorithm draws a random sample according to $\mathbf{m}_r$, applies the RPNI algorithm to construct a DFA, and tests the DFA using a randomly drawn test sample. If the DFA is consistent with the test sample then the algorithm outputs the DFA and halts. Otherwise the algorithm continues with the next guess for $N$. This technique of estimating the unknown number of states is called the *doubling technique* (Angluin, 1987).

**Theorem 3.** *The concept class $\mathcal{C}$ of DFA is learnable in polynomial time under the PACS model.*

**Proof:** Figure 10 shows a PAC learning algorithm for DFA.

```
Algorithm $\mathcal{A}_3$

Input: $\epsilon, \delta$
Output: A DFA $M$

begin
    1)  $i = 1, EX = \phi, p(0, 1/\delta) = 0$
    2)  repeat
            Draw $p(i, 1/\delta) - p(i-1, 1/\delta)$ examples according to $\mathbf{m}_r$
            Add the examples just drawn to the set $EX$
            Let $S$ be the subset of examples in $EX$ of length at most $2i - 1$
            $M = RPNI(S)$
            Draw $q(i, 1/\epsilon, 1/\delta)$ examples according to $\mathbf{m}_r$ and call this set $T$
            if $consistent(M, T)$
            then Output $M$ and halt
            else $i = i * 2$
            end if
        until eternity
end
```

*Figure 10.* A PAC algorithm for learning DFA.

In algorithm $\mathcal{A}_3$ the polynomial $p$ is defined such that a sample $S$ of size $p(N, \frac{1}{\delta})$ contains the characteristic set of simple examples $S^r_{sim,rep}$ with probability greater than $1 - \delta$. Corollary 1 gives us a bound on the size of $p(N, \frac{1}{\delta})$ that would satisfy this constraint. The polynomial $q$ is defined as $q(i, \frac{1}{\epsilon}, \frac{1}{\delta}) = \frac{1}{\epsilon}[2\ln(i+1) + \ln(\frac{1}{\delta})]$.

Consider the execution of the algorithm $\mathcal{A}_3$. At any step $i$ where $i \geq N$, the set $S$ will include the characteristic set of simple examples $S^r_{sim,rep}$ with probability greater than $1 - \delta$ (as proved in Lemma 4 and Corollary 1). In this case the RPNI algorithm will return a DFA $M$ that is equivalent to the target $A$ and hence $M$ will be consistent with the test sample $T$. Thus, with probability at least $1 - \delta$, the algorithm will correctly output the target DFA and halt.

Consider the probability that at some step $i$ the algorithm returns a DFA $M$ with an error greater than $\epsilon$ and halts.

$$\begin{aligned}
\Pr(M \text{ and } A \text{ are consistent on some } \alpha) &\leq 1 - \epsilon \\
\Pr(M \text{ and } A \text{ are consistent on all } \alpha \in T) &\leq (1-\epsilon)^{|T|} \\
&\leq (1-\epsilon)^{\frac{1}{\epsilon}[2\ln(i+1)+\ln(\frac{1}{\delta})]} \\
&\leq e^{-[2\ln(i+1)+\ln(\frac{1}{\delta})]} \\
&\quad \text{since } 1 - x \leq e^{-x} \text{ if } x \geq 0 \\
&\leq \frac{\delta}{(i+1)^2}
\end{aligned}$$

The probability that the algorithm returns a DFA with error greater than $\epsilon$ at some step $i$ and halts is less than $\sum_{i=1}^{\infty} \frac{\delta}{(i+1)^2}$ which is in turn strictly less than $\delta$. Thus, we have shown that with probability greater than $1 - \delta$ the algorithm returns a DFA with error at most $\epsilon$. Further, the run time of the algorithm is polynomial in $N$, $|\Sigma|$, $\frac{1}{\epsilon}$, $\frac{1}{\delta}$, and $m$ (where $m$ is the length of the longest test example seen by the algorithm). Thus, the class of DFA is efficiently PAC learnable under the PACS model. □

## 6. Relationship of the PACS model to other learning models

In this section we study the relationship of the PACS model to learning models such as Gold's model (1978) of *polynomial identifiability from characteristic samples*, Goldman and Mathias' *polynomial teachability* model (1993), and the model of learning from *example based queries* (Angluin, 1988). We explain how the PACS learning model naturally extends these models to a probabilistic framework. In the following discussion we will let $\mathcal{X}$ be the instance space, $\mathcal{C}$ be the concept class, and $\mathcal{R}$ be the set of representations of the concepts in $\mathcal{C}$.

### *6.1. Polynomial identifiability from characteristic samples*

Gold's model for polynomial identifiability of concept classes from characteristic samples is based on the availability of a polynomial sized characteristic sample for any concept in the concept class and an algorithm which when given a superset of a characteristic set is guaranteed to return, in polynomial time, a representation of the target concept.

*Definition 3* ([due to de la Higuera (1996)]). $\mathcal{C}$ is polynomially identifiable from characteristic samples iff there exist two polynomials $p_1()$ and $p_2()$ and an algorithm $\mathcal{A}$ such that

1. Given any sample $S = S^+ \cup S^-$ of labeled examples, $\mathcal{A}$ returns in time $p_1(\|S^+\| + \|S^-\|)$ a representation $r \in \mathcal{R}$ of a concept $c \in \mathcal{C}$ such that $c$ is consistent with $S$.
2. For every concept $c \in \mathcal{C}$ with corresponding representation $r \in \mathcal{R}$ there exists a characteristic sample $S_c = S_c^+ \cup S_c^-$ such that $\|S_c^+\| + \|S_c^-\| = p_2(|r|)$ and if $\mathcal{A}$ is provided with a sample $S = S^+ \cup S^-$ where $S_c^+ \subseteq S^+$ and $S_c^- \subseteq S^-$ then $\mathcal{A}$ returns a representation $r'$ of a concept $c'$ that is equivalent to $c$.

Using the above definition Gold's result can be restated as follows.

**Theorem 4** (due to Gold (1978)). *The class of DFA is polynomially identifiable from characteristic samples.*

The problem of identifying a minimum state DFA that is consistent with an arbitrary labeled sample $S = S^+ \cup S^-$ is known to be NP-complete (Gold, 1978). This result does not contradict the one in Theorem 4 because a characteristic set is not any arbitrary set of examples but a special set that enables the learning algorithm to correctly infer the target concept in polynomial time (see the RPNI algorithm in Section 3).

### *6.2. Polynomial teachability of concept classes*

Goldman and Mathias (1993) developed a teaching model for efficient learning of target concepts. Their model takes into account the quantity of information that a good teacher must provide to the learner. An additional player called the *adversary* is introduced in this model to ensure that there is no collusion whereby the teacher gives the learner an encoding of the target concept. A typical teaching session proceeds as follows:

1. The adversary selects a target concept and gives it to the teacher.
2. The teacher computes a set of examples called the *teaching set*.
3. The adversary adds correctly labeled examples to the teaching set with the goal of complicating the learner's task.
4. The learner computes a hypothesis from the augmented teaching set.

Under this model, a concept class for which the computations of both the teacher and the learner takes polynomial time and the learner always learns the target concept is called *polynomially T/L teachable*. Without the restrictive assumption that the teacher's computations be performed in polynomial time, the concept class is said to be *semi-polynomially T/L teachable*. When this model is adapted to the framework of learning DFA the length of the examples seen by the learner must be included as a parameter in the model. In the context of learning DFA the number of examples is infinite (it includes the entire set $\Sigma^*$) and further the lengths of these examples grow unboundedly. A scenario in which the teacher constructs a very small teaching set whose examples are unreasonably long is clearly undesirable and must be avoided. This is explained more formally in the following definition.

*Definition 4* (due to de la Higuera (1996)). A concept class $\mathcal{C}$ is semi-polynomially T/L teachable iff there exist polynomials $p_1()$, $p_2()$, and $p_3()$, a teacher $T$, and a learner $L$, such that for any adversary $ADV$ and any concept $c$ with representation $r$ that is selected by $ADV$, after the following teaching session the learner returns the representation $r'$ of a concept $c'$ that is equivalent to $c$.

1. $ADV$ gives $r$ to $T$.
2. $T$ computes a teaching set $S$ of size at most $p_1(|r|)$ such that each example in the teaching set has length at most $p_2(|r|)$.
3. $ADV$ adds correctly labeled examples to this set, with the goal of complicating the learner's task.
4. The learner uses the augmented set $S$ to compute a hypothesis $r'$ in time $p_3(\|S\|)$.

Note that from Gold's result (Theorem 4) it follows that DFA are semi-polynomially T/L teachable. Further, we demonstrated in lemma 1 that for any DFA there exists a procedure to enumerate a characteristic set corresponding to that DFA. This procedure can be implemented in polynomial time thereby proving a stronger result that DFA are polynomially T/L teachable. It was proved that the model for polynomial identification from characteristic samples and the model for polynomial teachability are equivalent to each other (i.e., by identifying the characteristic set with the teaching sample it was shown that a concept class is polynomially identifiable from characteristic samples *iff* it is semi-polynomially T/L teachable) (de la Higuera, 1996).

**Lemma 5.** *Let $c \in \mathcal{C}$ be a concept with corresponding representation $r \in \mathcal{R}$. If there exists a characteristic sample $S_c$ for $c$ and a polynomial $p_1()$ such that $S_c$ can be computed from $r$ and $\|S_c\| = p_1(|r|)$ then each example in $S_c$ is simple in the sense that $\forall \alpha \in S_c,\ K(\alpha \mid r) \leq \mu \lg(|r|)$ where $\mu$ is a constant.*

**Proof:** Fix an ordering of the elements of $S_c$ and define an index to identify the individual elements. Since $\|S_c\| = p_1(|r|)$ an index that is $O(\lg(|r|))$ bits long is sufficient to uniquely identify each element of $S_c$[5]. Since $S_c$ can be computed from $r$ we can construct a Turing machine that given $r$ reads as input an index of length $O(\lg(|r|))$ and outputs the corresponding string of $S_c$. Thus, $\forall \alpha \in S_c,\ K(\alpha \,|\, r) \leq \mu \lg(|r|)$ where $\mu$ is a constant independent of $\alpha$. □

Let us designate the characteristic set of simple examples $S_c$ identified above to be the set of simple representative examples $S^r_{sim,rep}$ for the concept $c$ represented by $r$. Lemma 4 and Corollary 1 together show that for an integer $l \geq |r|$ and $0 < \delta < 1$ if a sufficiently large sample $S$ (of size polynomial in $l$ and $1/\delta$) is drawn at random according to $\mathbf{m}_r$ then with probability greater than $1 - \delta$, $S^r_{sim,rep} \subseteq S$.

**Theorem 5.** *Any concept class that is polynomially identifiable from characteristic samples or equivalently semi-polynomially T/L teachable is probably exactly learnable under the PACS model.*

**Proof:** The proof follows directly from the results of Lemma 5, Lemma 4, and Corollary 1. □

### *6.3. Learning from example based queries*

A variety of concept classes are known to be learnable in deterministic polynomial time when the learner is allowed access to a teacher (or an oracle) that answers *example based queries* (Angluin, 1988). Example based queries include *equivalence*, *membership, subset, superset, disjointedness, exhaustive, justifying assignment,* and *partial equivalence* queries.

*Definition 5* ((due to Goldman and Mathias (1993)). An example based query is any query of the form

$$\forall (x_1, x_2, \ldots, x_k) \in \mathcal{X}^k \text{ does } \phi_r(x_1, x_2, \ldots, x_k) = 1?$$

where $r$ is the target concept and $k$ is a constant.

$\phi$ may use the instances $(x_1, \ldots, x_k)$ to compute additional instances on which to perform membership queries. The teacher's response to example based queries is either *yes* or a counter example consisting of $(x_1, x_2, \ldots, x_k) \in \mathcal{X}^k$ (along with the correct classification corresponding to each of the $x_i$'s) for which $\phi_r(x_1, x_2, \ldots, x_k) = 0$ and the labeled examples for which membership queries were made in order to evaluate $\phi_r$.

**Theorem 6** (due to Goldman and Mathias (1993))**.** *Any concept class that is learnable in deterministic polynomial time using example based queries is semi-polynomially T/L teachable.*

The above theorem enables us to connect learning from example based queries to PACS learning as follows.

**Theorem 7.** *Any concept class that is learnable in deterministic polynomial time using example based queries is probably exactly learnable under the PACS model.*

**Proof:** Follows directly from Theorems 5 and 6. □

Recently Castro and Guijarro (1998) have independently shown that any concept class that is learnable using membership and equivalence queries is also learnable under the PACS model. Further, they have intuitively demonstrated how this result can be extended to all example based queries. Theorem 7 above is an alternate proof of the relationship between query learning and PACS learning.

## 7. Collusion and PACS learning

Learning models that involve interaction between a knowledgeable teacher and a learner can admit *collusion* wherein the teacher directly passes information about the representation of the target function as part of the training set (Jackson & Tomkins, 1992; Goldman & Mathias, 1996). Consider for simplicity that the instance space is $\{0, 1\}^n$ (i.e., the training examples are $n$ bits long). The teacher and learner can *a-priori* agree on some suitable binary encoding of concepts. The teacher can then break the representation of the target concept $r$ in to groups of $n$ bits and use the training set to pass these groups as appropriately labeled examples to the learner. The learner could quickly discover the target concept without even considering the labels of the training examples. The *teaching model* due to Jackson and Tomkins (1992) prevents this coding of the target concept by requiring that the learner must still succeed if the teacher is replaced by an adversary (who does not code the target concept as the teacher above). Further, they argue that in their model the learner can stop only when it is convinced that there is only one concept consistent with the information received from the teacher i.e., the teacher does not tell the learner when to stop. Otherwise learning would be trivialized in that the teacher passes groups of $n$ bits to the learner (as training examples) and when sufficient number of bits have been passed to the learner so as to reconstruct the representation $r$ of the target concept, the teacher tells the learner to stop. Goldman and Mathias' work (1996) on *polynomial teachability* shows that an *adversary* whose task is to embed the training set (also called *teaching set*) provided by the teacher into a larger set of correctly labeled examples is sufficient to prevent the type of collusion discussed above. An interesting quirk of the PACS learning model is the fact that the standard encoding of the target concept $r$ is itself a simple example because by definition $K(r \mid r)$ is low. Thus, $r$ has a high probability under the universal distribution $m_r$. The PACS learning scenario wherein examples are drawn at random according to the universal distribution $\mathbf{m}_r$ is similar to the teaching framework in the above teaching models where a knowledgeable teacher draws a helpful set of examples that would enable the learner to learn the target concept efficiently. The representation of the target concept $r$ (that is drawn with high probability according to $\mathbf{m}_r$) could be broken into groups of $n$ bits and passed to the learner as appropriately labeled training examples. This form of collusion wherein the teacher directly passes an encoding of the target concept would not succeed in the presence of an adversary who embeds the training set provided by the teacher into a larger set of correctly labeled examples because in this case the learner who is operating by accumulating bits would not know precisely when it has sufficient information to reconstruct the target.

Another (perhaps more subtle) form of collusion comes to light when considering the problem of learning DFA under the PACS model. In DFA learning individual examples of the teaching set can be of different lengths. As discussed above, the canonical encoding of the target DFA $A$ (a string $r$ of length $O(N \lg N)$) appears with high probability when the examples are drawn at random according to $\mathbf{m}_r$. The fact that DFA learning does not require fixed length examples means that $r$ would itself appear with high probability as part of a polynomial sized training set. Of course, the learner cannot directly identify which example string in the training set represents the target DFA. However, assuming that the teacher and the learner have *apriori* agreed on an encoding scheme for the DFA, the learner can decode each labeled example and determine whether it represents a valid DFA. For each example that represents a valid DFA, the learner can test whether the DFA obtained by decoding the example is consistent with the training set and output (say) the first DFA in lexicographic order that is consistent with the training set. With high probability the learner would output the target DFA. This constitutes a PACS algorithm for learning DFA that is computationally more efficient than the RPNI based PACS algorithm presented in this paper. In this form of collusion the learner is provided with an encoding of the target concept but must perform some additional computation to decode the examples and check that the extracted DFA is consistent with the training set.

It is clear that the PACS learning framework admits collusion. Any learnability results within models that admit collusion can be criticized on the grounds that the learning algorithm might be collusive. One method of avoiding collusive learning is to tighten the learning framework suitably. Collusion cannot take place if the representation of the target concept cannot be directly encoded as part of the training set or if the learner cannot efficiently decode the training examples and identify the one that is consistent with the training set. In the event that the learning framework cannot be suitably tightened to avoid collusion, one might provide a learning algorithm that does not rely on collusion between the teacher and the learner. The RPNI based algorithm for learning DFA under the PACS model presented in this paper is an example of a non-collusive algorithm in a learning framework that admits collusion. Obtaining a general answer to the question of collusion in learning would require the development of much more precise definitions of collusion and collusion-free learning than are currently available. A detailed exploration of these issues is beyond the scope of this paper.

## 8. Discussion

The problem of exact learning of the target DFA from an arbitrary set of labeled examples and the problem of approximating the target DFA from labeled examples under Valiant's PAC learning framework are both known to be hard problems. Thus, the question as to whether DFA are efficiently learnable under some restricted yet fairly general and practically useful classes of distributions is clearly of interest. In this paper, we have provided a framework for efficient PAC learning of DFA from simple examples.

We have demonstrated that the class of logarithmic Kolmogorov complexity DFA is polynomially learnable under the universal distribution **m** (the simple PAC learning model) and the entire class of DFA is shown to be learnable under the universal distribution $\mathbf{m}_r$ (the PACS learning model). When an upper bound on the number of states of the target DFA is unknown, the algorithm for learning DFA under $\mathbf{m}_r$ can be used iteratively to efficiently PAC learn the concept class of DFAs for any desired error and confidence parameters[6]. These results have an interesting implication on the framework for incremental learning of the target DFA. In the RPNI2 incremental algorithm for learning DFA, the learner maintains a hypothesis that is consistent with all labeled examples seen thus far and modifies it whenever a new inconsistent example is observed (Dupont, 1996). The convergence of this algorithm relies on the fact that sooner or later, the set of labeled examples seen by the learner will include a characteristic set. If in fact the stream of examples provided to the learner is drawn according to a simple distribution, our results show that in an incremental setting the characteristic set would be made available relatively early (during learning) with a sufficiently high probability and hence the algorithm will converge quickly to the desired target. Finally, we have shown the applicability of the PACS learning model in a more general setting by proving that all concept classes that are polynomially identifiable from characteristic samples according to Gold's model, semi-polynomially T/L teachable according to Goldman and Mathias' model, and learnable in deterministic polynomial time from example based queries are also probably exactly learnable under the PACS model.

Li and Vitányi (1991) have shown that the class of *simple* k-reversible DFA is learnable under the simple PAC model. A k-reversible DFA is *simple* if each state of the DFA lies on a path (from the initial state to an accepting state) of Kolmogorov complexity $O(\lg N)$. They have shown that the class of *simple* k-reversible DFA includes k-reversible DFA whose canonical representations have Kolmogorov complexity $O(\lg N)$ and also some k-reversible DFA whose canonical representations have a higher Kolmogorov complexity, for example, $O(\lg^2 N)$. We have shown that the class of logarithmic Kolmogorov complexity DFA is learnable under the simple PAC model. It is of interest to explore whether classes of DFA of higher Kolmogorov complexity (such as $O(\lg^k N)$ where $k > 1$) are efficiently learnable under the simple PAC model.

The class of simple distributions includes a large variety of probability distributions (including all computable distributions). It has been shown that a concept class is efficiently learnable under the universal distribution if and only if it is efficiently learnable under each simple distribution provided that sampling is done according to the universal distribution (Li & Vitányi, 1991). This raises the possibility of using sampling under the universal distribution to learn under all computable distributions. However, the universal distribution is not computable. Whether one can instead get by with a polynomially computable approximation of the universal distribution remains an open question. It is known that the universal distribution for the class of polynomially-time bounded simple distributions is computable in exponential time (Li & Vitányi, 1991). This opens up a number of interesting possibilities for learning under simple distributions. Denis and Gilleron (1997) have proposed a model of learning under *helpful distributions*. A helpful distribution is one in which examples belonging to the characteristic set for the concept (if there exists one) are assigned non-zero probability. A systematic characterization of the class of helpful distributions would perhaps give us a more practical framework for learning from simple examples.

A related question of interest has to do with the nature of environments that can be modeled by simple distributions. In particular, if Kolmogorov complexity is an appropriate measure of the intrinsic complexity of objects in nature and if nature (or the teacher) has a propensity for simplicity, then it stands to reason that the examples presented to the learner by the environment are likely to be generated by a simple distribution. Against this background, empirical evaluation of the performance of the proposed algorithms using examples that come from natural domains is clearly of interest.

## Acknowledgments

The authors wish to thank Jack Lutz for introducing them to Kolmogorov complexity, Giora Slutzki for several helpful discussions on automata theory, Pierre Dupont for a careful review of an earlier draft of this paper and several helpful suggestions, Colin de la Higuera and François Denis for discussions which helped clarify some of the issues related to collusion, and several anonymous referees for their insightful comments that have helped us to improve the paper significantly. Vasant Honavar is grateful to the National Science Foundation for the research grants IRI-9409580 and IRI-9643299 that partially supported this work.

## Notes

1. Define $\langle x, y\rangle = bd(x)01y$ where $bd$ is the bit doubling function defined as $bd(0) = 00$, $bd(1) = 11$, and $bd(ax) = aabd(x), a \in \{0, 1\}$.
2. The results of this section were presented earlier in Parekh and Honavar (1999).
3. This enumeration strategy applies to any DFA and is not restricted to simple DFA alone.
4. The results of this section were presented earlier in Parekh and Honavar (1997).
5. Note that if the sum of the lengths of the examples belonging to a set is $k$ then the number of examples in that set is at most $k + 1$.
6. Recently it has been shown that if a concept class is learnable under the PACS model using an algorithm that satisfies certain properties then logarithmic Kolmogorov complexity concepts of that concept class are learnable under the simple PAC learning model (Castro & Guijarro, 1998).

## References

Angluin, D. (1981). A note on the number of queries needed to identify regular languages. *Information and Control, 51*, 76–87.

Angluin, D. (1987). Learning regular sets from queries and counterexamples. *Information and Computation, 75*, 87–106.

Angluin, D. (1988). Queries and concept learning. *Machine Learning, 2:4*, 319–342.

Castro, J., & Guijarro, D. (1998). Query, pacs and simple-pac learning. Technical Report LSI-98-2-R, Universitat Polytéctica de Catalunya, Spain.

Chomsky, N. (1956). Three models for the description of language. *PGIT, 2:3*, 113–124.

Denis, F., D'Halluin, C., & Gilleron, R. (1996). Pac learning with simple examples. *STACS'96—Proceedings of the* $13^{th}$ *Annual Symposium on the Theoretical Aspects of Computer Science* (pp. 231–242).

Denis, F., & Gilleron, R. (1997). Pac learning under helpful distributions. In *Proceedings of the Eighth International Workshop on Algorithmic Learning Theory (ALT'97), Lecture Notes in Artificial Intelligence 1316* (pp. 132–145), Sendai, Japan.

Dupont, P. (1996). Incremental regular inference. In L. Miclet, & C. Higuera, (Eds.), *Proceedings of the Third ICGI-96, Lecture Notes in Artificial Intelligence 1147* (pp. 222–237), Montpellier, France, Springer.

Dupont, P. (1996). *Utilisation et apprentissage de modèles de language pour la reconnaissance de la parole continue*. PhD thesis, Ecole Normale Supérieure des Télécommunications, Paris, France.

Dupont, P., Miclet, L., & Vidal, E. (1994). What is the search space of the regular inference? In *Proceedings of the Second International Colloquium on Grammatical Inference (ICGI'94)* (pp. 25–37). Alicante, Spain.

Gold, E. (1978). Complexity of automaton identification from given data. *Information and Control, 37:3*, 302–320.

Goldman, S., & Mathias, H. (1993). Teaching a smarter learner. In *Proceedings of the Workshop on Computational Learning Theory (COLT'93)* (pp. 67–76). ACM Press.

Goldman, S., & Mathias, H (1996). Teaching a smarter learner. *Journal of Computer and System Sciences, 52*, 255–267.

Colin de la Higuera (1996). Characteristic sets for polynomial grammatical inference. In L. Miclet, & C. Higuera, (Eds.), *Proceedings of the Third ICGI-96, Lecture Notes in Artificial Intelligence 1147* (pp. 59–71). Montpellier, France, Springer.

Hopcroft, J., & Ullman, J. (1979). *Introduction to automata theory, languages, and computation*. Reading, MA: Addison Wesley.

Jackson, J., & Tomkins, A. (1992). A computational model of teaching. In *Proceedings of the Workshop on Computational Learning Theory (COLT'92)* (pp. 319–326). ACM Press.

Kearns, M., & Valiant, L. G. (1989). Cryptographic limitations on learning boolean formulae and finite automata. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing* (pp. 433–444). New York: ACM.

Lang, K. (1992). Random DFAs can be approximately learned from sparse uniform sample. In *Proceedings of the* $5th$ *ACM workshop on Computational Learning Theory* (pp. 45–52).

Li, M., & Vitányi, P. (1991). Learning simple concepts under simple distributions. *SIAM Journal of Computing, 20:5*, 911–935.

Li, M., & Vitányi, P. (1997). *An introduction to Kolmogorov complexity and its applications*, (2nd ed.) New York: Springer Verlag.

Oncina, J., & Garcia, P. (1992). Inferring regular languages in polynomial update time. In N. Pérez et al. (eds.), *Pattern recognition and image analysis* (pp. 49–61). Singapore: World Scientific.

Pao, T., & Carr, J. (1978). A solution of the syntactic induction-inference problem for regular languages. *Computer Languages, 3*, 53–64.

Parekh, R., & Honavar, V. (1993). Efficient learning of regular languages using teacher supplied positive examples and learner generated queries. *In Proceedings of the Fifth UNB Conference on AI* (pp. 195–203). Fredricton, Canada.

Parekh, R., & Honavar, V. (1997). Learning DFA from simple examples. In *Proceedings of the Eighth International Workshop on Algorithmic Learning Theory (ALT'97), Lecture Notes in Artificial Intelligence 1316* (pp. 116–131). Sendai, Japan, Springer. Also presented at the *Workshop on Grammar Inference, Automata Induction, and Language Acquisition* (ICML'97), Nashville, TN, July 12, 1997.

Parekh, R & Honavar, V. (1999). Simple DFA are polynomially probably exactly learnable from simple examples. In *Proceedings of the Sixteenth International Conference on Machine Learning (ICML'99)* (pp. 298–306). Bled, Slovenia.

Pitt, L. (1989). Inductive inference, DFAs and computational complexity. In *Analogical and Inductive Inference, Lecture Notes in Artificial Intelligence, 397* (pp. 18–44). Springer-Verlag.

Pitt, L., & Warmuth, M. K. (1988). Reductions among prediction problems: on the difficulty of predicting automata. In *Proceedings of the 3rd IEEE Conference on Structure in Complexity Theory* (pp. 60–69).

Pitt, L., & Warmuth, M. K. (1989). The minimum consistency DFA problem cannot be approximated within any polynomial. In *Proceedings of the 21st ACM Symposium on the Theory of Computing* (pp. 421–432). ACM.

Rivest, R. L. & Schapire, R. E. (1993). Inference of finite automata using homing sequences. *Information and Computation, 103:2*, 299–347.

Trakhtenbrot, B., & Barzdin, Ya. (1973). *Finite Automata: Behavior and Synthesis*. Amsterdam, North Holland.

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

Received December 12, 1998
Revised February 22, 2000
Accepted March 30, 2000
Final Manuscript April 24, 2000