Inductive Inference, DFAs, and Computational Complexity

Leonard Pitt*
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, Illinois

Abstract

This paper surveys recent results concerning the inference of deterministic finite automata (DFAs). The results discussed determine the extent to which DFAs can be feasibly inferred, and highlight a number of interesting approaches in computational learning theory.

1 Introduction

Beginning with Gold's seminal paper on the identification of formal languages from examples [26], there has been extensive research into the problem of inferring DFAs (deterministic finite automata) from examples supplied to an inference algorithm. The problem of identifying an unknown DFA from examples has been central to the study of inductive inference for several reasons: Since a main focus of research in inductive inference involves determining those types of rules that can be inferred from examples, much attention has been given to those rules computable by the simplest types of formal devices. Similarly, the associated class of languages that are accepted by DFAs (i.e., the class of regular languages) occupies the "ground floor" of the Chomsky hierarchy, and research into the inferability of regular languages provides an excellent entry point into the investigation of the inferability of formal languages in general. Further, approaches and solutions to subproblems and problems related to DFA inference have often produced techniques applicable to other inference domains. In short, the study of the inferability of DFAs is an excellent means for studying a number of general aspects of inductive inference.

This paper surveys a number of very recent results that (for the most part) determine the extent to which DFAs can be inferred in a computationally efficient manner. Many of the results discussed involve interesting new computational models of inference as well as intriguing connections with problems in combinatorial optimization and cryptography.

The rest of this paper is organized as follows. Section 2 contains basic definitions that will be used throughout the paper. Section 3 begins by reviewing the definitions of identification in the limit due to Gold [26], and discusses the problem of augmenting the definition so as to incorporate a notion of computational efficiency. Various definitions are presented, and results of Angluin [4, 7] are summarized, showing that there is no polynomial time algorithm for exactly identifying the class of DFAs from examples alone. In Section 4 the "distribution-free" inference model of Valiant [58], and generalizations of this model are considered. The results of Pitt and

*Supported in part by NSF grant IRI-8809570, and by the Department of Computer Science, University of Illinois at Urbana-Champaign.

Warmuth [43] and Kearns and Valiant [34] are reviewed, which show, based on various complexity-theoretic assumptions, that despite these less demanding inference criteria, the problem of inferring DFAs from examples alone remains intractable. The related optimization problem of finding an approximately small DFA that is consistent with given data is also considered, and strongly negative results are reviewed [34, 42]. Section 5 considers inference algorithms that are provided with, or have the ability to obtain, additional information about the DFA to be inferred, as well as the problem of inferring certain restricted classes of DFAs. We describe a number of results (Angluin [6, 9], Rivest and Schapire [50, 51, 52], Ibarra and Jiang [33], Helmbold, Sloan, and Warmuth [30, 31], and Abe [2]) showing that the inference problem is efficiently solvable in these more favorable settings. Finally, Section 6 provides a brief summary, and suggests some problems for further research.

2 Definitions

P is the class of languages accepted in polynomial time by deterministic Turing machines, NP is the class of languages accepted in polynomial time by nondeterministic Turing machines, and RP is the class of languages accepted by probabilistic polynomial time Turing machines with one-sided error bounded away from 0, as defined in [24]. It is known that $P \subseteq RP \subseteq NP$, and it is not known if any of the inclusions are proper. Most of the inference problems considered in this paper would be trivial if $RP = NP$. The complexity theoretic assumption that $RP \neq NP$ is widely used; if $RP = NP$, there would be "efficient" (randomized) algorithms for solving problems for which there is significant evidence of intractability. See [23, 24, 59] for further justification and background.

An *alphabet* Σ is a finite set of symbols. The set Σ^* consists of all finite length sequences formed by concatenating zero or more elements of Σ. Elements of Σ^* are called *strings*. The unique string of length 0 is denoted λ. A *language* over alphabet Σ is any subset L of Σ^*. If L_1 and L_2 are languages over the same alphabet Σ, then $L_1 \oplus L_2$ is the *symmetric difference* of L_1 and L_2, defined by $L_1 \oplus L_2 = (L_1 - L_2) \cup (L_2 - L_1)$.

2.1 Deterministic Finite Automata

All of the background material regarding DFAs that is assumed in this paper may be found in [32]. Here we briefly review the formal definitions and notation that will be used.

A *deterministic finite automaton* (DFA), also called a *finite state machine*, is a 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$, where Q is a finite set of *states*, Σ is an alphabet of *input symbols*, δ is the *transition function* $\delta : Q \times \Sigma \to Q$, where $\delta(q, a) = q'$ indicates that if M is in state $q \in Q$, and the next input symbol to be read is a, then the next state of M is q'. $q_0 \in Q$ is a unique *initial state*, and $F \subseteq Q$ is a set of *final*, or *accepting* states. The transition function δ is extended inductively in the usual way to domain $Q \times \Sigma^*$ by $\delta(q, \lambda) = q$, and $\delta(q, ua) = \delta(\delta(q, u), a)$ for any string $u \in \Sigma^*$ and $a \in \Sigma$.

The DFA M *accepts* a string w iff $\delta(q_0, w) \in F$, i.e., iff the sequence of state transitions determined by the input string w results in a final state of M. The language of M, denoted $L(M)$, is the set of strings that M accepts. A language L is *regular* iff there exists a DFA M such that $L = L(M)$. Thus the class of regular languages is the class of languages accepted by DFAs.

Two DFAs M_1 and M_2 are *equivalent* iff $L(M_1) = L(M_2)$. For any M, there is a unique equivalent DFA M' (up to an isomorphism defined by renaming of states) such that the number of states of M' is minimum over all DFAs equivalent to M. Thus for any regular language L, there is a unique (up to isomorphism) *canonical* DFA that accepts L.

The *size* of a DFA is the length of its representation in some natural encoding. Very roughly, most size measures will be proportional to the number of states times the size of the alphabet, since each transition must be specified. A more crude measure of size is simply the number of states,

which is polynomially related to most reasonable encodings. A related measure of complexity, based on the *diversity* of the DFA, is discussed in Section 5.2.

A *nondeterministic* finite automaton (NFA) is defined similarly to a DFA, except that δ is a function with domain $Q \times \Sigma^*$, and range 2^Q. The reader may wish to consult [32] for further details. The class of languages accepted by NFAs also corresponds exactly to the class of regular languages. The regular languages are also representable by the class of *regular grammars*, or by *regular expressions.*

2.2 Classes of Representations

For any class of languages to be inferred, there must be an associated collection of names, or *class of representations*; otherwise an inference algorithm would have no means for indicating its conjecture as to the language being inferred. We are mainly interested in polynomial-time (feasible) inference. Consequently, we wish to associate a measure of complexity with any language to be inferred, and allow an inference algorithm a number of computation steps that depends polynomially on this complexity measure (and possibly other parameters). Rather than assign one measure of complexity to each *language*, it is natural to let the measure of complexity depend on the choice of *representation* of the language in question. A reasonable measure of the complexity is the length of this representation. By assuming a particular class of representations (for example, the DFAs represent the regular languages) we conceivably obtain different results than might be obtained by considering a different class of representations (e.g., NFAs), since the lengths of the descriptions of the same language under different choices of representation schemes might not be polynomially related (e.g., for some languages, NFAs are exponentially more concise than DFAs). We define these notions more precisely.

Definition 1 *A* class of representations *for languages over alphabet Σ is a language R (over some alphabet Γ) such that*

1. *R is recursive.*
2. *Each $r \in R$ denotes a language $L(r) \subseteq \Sigma^*$.*
3. *There exists an algorithm that on input of any string $w \in \Sigma^*$ and representation $r \in R$, outputs "yes" iff $w \in L(r)$, and outputs "no" iff $w \notin L(r)$.*

If there is a polynomial-time algorithm witnessing condition 1 above, and if there exists a polynomial-time algorithm that satisfies condition 3, then we say that the class of representations R is polynomially reasonable.

Since the only classes of representations that we consider are polynomially reasonable, we omit the phrase "polynomially reasonable" except in those cases where emphasis is desired.

If R is a class of representations, then the *size* of an element $r \in R$ is simply the length of the string r. A *(labeled) example* of a representation r is a pair (w, b) where $w \in \Sigma^*$, and $b =$ "+" if $w \in L(r)$, and $b =$ "−" if $w \notin L(r)$. If (w, b) is an example of some $r \in R$, then $r' \in R$ is *consistent* with (w, b) if $w \in L(r') \Leftrightarrow b =$ "+". Examples are also called *labeled strings.*

2.3 Inference Problems

Following Gold [26] (see also Angluin and Smith [12]), in order to properly define an inference problem, we must carefully describe:

1. The (input) class of representations that defines the class of languages from which examples are taken. This class is called the *target class*. The target class implicitly specifies an associated measure of *size* of a given representation which acts as a parameter of complexity for the inference algorithm. Throughout the paper, unless otherwise specified, the class to be inferred is the class of DFAs. Size measures for DFAS were discussed briefly in Section 2.1. The particular DFA to be inferred during a given run of an inference algorithm is referred to as the *target* DFA.

2. The hypothesis space (i.e., the class of representations) used by the inference algorithm, which is not necessarily the same as the input class. Unless otherwise specified, the output(s) of inference algorithms will be assumed to be DFAs. However, we will also consider algorithms that may output other representations of the regular sets, as well as algorithms that may output elements from an arbitrary polynomially reasonable hypothesis space.

3. The manner in which examples are presented to the inference algorithm. Possibilities include (but are not limited to): (a) Examples are presented in some particular order (e.g., lexicographic). (b) Examples are presented in an arbitrary order, as long as each string of Σ^* appears in the sequence of examples at least once. (c) Examples are randomly generated according to some probability distribution on Σ^* which may or may not be known to the inference algorithm.

4. The class of allowable inference algorithms. In this paper we mainly consider deterministic and randomized inference algorithms that obey various bounds on their running time.

5. The criterion of successful inference. Sections 3 and 4 discuss a number of criteria for polynomial time inference.

6. Other means (if any) by which the inference algorithm may obtain additional information about the target DFA. For example, we consider algorithms that make various types of queries regarding the language accepted by the target DFA, and algorithms that are given some initial additional information about the target DFA.

Finding natural formal definitions that capture the notion of efficient inference of DFAs is not at all straightforward. In particular, if our notion of "efficient" is the usual one of polynomial-time computation, then it is natural to allow the inference algorithm time polynomial in both the size of the target DFA, as well as in the lengths of the example strings seen. Because the example space Σ^* contains strings of arbitrary length, dependence on this latter quantity introduces a number of definitional problems. In part, a main catalyst for many of the results we discuss was the development of appropriate and natural definitions for computationally efficient DFA inference.

3 Exact Identification

In this section we consider the case where successful inference requires the inference algorithm to *exactly* identify the language represented by some target DFA. After discussing a number of different protocols and developing definitions for polynomial time identification in the limit, we review the results of Angluin [4, 7] which assert that this type of exact identification is too demanding to allow for polynomial time identification of DFAs.

3.1 Identification in the Limit

We begin with the framework of identification in the limit introduced by Gold [26]. A *presentation* of the language accepted by a given target DFA M is defined to be any infinite sequence of examples such that for every string $w \in L(M)$, the example $(w, +)$ occurs at least once in the sequence, for every string $w \in \Sigma^* - L(M)$, the example $(w, -)$ appears at least once in the sequence, and no other (that is, incorrectly labeled) examples appear in the sequence.

In this model, a (deterministic) inference algorithm A is given as input an arbitrary presentation written on a read-only input tape. After reading each next example of the presentation, A outputs an hypothesis DFA. A is said to *identify* the DFA M *in the limit* iff on input of any presentation of $L(M)$, the infinite sequence of DFAs output by A satisfies the following property: There exists a particular DFA M' such that for all sufficiently large i, the i-th output of A is M', and furthermore, $L(M') = L(M)$. Thus the sequence of outputs of A must converge to a DFA that accepts the same language as the target DFA. Note that the point of convergence may depend on the particular presentation. The class of DFAs is said to be *identifiable in the limit* iff there exists a single inference algorithm A such that for all DFAs M, A identifies M in the limit. This criterion of successful inference is also known as *EX-identification* [20].

We also consider a presentation of only the *positive* examples of the DFA M. A *positive presentation* of M is the same as a presentation of M, except that the sequence contains all and *only* the strings that are in the language $L(M)$. The class of DFAs is said to be identifiable in the limit *from positive examples only* iff there exists a single inference algorithm that identifies every DFA from any positive presentation.

Theorem 2 [26] *DFAs are identifiable in the limit, and are not identifiable in the limit from positive examples only.*

The proof of the first part is a simple application of the technique of *identification by enumeration*: On input of a given presentation, at stage i of its computation, the inference algorithm outputs the lexicographically first DFA that is consistent with the first i elements in the presentation. This approach is easily modified so that at any point, the conjectured DFA is a smallest canonical DFA that is consistent with all examples seen to that point. The second part follows from a more general theorem that states that any class of languages containing all of the finite languages, and at least one infinite language, is not identifiable from positive examples alone. The theorem holds for any class of representations for the regular languages.

3.2 Polynomial-time Identification in the Limit

The solution (in the case of positive and negative examples) of identification by enumeration discussed above seems at once unsatisfactory from the standpoint of computational efficiency. We are led to the question of whether there is a *polynomial time* procedure that identifies the class of regular languages in the limit. A problem with this question is that it is ill-posed — polynomial time in *what*? In this subsection we successively propose and analyze various definitions for polynomial time limiting identification. Ultimately, we arrive at what we believe to be one of few possible natural definitions.

In the discussion to follow, let M be the target DFA, and let M have size n. Let $m_1, m_2, \ldots$ be the lengths, respectively, of each element in some infinite presentation of examples of M. We would like to allow for "polynomial-time". Certainly we should allow the inference algorithm time to write down a correct hypothesis, which will be at least n in the case that M is a canonical DFA.

Consider the augmented definition of identification in the limit where, in addition to convergence to a correct hypothesis, we insist that the inference algorithm must have total computation time at most $p(n)$ for some polynomial p. This definition is clearly too restrictive, since the very first example may have length $m_1 > p(n)$, and thus the inference algorithm would not even be able to examine the first example.

Another definition allows for *polynomial update time*: The inference algorithm is allowed at most $q(n, m_1 + m_2 + \cdots + m_i)$ steps to produce it's i-th hypothesis, where q is any polynomial function of two variables. This requirement alone is not sufficiently restrictive, since the exhaustive search strategy of identification by enumeration (which we are explicitly trying to exclude) may be implemented so as to have polynomial update time. Demanding quick processing of each example is no guarantee that the total amount of time spent by the inference algorithm will be small.

Bounding the number of "mind changes" before the inference algorithm has converged to a correct hypothesis has also been considered as a measure of complexity of inference (see, for example, [12, 20, 21]). We might require that the number of changes of hypothesis be at most $p(n)$ for some polynomial p, in addition to requiring polynomial update time. This criterion of success is satisfied by the following algorithm. At most one DFA of size i, for $i = 0, 1, 2, \ldots n$, is ever conjectured; thus the number of mind changes is at most n. Initially the algorithm hypothesizes the null DFA. If the current hypothesis is a DFA M_i of size i, then M_i is repeatedly conjectured until for some $j > i$, enough additional examples have been collected so that in time polynomial in the length of the additional examples, it can be verified that there is exactly one canonical DFA M_j (up to isomorphism), of size j such that M_j is the smallest canonical DFA consistent with all examples seen so far. At this point, M_j is conjectured. While this algorithm achieves the desired bounds for mind changes and update time, the DFAs it produces are not particularly useful until a number of examples that is exponential in n have been seen. The algorithm achieves a bounded number of mind changes only by tolerating intermediate hypotheses that are clearly inconsistent with many of the examples already seen. Bounding the number of mind changes does not exclude algorithms that are intuitively inefficient.

In [21], general definitions are developed for the complexity of inductive inference. These definitions correspond to the total amount of computation time spent by the inference algorithm before it converges to a correct hypothesis. We have already noted that we need to allow the time bound to depend on the length of the examples seen. Thus a reasonable definition might be that an inference algorithm runs in polynomial time iff the total amount of computation time is at most $p(n, m_1 + \cdots + m_k)$, where p is a polynomial, and where the algorithm converges after seeing the k-th example. (The convergence point k may depend on the presentation.) This definition is also unsatisfactory, because any inference algorithm (in particular, the exhaustive search strategy) could simply delay convergence (while doing virtually no computation) until a sufficiently long example appears so that the algorithm may meet the polynomial time bound.

An attempt to fix the previous definition by requiring that the point of convergence be polynomial in n results in a definition that is not satisfied by any inference algorithm. The presentation may be such that a key example that distinguishes two otherwise equivalent DFAs does not appear until after the required convergence point. We note Gold's result [26] that the method of identification by enumeration is *optimally text efficient* in the sense that no other inference strategy converges at least as soon as it does on all presentations, and converges before it does on at least one presentation.

It seems that the most serious impediment in devising a reasonable definition of polynomial time limiting identification is that the inference algorithm has no control over the data presentation, thus no guarantee as to point of convergence can be made. However, an hypothesis that is consistent with all past data and is correct for a reasonably long stream of additional examples may be useful

in practice, especially if some bound may be given on the number of times the current hypothesis will contradict the next example.

Let A be an inference algorithm for DFAs. After seeing i examples, A conjectures some DFA M_i. We say that A makes an *implicit error of prediction* at step i if M_i is not consistent with the $(i+1)$-st example. (The algorithm is not *explicitly* making a prediction, but rather the conjectured DFA *implicitly* predicts the classification of the next example string.) We propose the following as a definition of polynomial time identification in the limit. The definition is stated in terms of DFAs, but may be applied to any inference domain.

Definition 3 *DFAs are* identifiable in the limit in polynomial time *iff there exists an inference algorithm A for DFAs such that A has polynomial update time, and A never makes more than a polynomial number of implicit prediction errors. More specifically, there must exist polynomials p and q such that for any n, for any DFA M of size n, and for any presentation of M, the number of implicit errors of prediction made by A is at most $p(n)$, and the time used by A between receiving the i-th example and outputting the i-th conjectured DFA is at most $q(n, m_1 + \cdots + m_i)$, where m_j is the length of the j-th example.*

A model of *prediction* in the limit has also been investigated [12, 13, 14, 16, 37, 44, 45]. This model (as applied to DFAs) dictates that the inference algorithm receives an *unlabeled* string w, and must predict whether or not w is in the language of the target DFA M. After the prediction, the algorithm is told whether in fact $w \in L(M)$. There is no requirement that the inference algorithm ever conjecture a DFA, or have one "in mind" during the inference process. Littlestone [37] considers bounding the number of (explicit) prediction errors made by such a prediction algorithm in the worst case over all presentations of a language to be predicted. Definition 3 captures the spirit of Littlestone's model in the case where the inference algorithm is required to output a correct DFA (in the limit). The definition of polynomial time identification in the limit is also closely related to the definitions of Angluin [7, 11] considered in the next section, and the development of Definition 3 nicely motivates her model of polynomial time identification using equivalence queries.

3.3 Polynomial-time Identification using Equivalence Queries

Angluin [11] considers inference algorithms that may make *equivalence queries* to obtain information about the language of the target DFA M. An equivalence query is a description of a conjectured DFA M', and if $L(M) = L(M')$, then the reply to the inference algorithm is "yes", and the inference is completed. Otherwise, the inference algorithm is told "no", and supplied with a *counterexample* — an arbitrary string $w \in L(M) \oplus L(M')$, the symmetric difference of $L(M)$ and $L(M')$. Angluin gives the following definition:

Definition 4 *An algorithm A* identifies the class of DFAs using equivalence queries in polynomial time *if and only if there exists a polynomial $p(n, m)$ such that for any DFA M, when A is run with an oracle to answer equivalence queries for $L(M)$, it halts and outputs a DFA M' such that $L(M') = L(M)$. Moreover,* at any point during the run, *the time used by A* to that point *is bounded by $p(n, m)$, where n is the size of M, and m is the length of the longest counterexample returned by any equivalence query* seen to that point *in the run.*

Note that not only is the total running time bounded, but the amount of time used at any point is required to be polynomial in the counterexamples provided to that point. Angluin shows that if only the total running time is polynomially bounded, then a variant of identification by enumeration would satisfy the less restrictive definition by first finding a correct DFA using equivalence queries (ignoring any time constraints), and then forcing the oracle to return a sufficiently long string as

a counterexample so that the total computation time becomes polynomial. This is achieved by conjecturing a DFA that accepts a language identical to the correct DFA except for some very long string w. (If one objects to the algorithm continuing after it has obtained a correct hypothesis, the construction may be modified so that the correct DFA is not produced until after the very long counterexample is obtained.)

It is interesting to compare Definition 3 with Definition 4. Suppose that in the former we ignore all hypotheses of the inference algorithm that do not result in an implicit error of prediction, and compress the run of the inference algorithm to contain only those trials where the hypothesized DFA is incorrect on the next example. Then this corresponds to identification with equivalence queries, since after each conjecture, a counterexample is provided (or, if the conjecture is correct, the inference task ends). More formally, it is not difficult to show that if DFAs are polynomial time identifiable in the limit, then they are identifiable in polynomial time using equivalence queries. The definition of identification in polynomial time using equivalence queries is less restrictive in that the number of implicit prediction errors may depend also on the size of counterexamples that are returned, whereas in Definition 3, only the update time may depend on this parameter — the number of implicit errors is bounded only by a polynomial function of the size of the target DFA.

The following theorem is due to Angluin.

Theorem 5 [4, 7] *The class of DFAs is not polynomial time identifiable using equivalence queries.*

As a corollary we have

Corollary 6 *The class of DFAs is not polynomial time identifiable in the limit.*

The proof of Theorem 5 is by a general method dubbed "approximate fingerprinting". For each n, an exponentially sized subclass H_n of DFAs of size n is constructed with the following properties. For any DFA M (not necessarily from H_n), for any polynomial q, and for all sufficiently large n, there is a string x_M with length bounded by a polynomial in n, such that the fraction of DFAs of H_n that agree with M on the string x_M is less than $\frac{1}{q(n)}$. Thus by answering "no" to the equivalence query "M", and providing the inference algorithm with the counterexample x_M, an adversary may ensure that less than a polynomially small fraction of the set H_n may be eliminated from consideration by the inference algorithm. The adversary can repeat this approach, and force any algorithm to make more than $q(n)$ queries before it can eliminate all but one element of H_n. Since q was an arbitrary polynomial, the result follows.

Angluin uses the method of approximate fingerprints to obtain other negative results in the model of polynomial time identification using equivalence queries. Similar theorems are proven for NFAs, context-free grammars, bottom-up tree automata, DNF formulas, and μ-formulas.

The proof of Theorem 5 is information-theoretic; it makes no complexity-theoretic assumptions whatsoever. In fact, the polynomial bound on the running time of the inference algorithm is used only to bound the size of the hypothesized DFAs. Thus the theorem essentially states that a polynomial number of polynomially sized conjectures (together with the corresponding counterexamples) need not provide enough information to uniquely distinguish a particular DFA from among the class H_n. Consequently, even an oracle for PSPACE does not help the inference algorithm.

The theorem is *representation dependent* — the proof relies on the fact that the inference algorithm is only allowed to make equivalence queries using DFAs. In [11] a very general "majority vote" algorithm (see also [14, 37]) is shown to achieve a number of equivalence queries that is logarithmic in the size of the class of target representations. This general strategy uses equivalence queries that are not limited to any one class of representations. Since the classes H_n have only exponentially many elements, the majority vote strategy identifies these subclasses of DFAs using

only polynomially many equivalence queries. Theorem 5 points out that the majority vote algorithm has no efficient implementation that uses DFA equivalence queries alone. By the comments above, the reason for this must be that the languages queried by the majority vote strategy are not representable by polynomially sized DFAs.

Another interesting property of the proof of Theorem 5 is that the target DFAs under consideration (i.e., elements of H_n) accept only finite languages. Thus polynomial time identification of DFAs with equivalence queries is no easier in the restricted case of DFAs accepting only finite languages. Independently, Ibarra and Jiang [33] show that the inference problem for finite languages, and fixed-length languages (all strings are the same length), is no easier than the inference problem for DFAs accepting arbitrary languages.

A final note regarding this result is that the subclass of DFAs used in the proof are all *zero-reversible*: each DFA has at most one final state, and for any state q and symbol a there is at most one state p such that $\delta(p, a) = q$. Thus even the class of zero-reversible automata are not identifiable in the limit using equivalence queries. This contrasts with the algorithm of Angluin [5], that has polynomial update time, and identifies this class in the limit from positive examples only.

A number of other results have been shown in the model of polynomial time identification with equivalence queries. In [11] similar information-theoretic lower bounds are presented for identification of other types of languages, as well as some positive results that rely on additional types of queries (see also [9]). We will discuss the positive results for DFAs in Section 5.

4 Approximate Identification

In this section we consider a less demanding criterion of *approximate* identification, and generalizations of this criterion to approximate *prediction*. Despite these relaxed definitions, if the information available is limited to example strings, then the existence of an efficient inference algorithm seems unlikely. As opposed to the information-theoretic results of Theorem 5, the negative results reviewed here necessarily depend on various complexity-theoretic assumptions.

4.1 PAC-Identification

In [58], Valiant introduces a distribution-independent model of randomized inference that has been called *PAC-identification*. Subsequently, the model has been refined and extended in a number of ways; the reader should consult [17, 29, 43] for interesting variations. Here we consider the version(s) most applicable to the class of DFAs.

"PAC" abbreviates "probably approximately correct". The goal of a PAC-identification algorithm is to obtain, with high probability, a DFA that is approximately correct when compared to the target DFA. Assume that there is an unknown and arbitrary probability distribution D on strings of Σ^* of length at most m. (Below we justify this bound on string length.) A PAC-inference algorithm A may sample strings according to the distribution D, and for each string w that is generated, A is told whether or not $w \in L(M)$, where M is the target DFA. A is required to produce, in polynomial time and with high probability, a DFA that is likely to correctly classify a randomly chosen example according to the distribution D on which it has trained. "With high probability", and "likely" in the previous sentence are precisely quantified by two parameters that are supplied to A: an *accuracy* parameter ϵ, and a *confidence* parameter δ.

Definition 7 *DFAs are* PAC-identifiable *iff there exists a (possibly randomized) algorithm A such that on input of any parameters ϵ and δ, for any DFA M of size n, for any number m, and for any probability distribution D on strings of Σ^* of length at most m, if A obtains labeled examples*

of M generated according to distribution D, then A produces a DFA M' such that, with probability at least $1-\delta$, the probability (with respect to distribution D) of the set $\{w : w \in L(M) \oplus L(M')\}$ is at most ϵ. The run time of A (and hence the number of randomly generated examples obtained by A) is required to be polynomial in $n, m, \frac{1}{\epsilon}, \frac{1}{\delta}$, and $|\Sigma|$.

In many other PAC-identification domains that have been studied (e.g., the inference of Boolean functions), all example strings for a given target function have the same length (e.g., the number of Boolean variables over which the function is defined). Again, in the case of DFAs, definitional problems arise because example strings may have different lengths. In the case of PAC-identification, there are especially subtle issues in arriving at a natural definition due to the fact that examples are chosen randomly. These issues are discussed in more detail in [43]. In the definition above, the distribution is limited to a finite sublanguage of the regular language accepted by the target DFA. This definition seems to be the best of several possible alternatives discussed in [43]. Further note that for any distribution D on the countable space Σ^*, and for any arbitrarily small $\gamma > 0$, there is a length m such that the probability of any string of length greater than m occurring is at most γ. Thus there is a distribution D' assigning zero probability to strings of length greater than m such that D' "approximates" D within γ. Consequently, the restriction to such distributions is not unreasonable. It is also of interest that similarly restricting the class of target languages to finite sublanguages does not simplify the inference task in the case of polynomial time identification with equivalence queries [4, 33]. See Section 3.3 for additional discussion.

A relaxation of the PAC-identification criterion allows the inference algorithm to output representations chosen from some other hypothesis class, as long as the language associated with the representation output by the algorithm has error at most ϵ when compared to the target DFA and the distribution on which the inference has occurred. For example, an algorithm might output an NFA N as its conjecture. If H is a class of representations (e.g., NFAs, regular expressions, context-free grammars, etc.), then we say that DFAs are PAC-identifiable *in terms of* H iff the above definitions hold, but the inference algorithm outputs a representation from the class H. The choice of hypothesis space can often be the determining factor as to whether or not a feasible algorithm exists. For example, it is known that the class k-term-DNF of Boolean formulas in disjunctive normal form with at most k terms ($k \geq 2$ and constant) is not PAC-identifiable unless $RP = NP$, but there *is* a PAC-algorithm for this class in terms of the class k-CNF of conjunctive normal form expressions where each clause has size at most k [41].

Finally, there is a notion of polynomial time *predictability* that corresponds to the criterion of PAC-identification. We say that DFAs are *(polynomially) approximately predictable* iff there exists any (polynomially reasonable) hypothesis class H such that DFAs are PAC-identifiable in terms of H. This is equivalent to a model discussed in [43], where, as in the definitions of prediction in the limit described earlier, instead of outputting a DFA as a conjecture, the inference algorithm need only classify unlabeled examples correctly. In particular, consider the protocol where the inference algorithm A may access randomly generated examples (as in PAC-identification), and then halts in a special *predict state*. A new (unlabeled) string is drawn according to the distribution, and A predicts whether $w \in L(M)$ or $w \notin L(M)$. If A is incorrect, then A makes an (explicit) error of prediction. The following definition of approximate prediction is equivalent to that of PAC-identification in terms of an arbitrary (polynomially reasonable) hypothesis space. (The parameter δ has been absorbed into the parameter ϵ.)

Definition 8 *DFAs are* polynomially approximately predictable *iff there exists a (possibly randomized) algorithm A such that on input of any parameter ϵ, for any target DFA M of size n, for any number m, and for any probability distribution D on strings of Σ^* of length at most m, if A*

obtains labeled examples of M generated according to distribution D, then A halts in a predict state such that the probability is at most ϵ that A makes an error of prediction on a new string generated according to distribution D. The run time of A is required to be polynomial in $n, m, \frac{1}{\epsilon}$, and $|\Sigma|$.

A number of relationships between these (and other) criteria have been given [11, 29, 37, 43]. Particularly relevant to our investigation here is the following theorem due to Angluin.

Theorem 9 [11] *For all polynomially reasonable classes of representations R, and for DFAs in particular, if R is polynomial time identifiable with equivalence queries, then R is PAC-identifiable, and hence polynomially approximately predictable. Further, there are classes of representations that are PAC-identifiable, but not polynomial time identifiable with equivalence queries.*

4.2 Occam's Razor: Finding Small DFAs

The *minimum consistent DFA problem* it that of finding the smallest DFA that is consistent with each of a finite collection of examples. We will see below how this problem is motivated by the results of Blumer, Ehrenfeucht, Haussler, and Warmuth [18] in the context of PAC-identification. However, before the PAC-identification criterion was proposed, the minimum consistent DFA problem had already received significant attention. This is not surprising, in that independent of any formal justification from the standpoint of computational inference models, the principle of *Occam's Razor* [40], is appealing both intuitively and philosophically as an inference technique. In the context of inference, Occam's Razor is taken to mean that *among competing hypotheses, the simplest is preferable.* Taking the standard information-theoretic interpretation of "simplest" to mean "fewest number of bits", Occam's Razor suggests that we attempt to identify a regular language by hypothesizing, at any point, the smallest DFA consistent with the set of examples seen up to that point. This general approach has been taken in a number of practical and theoretical settings [2, 12, 36, 48, 49, 56].

Gold [25] showed that an algorithm for the minimum consistent DFA problem could be used to achieve identification in the limit, and that such a technique would be optimally data efficient. However, Gold also showed that it is NP-hard to find the smallest DFA consistent with a given sample. Trakhtenbrot and Barzdin [57] show that if the sample consists of all strings up to a given length, then there is a polynomial-time algorithm for finding the smallest consistent DFA. Angluin [10] extends Gold's NP-hardness results by showing that the problem remains NP-hard even if all but ϵ of the strings up to a given length are given as examples. Angluin also shows that the problem of finding a smallest consistent regular expression is NP-hard.

That the minimum consistent DFA problem is relevant to inference is more formally supported by the work of Blumer, Ehrenfeucht, Haussler, and Warmuth [18]. To understand their results, we need the following definition.

Definition 10 *An* Occam algorithm *for the class of representations R in terms of the class H is an algorithm O such that for some constants $k \geq 0$ and $0 \leq \alpha < 1$, the following guarantee holds. Let s, n, and m be any numbers, and let $r \in R$ have size n and represent the language $L(r)$. Then on input of any set of s examples of $L(r)$, each of length at most m, O outputs an element $h \in H$ of size at most $n^k m^k s^\alpha$ that is consistent with each of the s examples. An Occam algorithm for a class R in terms of the same class R is called an Occam algorithm for R.*

Blumer et al prove the following sufficient condition for PAC-identification:

Theorem 11 [17, 18] *For any classes of representations R and H, if there exists a polynomial-time Occam algorithm for R in terms of H, then R is PAC-identifiable in terms of H.*

Corollary 12 *For any (polynomially reasonable) classes of representations R and H, if there exists a polynomial-time Occam algorithm for R in terms of H, then R is polynomially approximately predictable.*

Theorem 11 and Corollary 12 also hold if the Occam algorithm is randomized, and is only required to produce an element $h \in H$ as specified with probability exceeding some constant γ.

If we let both R and H be the class of DFAs, then Theorem 11 gives the following sufficient condition for PAC-identification of DFAs.

Corollary 13 *DFAs are PAC-identifiable if there exists a (possibly randomized) polynomial-time Occam algorithm for DFAs.*

Hence the PAC-identifiability of DFAs would follow from the existence of a random polynomial-time algorithm that could *approximately* solve the minimum consistent DFA problem. Board and Pitt [19] recently prove that if a hypothesis class H is "closed under exceptions" then the converse of Theorem 11 holds. They observe that DFAs have this property, and obtain the following theorem.

Theorem 14 [19] *If DFAs are PAC-identifiable then there exists a randomized polynomial-time Occam algorithm for DFAs.*

Together with Corollary 13, Theorem 14 shows that the PAC-identifiability of DFAs is equivalent to the existence of a random polynomial-time approximation algorithm for the minimum consistent DFA problem that makes only a very weak approximation guarantee. More succinctly, for DFAs, PAC-identification is equivalent to data compression.

Consequently, this approach to inferring DFAs has recently received more scrutiny. Li and Vazirani [35] provided the first nonapproximability result extending the NP-completeness result of [25], by showing that it is NP-hard to produce a consistent DFA that is at most $\frac{9}{8}$ times larger than the smallest consistent DFA [35]. Recently, Pitt and Warmuth [42] show (assuming $P \neq NP$) that there is no polynomial-time approximation algorithm that is guaranteed to produce a consistent DFA of size bounded above by *any* polynomial in the size of the smallest DFA; in particular, they prove the following theorem.

Theorem 15 [42] *For any constant $\epsilon > 0$, the following optimization problem is NP-hard. Given a set of examples of some DFA with at most n states, produce a DFA with at most $n^{(1-\epsilon)\log\log n}$ states.*

Theorem 15 appears to be one of few very strong nonapproximability results for naturally arising combinatorial optimization problems. Theorem 15 does *not* imply that DFAs are not PAC-identifiable: The theorem does not preclude the existence of an Occam algorithm, since the unachievable approximation bound is a function only of the size of the smallest DFA, whereas an Occam algorithm allows the size of the DFA produced to also depend polynomially on the length m of each example, as well as sublinearly on the number s of examples for which a consistent DFA is sought. Note also that the result is *representation dependent*, in that it does not preclude the existence of an approximation algorithm that produces a consistent hypothesis h from some class H other than DFAs, such that the size of h is polynomially bounded in the size of the smallest consistent DFA.[1] Such an algorithm would show that DFAs were PAC-identifiable in terms of the

[1] However, for the case that H is the class of NFAs, both the results of [35] and the results of [42] remain true. If H is the class of regular expressions, regular grammars, DFAs, or NFAs over the fixed alphabet $\{0, 1\}$, then a slightly weaker bound than that of Theorem 15 holds [42].

hypothesis space H, and thus, by Corollary 12, were polynomially approximately predictable. In fact, in Section 5.4 we will see that the *subclass* of DFAs for which the nonapproximation result of Theorem 15 holds *is* polynomially approximately predictable.

Based on cryptographic and number-theoretic assumptions that are (ostensibly) stronger than the $P \neq NP$ assumption, Kearns and Valiant [34] show that DFAs are not polynomially approximately predictable (Theorem 19 below). By Corollary 12 they conclude that there is no polynomial-time Occam algorithm for DFAs *in terms of any polynomially reasonable class of representations.* They thus obtain the following representation independent nonapproximability result for the minimum consistent DFA problem.

Theorem 16 [34] *For any constants $k \geq 0$ and $\alpha < 1$, the following optimization problem is as hard as any of the cryptographic or number-theoretic problems mentioned in the statement of Theorem 19: Given a set of s examples, each of length at most m, of some DFA with at most n states, produce a polynomial-time algorithm that is consistent with all s examples, and has size at most most $n^k m^k s^\alpha$.*

4.3 PAC-identifiability and Predictability: Negative Results

It is not difficult to show that if $RP = NP$ then any polynomially reasonable class of representations is PAC-identifiable. Thus any negative results for the PAC-identification criterion must make the standard complexity-theoretic assumption that $RP \neq NP$.

There *are* negative results based only on this assumption (for example, see [28, 41]). These negative results are typically obtained by showing that the consistency problem for the class of representations is NP-hard [17, 28, 41]. (The consistency problem is that of producing *any* element from the class that is consistent with a given set of examples.) Most such results involve a class of representations that is restricted in some way (for example, DNF with a constant upper bound on the number of terms). For any class of representations that is powerful enough to express arbitrary disjunctions, this approach to proving non PAC-identifiability will not work. For example, the consistency problem for DNF formulas (without a bound on the number of terms) is trivial. The consistency problem for DFAs is also easily solved by constructing a DFA that accepts exactly the positively labeled strings in the given sample. Thus this approach is not a useful one in the present context. Note however, that we may obtain a negative result for the class of DFAs restricted by size: The nonapproximability result of Theorem 15 shows, based only on the assumption that $RP \neq NP$, that DFAs with n states are not PAC-identifiable in terms of DFAs with at most $n^{(1-\epsilon)\log\log n}$ states.

For classes of representations that are not restricted in some way, it appears to be very difficult to prove negative results in the PAC-identification model based only on the assumption that $RP \neq NP$. An alternative approach, analogous to the one taken in computational complexity theory, is to relate the relative difficulty of PAC-identification of various classes. Perhaps DFAs are at least as hard to PAC-identify as a wide variety of other classes of representations.

This approach was taken by Pitt and Warmuth [43]. In the context of approximate prediction (i.e., PAC-identification in terms of an arbitrary hypothesis space), a notion of *prediction preserving reduction* is given (see also [37]). The existence of a (polynomial time) prediction preserving reduction from inference problem A to inference problem B guarantees that if B is polynomially approximately predictable, then A is also polynomially approximately predictable. A number of interesting reductions are given.

Prediction problems are classified based on the computational complexity of their *evaluation problems.* The evaluation problem for a class R is to determine, given any $r \in R$ and $w \in \Sigma^*$,

whether or not $w \in L(r)$. Since all classes we consider are polynomially reasonable, by condition 3 of Definition 1, the evaluation problems for these classes are all in the complexity class P. Pitt and Warmuth prove the following "completeness" theorem for DFA prediction:

Theorem 17 [43] *If DFAs are polynomially approximately predictable, then so is every class of representations whose evaluation problem is in the complexity class Deterministic LogSpace.*

The theorem is proved by showing that (1) if R is a class of representations whose evaluation problem is in Deterministic LogSpace, then there is a prediction preserving reduction from R to the class of logspace bounded deterministic Turing machines; and (2) there is a prediction preserving reduction from logspace bounded deterministic Turing machines to DFAs. Thus not only would the polynomial time approximate predictability of DFAs imply the same for logspace Turing machines, but also for a wide variety of classes of representations. In particular, since the evaluation problem for Boolean formulas is in Deterministic LogSpace, the following corollary is obtained.

Corollary 18 [43] *If DFAs are polynomially approximately predictable, then Boolean formulas are polynomially approximately predictable.*

Corollary 18 is particularly interesting because the class of DNF formulas is a very restricted subclass of Boolean formulas, and whether DNF formulas are PAC-identifiable (and therefore, approximately predictable) has remained a basic open problem since it was posed by Valiant in 1984 [58]. By relying on other unresolved inference problems, the fact that DFAs are "prediction-complete" for Deterministic LogSpace provides some evidence for the difficulty of their prediction. Similar results are obtained showing that other types of automata are prediction-complete for a range of complexity classes.

Any "evidence of intractability" for a problem A that is obtained via a reduction from a problem B (or from a large class of problems) is only plausible to the extent that we believe B to be an intractable problem to begin with. In the area of cryptography, a number of cryptosystems exist that appear to be difficult to invert. Indeed, one of the main research goals in this area is to devise cryptographic schemes that are provably hard to invert. Additional and more convincing evidence of the intractability of DFA prediction can be obtained by relying on assumptions concerning the difficulty of breaking various cryptographic systems. Kearns and Valiant [34] take exactly this approach, by considering certain cryptographic and number-theoretic problems. Before describing their results, we briefly explain the problems on which they are based.

The *RSA encryption function*, due to Rivest, Shamir, and Adleman [53], is the basis for a well known public key cryptosystem that has received much analysis. There is currently no known polynomial-time algorithm for inverting the RSA function. Similarly, the encryption scheme of Rabin [47] is thought to be hard to invert. In Rabin's encryption scheme, the problem of decryption can be shown to be equivalent (with respect to polynomial-time probabilistic computation) to the problem of *factoring Blum integers*, which is that of determining the factors p and q, each l-bit primes congruent to 3 mod 4, given only the product $N = pq$. The evidence of difficulty of inverting Rabin's encryption scheme rests on the fact that the problem of factoring Blum integers has received much scrutiny, yet no known polynomial-time probabilistic algorithm for it has been discovered. Finally, the *quadratic residue problem* is also a well studied problem of number theory for which no polynomial-time solution is known: Let Z_N^* be the multiplicative group modulo N (Z_N^* contains only elements of Z_N that are relatively prime to N) and let $J(x, N)$ denote the Jacobi symbol of x with respect to N [39]. Then the *quadratic residue problem* is the following. Given a number N that is the product of two (unknown) primes p and q, each of length l, and given a number $x \in Z_N^*$ such that $J(x, N) = 1$, determine if there exists $a \in Z_N^*$ such that $x = a^2 \bmod N$.
Kearns and Valiant [34] prove the following theorem.

Theorem 19 [34] *If DFAs are polynomially approximately predictable, then there is a probabilistic polynomial time algorithm for inverting the RSA encryption function, for factoring Blum integers, and for deciding quadratic residues.*

We describe the two main steps involved in the proof of Theorem 19, using the RSA function as an example: (1) A class of representations R is defined such that the problem of polynomially approximately predicting the class R is equivalent (with respect to probabilistic polynomial time computation) to the problem of inverting the RSA encryption function. (2) It is shown that the class R has an evaluation problem that is contained in the complexity class Deterministic LogSpace. (Actually, they give the stronger result that the evaluation problem for R is contained in NC^1 — the class of polynomial size, log depth circuits of standard fan-in two Boolean gates.) Theorem 19 then follows from Theorem 17.

Let us look at these two steps in a bit more detail. First we state a few facts about the RSA scheme. (The following is not meant to fully describe the RSA encryption function — we present only what is necessary for our exposition.) A particular RSA function is given by specifying three numbers p, q, and e, with special properties. Let $N = pq$. The encryption function is defined by $RSA(N, e, x) = x^e \bmod N$. If p, q, and e are known, then inverting $RSA(N, e, x)$ (i.e., finding x) may be done in polynomial time. If only N and e are known, but not p and q, then inverting $RSA(N, e, x)$ is believed to be difficult for x chosen randomly from Z_N^* according to a uniform distribution.

It has been shown that finding the least significant bit of x given $RSA(N, e, x)$ is as hard as determining the entire string x [3]. This suggests the following inference problem R based on the RSA function. The class of representations R consists of triples (p, q, e) with the required special properties. A positive example of (p, q, e) is any number $RSA(N, e, x)$ such that the least significant bit of x is 1. A negative example is a number $RSA(N, e, x)$ such that the least significant bit of x is 0. Thus learning to predict positive and negative examples of a target representation (p, q, e) is equivalent to the problem of determining the least significant bit of the encrypted number x, which in turn is equivalent to decrypting the RSA function.

For step (2) described above, it must be shown that the evaluation problem for R is solvable in deterministic logspace. In this case, the evaluation problem is to determine, given (p, q, e), and $z = RSA(N, e, x)$, whether or not z is a positive example, i.e., whether or not the least significant bit of x is a 1. This can be done in polynomial time, but unfortunately, this problem does not appear to be achievable in deterministic logspace. Kearns and Valiant overcome this obstacle by describing a different prediction problem for R, where each example consists not only of an encrypted string $RSA(N, e, x)$, but also contains additional information that is sufficient to lower the complexity of the resulting evaluation problem down to the complexity class NC^1. Although the evaluation problem becomes easier, the additional information does not make the prediction problem any easier, since it is shown that this information is readily obtained by any probabilistic polynomial time algorithm from the encrypted message alone.

Thus classes of representations are defined that are as hard to approximately predict as the above number-theoretic problems are to solve, and whose evaluation problem lies in the complexity class NC^1. They also prove the following related theorem, by exhibiting a prediction-preserving reduction from the class of NC^1 circuits to the class of Boolean formulas.

Theorem 20 [34] *If the class of Boolean formulas are polynomially approximately predictable, then there is a probabilistic polynomial time algorithm for inverting the RSA encryption function, for factoring Blum integers, and for deciding quadratic residues.*

Each of the number-theoretic problems on which Theorem 19 (and Theorem 20) is based have the additional interesting property that if there is an algorithm that solves the problem for some

small fraction of inputs according to a uniform distribution, then there is an algorithm that solves the problem for *most* inputs according to a uniform distribution. As a result, Theorem 19 can be strengthened to show that DFAs are not predictable even in a very weak sense.

Definition 21 *DFAs are* polynomially weakly predictable *iff there exists a (possibly randomized) algorithm A and a polynomial q such that for any target DFA M of size n, for any number m, and for any probability distribution D on strings of Σ^* of length at most m, if A obtains labeled examples of M generated according to distribution D, then A halts in a predict state such that the probability is at most $\frac{1}{2} - \frac{1}{q(n)}$ that A makes an error of prediction on a new string generated according to distribution D. The run time of A is required to be polynomial in n, m, and $|\Sigma|$.*

Thus weak predictability only requires that the inference algorithm arrive at a state where it can predict only slightly better than guessing randomly.

Corollary 22 [34] *If DFAs are polynomially weakly predictable then there is a probabilistic polynomial time algorithm for inverting the RSA encryption function, for factoring Blum integers, and for deciding quadratic residues.*

This corollary also follows from Theorem 19 by a recent result of Schapire [55], showing that any class is polynomially approximately predictable if and only if it is polynomially weakly predictable.

The cryptographic and number-theoretic problems above are assumed hard when the input is drawn according to a uniform distribution. If we trace through all necessary prediction-preserving reductions in the proof of Theorem 19, we may observe that the results of Theorem 19 hold even when the distribution on example strings of the target DFA is *flat* — each positive example (that has nonzero probability) has the same probability, and similarly for the negative examples.

Finally, by Theorem 9, assuming that the above mentioned cryptographic and number-theoretic problems are intractable, the negative results for polynomial time identification of DFAs with equivalence queries (Theorem 5) follows from Theorem 19 even when the equivalence queries are allowed to be from any polynomially reasonable hypothesis class. Thus we may strengthen Theorem 5 to be representation independent by making stronger assumptions.

5 Learning with Additional Information

Any of the inference models that we have discussed may be augmented by either allowing the inference algorithm to obtain additional information concerning the target DFA, or by explicitly providing such information at the outset. In light of the negative results presented in Sections 3 and 4, it is especially important to determine the type (and amount) of additional information that is necessary and sufficient for feasible inference. In this section we review a number of positive results for DFA inference in the presence of additional information. We also consider some recent results on inferring restricted types of DFAs.

5.1 Identification with Membership Queries

When example strings are provided in a deterministic presentation (as with identification in the limit), or by a randomized source (as in PAC-identification), the inference algorithm is at the mercy of the data-stream. By instead (or in addition) allowing the inference algorithm the ability to play a more active role in obtaining examples, the task may become much easier. A *membership query* made by an inference algorithm is a string $w \in \Sigma^*$. The reply to a membership query is "yes" if

$w \in L(M)$, or "no" if $w \notin L(M)$, where M is the target DFA. Thus a membership query provides a way for the inference algorithm to quickly extract desired information about the unknown language.

DFAs are trivially identifiable in the limit from membership queries alone — an inference algorithm simply queries every string of Σ^* in some canonical order, obtaining a presentation of M, and then Theorem 2 applies. In the absence of any additional information, it is known that DFAs cannot be identified by an algorithm in finite time, i.e., by one that halts after making a single conjecture [38]. However, if an upper bound n on the number of states of M is provided, then a canonical DFA may be found by querying sufficiently many strings so that at most one canonical acceptor of size less than or equal to n is consistent with the results of the queries. However, this approach is not efficient — it has been shown that even with an upper bound on the number of states in the target DFA, any algorithm that identifies DFAs from membership queries alone must make an exponential number of queries in the worst case [9].

5.1.1 Membership queries and state representatives

Let $M = (Q, \Sigma, \delta, q_0, F)$ be any DFA, and let L be the language accepted by M. A state q of M is *reachable* if there exists $w \in \Sigma^*$ such that $\delta(q_0, w) = q$. We also say that w is a *representative* for the state q. A *complete set of state representatives* for M is a set U of strings such that for each reachable state q of M, there is a representative $u \in U$ for q. Angluin [9] proves that this information together with membership queries allows for polynomial time identification of the target DFA:

Theorem 23 [9] *The class of DFAs are identifiable in polynomial time from a set of state representatives and using membership queries. More specifically, there is a polynomial time algorithm A such that if M is any DFA of n states, then if A is given as input any set of state representatives U for M, then A makes at most $O(|\Sigma|n^2)$ membership queries, and outputs the canonical acceptor equivalent to M.*

Angluin actually proves that only representatives for "live" states of M are needed. Our consideration of the slightly weaker result above allows for a clearer presentation. The number of queries made by Angluin's algorithm (which is slightly better than the bound stated here) is shown to be optimal to within a constant factor [9].

We sketch the ideas in the proof of Theorem 23, which was motivated by the approach of Gold [27]. We first briefly review the *state minimization* algorithm for DFAs [32]. Given any DFA $M = (Q, \Sigma, \delta, q_0, F)$, a pair of states q_1, q_2 of Q is said to be *distinguishable* if there exists a string w such that $\delta(q_1, w) \in F$ but $\delta(q_2, w) \notin F$, or vice versa. Note that if $q_1 \in F$, and $q_2 \notin F$ (or vice versa), then q_1 and q_2 are distinguished by the empty string λ. Suppose that q_1 and q_2 are distinguished by string w. Then if for some $a \in \Sigma$ and $p_1, p_2 \in Q$, $\delta(p_1, a) = q_1$ and $\delta(p_2, a) = q_2$, then p_1 and p_2 are distinguished by string aw. It follows that if a pair of states is distinguishable, then the pair is distinguishable by a string of length at most $n = |Q|$. The standard polynomial-time state minimization algorithm works by starting with the given DFA M, and iteratively finding pairs of states that are distinguished by strings of length 0, length 1, length 2, ... In particular, the algorithm iteratively searches for a pair of states p_1, p_2 and a symbol a such that the pair of states $\langle \delta(p_1, a), \delta(p_2, a) \rangle$ has already been distinguished, then it marks the pair $\langle p_1, p_2 \rangle$ as distinguished. The algorithm terminates when no triple (p_1, p_2, a) can be found with the above properties. It can be shown that if any pair of states that have not been distinguished from each other are merged into a single state, then the canonical acceptor is obtained. The algorithm is easily modified to save the distinguishing strings that were found for each pair of distinguished states.

Now suppose that a complete set of state representatives U is given. Let q_u denote that state that string u represents, i.e., $q_u = \delta(q_0, u)$ in the target DFA M. The approach is to find the canonical acceptor that is equivalent to M by adapting the state minimization algorithm to: (1) determine which pairs of states q_u, q_v are distinguishable, for any $u, v \in U$; and (2) determine the transitions $\delta(q_u, a)$ for each $u \in U$ and $a \in \Sigma$.

The general step of the state minimization algorithm requires us to determine for arbitrary q_u, q_v, and a, whether $\delta(q_u, a)$ and $\delta(q_v, a)$ are distinguished states. But M is unknown, and we do not know which state $\delta(q_u, a)$ actually is. In order to apply the general iterative step of the state minimization algorithm, for any $u \in U$ and $a \in \Sigma$, we must be able to determine a $w \in U$ such that $\delta(q_u, a) = q_w$. It turns out that it is possible to determine this transition function while *simultaneously* determining state distinguishability and using membership queries. For example, we may deduce that $\delta(q_u, a) \neq q_v$ if we obtain different responses to membership queries about the strings ua and v. While this doesn't determine $\delta(q_u, a)$, it does rule out one possibility. By saving all distinguishing strings $w_1, w_2, \ldots$, that are found, and comparing the results of the membership queries $uaw_1, uaw_2, \ldots$, with the results (respectively) of the membership queries $vw_1, vw_2, \ldots$ for each $v \in U$, ultimately enough evidence is obtained so that for only one v is it possible that $\delta(q_u, a) = q_v$.

5.1.2 Equivalence queries and membership queries

In a more recent paper, Angluin [6] shows that the ability to make equivalence queries can compensate for the lack of state representatives:

Theorem 24 [6] *The class of DFAs are polynomial time identifiable using equivalence queries and membership queries.*

To see how this is proved, consider how the algorithm sketched above (call it algorithm A) might fail if it is given only a set of representatives U for a proper subset of the states of the target DFA M. While inferring the transition function δ (see above), it might be determined that for *no* $v \in U$ is it possible that $\delta(q_u, a) = q_v$. This would mean that U is not *closed* in the sense that the responses to the membership queries $uaw_1, uaw_2, \ldots$ do not exactly match the results of the membership queries $vw_1, vw_2, \ldots$ for any $v \in U$. In this case, ua must be a representative for some state not already represented in U, and we may add ua to U and run A again from the beginning. This can happen at most n times, where n is the number of states of the canonical DFA equivalent to M. Ultimately, a set U is obtained that is closed, and no such witness ua is obtained. In this case, A will run to completion, and produce a DFA M'. It turns out that any string that is a counterexample to the correctness of M' must have a prefix that is a representative for a state of M that has no representative in U. Thus, if the equivalence query "M'" results in a counterexample w, we add all prefixes of w to U, and begin the entire process again. This procedure is iterated (at most polynomially many times) until a canonical DFA is produced. Because the number of equivalence queries used in Angluin's algorithm depends only on n, and not on the length of counterexamples seen, we have the following slightly stronger corollary.

Corollary 25 *The class of DFAs is polynomial time identifiable in the limit from membership queries (and an arbitrary presentation).*

From Theorem 9, we also have

Corollary 26 *The class of DFAs is PAC-identifiable using membership queries (in addition to randomly generated examples).*

Suppose we extend the definitions for prediction preserving reductions (discussed in Section 4.3) in a natural way so as to allow for inference algorithms that make membership queries in addition to receiving randomly generated examples. A particularly intriguing open question is whether an analogue of Corollary 18 exists, i.e., whether an algorithm for predicting DFAs that uses membership queries could be used to obtain an algorithm for predicting Boolean formulas using membership queries. If so, then Theorem 24 could be used to show that Boolean formulas would be polynomially approximately predictable if membership queries were available in addition to randomly generated examples.

Extensions to Theorem 24 have been obtained in other domains. Berman and Roos [15] show that deterministic one-counter languages can be identified in polynomial time using equivalence and membership queries. Sakakibara shows that context-free grammars may be identified in polynomial time using *structural equivalence queries* and *structural membership queries* [54]. Angluin [11] gives a general investigation of different types of queries as applied to various inference domains.

5.2 Experiments

A membership query allows the inference algorithm to *reset* the automaton to the initial state, and thus observe the behavior of the automaton on a newly queried string. For some inference applications it may be unreasonable to assume that such a reset is available. Rivest and Schapire [50, 51, 52] view the inference process as a single continuous experiment, where each input given to the DFA causes a state transition which may not be reversible. Even if the transition is reversible, the sequence of characters necessary to return the DFA to the initial state may not be known to the inference algorithm.

Let $M = (Q, \Sigma, \delta, q_0, F)$ be the target DFA. It will be convenient to view M as a *Moore* machine (see [32]), where F is treated also as an *output function*. Thus we write $F(q) = 1$ if $q \in F$, and $F(q) = 0$ if $q \notin F$. Consider the inference protocol where M is initially in state q_0. At any time during the inference process, the algorithm may execute an *experiment* $a \in \Sigma$. If the current state of M is q, then after the experiment a, the DFA is in state $q' = \delta(q, a)$, and the inference algorithm may observe the output $F(q')$. Experiments are a restricted form of membership queries – while initially, an inference algorithm can determine whether a given string w is accepted, once the single query w has been made, only strings which *extend* w may be subsequently queried. In particular, experiments only allow the inference algorithm to make membership queries on any collection of strings of the form a_1, a_1a_2, $a_1a_2a_3, \ldots$, where each a_i is an element of Σ.

5.2.1 Experiments and equivalence queries

Since experiments are a restricted type of membership query, by the comments at the beginning of Section 5.1, there is no algorithm for inferring DFAs in polynomial time from experiments alone. Thus Rivest and Schapire consider inference of DFAs from equivalence queries and experiments. In their model, an equivalence query is slightly different than in the model of Angluin: Let $M = (Q, \Sigma, \delta, q_0, F)$ be the target DFA, and for any $q \in Q$, let $M_q = (Q, \Sigma, \delta, q, F)$. At any point of the inference algorithm, if q is the current state of M, then the answer to an equivalence query "M'" is "yes" if $L(M') = L(M_q)$. Thus each equivalence query is answered as if the current state was the initial state of the target DFA.

A DFA is *strongly connected* iff for any pair of states q_1, q_2, there exists a string $w \in \Sigma^*$ such that $\delta(q_1, w) = q_2$. Without the assumption that the target DFA is strongly connected, polynomial time identification using equivalence queries and experiments is not possible, because the very first experiment could force the DFA into a state from which no other states are reachable, rendering

any further experiments useless, and then Theorem 5 may be applied. While a deterministic algorithm for polynomial time identification of strongly connected DFAs using equivalence queries and experiments is not known, there is a *randomized* polynomial-time algorithm that can guarantee successful identification with high probability.

Theorem 27 [51] *The class of strongly connected DFAs are* random-polynomial time *identifiable using equivalence queries and experiments. In particular, there is a randomized algorithm A such that on input of any $\delta > 0$, for any strongly connected target DFA M, using equivalence queries and experiments, A will successfully identify M with probability at least $1 - \delta$. The running time of A is polynomial in n and m and logarithmic in $\frac{1}{\delta}$, where n is the size of M, and m is the length of the longest counterexample returned from any equivalence query.*

We give some of the main ideas in the proof of Theorem 27. For any $q \in Q$ and $w = a_1 a_2 \ldots a_m$, where each a_i is in Σ, define the *output sequence* $q(w)$ to be the sequence

$$q(w) = \langle F(q), F(\delta(q, a_1)), F(\delta(q, a_1 a_2)), \ldots F(\delta(q, a_1 a_2 \ldots a_m)) \rangle.$$

A *homing sequence* $h \in \Sigma^*$ is a string such that for all $q_1, q_2 \in Q$, $q_1(h) = q_2(h) \Rightarrow \delta(q_1, h) = \delta(q_2, h)$. Thus, from any state, if the homing sequence h is given to M, the output sequence obtained uniquely determines the resulting state of M. If h is a homing sequence, then let q_σ denote the unique state $\delta(q', h)$ for all q' such that $q'(h) = \sigma$.

The main idea of the proof is to use a homing sequence instead of a reset, and apply the algorithm of Angluin (call it B) that identifies DFAs from equivalence queries and membership queries (described in the proof of Theorem 24). Suppose that the inference algorithm has available a homing sequence h. Let q be the current state, and suppose that h is applied and output sequence $\sigma = q(h)$ is obtained, indicating that $q_\sigma = \delta(q, h)$ is the resulting state of M. Algorithm B is now run, with q_σ as an initial state. Whenever B attempts to "reset" from some state q', and do a membership query on a new string, the homing sequence h is executed, and if we are lucky, the output is again $\sigma = q'(h)$, and (since h is a homing sequence) the current state must again be q_σ, and the membership query may then be made. The problem with this approach is that σ might not be obtained upon execution of h, and instead some other output sequence σ' might be obtained. The solution is to run an independent copy B_σ of the algorithm B for each possible output sequence σ of the homing sequence. Whenever the homing sequence is executed, some string σ is output, and the DFA M returns to state q_σ, which is the initial state of the DFA that algorithm B_σ is attempting to infer. Thus, the next membership query of B_σ may be successfully executed. After each execution of h, at least one of the algorithms $\{B_\sigma\}$ makes progress towards completion. Since the number of distinct output sequences σ is at most $n = |Q|$, the algorithm makes no more than n times the number of equivalence and membership queries as does algorithm B.

Finally, it is shown how to *infer* a homing sequence h while at the same time inferring the unknown DFA M. A candidate homing sequence h is maintained (initially, $h = \lambda$). At any point during the run of any of the algorithms $\{B_\sigma\}$, it might be determined that h is inconsistent, i.e., some $q_1, q_2 \in Q$, and $x \in \Sigma^*$ are found such that $q_1(h) = q_2(h)$ but $\delta(q_1, hx) \neq \delta(q_2, hx)$. In this case, it is shown that hx is a "better approximation" to a homing sequence than h. At this point, all prior computation is discarded, and the entire algorithm is rerun with the new candidate homing sequence hx. It is shown that this can happen at most $n = |Q|$ times. Finally, it is possible that although h is inconsistent, this fact is not determined during the run of the algorithm. In this case, it is shown that a randomized strategy may be used to find an inconsistency of h with high probability.

5.2.2 Permutation automata

A *permutation DFA* is a DFA $M = (Q, \Sigma, \delta, q_0, F)$ such that for every $a \in \Sigma$, the function $\delta(\cdot, a)$ is a permutation of Q. Rivest and Schapire strengthen Theorem 27 to the case of inferring permutation automata.

Theorem 28 [51] *The class of permutation DFAs are* random-polynomial time *identifiable using experiments alone. In particular, there exists a randomized algorithm A such that on input of any $\delta > 0$, for any permutation DFA M of size n, in time polynomial in n and $\frac{1}{\delta}$, using experiments alone, A will output a DFA M' such that with probability at least $1 - \delta$, $L(M') = L(M)$.*

To prove Theorem 28, first consider DFAs such that the output at any state is the name of that state. Such a DFA is easily identified in polynomial time. Next it is shown how a homing sequence may be used to obtain a name for the current state: For permutation DFAs, a homing sequence h is also a *distinguishing sequence*, i.e., $q_1(h) = q_2(h)$ iff $q_1 = q_2$. Thus the identity of the current state q may be determined by executing h and observing the output sequence $q(h)$. At this point however, the current state has changed from q to $q' = \delta(q, h)$. How is the identity of the current state q (which is given by the string $q(h)$) obtained *without* first executing h? If h is executed from $q' = \delta(q, h)$, we may record in a "lookahead table" that if the sequence $q(h)$ is observed, then the result of applying h *once again* will be $q'(h)$. Thus the next time h is executed and $q(h)$ is output, the inference algorithm knows that the identity of the current state is given by the string $q'(h)$. Finally, it is shown that for permutation DFAs, a homing sequence may be obtained with probability $1 - \delta$ by randomly choosing any string that is sufficiently (although not more than polynomially) long.

Theorem 28 is especially interesting in light of our comments at the beginning of Section 5.1, that DFAs are not polynomial time identifiable from membership queries alone. Theorem 28 shows that for permutation automata, a randomized algorithm can achieve polynomial time identification (with high probability) using only a restricted type of membership query.

5.2.3 Diversity

The actual definitions of Rivest and Schapire generalize those described above, and are meant to model the situation of a robot (algorithm) attempting to infer its finite state environment. The robot may take basic actions (elements of Σ) which cause a state transition of the environment. There is no set F of accept states, but instead there is a finite set of predicates $P = \{p_1, \ldots, p_k\}$, such that for each i, $p_i : Q \to \{\text{true, false}\}$. For any $q \in Q$, the vector $\langle p_1(q), \ldots, p_k(q)\rangle$ may be viewed as the k-ary output of a Moore machine at the state q. The predicates correspond to the "sensations" that are available to the robot from a given state of the environment. In the protocol of Rivest and Schapire, the k outputs are not given at each state, but must be explicitly obtained by executing a *test*. A test is an element of $\Sigma^* \times P$. The value of a test $t = (w, p)$ from state q is denoted qt and defined by $qt = p(\delta(q, w))$. The goal of the inference algorithm is to eventually be able to predict the outcome of any test.

A different measure of the *size* of the DFA (environment) is introduced. Define two tests t_1 and t_2 to be *equivalent* iff for all $q \in Q$, $qt_1 = qt_2$. The *diversity* of the environment is the number of distinct equivalence classes of tests under this equivalence relation. It is shown that the diversity of a DFA is at least logarithmic, and at most exponential, in the number of states of the DFA. A DFA with many states but low diversity is one which is fairly "structured". Algorithms for inferring DFAs using diversity as a measure of size are presented, and theorems analogous to Theorem 27 and Theorem 28 are proven. These results subsume the results in [52] on the inference of "visible simple assignment automata". See [8, 50] for further discussion of the diversity measure.

5.3 Equivalence Queries with Ordered Counterexamples

Ibarra and Jiang [33] present an interesting theorem that contrasts nicely with the negative result of Theorem 5. An *LF-equivalence query* is an equivalence query, except if the queried DFA M' is not equivalent to the target DFA M, then the counterexample supplied is the *lexicographically first* element of $L(M) \oplus L(M')$ according to some standard enumeration of Σ^*. Similarly, define an *S-equivalence* query to be one for which every counterexample is a *shortest* possible counterexample.

Theorem 29 [33] *The class of DFAs is polynomial time identifiable using LF-equivalence queries. The class of DFAs is* not *polynomial time identifiable using S-equivalence queries.*

They also give general reductions showing that in the model of polynomial time identification with equivalence queries, identifying the class of DFAs accepting finite languages is no easier than identifying the full class of DFAs. These general reductions play an important role in proving Theorem 29.

Porat and Feldman [46] also consider the problem of identifying DFAs from ordered examples. They present an algorithm that identifies the class of DFAs in the limit from a lexicographic presentation. While their algorithm does not satisfy Definition 3, it is shown to have a number of desirable properties that are motivated from applications to "connectionist" learning. For example, the total amount of space used by the algorithm at any point is bounded by the size of the canonical acceptor. The algorithm is shown to have polynomial update time in an *amortized* sense.

5.4 Learning Subclasses of Regular Languages

A *commutative* regular language is a regular language L such that for all w, $w \in L \Rightarrow \pi(w) \in L$, where $\pi(w)$ is any permutation of the characters of w. A commutative DFA is a DFA that accepts a commutative regular language. Recall that a zero-reversible DFA has at most one final state, and for any state q and symbol a there is at most one state p such that $\delta(p, a) = q$. Theorem 15 in fact holds for zero-reversible commutative DFAs, indicating that if this restricted subclass of DFAs are PAC-identifiable, then the inference algorithm would necessarily have to output very large DFAs or NFAs. However, by considering a different hypothesis class for these languages, it can be shown that this class is polynomially approximately predictable.

Let Z denote the set of integers. Helmbold, Sloan, and Warmuth [30, 31] consider the problem of inferring the class of subsets of Z^k that are closed under addition and subtraction (i.e., the class of submodules of the free Z-module of rank k). They show that their algorithm for identifying submodules of Z^k can be used to predict zero-reversible commutative regular languages:

Theorem 30 [30] *The class of zero-reversible commutative DFAs is polynomially approximately predictable.*

The nonapproximability of the minimum consistent DFA problem (Theorem 15) for this subclass of DFAs is overcome by representing the hypothesis not as a DFA, but as a submodule that concisely encodes the randomly generated examples.

Abe [2] considers the related problem of inferring *semi-linear sets*. A semi-linear set of dimension d is the union of a finite number of *linear sets* of dimension d. A linear set of dimension d is a set of the form $\{\vec{v} \in \mathrm{N}^d : (\exists c_1, \ldots, c_k \in \mathrm{N})\ \vec{v} = v_0 + \sum_{i=1}^{k} c_i \vec{v}_i\}$ for some fixed generating vectors $\vec{v}_0, \ldots, \vec{v}_k \in \mathrm{N}^d$. Semi-linear sets of d dimensions are exactly the sets of "letter counts" (or Parikh-images) of regular sets over an alphabet of size d.

There are two measures of size that we may associate with the representation of a semi-linear set of d dimensions. The *unary size* is the sum of the *values* of all components of each generating vector.

This corresponds roughly to the total number of characters needed to write the representation where integers are encoded in unary. The *binary size* is the sum of the *logarithms* of all components of each generating vector, corresponding to the number of bits needed to encode the vectors in binary.

Abe shows that, according to the unary measure, semi-linear sets of dimension 1 and 2 are PAC-identifiable. It is shown that polynomial approximate prediction of semi-linear sets of varying dimension d is as hard as polynomial approximate prediction of DNF formulas – which remains an intriguing open problem. Similarly, DNF prediction also reduces to the prediction of semi-linear sets of 2 dimensions when the binary size measure is used.

Finally, at the time of this writing Abe claims to have extended his results to show that (not necessarily zero-reversible) commutative DFAs over arbitrary alphabets are polynomially predictable. See [1, 2] for further details and discussion.

As a final example of a positive result for inference of restricted types of DFAs, consider the k-bounded regular languages: A regular language L is *k-bounded* if there exists strings $w_1, \ldots, w_k$ such that $L \subseteq w_1^* w_2^* \cdots w_k^*$. A DFA is k-bounded if it accepts a k-bounded regular language. Ibarra and Jiang [33] prove the following theorem.

Theorem 31 *The class of k-bounded DFAs is polynomial time identifiable using equivalence queries.*

6 Conclusion

The negative results outlined in Section 3 and Section 4 show, based on various complexity-theoretic assumptions, that the problem of inferring DFAs from examples alone is intractable according to a number of reasonable definitions of polynomial time inference. On the other hand, Section 5 outlines a number of interesting algorithms which efficiently infer DFAs, and restricted classes of DFAs, assuming that the inference algorithm may take a more active role in obtaining additional information about the target DFA.

Collectively, the results surveyed provide an introduction to many issues that are prevalent in the investigation of feasible inference in other domains. Several main themes deserve mention: (1) The choice of hypothesis space plays a key role in determining whether a class of languages may be inferred. (2) Similarly, the ability of the inference algorithm to make membership queries, or actively execute some other type of experiment on the object of inference, may allow for feasible inference when "passively obtained" examples alone are not sufficient. (3) The ability to compress the set of examples seen by finding reasonably small hypotheses is sufficient for PAC-identification, and in some cases, necessary. Thus many PAC-identification problems are equivalent to finding weak approximation algorithms for associated combinatorial optimization problems.

This survey is not meant to be exhaustive, although an effort was made to include as many recent theorems (and relevant past theorems) on DFA inference as possible considering the space limitations. It would be interesting to compare many of the results discussed here with heuristic approaches that have been presented in earlier work in more practical settings. Many of these approaches are discussed in the survey by Angluin and Smith [12].

Hopefully, we have convinced the reader that the general area of DFA inference is rich with interesting problems and solutions; we close with a list of open problems for further consideration.

1. Can it be shown that DFAs are not PAC-identifiable based only on the assumption that $RP \neq NP$? Such a result could be proved by showing that there is no Occam algorithm for DFAs (in terms of DFAs) unless $RP \neq NP$. Stronger still, can the nonpredictability result of Theorem 19 be strengthened by replacing the cryptographic assumptions with only the assumption that $RP \neq NP$? This would be the strongest negative result possible in the PAC-identification model.

2. Are DFAs PAC-identifiable (or at least polynomially approximately predictable) if examples are drawn from a uniform distribution, or some other known simple distribution?

3. Can matching lower and upper bounds be found on the number of examples, queries, and other types of information needed to infer DFAs according to the models presented? Such bounds might be obtained by computing the VC-dimension [17] of the classes considered, and applying techniques used for other domains [22].

4. Are (not necessarily zero-reversible) commutative DFAs (or NFAs) PAC-identifiable?

5. Is the class of Boolean formulas polynomially approximately predictable if membership queries are also allowed? Perhaps a method will be found by enhancing the reduction showing that Boolean formulas are no harder to predict than DFAs from examples alone. How about the easier problem of predicting DNF formulas with membership queries allowed?

Acknowledgements

I thank Dana Angluin and Manfred Warmuth for helpful and enjoyable discussions during the preparation of this manuscript. Also, it is a pleasure to thank all of my colleagues who have made this presentation possible by their fine work in this area. It has been an exciting time to be involved in research in the field of computational learning theory; I eagerly look forward to collective progress toward the solutions of many more intriguing questions.

References

[1] N. Abe. Polynomial learnability as a formal model of natural language acquisition. 1989. Ph.D. Thesis, in preparation, Department of Computer and Information Science, University of Pennsylvania.

[2] N. Abe. Polynomial learnability of semilinear sets. In *Proceedings of the 1989 Workshop on Computational Learning Theory*, Morgan Kaufmann, San Mateo, CA, August 1989.

[3] W. Alexi, B. Chor, O. Goldreich, and C. P. Schnorr. RSA and Rabin functions: certain parts are as hard as the whole. *S.I.A.M. Journal on Computing*, 17(2):194–209, 1988.

[4] D. Angluin. Equivalence queries and approximate fingerprints. In *Proceedings of the 1989 Workshop on Computational Learning Theory*, Morgan Kaufmann, San Mateo, CA, August 1989.

[5] D. Angluin. Inference of reversible languages. *J. ACM*, 29(3):741–765, 1982.

[6] D. Angluin. Learning regular sets from queries and counterexamples. *Information and Computation*, 75(2):87–106, 1987.

[7] D. Angluin. *Negative Results for Equivalence Queries.* Technical Report YALEU/DCS/RR-648, Department of Computer Science, Yale University, September 1988.

[8] D. Angluin. A note on diversity. December 1987. Unpublished manuscript.

[9] D. Angluin. A note on the number of queries needed to identify regular languages. *Information and Computation*, 51:76–87, 1981.

[10] D. Angluin. On the complexity of minimum inference of regular sets. *Inform. Contr.*, 39(3):337–350, 1978.

[11] D. Angluin. Queries and concept learning. *Machine Learning*, 2:319–342, 1987.

[12] D. Angluin and C. H. Smith. Inductive inference: theory and methods. *Computing Surveys*, 15(3):237–269, 1983.

[13] J. M. Barzdin. *Prognostication of automata and functions*, pages 81–84. Elsevier North-Holland, New York, 1972.

[14] J. M. Barzdin and R. V. Freivald. On the prediction of general recursive functions. *Soviet Mathematics Doklady*, 13:1224–1228, 1972.

[15] P. Berman and R. Roos. Learning one-counter languages in polynomial time. In *Proceedings of the 28th Annual IEEE Symposium on Foundations of Computer Science*, IEEE Computer Society Press, Washington, D.C., October 1987.

[16] L. Blum and M. Blum. Toward a mathematical theory of inductive inference. *Inform. Contr.*, 28:125–155, 1975.

[17] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. Warmuth. *Learnability and the Vapnik-Chervonenkis Dimension.* Technical Report UCSC-CRL-87-20, Department of Computer and Information Sciences, University of California, Santa Cruz, November 1987. To appear, *J. ACM.*

[18] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. Warmuth. Occam's razor. *Inform. Process. Letters*, 24:377–380, 1987.

[19] R. A. Board and L. Pitt. Sufficent conditions for the necessity of Occam algorithms. 1989. Manuscript in preparation.

[20] J. Case and C. Smith. Comparison of identification criteria for machine inductive inference. *Theoretical Computer Science*, 25:193–220, 1983.

[21] R. Daley and C. H. Smith. On the complexity of inductive inference. *Information and Control*, 69:12–40, 1986.

[22] A. Ehrenfeucht, D. Haussler, M. Kearns, and L. G. Valiant. A general lower bound on the number of examples needed for learning. In *Proceedings of the 1988 Workshop on Computational Learning Theory*, pages 139–154, Morgan Kaufmann, San Mateo, CA, August 1988.

[23] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness.* W. H. Freeman, San Francisco, California, 1979.

[24] J. Gill. Probabilistic Turing machines. *SIAM J. Computing*, 6(4):675–695, 1977.

[25] E. M. Gold. Complexity of automaton identification from given data. *Inform. Contr.*, 37:302–320, 1978.

[26] E. M. Gold. Language identification in the limit. *Inform. Contr.*, 10:447–474, 1967.

[27] E. M. Gold. System identification via state characterization. *Automatica*, 8:621–636, 1972.

[28] D. Haussler. *Learning Conjunctive Concepts in Structural Domains.* Technical Report UCSC-CRL-87-01, Department of Computer and Information Sciences, University of California, Santa Cruz, February 1987. To appear in *Machine Learning.*

[29] D. Haussler, M. Kearns, N. Littlestone, and M. K. Warmuth. Equivalence of models for polynomial learnability. In *Proceedings of the 1988 Workshop on Computational Learning Theory*, pages 42–55, Morgan Kaufmann, San Mateo, CA, August 1988.

[30] D. Helmbold, R. Sloan, and M. K. Warmuth. *Learning Submodules and Reversible Commutative Regular Languages.* 1989. Technical report to appear; Department of Computer and Information Sciences, University of California at Santa Cruz.

[31] D. Helmbold, R. Sloan, and M. K. Warmuth. Learning nested differences of intersection closed concept classes. In *Proceedings of the 1989 Workshop on Computational Learning Theory*, Morgan Kaufmann, San Mateo, CA, August 1989.

[32] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, Reading, Massachussetts, 1979.

[33] O. H. Ibarra and T. Jiang. Learning regular languages from counterexamples. In *Proceedings of the 1988 Workshop on Computational Learning Theory*, pages 371–385, Morgan Kaufmann, San Mateo, CA, August 1988.

[34] M. Kearns and L. G. Valiant. Cryptographic limitations on learning Boolean formulae and finite automata. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, pages 433–444, Assoc. Comp. Mach., New York, May 1989.

[35] M. Li and U. Vazirani. On the learnability of finite automata. In *Proceedings of the 1988 Workshop on Computational Learning Theory*, pages 359–370, Morgan Kaufmann, San Mateo, California, August 1988.

[36] M. Li and P. Vitanyi. Inductive reasoning and Kolmogorov complexity. In *Proceedings of the 4rd Annual IEEE Conference on Structure in Complexity Theory*, IEEE Computer Society Press, Washington, D.C., June 1989.

[37] N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2:285–318, 1987.

[38] E. F. Moore. *Gedanken-experiments on sequential machines*, pages 129–153. Princeton University Press, Princeton, NJ, 1956.

[39] I. Niven and H. S. Zuckerman. *An Introduction to the Theory of Numbers.* John Wiley and Sons, Inc., New York, 1972.

[40] William of Occam. "Entities should not be multiplied unnecessarily". c. 1320. Comment regarding superfluous elaboration.

[41] L. Pitt and L. G. Valiant. Computational limitations on learning from examples. *J. ACM*, 35(4):965–984, 1988.

[42] L. Pitt and M. K. Warmuth. *The Minimum Consistent DFA Problem Cannot be Approximated within any Polynomial.* Technical Report UIUCDCS-R-89-1499, University of Illinois at Urbana-Champaign, February 1989. A preliminary version appears in the 21st annual ACM Symposium on Theory of Computing, May, 1989.

[43] L. Pitt and M. K. Warmuth. *Prediction Preserving Reducibility.* Technical Report UCSC-CRL-88-26, University of California, Santa Cruz, November 1988. Preliminary version appeared in Proceedings of the 3rd Annual IEEE Conference on Structure in Complexity Theory, pp. 60-69, June, 1988.

[44] K. M. Podnieks. Comparing various concepts of function prediction, part 1. *Latv. Gosudarst. Univ Uch. Zapiski*, 210:68–81, 1974. (in Russian).

[45] K. M. Podnieks. Comparing various concepts of function prediction, part 2. *Latv. Gosudarst. Univ Uch. Zapiski*, 233:33–44, 1975. (in Russian).

[46] S. Porat and J. A. Feldman. Learning automata from ordered examples. In *Proceedings of the 1988 Workshop on Computational Learning Theory*, pages 386–396, Morgan Kaufmann, San Mateo, CA, August 1988.

[47] M. O. Rabin. *Digital Signatures and Public Key Functions as Intractable as Factoring.* Technical Report TM-212, Laboratory for Computer Science, M.I.T., 1979.

[48] J. Rissanen. *Minimum Description Length Principle*, pages 523–527. Wiley, New York, 1985.

[49] R. L. Rivest. Learning decision lists. *Machine Learning*, 2:229–246, 1987.

[50] R. L. Rivest and R. E. Schapire. Diversity-based inference of finite automata. In *Proceedings of the 28th Annual IEEE Symposium on Foundations of Computer Science*, pages 296–304, IEEE Computer Society Press, Washington, D.C., October 1987.

[51] R. L. Rivest and R. E. Schapire. Inference of finite automata using homing sequences. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, pages 411–420, Assoc. Comp. Mach., New York, 1989.

[52] R. L. Rivest and R. E. Schapire. Inference of visible simple assignment automata with planned experiments. October 1987. Unpublished manuscript, MIT Lab. for Computer Science, Cambridge, MA 02139.

[53] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public key cryptosytems. *Communcations of the A.C.M.*, 21(2):120–126, 1978.

[54] Y. Sakakibara. Learning context-free grammars from structural data in polynomial time. In *Proceedings of the 1988 Workshop on Computational Learning Theory*, pages 330–344, Morgan Kaufmann, San Mateo, CA, August 1988.

[55] R. Schapire. The strength of weak learnability. In *Proceedings of the 1989 Workshop on Computational Learning Theory*, Morgan Kaufmann, San Mateo, CA, August 1989.

[56] R. Solomonoff. A formal theory of inductive inference, parts 1 and 2. *Information and Control*, 7:1–22 and 224–254, 1964.

[57] B. A. Trakhtenbrot and Ya. M. Barzdin. *Finite Automata.* North-Holland, Amsterdam, 1973. pp. 98-99.

[58] L. G. Valiant. A theory of the learnable. *Comm. Assoc. Comp. Mach.*, 27(11):1134–1142, 1984.

[59] C. K. Yap. *Theory of Complexity Classes.* Oxford University Press, Oxford. To appear.