Machine Learning, 5, 121-150 (1990)

Negative Results for Equivalence Queries

DANA ANGLUIN (ANGLUIN@CS.YALE.EDU)
Department of Computer Science, Yale University, P.O. Box 2158, New Haven, CT 06520, USA

Abstract. We consider the problem of exact identification of classes of concepts using only equivalence queries. We define a combinatorial property, *approximate fingerprints*, of classes of concepts and show that no class with this property can be exactly identified in polynomial time using only equivalence queries. As applications of this general theorem, we show that there is no polynomial time algorithm using only equivalence queries that exactly identifies deterministic or nondeterministic finite state acceptors, context free grammars, or disjunctive or conjunctive normal form boolean formulas.

Keywords. Concept learning, queries.

1. Introduction

Consider the problem of exactly identifying deterministic finite state acceptors (dfas) using various types of queries. There is a polynomial time solution that uses both equivalence and membership queries (Angluin, 1987). An easy argument based on *passwords* shows that there is no polynomial time solution using only membership queries even if we are given a bound on the size of the unknown dfa (Angluin, 1982b). However, it was an open question whether there might be a polynomial time solution using equivalence queries only. In this paper we answer this question in the negative.

We use similar techniques to show that there is no polynomial time algorithm that uses only equivalence queries and exactly identifies nondeterministic finite state acceptors (nfas), context free grammars (cfgs), or disjunctive normal form (DNF) or conjunctive normal form (CNF) formulas. A similar result holds also for read-once or μ-formulas (Angluin, Hellerstein, & Karpinski, 1989).

These results are not conditioned on unproven complexity theoretic assumptions, but they depend critically on the form of the representation of hypotheses as inputs to the equivalence queries. They also do not imply negative results for the polynomial time probably approximately correct identifiability or the polynomial time predictability of these classes.

2. Equivalence queries

We consider a learner faced with the task of identifying an unknown concept chosen from some specified class C of concepts. A particular method of representing concepts from C is also specified. For example, we might consider the class of regular sets over a fixed alphabet represented using deterministic finite state acceptors. The goal of the learner is exact identification, that is, to output a representation for (exactly) the unknown concept. The information available to the learner about the unknown concept is the answers to

equivalence queries. That is, the learner may propose a representation h (in the specified system) of a concept in C and the reply is either *yes* or *no* and a counterexample. The reply *yes* means that h represents the unknown concept. Otherwise the answer is *no* and an arbitrary element that is classified differently by h and the unknown concept.

For each k, there is a polynomial time learning algorithm using only equivalence queries that exactly identifies all the k-CNF formulas (propositional formulas in conjunctive normal form with at most k literals per clause) (Angluin, 1988a). A polynomial time learning algorithm using only equivalence queries in a *reasonable* representation system can be transformed into a polynomial time probably approximately correct identification algorithm in the distribution-free sense of Valiant, but not necessarily vice versa (Angluin, 1988a). In this transformation, equivalence queries are replaced by stochastic equivalence testing, which either supplies a counterexample to the proposed hypothesis, or guarantees that with high probability the proposed hypothesis is *close enough* to the unknown concept to be accepted.

3. The basic idea of approximate fingerprints

We present a general technique for proving that certain kinds of hypothesis spaces cannot have polynomial time exact identification algorithms using only equivalence queries. The basic idea of the method is to exhibit a phenomenon we term *approximate fingerprints* in the class H of hypotheses being considered. That is, we exhibit some target set T of hypotheses such that for each hypothesis h from H, there is an example w_h such that *very few* hypotheses in T classify w_h the same way that h does. Although the behavior of h in classifying w_h does not uniquely identify a hypothesis in T, it narrows the possibilities to a superpolynomially small fraction of T.

Approximate fingerprints can be used by an adversary to generate *uninformative* counterexamples to equivalence queries—if a hypothesis h is proposed by the learner, then answering the equivalence query with *no* and the counterexample w_h means that the adversary has eliminated h and only a superpolynomially small fraction of the target set T as candidates for the correct hypothesis, namely those that agree with h in its classification of the example w_h. This can be repeated by the adversary a superpolynomial number of times without exhausting the set of consistent candidates. Thus, the learner cannot be certain of the correct hypothesis in T after only polynomially many equivalence queries.

This property can be viewed as a generalization of the following example considered by Angluin (1988a). Let the class C_n consist of 2^n concepts, each consisting of a single string of n bits. For each string w of n bits, the hypothesis $\{w\}$ has as its approximate fingerprint the string w. In this case, $\{w\}$ is the only hypothesis in C_n that classifies w as a positive example. Hence, an adversary that always gives w as the counterexample to the hypothesis $\{w\}$ can force the learner to make $2^n - 1$ equivalence queries in the worst case to achieve exact identification of every concept in C_n. What we show below is that a more general form of this simple phenomenon afflicts the richer hypothesis classes of dfas, nfas, cfgs, and DNF and CNF formulas.

Note that the difficulty in this simple case is removed if the empty set is permitted as a hypothesis. Then the learner proposes the empty set, and is either correct, or the only

possible counterexample is a string w that *gives away* the identity of the unknown concept. In this case, the hypothesis of the empty set is the *majority vote* of the target class of singletons. That is, it contains a string w if and only if the majority of the concepts in the target class contain w.

The target classes of concepts used to demonstrate the approximate fingerprint property below have a related structure. Each concept in the target class has a representation of polynomial size; the concept that consists of the majority vote of the target concepts does not. Angluin (1988a) describes some other relations between the idea of a majority vote concept and equivalence queries.

4. Preliminaries

4.1. Representations of concepts

We are interested in representations of concepts, which we take to be specified by a quadruple

$$\mathbf{R} = \langle \Sigma, \Delta, R, \mu \rangle$$

where Σ and Δ are finite alphabets, R is a subset of Δ^*, and μ is a map from R to subsets of Σ^*. Σ is the alphabet of examples, and a *concept* is any subset of Σ^*. R is the set of representations, and μ is the map that specifies which concept is represented by a given representation. The class of concepts represented by $\mathbf{R}$ is just $\{\mu(r) : r \in R\}$. For any concept, c, χ_c denotes the characteristic function of c.

For dfas the strings in R represent deterministic finite state acceptors over the input alphabet Σ in some straightforward way. In particular, we assume that there exists a polynomial $p(s)$ such that every dfa over Σ with at most s states can be represented by a string in R of length at most $p(s)$. Moreover, we assume that for every string $r \in R$, the number of states of the dfa represented by r is at most $|r|$.

Then for each $r \in R$, $\mu(r)$ denotes the set of strings accepted by the dfa represented by r. The class of concepts represented in this case is the class of all regular sets over the alphabet Σ. We assume a similar representation for nfas.

For cfgs, we may assume that each grammar is in Chomsky normal form, since there is a polynomial time algorithm to convert an arbitrary context free grammar into Chomsky normal form. The *size* of a Chomsky normal form grammar is the number of nonterminals it contains. In this case, we assume that the representation is such that there is a polynomial $p(s)$ such that every Chomsky normal form grammar over Σ of size at most s can be represented by a string in R of length at most $p(s)$. Also, we assume that for every $r \in R$, the Chomsky normal form grammar represented by r has size at most $|r|$.

For DNF formulas we assume that $\Sigma = \{0, 1\}$. Each $r \in R$ will represent a DNF formula over the variables $X_1, X_2, \ldots$, together with an integer n such that if X_i occurs in the formula then $n \geq i$. The value of $\mu(r)$ is the set of all binary strings w of length n such that the formula represented by r is assigned 1 by the assignment $X_i = w[i]$ for all $i = 1, \ldots, n$. We assume that there exists a polynomial $p(n)$ such that every DNF formula over V_n of at most n terms can be represented by a string from R of length at most $p(n)$. We also assume that for every string $r \in R$, the DNF formula represented by r has at most $|r|$ terms. An analogous representation is assumed for CNF formulas.

4.2. Exact identification with equivalence queries

Fix a representation of concepts $\mathbf{R} = \langle \Sigma, \Delta, R, \mu \rangle$. For the exact identification problem there is an unknown concept c_*, and information may be gathered about c_* by asking equivalence queries. The input to an equivalence query is any $r \in R$ and the response is *yes* if $\mu(r) = c_*$ and *no* otherwise. In addition to the answer *no*, a *counterexample* is supplied, that is, a string $w \in \Sigma^*$ such that $w \in c_* \oplus \mu(r)$, where $A \oplus B$ denotes the symmetric difference of the sets A and B. Thus, w is a string classified differently by the concepts $\mu(r)$ and c_* and constitutes a counterexample to the hypothesis that r represents the unknown concept. The choice of a counterexample is assumed to be arbitrary.

A deterministic algorithm A exactly identifies $\mathbf{R}$ using only equivalence queries if and only if for every c_* represented by $\mathbf{R}$, when A is run with an oracle for equivalence queries for c_*, A eventually halts and outputs some r such that $\mu(r) = c_*$. Such an algorithm runs in polynomial time if and only if there is a polynomial $p(l, m)$ such that for every c_* represented by $\mathbf{R}$, if l is the minimum length of any $r \in R$ such that $\mu(r) = c_*$ then when A is run with an oracle for equivalence queries for c_* at any point in the run the time used by A is bounded by $p(l, m)$, where m is the maximum length of any counterexample returned by equivalence queries so far in the run, or $m = 0$ if no equivalence queries have been used. (Note: this definition corrects a loophole in the definition used in the paper (Angluin, 1987), which permitted a degenerate solution with equivalence queries (Angluin, 1988b).

4.3. Approximate fingerprints

Let T denote a class of concepts, w a string from Σ^*, b an element of $\{0, 1\}$, and $\alpha > 0$ a real number. We say that the pair $\langle w, b \rangle$ is an *α-approximate fingerprint with respect to T* provided that

$$|\{c \in T : \chi_c(w) = b\}| < \alpha |T|.$$

That is, the number of concepts in T that agree with the classification b of w is strictly less than the fraction α of the total number of concepts in T. The set $\{c \in T : \chi_c(w) = b\}$ may be empty, if no concept in T agrees with the classification b of w.

A *sequence of concept classes* is a sequence

$$C_1, C_2, C_3, \ldots$$

such that each C_n is a class of concepts. Such a sequence is *bounded by $f(n)$ with respect to* a given representation $\mathbf{R}$ if for all sufficiently large n and every $c \in C_n$, there exists a representation $r \in R$ such that $\mu(r) = c$ and $|r| \leq f(n)$. That is, for all sufficiently large n, every concept in C_n has a representation in R of length at most $f(n)$. A representation of concepts $\mathbf{R}$ is said to have the *approximate fingerprint property* if and only if there exist positive nondecreasing polynomials $p_1(n)$ and $p_2(n)$ such that for every positive nondecreasing polynomial $q(n)$ there exists a sequence of concept classes $T_1, T_2, T_3, \ldots$ bounded by $p_1(n)$

with respect to **R** such that for all sufficiently large n, T_n contains at least two concepts and if $r \in R$ and $|r| \leq q(n)$ then there exists a string $w \in \Sigma^*$ of length at most $p_2(n)$ such that $\langle w, \chi_{\mu(r)}(w)\rangle$ is a $1/q(n)$-approximate fingerprint with respect to T_n. That is,

$$|\{c \in T_n : \chi_c(w) = \chi_{\mu(r)}(w)\}| < |T_n|/q(n).$$

4.4. The basic theorem for approximate fingerprints

This theorem and its proof formalize the intuitive description given above of how an adversary can use approximate fingerprints to thwart a learning algorithm that uses only equivalence queries.

THEOREM 1. Let **R** be a representation of concepts with the approximate fingerprint property. Then there is no polynomial time algorithm for exact identification of **R** using only equivalence queries.

Proof. Suppose to the contrary that A is an algorithm that exactly identifies **R** using only equivalence queries, and A runs in time bounded by the polynomial $p(l, m)$. Without loss of generality we may assume that $p(l, m)$ is positive and nondecreasing in both arguments.

Since **R** has the approximate fingerprint property, we may assume that there are positive nondecreasing polynomials $p_1(n)$ and $p_2(n)$ satisfying the definitions above. Let $q(n) = 2p(p_1(n), p_2(n))$.

Then there exists a sequence of concept classes $T_1, T_2, \ldots$ bounded by $p_1(n)$ such that for all sufficiently large n, T_n contains at least two concepts and for any $r \in R$ with $|r| \leq q(n)$ there exists a string $w \in \Sigma^*$ such that $|w| \leq p_2(n)$ and $\langle w, \chi_{\mu(r)}(w)\rangle$ is a $1/q(n)$-approximate fingerprint with respect to T_n. Suppose n is sufficiently large that this holds, and that every concept in T_n has a representation of length at most $p_1(n)$.

Consider the adversary that answers an equivalence query with input $r \in R$ as follows. If $|r| \leq q(n)$ then the reply is *no* and the counterexample is some string $w \in \Sigma^*$ such that $|w| \leq p_2(n)$ and $\langle w, \chi_{\mu(r)}(w)\rangle$ is a $1/q(n)$-approximate fingerprint with respect to T_n. If $|r| > q(n)$, then the adversary answers *yes*.

Run the algorithm A with this adversary answering each equivalence query. We claim that for each $1 \leq i \leq q(n)/2$, at least i equivalence queries are eventually made, and after i equivalence queries have been answered, more than $(1 - i/q(n))|T_n|$ concepts in T_n are consistent with the answers given so far, and all the counterexamples given so far are of length at most $p_2(n)$. We prove this by induction on i.

Since A exactly identifies **R**, it must identify each concept in T_n, and since each concept in T_n can be represented by an $r \in R$ of length at most $p_1(n)$, the running time of A at each point must be bounded by $p(p_1(n), m)$, where m is the maximum length of any counterexample seen to that point, or $m = 0$ if no counterexamples have been seen.

Since $|T_n| \geq 2$, A must ask at least one equivalence query. Consider the first equivalence query asked by A, say with r. Clearly, $|r| \leq p(p_1(n), 0) \leq q(n)$, so the adversary answers with *no* and a counterexample w such that $|w| \leq p_2(n)$ and

$$|\{c \in T_n : \chi_c(w) = \chi_{\mu(r)}(w)\}| < |T_n|/q(n).$$

The number of concepts in T_n consistent with this answer is greater than $(1 - 1/q(n))|T_n|$.

Suppose the claim is true for some $1 \leq i < q(n)/2$. Since more than $(1 - i/q(n))|T_n| \geq (1/2)|T_n|$ concepts in T_n are consistent with the answers given so far, and since $|T_n| \geq 2$, at least two distinct concepts in T_n are consistent with the answers given so far. Hence, A must ask another equivalence query, say with input $r \in R$. Since all the counterexamples so far have been of length at most $p_2(n)$, the running time of A, and therefore the length of r, must be bounded by $p(p_1(n), p_2(n)) \leq q(n)$. Thus, the adversary answers with *no* and a counterexample w of length at most $p_2(n)$ such that

$$|\{c \in T_n : \chi_c(w) = \chi_{\mu(r)}(w)\}| < |T_n|/q(n).$$

This answer is inconsistent with fewer than $|T_n|/q(n)$ concepts in T_n, so no more than $(1 - (i + 1)/q(n))|T_n|$ concepts in T_n are consistent with the replies to the first $i + 1$ queries.

This completes the induction. Now consider the situation after the first $q(n)/2$ queries have been answered. More than $(1/2)|T_n|$ concepts in T_n are consistent with all the replies that have been given, and since $|T_n| \geq 2$, this means that at least two distinct concepts from T_n are consistent with all the replies so far. Moreover, all the counterexamples that have been given are of length at most $p_2(n)$, which means that the running time permitted to A is bounded by $p(p_1(n), p_2(n)) = q(n)/2$. Hence, if A does not ask another query, it will give an incorrect answer for at least one concept in T_n. But if A asks another query, it will exceed its time bound. This contradiction shows that no such algorithm A is possible. Q.E.D.

Note that the running time of A is only used to give a bound on the number of equivalence queries and the length of the input to each. Thus in fact there is no algorithm to exactly identify **R** that uses a polynomially bounded number of equivalence queries of polynomially bounded size. In subsequent sections we demonstrate the approximate fingerprint property for some concrete classes of interest.

5. DNF formulas have approximate fingerprints

We concentrate on DNF formulas; the results for CNF formulas follow by duality. The key property of DNF formulas for our purposes is that for every DNF formula ϕ, there is an assignment with *few* 1's that satisfies ϕ or an assignment with *few* 0's that falsifies ϕ. Moreover, in a certain target sequence of concept classes, not many formulas share the value of ϕ on this assignment. This assignment serves as an approximate fingerprint.

5.1. Preliminaries for DNF formulas

If n is a positive integer, let V_n denote the set of variables $X_1, \ldots, X_n$. The *literals* over V_n is the set of all elements of V_n and their negations. The negation of X_i is denoted $\neg X_i$.

A *term* over V_n is a set of literals over V_n that does not contain both X_i and $\neg X_i$ for any i. A term is *monotone* if it contains no negations of variables. The *size* of a term τ is the cardinality of τ.

A DNF formula over V_n is a set of terms over V_n. A DNF formula is *monotone* if it contains only monotone terms. Note that we do not distinguish between formulas that can

be obtained from each other by commuting elements within a sum or product. The *size* of a DNF formula ϕ, denoted *size*(ϕ), is the sum of the sizes of the terms it contains.

An *assignment* a to V_n is a mapping of V_n to $\{0, 1\}$. It is extended to literals, terms, and DNF formulas in the usual way. Thus, $a(\neg X_i) = 1 - a(X_i)$. If τ is a term, then $\alpha(\tau) = \wedge_{L \in \tau} a(L)$. If ϕ is a DNF formula, $a(\phi) = \vee_{\tau \in \phi} a(\tau)$. Two DNF formulas ϕ and ψ over V_n are *logically equivalent*, denoted $\phi \equiv \psi$, if and only if for all assignments a to V_n,

$$a(\phi) = a(\psi).$$

Let n, t, and s be positive integers. Let $T(n, s)$ denote the set of all monotone terms over V_n of size s. Then

$$|T(n, s)| = \binom{n}{s}.$$

Let $M(n, t, s)$ denote the set of all monotone DNF formulas over the variables V_n that have t distinct terms, each chosen from $T(n, s)$. Then

$$|M(n, t, s)| = \binom{|T(n, s)|}{t}.$$

For example, one formula in $M(4, 3, 2)$ is

$$X_1X_3 + X_2X_3 + X_2X_4.$$

Note that the size of any formula from $M(n, t, s)$ is st. All the formulas in $M(n, t, s)$ are logically distinct.

5.2. *Counting falsified DNF formulas*

One basic tool of our analysis is an estimate of the fraction of formulas in $M(n, t, s)$ that are falsified by a given assignment a over V_n. Let $p(a)$ denote the number of 1's in a, that is,

$$p(a) = |\{i : a(X_i) = 1\}|.$$

Let $f_0[n, t, s](a)$ denote the fraction of formulas in $M(n, t, s)$ that are assigned 0 by a. That is,

$$f_0[n, t, s](a) = |\{\phi \in M(n, t, s) : a(\phi) = 0\}|/|M(n, t, s)|.$$

Also, let $f_1[n, t, s](a)$ denote the fraction of formulas in $M(n, t, s)$ that are assigned 1 by a, so $f_1[n, t, s](a) = 1 - f_0[n, t, s](a)$.

In this section we shall use an in-line notation for binomial coefficients. For all positive integers n and m,

$$C(n, m) = \binom{n}{m}.$$

Two simple lemmas on binomial coefficients follow.

LEMMA 2. For all nonnegative integers i, j, k such that $i \leq j \leq k$,

$$((j - i)/k)^i \leq C(j, i)/C(k, i) \leq (j/k)^i.$$

LEMMA 3. For all positive integers n and s such that $s \leq n$, we have $C(n, s) \geq (n/s)^s$.

The following lemma supplies the bounds we need on $f_0[n, t, s](a)$.

LEMMA 4. Let n, t, and s be positive integers such that $t \leq n$ and $s \leq n$. Let a be an assignment to V_n. Then

$$f_0[n, t, s](a) \leq (1 - ((p(a) - s)/n)^s)^t,$$

and

$$f_0[n, t, s](a) \geq (1 - (p(a)/n)^s - t(s/n)^s)^t.$$

Proof. A term in $T(n, s)$ is assigned 1 by a if and only if each of its variables is chosen from the set of $p(a)$ variables assigned 1 by a. Thus the number of terms in $T(n, s)$ assigned 1 by a is $C(p(a), s)$. Hence the number of terms in $T(n, s)$ assigned 0 by a is $C(n, s) - C(p(a), s)$. A formula in $M(n, t, s)$ is assigned 0 by a if and only if each of its terms is chosen from the set of terms in $T(n, s)$ assigned 0 by a, so the number of formulas in $M(n, t, s)$ assigned 0 by a is

$$C(C(n, s) - C(p(a), s), t).$$

Recalling that $|M(n, t, s)| = C(C(n, s), t)$, we have

$$f_0[n, t, s](a) = C(C(n, s) - C(p(a), s), t)/C(C(n, s), t).$$

Applying Lemma 2,

$$f_0[n, t, s](a) \leq ((C(n, s) - C(p(a), s))/C(n, s))^t,$$

and

$$f_0[n, t, s](a) \geq ((C(n, s) - C(p(a), s) - t)/C(n, s))^t.$$

Applying Lemma 2 again to the quantity $C(p(a), s)/C(n, s)$, we have

$$((p(a) - s)/n)^s \leq C(p(a), s)/C(n, s) \leq (p(a)/n)^s.$$

Combining these two, we have

$$f_0[n, t, s](a) \leq (1 - ((p(a) - s)/n)^s)^t,$$

and

$$f_0[n, t, s](a) \geq (1 - (p(a)/n)^s - t/C(n, s))^t.$$

Finally, applying Lemma 3, $C(n, s) \geq (n/s)^s$, so

$$f_0[n, t, s](a) \geq (1 - (p(a)/n)^s - t(s/n)^s)^t.$$

Q.E.D.

5.3. Few 1's or few 0's

The second tool that we use is the following lemma that says that for sufficiently large n and t, any DNF formula over V_n with t terms is satisfied by an assignment with at most $\sqrt{n}$ 1's or falsified by an assignment with at most $1 + (\sqrt{n}) \log t$ 0's.

LEMMA 5. Let $n \geq 4$ and $t \geq 1$ be integers. Let ϕ be any DNF formula over V_n with t terms. Either there is an assignment a_1 that satisfies ϕ and has $p(a_1) < \sqrt{n}$, or there is an assignment a_0 that falsifies ϕ and has $p(a_0) \geq n - 1 - (\sqrt{n}) \log t$.

This is a corollary of the following simple lemma.

LEMMA 6. Let n be any positive integer. Let α be a real number such that $0 < \alpha < 1$. Suppose ϕ is a DNF formula over V_n with $t \geq 1$ terms such that each term contains at least αn distinct positive literals. Then there is a set $V \subseteq V_n$ of at most $1 + \lfloor \log_b t \rfloor$ variables such that every term of ϕ contains a positive occurrence of some element of V, where $b = 1/(1 - \alpha)$.

Proof. We construct V as follows. Let X_i be a variable that maximizes the number of terms of ϕ that contain a positive occurrence of X_i. Add X_i to V and remove from ϕ any term that contains a positive occurrence of X_i. Iterate this process until no terms are left in ϕ.

Since every term of ϕ contains at least αn positive occurrences of variables, at least one variable must occur positively in a fraction α of the terms remaining at the end of every iteration of the process, so after r elements have been added to V, there must be at most $(1 - \alpha)^r t$ terms left in ϕ. Hence, for $b = 1/(1 - \alpha)$, when

$$r = 1 + \lfloor \log_b t \rfloor,$$

we have

$$(1 - \alpha)^r t < (1 - \alpha)^{\log_b t} t = 1.$$

Thus fewer than 1 term must be left in ϕ; that is, ϕ must be empty. Q.E.D.

Proof of Lemma 5. Let $n \geq 4$ let $t \geq 1$, and let ϕ be any DNF formula with t terms over V_n. If some term τ of ϕ has fewer than $\sqrt{n}$ positive occurrences of literals, let $a_1(X_i) = 1$ if and only if X_i occurs positively in τ. Then $a_1(\tau) = 1$. Hence $p(a_1) < \sqrt{n}$ and $a_1(\phi) = 1$.

If every term of ϕ has at least $\sqrt{n}$ positive occurrences of literals, then we apply the preceding lemma with $\alpha = 1/\sqrt{n}$ to conclude that there is a set of variables V such that every term of ϕ has a positive occurrence of some variable from V and $|V| \leq 1 + \lfloor \log_b t \rfloor$ where $b = 1/(1 - \alpha)$. Take $a_0(X_i) = 0$ if and only if $X_i \in V$. Then $a_0(\phi) = 0$, and $p(a_0) = n - |V|$. For $n \geq 4$, $\log_b t \leq (\sqrt{n} - 1) \log t$, so $p(a_0) \geq n - 1 - (\sqrt{n} - 1) \log t$. Thus, for $t \geq 1$,

$$p(a_0) \geq n - 1 - (\sqrt{n}) \log t.$$

Q.E.D.

5.4. A little lemma

LEMMA 7. Let n be any positive integer and x any real number such that $0 \leq x \leq 1/n$. Then $1 - (1 - x)^n \leq 2nx$.

The proof is not difficult using the binomial theorem.

5.5. DNF formulas: the main result

THEOREM 8. The class of DNF formulas has the approximate fingerprint property.

COROLLARY 9. There is no polynomial time algorithm that exactly identifies all DNF formulas using only equivalence queries.

Proof of Theorem 8. The positive nondecreasing polynomial $p_1(n)$ is chosen so that any DNF formula over V_n with at most n terms can be represented by a string of length at most $p_1(n)$. This is possible because of our assumptions on the representation of DNF formulas by strings. We take the polynomial $p_2(n) = n$.

Now suppose we are given any positive nondecreasing polynomial $q(n)$. We must define a sequence of target classes $T_1, T_2, \ldots$ with the required properties. Choose the constant $S \geq 4$ so that for all sufficiently large n,

$$2n(1/\sqrt{n})^S < 1/2q(n).$$

This is possible because the left side is $2/n^{S/2-1}$ and $q(n)$ is a fixed polynomial. Then for all sufficiently large n,

$$2n^2(S/n)^S < 1/2q(n).$$

This is true because the left side is the product of $2/n^{S/2-1}$ and $S^S/n^{S/2-1}$, and for fixed $S \geq 1$, the latter quantity goes to zero as n goes to infinity. Thus, for all sufficiently large n,

$$2n((1/\sqrt{n})^S + n(S/n)^S) < 1/q(n). \tag{1}$$

Now choose the constant $T \geq 1$ such that for all sufficiently large n,

$$((2S \log q(n))/\sqrt{n} + 2S(S+1)/n)^T < 1/q(n). \tag{2}$$

This is possible because for fixed S and $q(n)$ the left hand side is $O((C \log n/\sqrt{n})^T)$ for a fixed constant C.

For each n, the class T_n is all the concepts represented by formulas from $M(n, T, S)$. Provided $n \geq T$, each formula in $M(n, T, S)$ has at most n terms and can be represented by a string of length at most $p_1(n)$, so the sequence of concepts $T_1, T_2, \ldots$ is bounded by $p_1(n)$, as required.

In addition, provided $n > S$ and $n > T$, there are at least n terms in $T(n, S)$ and at least n formulas in $M(n, T, S)$, so the class T_n contains at least two distinct concepts. We will show that for all sufficiently large n and for any string r of length at most $q(n)$ representing a DNF formula there exists a string w of length $p_2(n) = n$ such that $\langle w, \chi_{\mu(r)}(w)\rangle$ is a $1/q(n)$-approximate fingerprint with respect to T_n.

Choose n sufficiently large that $n > T$, $n > S$, the inequalities (1) and (2) hold, and also

$$(1/\sqrt{n})^S + T(S/n)^S \leq 1/T, \tag{3}$$

and

$$(\sqrt{n} \log q(n))/n + (S+1)/n \leq 1/S. \tag{4}$$

Suppose that r is a string of length at most $q(n)$ representing a DNF formula.

First, if $1^n \notin c(r)$, then since every formula in T_n is satisfied by 1^n, the fraction of formulas in T_n that agree with r in the classification of 1^n is 0; so in this case, $\langle 1^n, \chi_{\mu(r)}(1^n)\rangle$ is a $1/q(n)$-approximate fingerprint with respect to T_n. Thus, if r represents a formula over V_k for some $k \neq n$ or if r represents a formula over V_n that is not satisfied by the assignment of all 1's, then for the choice $w = 1^n$, $\langle w, \chi_{\mu(r)}(w)\rangle$ is a $1/q(n)$-approximate fingerprint with respect to T_n.

So assume r represents a formula ϕ over V_n that is satisfied by the assignment of all 1's. In particular, ϕ must contain at least one term.

By our assumptions on the representation of DNF formulas, the number of terms in ϕ is bounded by $|r|$. Thus ϕ contains at most $q(n)$ terms. Since $n \geq 4$, we may apply Lemma 5 to find either an assignment a_1 with $p(a_1) < \sqrt{n}$ and $a_1(\phi) = 1$ or an assignment a_0 with $p(a_0) \geq n - 1 - (\sqrt{n}) \log q(n)$ and $a_0(\phi) = 0$. In each of these cases we will show that the assignment gives an approximate fingerprint with respect to T_n.

Consider first the case of a_1 with $p(a_1) < \sqrt{n}$. We need to show that the fraction of formulas in $M(n, T, S)$ that are satisfied by a_1 is less than $1/q(n)$. Lemma 4 gives a lower

bound on the fraction of formulas in $M(n, T, S)$ that are assigned 0 by a_1, and therefore an upper bound on the fraction of formulas in $M(n, T, S)$ that are assigned 1 by a_1 of

$$f_1[n, T, S](a_1) \leq 1 - (1 - (p(a_1)/n)^S - T(S/n)^S)^T.$$

Since $p(a_1) \leq \sqrt{n}$,

$$f_1[n, T, S](a_1) \leq 1 - (1 - (1/\sqrt{n})^S - T(S/n)^S)^T.$$

By our choice of n, inequality (3) holds, that is,

$$(1/\sqrt{n})^S + T(S/n)^S \leq 1/T,$$

so Lemma 7 applies, yielding

$$f_1[n, T, S](a_1) \leq 2T((1/\sqrt{n})^S + T(S/n)^S).$$

Since $n > T$,

$$f_1[n, T, S](a_1) \leq 2n((1/\sqrt{n})^S + n(S/n)^S).$$

By our choices of S, T and n, inequality (1) holds, that is,

$$2n((1/\sqrt{n})^S + n(S/n)^S) < 1/q(n),$$

and therefore

$$f_1[n, T, S] < 1/q(n).$$

Thus in this case, if w is the string representing the assignment a_1, $\langle w, \chi_{\mu(r)}(w)\rangle$ is a $1/q(n)$-approximate fingerprint with respect to T_n.

For the other case, suppose a_0 is an assignment that falsifies ϕ and $p(a_0) \geq n - 1 - \sqrt{n} \log q(n)$. We need to show that the fraction of $M(n, T, S)$ falsified by a_0 is less than $1/q(n)$. Lemma 4 gives an upper bound on the fraction $f_0[n, T, S](a_0)$ of formulas in $M(n, T, S)$ that are falsified by a_0 of

$$f_0[n, T, S](a_0) \leq (1 - ((p(a_0) - S)/n)^S)^T.$$

Since $p(a_0) \geq n - 1 - \sqrt{n} \log q(n)$,

$$f_0[n, T, S](a_0) \leq (1 - ((n - 1 - \sqrt{n} \log q(n) - S)/n)^S)^T.$$

Thus

$$f_0[n, T, S](a_0) \leq (1 - (1 - (\sqrt{n} \log q(n))/n - (S + 1)/n)^S)^T.$$

By our choice of n, inequality (4) holds, that is,

$$(\sqrt{n} \log q(n))/n + (S + 1)/n \leq 1/S,$$

so Lemma 7 applies, yielding

$$f_0[n, T, S](a_0) \leq (2S((\sqrt{n} \log q(n))/n + (S + 1)/n))^T.$$

By our choices of S, T, and n, inequality (2) holds, that is,

$$((2S \log q(n))/\sqrt{n} + 2S(S + 1)/n)^T < 1/q(n),$$

so

$$f_0[n, T, S](a_0) < 1/q(n).$$

Hence if w is the string of length n representing the assignment a_0, $\langle w, \chi_{\mu(r)}(w)\rangle$ is a $1/q(n)$-approximate fingerprint with respect to T_n.

Thus we have shown that for all sufficiently large n, if r is a string of length at most $q(n)$ representing a DNF formula, then there is a string w of length n such that $\langle w, \chi_{\mu(r)}(w)\rangle$ is a $1/q(n)$-approximate fingerprint with respect to T_n. Since $q(n)$ was arbitrary, the class of DNF formulas has the approximate fingerprint property. Q.E.D.

6. Dfas have approximate fingerprints

In this section and the following two sections we prove that dfas, nfas, and cfgs have approximate fingerprints. The proof for dfas is fairly simple, the proof for nfas is slightly more complex, and the proof for cfgs is decidedly complicated; however, there is a progression in technique that may be helpful for understanding.

Let Σ be the finite alphabet $\{0, 1\}$. Let Σ^k denote the set of all strings over Σ of length k. If $x, y \in \Sigma^k$, let $d(x, y)$ denote the Hamming distance between x and y, that is, the number of bit positions $1 \leq i \leq k$ such that x and y are different in the i-th bit.

For any $n \geq 1$ and $1 \leq i \leq n$ define the language

$$L(i, n) = \{ubvbw : u \in \Sigma^{i-1}, b \in \Sigma, v \in \Sigma^{n-1}, w \in \Sigma^{n-i}\}.$$

That is, $L(i, n)$ is the set of all strings of length $2n$ such that the i-th bit is equal to the $(n + i)$-th bit. Note that $L(i, n)$ is accepted by a dfa with $3n + 2$ states.

The target class T_n will consist of all languages L that can be obtained as a concatenation

$$L(i_1, n) \cdot L(i_2, n) \cdots L(i_n, n)$$

such that $1 \leq i_j \leq n$ for each j. Thus there are n^n distinct languages in the class T_n. Each one can be accepted by a dfa of $3n^2 + 2$ states, so this sequence of concept classes is polynomially bounded.

Now we consider two further sets of strings:

$$A_n = \{x_1x_1x_2x_2 \cdots x_nx_n : \text{for each } i,\ x_i \in \Sigma^n\},$$

and

$$B_n = \{x_1y_1x_2y_2 \cdots x_ny_n : \text{for each } i,\ x_i \in \Sigma^n,\ y_i \in \Sigma^n,\ d(x_i, y_i) > n/4\}.$$

Note that A_n is the intersection of all the languages in T_n, and the smallest dfa to accept A_n has a number of states that grows exponentially in n.

For every string $w \in B_n$, there are at most $(3n/4)^n$ languages $L \in T_n$ that contain w. This is an exponentially small fraction of the languages in T_n since $(3n/4)^n$ divided by n^n is $(3/4)^n$.

Now consider some fixed polynomial $q(n)$. We prove that for all sufficiently large n, if M is any dfa with at most $q(n)$ states, then either M rejects a string w in A_n or accepts a string w in B_n. In either case, if n is sufficiently large then $\langle w, M(w)\rangle$ is a $1/q(n)$-approximate fingerprint with respect to T_n, since in the first case no concepts in T_n agree with M's classification of w, and in the second case, an exponentially small fraction of the concepts in T_n do. We formalize this as follows.

LEMMA 10. There exists a constant c_0, $0 < c_0 < 1$, such that for all sufficiently large positive integers n,

$$\sum_{i=0}^{\lceil n/4 \rceil} \binom{n}{i} < 2^{c_0 n}.$$

The proof is by straightforward application of Stirling's approximation for $n!$ An immediate consequence is the following.

LEMMA 11. For the constant c_0 of Lemma 10 and all sufficiently large n, if $x \in \Sigma^n$ then there are fewer than $2^{c_0 n}$ strings $y \in \Sigma^n$ such that $d(x, y) \leq n/4$.

Proof. Let c_0 and n be as indicated, and let x be any element of Σ^n. Then a string $y \in \Sigma^n$ is such that $d(x, y) \leq n/4$ if and only if y may be obtained from x by complementing at most $n/4$ of the bits of x. Hence, any such y can be obtained by selecting a subset of $i \leq n/4$ bits of x and complementing them. Thus, there are at most

$$\sum_{i=0}^{\lceil n/4 \rceil} \binom{n}{i}$$

such strings y, and this is less than $2^{c_0 n}$ by Lemma 10. Q.E.D.

LEMMA 12. Let $q(n)$ be any positive nondecreasing polynomial in n. For sufficiently large n, if M is any dfa of size at most $q(n)$ and q_i is any state of M, then there exist two strings x and y of length n such that $d(x, y) > n/4$ and $\delta(q_i, x) = \delta(q_i, y)$, where δ is the transition function of M.

Proof. Consider $\delta(q_i, x)$ for all $x \in \Sigma^n$. There are at most $q(n)$ states in M, so there is at least one state, say q_j such that $\delta(q_i, x) = q_j$ for at least $2^n/q(n)$ values of $x \in \Sigma^n$.

Choose any $x \in \Sigma^n$ such that $\delta(q_i, x) = q_j$. By Lemma 11, there are fewer than $2^{c_0 n}$ strings $x' \in \Sigma^n$ such that $d(x, x') \leq n/4$.

Since for all sufficiently large n,

$$2^n/q(n) > 2^{c_0 n},$$

there must be at least one string $y \in \Sigma^n$ such that $\delta(q_i, y) = q_j$ and $d(x, y) > n/4$.Q.E.D.

LEMMA 13. Let $q(n)$ be any positive nondecreasing polynomial in n. Then for all sufficiently large n, if M is any dfa of size at most $q(n)$ that accepts all the strings in A_n, then M accepts some string in B_n.

Proof. Let n be sufficiently large that the conclusion of Lemma 12 holds for $q(n)$ and n. Assume that M is any dfa of size at most $q(n)$ and $A_n \subseteq L(M)$. Let δ denote the transition function of M and let q_0 denote the initial state of M. We construct a string in B_n accepted by M by induction.

By Lemma 12 there are two strings x_1 and y_1 in Σ^n such that $\delta(q_0, x_1) = \delta(q_0, y_1)$ and $d(x_1, y_1) > n/4$. Let $q_1 = \delta(q_0, x_1x_1)$. Note that

$$\delta(q_0, x_1x_1) = \delta(q_0, y_1x_1) = q_1.$$

Assume that for some k, $1 \leq k < n$, and for all i, $1 \leq i \leq k$, there exist states q_i and strings x_i and y_i in Σ^n such that

$$\delta(q_{i-1}, x_ix_i) = \delta(q_{i-1}, y_ix_i) = q_i$$

and $d(x_i, y_i) > n/4$. Then, by Lemma 12, there exist strings x_{k+1} and y_{k+1} in Σ^n such that

$$\delta(q_k, x_{k+1}) = \delta(q_k, y_{k+1})$$

and $d(x_{k+1}, y_{k+1}) > n/4$. Let

$$q_{k+1} = \delta(q_k, x_{k+1}x_{k+1}) = \delta(q_k, y_{k+1}x_{k+1})$$

and the induction hypothesis is satisfied for $k + 1$.

Thus for $i = 1, \ldots, n$, there exist states q_i and strings x_i and y_i in Σ^n such that

$$\delta(q_{i-1}, x_ix_i) = \delta(q_{i-1}, y_ix_i) = q_i$$

and $d(x_i, y_i) > n/4$. Now,

$$x_1x_1x_2x_2 \cdots x_nx_n \in A_n$$

and

$$y_1x_1y_2x_2 \cdots y_nx_n \in B_n.$$

Moreover,

$$\delta(q_0, x_1x_1x_2x_2 \cdots x_nx_n) = \delta(q_0, y_1x_1y_2x_2 \cdots y_nx_n) = q_n.$$

Since M accepts every string in A_n by hypothesis, the state q_n must be accepting, so M accepts the string

$$y_1x_1y_2x_2 \cdots y_nx_n \in B_n.$$

Q.E.D.

THEOREM 14. The class of dfas has the approximate fingerprint property.

COROLLARY 15. There is no polynomial time algorithm that exactly identifies the class of dfas using only equivalence queries.

Proof of Theorem 14. The sequence of concept classes $T_1, T_2, \ldots$ is polynomially bounded. Provided $n \geq 2$, T_n contains at least two distinct concepts. For any positive nondecreasing polynomial $q(n)$ we may choose n sufficiently large that $(3/4)^n < 1/q(n)$ and any dfa of at most $q(n)$ states must either fail to accept some string in A_n or accept some string in B_n.

Let r be any string of length at most $q(n)$ representing a dfa M. Then by our assumptions on the representation of dfas, M has at most $q(n)$ states, and therefore either fails to accept a string in A_n or accepts a string in B_n.

Suppose M fails to accept a string $w \in A_n$. Note that the length of w is $2n^2$. Since every concept in T_n contains A_n as a subset, the fraction of concepts in T_n that agree with M on the string w is zero, so $\langle w, \chi_{c(r)}(w)\rangle$ is a $1/q(n)$-approximate fingerprint with respect to T_n.

Suppose M accepts some string $w \in B_n$. Note that the length of w is $2n^2$. The fraction of concepts in T_n that contain w is at most $(3/4)^n$, and by our choice of n, this is less than $1/q(n)$. Thus in this case also $\langle w, \chi_{c(r)}(w)\rangle$ is a $1/q(n)$-approximate fingerprint with respect to T_n. Q.E.D.

7. Nfas have approximate fingerprints

We consider nondeterministic finite state acceptors (nfas) over the alphabet $\Sigma = \{0, 1\}$. Such an acceptor has a finite set of states, a distinguished start state, a set of accepting states, and a transition function that maps each element of $Q \times \Sigma$ to a subset of Q. The *size* of such an acceptor is the number of states it contains. Our goal in this section is to prove the following.

THEOREM 16. The class of nfas has the approximate fingerprint property.

COROLLARY 17. There is no polynomial time algorithm that exactly identifies all the nfas using only evidence queries.

The result for dfas does not directly imply the corresponding result for nfas or cfgs since the definition of approximate fingerprints is representation-dependent.

For the case of nfas, we use the same target classes T_n and the same sets A_n but replace the sets B_n by the following. The set B_n' consists of all strings

$$x_1 y_1 x_2 y_2 \cdots x_n y_n$$

such that for each i, $x_i \in \Sigma^n$ and $y_i \in \Sigma^n$, and for at least half the values of j between 1 and n, $d(x_j, y_j) > n/4$. For any string $w \in B_n'$ there are at most $n^{n/2}(3n/4)^{n/2}$ languages $L \in T_n$ that contain w. This is an exponentially small $((3/4)^{n/2})$ fraction of the concepts in T_n.

The analog of Lemma 13 that we use is the following.

LEMMA 18. Let $q(n)$ be any positive nondecreasing polynomial in n. Then for all sufficiently large n, if M is any nfa of size at most $q(n)$ that accepts all the strings in A_n, then M accepts some string in B_n'.

Proof. Let n be sufficiently large that the conclusion of Lemma 10 holds, and

$$2^{n^2(1-c_0)/2} > q(n)^{2n}, \tag{5}$$

where c_0 is the constant in Lemma 10. There exists such an n because $c_0 < 1$, so the left-hand side is of the form $2^{\epsilon n^2}$, while the righthand side is of the form $2^{Kn\log n + c}$, for positive constants ϵ, K, and c.

Let M be any nfa of size at most $q(n)$ that accepts all the strings in A_n. Let Q be the set of states of M, q_0 the initial state of M, F the set of accepting states, and δ the transition function. Extend δ so that for a state $q \in Q$ and a string $u \in \Sigma^*$, $\delta(q, u)$ is the set of states reachable from q on input string u.

Let $x_1, \ldots, x_n$ be any n-tuple of strings from Σ^n. The string $x_1 x_1 x_2 x_2 \cdots x_n x_n$ is in A_n and therefore is accepted by M. Define the function

$$h(i) = \lceil i/2 \rceil$$

for all $i = 1, \ldots, 2n$. There exists a sequence of states $q_1, \ldots, q_{2n}$ from Q such that $q_{2n} \in F$ and

$$q_i \in d(q_{i-1}, x_{h(i)})$$

for $i = 1, \ldots, 2n$. That is, $q_0, q_1, \ldots, q_{2n}$ is the sequence of states at distance n apart along some accepting computation of M on the string $x_1 x_1 \cdots x_n x_n$.

For each n-tuple $x_1, \ldots, x_n$ of strings from Σ^n, choose one such sequence of states $q_1, \ldots, q_{2n}$ and define

$$F(x_1, \ldots, x_n) = \langle q_1, q_2, \ldots, q_{2n} \rangle.$$

The domain of F has cardinality 2^{n^2} and the range of F has cardinality at most $q(n)^{2n}$, so there must be at least one point in the range that is the image of at least $2^{n^2}/q(n)^{2n}$ points in the domain. Let $\langle q_1, q_2, \ldots, q_{2n}\rangle$ be any such point in the range, and let S denote the set of n-tuples $x_1, \ldots, x_n$ that map to this point. Then

$$|S| \geq 2^{n^2}/q(n)^{2n}.$$

Let $S[i]$ denote the projection of S in the i-th component, that is, the set of $x_i \in \Sigma^n$ such that there exist strings $x_1, \ldots x_{i-1}$ and $x_{i+1}, \ldots, x_n$ in Σ^n such that the n-tuple $x_1, \ldots, x_n$ is in S. We define S to be *sparse for i* if and only if $|S[i]| < 2^{c_0 n}$, where c_0 is the constant in Lemma 10. Let D be the set of indices i such that S is not sparse for i.

We claim that $|D| \geq n/2$. Otherwise, the cardinality of S must be bounded above by

$$(2^n)^{n/2} \cdot (2^{c_0 n})^{n/2} = 2^{n^2(1+c_0)/2}.$$

But this implies that

$$2^{n^2(1+c_0)/2} \geq 2^{n^2}/q(n)^{2n},$$

that is,

$$2^{n^2(1-c_0)/2} \leq q(n)^{2n},$$

which contradicts our choice of n satisfying inequality (5).

Now choose any string $u = x_1x_1 \cdots x_nx_n$ such that the n-tuple $\langle x_1, \ldots, x_n\rangle$ is in S. For each $i = 1, \ldots, n$, define the string v_i as follows. If $i \notin D$, then let $v_i = x_ix_i$. If $i \in D$, $S[i]$ is not sparse, so $|S[i]| \geq 2^{c_0 n}$. Thus, by Lemma 11 there must be a string $y_i \in S[i]$ such that $d(x_i, y_i) > n/4$. Choose any such y_i, and let $v_i = x_iy_i$.

To see that the final string $v_1v_2 \cdots v_n$ has the desired properties, not that each string $v_i \in \Sigma^{2n}$. Also, since $|D| \geq n/2$, for at least $n/2$ indices, $v_i = x_iy_i$ with $d(x_i, y_i) > n/4$. To see that $v_1v_2 \cdots v_n$ is accepted by M, note that for each $i = 1, \ldots, n$,

$$q_{2i-1} \in \delta(q_{2i-2}, x_i),$$

and

$$q_{2i} \in \delta(q_{2i-1}, x_i),$$

and if $i \in D$,

$$q_{2i} \in \delta(q_{2i-1}, y_i).$$

Thus, there is an accepting computation of M on the string $v_1 \cdots v_n$ that passes through the states $q_0, q_1, \ldots, q_{2n}$. This concludes the proof of Lemma 18. Q.E.D.

Proof of Theorem 16. Since every dfa is an nfa, the sequence of concept classes $T_1, T_2, \ldots$ is polynomially bounded (with respect to representation by nfas). For any positive nondecreasing polynomial $q(n)$ we may choose n sufficiently large that $(3/4)^{n/2} < 1/q(n)$ and any nfa of at most $q(n)$ states must either fail to accept some string in A_n or accept some string in B'_n.

Let r be any string of length at most $q(n)$ representing an nfa M. Then by our assumptions on the representation of nfas, M has at most $q(n)$ states, and therefore either fails to accept a string in A_n or accepts a string in B'_n.

Suppose M fails to accept a string $w \in A_n$. Note that the length of w is $2n^2$. Since every concept of T_n contains A_n as a subset, the fraction of concepts in T_n that agree with M on the string w is zero, so $\langle w, \chi_{c(r)}(w)\rangle$ is a $1/q(n)$-approximate fingerprint with respect to T_n.

Suppose M accepts some string $w \in B'_n$. Note that the length of w is $2n^2$. The fraction of concepts in T_n that contain w is at most $(3/4)^{n/2}$, and by our choice of n, this is less than $1/q(n)$. Thus in this case also $\langle w, \chi_{c(r)}(w)\rangle$ is a $1/q(n)$-approximate fingerprint with respect to T_n. Q.E.D.

Sakakibara (1988) has shown that deterministic bottom-up tree automata can be identified in polynomial time using equivalence and membership queries. By simply tacking a linear tree structure onto the strings used in the proofs above, we have the following.

COROLLARY 19. There is no polynomial time algorithm that exactly identifies deterministic or nondeterministic bottom-up tree automata using only equivalence queries.

8. Cfgs have approximate fingerprints

We consider context free grammars (cfgs) over the terminal alphabet $\Sigma = \{0, 1\}$. We assume that all grammars are in Chomsky normal form, that is, every production is of the form $A \to BC$ where A, B, and C are nonterminal symbols, or $A \to a$, where A is a nonterminal symbol and a is a terminal symbol, or $S \to \epsilon$, where S is the start symbol and ϵ is the empty string. Since there is a polynomial time algorithm to translate an arbitrary cfg into one in Chomsky normal form, this assumption is not an essential restriction. Recall that the size of a Chomsky normal form cfg G is the number of nonterminal symbols it contains.

The goal of this section is the following.

THEOREM 20. The class of context free grammars has the approximate fingerprint property.

COROLLARY 21. There is no polynomial time algorithm that exactly identifies all the context free grammars using only equivalence queries.

For the case of cfgs the proofs are rather involved, but the basic outline is the same. The target class and the approximate fingerprints are easily described. For each $n \geq 1$ let

$$C_{1,n} = \{xx^r y : x \in \Sigma^n, y \in \Sigma^n\},$$

where x^r denotes the reverse of the string x, and let

$$C_{2,n} = \{yx^rx : x \in \Sigma^n, y \in \Sigma^n\}.$$

Each of $C_{1,n}$ and $C_{2,n}$ can be generated by a Chomsky normal form cfg with $4n + 1$ nonterminals.

The target class T_n^{cfg} contains each language of the form

$$C_{i_1,n} \cdot C_{i_2,n} \cdots C_{i_n,n}$$

where $i_j \in \{1, 2\}$ for each j. Thus, there are 2^n distinct languages in T_n^{cfg}. Each of them can be generated by a Chomsky normal form grammar with $6n$ nonterminals. Thus, this sequence of concept classes is polynomially bounded.

Let

$$I_n = C_{1,n} \cap C_{2,n} = \{xx^rx : x \in \Sigma^n\}.$$

Note that the smallest cfg to generate I_n has size exponential in n. Let

$$F_n = \Sigma^{3n} - I_n.$$

Any string in F_n is in at most one of $C_{1,n}$ and $C_{2,n}$.

The set of strings corresponding to A_n is just the intersection of all the sets in T_n^{cfg}, that is,

$$A_n^{cfg} = (I_n)^n = \{x_1x_1^rx_1x_2x_2^rx_2 \cdots x_nx_n^rx_n : \text{for each } i,\ x_i \in \Sigma^n\}.$$

Clearly, A_n^{cfg} is a subset of every language in T_n^{cfg}.

For any constant $0 < c < 1$, let $B_{c,n}^{cfg}$ consist of all string of the form

$$w_1w_2 \cdots w_n$$

such that each $w_i \in \Sigma^{3n}$ and for at least cn values of j, $w_j \in F_n$. Note that for each string $w \in B_{c,n}^{cfg}$, there are at most 2^{n-cn} languages $L \in T_n^{cfg}$ that contain w. This is an exponentially small (2^{-cn}) fraction of T_n^{cfg}.

The analog of Lemmas 13 and 18 that we use in this case is the following.

LEMMA 22. Let $p(n)$ be any positive nondecreasing polynomial in n. There exists a constant c_1 such that $0 < c_1 \leq 1$ such that for all sufficiently large n, if G is a Chomsky normal form context free grammar of size at most $p(n)$ then either G rejects some string in A_n^{cfg} or G accepts some string in $B_{c_1,n}^{cfg}$.

The proof of this lemma is given in the next two subsections. Here we show that it suffices to prove Theorem 20.

Proof of Theorem 20. The sequence of concept classes T_n^{cfg} is polynomially bounded, and for each $n \geq 1$, T_n^{cfg} contains at least two distinct languages.

Let c_1 be the constant of Lemma 22, and let $q(n)$ be any positive nondecreasing polynomial. Let n be chosen sufficiently large that $2^{-c_1 n} < 1/q(n)$ and if G is a Chomsky normal form grammar of size at most $q(n)$, then G either rejects a string in A_n^{cfg} or accepts a string in $B_{c_1,n}^{cfg}$.

Let r be any string of length at most $q(n)$ representing a Chomsky normal form context free grammar G. By our assumptions on the representation of grammars, G has size at most $q(n)$, so G must either reject a string in A_n^{cfg} or accept a string in $B_{c_1,n}^{cfg}$.

In the first case, let w be a string in A_n^{cfg} rejected by G. Note that w is of length $3n^2$. Since every concept in T_n^{cfg} accepts w, the fraction of concepts in T_n^{cfg} that agree with G on the classification of w is zero, so $\langle w, \chi_{c(r)}(w)\rangle$ is a $1/q(n)$-approximate fingerprint with respect to T_n^{cfg}.

In the second case, let w be an element of $B_{c_1,n}^{cfg}$ accepted by G. Note that $|w| = 3n^2$. The fraction of concepts in T_n^{cfg} that accept w is at most $2^{-c_1 n}$, and by our choice of n, $2^{-c_1 n} < 1/q(n)$. Hence, $\langle w, \chi_{c(r)}(w)\rangle$ is a $1/q(n)$-approximate fingerprint with respect to T_n^{cfg}.
Q.E.D.

We now turn to the proof of Lemma 22. The general plan is as follows. We assume that $L(G)$ contains all 2^{n^2} strings in A_n^{cfg}, and fix a set of parse trees for these strings with respect to G. We then find a lot of *pieces* of these parse trees with the same *environments* and different yields that can be substituted for each other to construct a legal parse tree with respect to G for a string in $B_{c_1,n}^{cfg}$.

The following lemma concerning I_n and F_n will be of use in the substitution arguments in the following sections.

LEMMA 23. Suppose $xyz \in I_n$, where y is of length at most $2n$. If y' is any string such that $y' \neq y$ and $|y'| = |y|$, then $xy'z \in F_n$.

Proof. There is some $w_1 \in \Sigma^n$ such that $xyz = w_1 w_1' w_1$. Since $|y| \leq 2n$, $|x| + |z| \geq n$. If $|x| \geq n$, then let x_1 be the prefix of x of length n and let z_2 be the empty string. If $|x| < n$, then let $x_1 = x$ and let z_2 be the suffix of z of length $n - |x|$. In either case, x_1 is the prefix of w_1 of length $|x_1|$, and z_2 is the suffix of w_1 of length $|z_2|$, and $|x_1| + |z_2| = n$, so $w_1 = x_1 z_2$.

Now suppose that $xy'z \notin F_n$. Since $|y| = |y'|$, $xy'z$ is of length $3n$, so it must be that $xy'z \in I_n$. Hence, for some $w_2 \in \Sigma^n$, $xy'z = w_2 w_2' w_2$. As above, $w_2 = x_1 z_2$, and therefore $w_1 = w_2$. Thus, $xyz = xy'z$ and $y = y'$, contradicting the hypotheses. Hence $xy'z \in F_n$, as claimed.
Q.E.D.

8.1. Pieces of binary trees

We need some technical tools for surgery on parse trees in the next subsection. We consider rooted ordered binary trees. Let T be such a tree. The subtree of T rooted at the node v is denoted $T(v)$.

Let v be a node of T and V' a subset of nodes that are in $T(v)$. Define $P(v, V')$ to be the rooted ordered binary tree obtained from T by taking $T(v)$ and removing all the proper

descendants of any node in V'. Such a rooted ordered binary tree will be called a *piece* of T. Each leaf of $P(v, V')$ is either an element of V' or a leaf in T. A leaf of $P(v, V')$ that is not a leaf of T is called a *special leaf* of $P(v, V')$. $P(v, V')$ is a *type-i* piece of T if and only if the number of special leaves of $P(v, V')$ is exactly i.

A type-0 piece of T is simply $T(v)$ for some node v of T. A type-1 piece of T is $T(v)$ for some node v minus all the proper descendants of one of the nonleaf nodes of $T(v)$. We are primarily interested in type-0 and type-1 pieces of binary trees.

Two pieces T_1 and T_2 of T are called *nonoverlapping* provided either they contain no nodes in common, or, if they contain any nodes in common, it is just one node x, and x is the root of one of the trees and a leaf of the other. A set of pieces of T is called *nonoverlapping* provided every pair of distinct pieces in the set is nonoverlapping.

LEMMA 24. If T is a rooted ordered binary tree with n leaves and $1 \leq k \leq n$, then there exists a node v of T such that $T(v)$ has at least k and fewer than $2k$ leaves.

Proof. Consider the root x of T. If $n < 2k$, then $v = x$ suffices. Otherwise, $n \geq 2k$. Then at least one of the two immediate descendants of x, say x_1, is such that $T(x_1)$ has at least k leaves. Iterate with the tree $T(x_1)$. Since the number of leaves in the subtree being considered must be strictly decreasing, this process must terminate with a node v such that the number of leaves in $T(v)$ is at least k and less than $2k$. Q.E.D.

LEMMA 25. Let T be a rooted ordered binary tree with $n \geq 2$ leaves. Let $2 \leq k \leq n$. Then there is a nonoverlapping set S of type-0 and type-1 pieces of T such that each piece has at least k and fewer than $2k$ leaves, and $|S| \geq (n - k)/4(k - 1)$.

Proof. We describe how to construct such an S. Initially let S be empty, and let $T' = T$.

T' is a rooted ordered tree with at least k leaves, so by Lemma 24, there is a node v of T' such that $T'(v)$ has at least k and fewer than $2k$ leaves. If $T'(v)$ has at most one leaf that is not a leaf of T, then it is a type-0 or type-1 piece of T and is added to S. In any case, T' is then set to T' with all the proper descendants of v removed, and if T' still has at least k leaves, this process is iterated.

It is clear that throughout this process, T' is a rooted ordered binary tree obtained from T by removing all the proper descendants of some set of nodes in T. Hence the trees added to S are distinct pairwise nonoverlapping type-0 and type-1 pieces of T with at least k and fewer than $2k$ leaves.

It remains to establish the bound claimed for the cardinality of S. Consider the number s of leaves of T' that are not leaves of T. Initially $s = 0$. When $T'(v)$ is a type-i piece, s is set to $s + 1 - i$, so it is increased by 1 when a type-0 piece is found, remains the same when a type-1 piece is found, and is decreased by 1 or more when any other type of piece is found. After the first iteration, s is always at least 1. Hence the number of iterations in which type-0 or type-1 pieces are found must exceed the number of iterations in which other types of pieces are found.

How many iterations must there be? Each iteration removes at least $k - 1 > 0$ and at most $2k - 2$ leaves from T', and the process continues until fewer than k leaves are left in T'. Initially T' has n leaves. Thus, if j is the total number of iterations, we must have

$$n - j(2k - 2) < k,$$

which implies that

$$j > (n - k)/(2k - 2).$$

Since at least half of these must be iterations in which a type-0 or type-1 piece is found and added to S, we have

$$|S| > (n - k)/4(k - 1),$$

which proves Lemma 25. Q.E.D.

8.2. Parse tree surgery

The goal of this subsection is to prove Lemma 22 and thus to conclude the proof of Theorem 20. If i and j are integers, then $[i, j]$ denotes the set of integers k such that $i \leq k \leq j$.

Let $n \geq 2$ be a positive integer, and let G be any Chomsky normal form context free grammar such that $(I_n)^n \subseteq L(G)$. Let $N(G)$ denote the size of G, that is, the number of nonterminals of G. For each n-tuple $w_1, \ldots, w_n$ of strings from Σ^n, the string

$$w_1 w_1^r w_1 w_2 w_2^r w_2 \cdots w_n w_n^r w_n$$

is an element of $L(G)$. Choose one fixed parse tree for this string with respect to the grammar G and denote it by $T(w_1, \ldots, w_n)$. This tree is a rooted ordered binary tree whose internal nodes are labelled by nonterminal symbols of G and whose leaves are labelled by terminal symbols of G.

Indexed parse trees.

For any parse tree T with respect to G, let $num(T)$ be the tree obtained from T by replacing each leaf label a by the ordered pair (a, i), where i is the number of this leaf in left-to-right order, starting with 1. The number i is called the *leaf index* of the leaf labelled by (a, i). $num(T)$ is called an *indexed parse tree*. If T is a rooted ordered binary tree, define $un(T)$ to be the tree obtained from T by replacing any leaf label (a, i) by the label a. Clearly, $un(num(T)) = T$ for any parse tree T.

Let $T_n(G)$ denote the set of all indexed parse trees $num(T(w_1, \ldots, w_n))$ as $w_1, \ldots, w_n$ ranges over all n-tuples of strings from Σ^n. There are 2^{n^2} elements of $T_n(G)$, each of which has $3n^2$ leaves.

Let T be any tree in $T_n(G)$. By Lemma 25 with $k = n$, there exists a set, which we denote by $P(T)$, of nonoverlapping type-0 and type-1 pieces of T such that each piece has at least n and fewer than $2n$ leaves and

$$|P(T)| \geq (3n^2 - n)/4(n - 1) > 3n/4.$$

Environments.

A *type-0 environment* is a triple $e = (A, i, j)$ such that A is a nonterminal of G, and i and j are positive integers such that $1 \leq i \leq j \leq 3n^2$. The *terminal indices* of e is the set of integers $[i, j]$.

A *type-1 environment* is a sextuple $e = (A, i, j, B, k, l)$ such that A and B are nonterminals of G, and i, j, k, and l are nonnegative integers such that one of the three possibilities below holds:

1. $1 \leq i \leq j < k \leq l \leq 3n^2$.
2. $i = j = 0$ and $1 \leq k \leq l \leq 3n^2$.
3. $1 \leq i \leq j \leq 3n^2$ and $k = l = 0$.

The *terminal indices* of e in case (1) is the set of integers $[i, j] \cup [k, l]$, in case (2) is the set of integers $[k, l]$, and in case (3) is the set of integers $[i, j]$.

Let E denote the set of all type-0 and type-1 environments that have at least $n - 1$ and at most $2n - 1$ terminal indices. Then $|E| \leq 81n^8(N(G))^2$ because each environment in E can be specified by a choice of four numbers in the range 1 through $3n^2$ and two nonterminals from G.

For each piece T' in $P(T)$ we define the *environment* of T', denoted $e(T')$, as follows. If T' is a type-0 piece, then $e(T')$ is (A, i, j), where A is the nonterminal labelling the root of T', i and j are leaf indices of the leftmost and rightmost leaves of T', respectively. If T' is a type-1 piece, then there is exactly one special leaf of T', and it is labelled with a nonterminal of G. In this case, $e(T')$ is (A, i, j, B, k, l) where A is the nonterminal labelling the root of T', B is the label of the special leaf of T', i and j are the leaf indices of the leftmost and rightmost leaves that are to the left of the special leaf of T', and k and l are leaf indices of the leftmost and rightmost leaves that are to the right of the special leaf of T'. If there are no leaves to the left of the special leaf of T' then i and j are 0; likewise, if there are no leaves to the right of the special leaf of T', k and l are 0. Note that in either case, $e(T') \in E$, and the set of terminal indices of $e(T')$ is the same as the set of leaf indices of T'.

For each $T \in T_n(G)$ define the set

$$E(T) = \{e(T') : T' \in P(T)\}$$

of all environments of pieces in $P(T)$. Since the pieces in $P(T)$ are nonoverlapping and each contains at least $n - 1$ nonspecial leaves, they all have distinct environments, so for each $T \in T_n(G)$,

$$|E(T)| = |P(T)| > 3n/4.$$

Transplanting pieces.

Pieces with the same environment may be interchanged to produce new legal parse trees, as indicated by the following.

LEMMA 26. Suppose T_1 and T_2 are elements of $T_n(G)$ such that $T_1' \in P(T_1)$, $T_2' \in P(T_2)$, and $e(T_1') = e(T_2')$. Then if T_3 is the tree obtained by substituting piece T_2' for piece T_1' *in* T_1, $un(T_3)$ is a legal parse tree with respect to G.

Proof. In fact, all that is necessary to produce a new legal parse tree is that the nonterminal symbols appearing in the environments of T_1' and T_2' be the same. The equality of the sets of terminal indices in the environments guarantees the stronger condition that the yield of T_3 is obtained from the yield of T_1 by substituting the section(s) of the yield of T_2 that correspond to the interval(s) of terminal indices specified in the environment. Q.E.D.

Segments.

We regard the set of numbers from 1 to $3n^2$ as divided up into n *segments* of $3n$ consecutive numbers each. Thus, the s-th segment consists of the numbers $3n(s-1)+1$ through $3ns$. If $y \in \Sigma^{3n^2}$, and s is a segment number, then we define the s-th segment of y to be the substring of length $3n$ beginning with position $3n(s-1)+1$ in y.

For each environment $e \in E$ we define the *segments impacted* by e to be the set of all segments that contain an element from the set of terminal indices of e. Note that at most four segments are impacted by any $e \in E$, since e has at most $2n-1$ terminal indices in at most two intervals $[i, j]$ and $[k, l]$. If T' is an element of $P(T)$ for some $T \in T_n(G)$, then the *segments impacted by* T' is the set of segments impacted by $e(T')$.

For each environment $e \in E$, we define the *major segment* of e to be the leftmost segment that contains at least $(n-1)/4$ members of the set of terminal indices of e. Since e must have at least $n-1$ terminal indices, and impacts at most four segments, there must be at least one segment that contains at least $(n-1)/4$ of the terminal indices of e. The *major segment* of a piece T' of $T \in T_n(G)$ is the major segment of its environment $e(T')$.

Proof of Lemma 22. Now we are ready to prove Lemma 22. Let $p(n)$ be any positive nondecreasing polynomial in n. Let n be sufficiently large that $n > 13$ and

$$2^{3n(n-1)/32C} > q(n)^r,$$

where $C = 3168$, $q(n) = 8ln^8p(n)^2$, and $r = \lceil 3n/4 \rceil$. This is possible because the lefthand side is of the form $2^{\epsilon n^2}$ and the righthand side is of the form $2^{Kn\log n+c}$ for positive constants ϵ, K, and c. Let G be any Chomsky normal form context free grammar of size at most $p(n)$ such that $(I_n)^n \subseteq L(G)$.

Since there are 2^{n^2} elements $T \in T_n(G)$, and at most $q(n)$ elements of E, there must be at least one environment $e_1 \in E$ that is a member of at least

$$2^{n^2}/q(n)$$

different sets $E(T)$. Besides e_1, each of these sets contains at least $r-1$ other environments, because $|E(T)| \geq r$ for all $T \in T_n(G)$. Thus, there must be at least one environment $e_2 \neq e_1$ that appears in at least

$$2^{n^2}/q(n)^2$$

of the sets $E(T)$ that e_1 appears in. Continuing in this way, we can find a set

$$E' = \{e_1, \ldots, e_r\}$$

of r distinct environments such that at for at least

$$2^{n^2}/q(n)^r$$

elements $T \in T_n(G)$, $E' \subseteq E(T)$.

Now we need to select a large subset E'' of E' with the property that for each $e \in E''$, the major segment of e is not impacted by any other element in E''. In particular, we have the following.

LEMMA 27. Let $C = 3168$. There exists a subset E'' of E' such that for every $e \in E''$, the major segment of e is not impacted by any other element of E'' and $|E''| \geq r/C$.

Proof. Note that E' is a subset of at least one $E(T)$, and hence must be nonoverlapping. Consider any segment s. It contains $3n$ indices, and can be the major segment of at most 12 environments from E', since

$$13(n - 1)/4 > 3n.$$

Let M denote the set of segments that are major segments of some environment in E'. Then

$$|M| \geq |E'|/12 = r/12.$$

Let E_{96} denote the set of $e \in E'$ such that the major segment of e is impacted by at most 96 of the elements of E'. We argue that $|E_{96}| \geq r/24$. Since each segment in M is the major segment of at least one environment in E', and each environment in E' has exactly one major segment, it suffices to show that at least $r/24$ segments in M are each impacted by at most 96 elements of E'.

E' contains r elements, each of which can impact at most 4 segments, so the total number of instances of a segment being impacted by an environment in E' is at most $4r$. Suppose fewer than $r/24$ of the segments in M are each impacted by at most 96 of the environments in E'. Since $|M| \geq r/12$, this means that more than $r/24$ of the segments in M are each impacted by more than 96 of the environments in E', so the total number of instances of segments being impacted by an environment in E' is greater than $96(r/24) = 4r$, a contradiction.

Thus, at least $r/24$ of the segments in M are each impacted by at most 96 of the environments in E', which implies that $|E_{96}| \geq r/24$, as claimed.

Now we construct E'' as follows. Let $F = E_{96}$. Choose any element $e \in F$, and let the major segment of e be s. Add e to E''. Now remove from F any environment that impacts segment s and any environment whose major segment is impacted by e. There are at most 96 environments in E_{96} that impact segment s. There are at most 3 segments other than s that are impacted by e. Each of these segments is the major segment of at most 12 elements

of E_{96}. Thus, there are at most 36 elements e', of E_{96} with the property that the major segment of e' is one of the segments other than s impacted by e. Hence at most 132 elements are removed from F. Iterate this process until no elements remain in F.

Since F initially contains at least $r/24$ elements, and at most 132 elements are removed at each iteration, in the end

$$|E''| \geq r/(24 \cdot 132) = r/3168,$$

and the major segment of each environment in E'' is not impacted by any of the other environments in E''. (Environments in E'' may share impacted non-major segments.)Q.E.D.

We need a further subset of E'', to guarantee the existence of distinct strings for substitution. Recall that $r = \lceil 3n/4 \rceil$. Thus, E'' contains at least $3n/4C$ elements. Let $T(E'')$ be the set of all $T \in T_n(G)$ such that $E'' \subseteq E(T)$. Note that since $E'' \subseteq E'$,

$$|T(E'')| \geq 2^{n^2}/q(n)^r.$$

For each environment $e \in E''$, consider the interval $[i, j]$ of terminal indices of e that are contained in the major segment of e. We know that the length of this interval is at least $(n - 1)/4$, by the definition of major segment. For each $T \in T(E'')$, consider the string of terminal symbols $a_i \cdots a_j$ that appear in the leaves with indices i through j, and define

$$m(e, T) = a_i \cdots a_j.$$

An environment $e \in E''$ is called *major-constant* if and only if $m(e, T)$ takes on only one value as T ranges over all elements of $T(E'')$. Let E''' be the set of environments of E'' that are not major-constant.

LEMMA 28. $|E'''| \geq 3n/8C$.

Proof. Suppose to the contrary that there are at least $3n/8C$ major-constant environments in E''. Each such environment has at least $(n - 1)/4$ terminal indices in its major segment and these are all nonoverlapping. Thus, for at least $3n(n - 1)/32C$ leaf indices, all the trees $T \in T(E'')$ have the same terminals in the leaf labels, so there are at most

$$2^{n^2}/2^{3n(n-1)/32C}$$

trees T in $T(E'')$. But we have shown above that there are at least

$$2^{n^2}/q(n)^r$$

trees in $T(E'')$, so

$$2^{3n(n-1)/32C} \leq q(n)^r,$$

which is a contradiction, by our choice of n. Hence, for at least $3n/8C$ elements $e \in E''$, e is not major-constant, that is, $|E'''| \geq 3n/8C$. Q.E.D.

We are ready to perform substitutions. We start with any $T_0 \in T(E'')$. Choose any environment $e_1 \in E'''$. There is a tree $T' \in T(E'')$ such that $m(e_1, T_0) \neq m(e_1, T')$. Let t_1 be the unique piece from $P(T_0)$ with environment e_1 and let t_1' be the unique piece from $P(T')$ with environment e_1. Replace t_1 in T_0 by t_1' and call the result T_1.

Note that $un(T_1)$ is a legal parse tree in G. Let y_0 denote the yield of the parse tree $un(T_0)$ and let y_1 denote the yield of $un(T_1)$. Let s be the major segment of e_1. The s-th segment of y_1 is obtained from the s-th segment of y_0 by substituting the string $m(e, T')$ for the string $m(e, T_0)$. Since the s-th segment of y_0 is in I_n, $m(e, T_0) \neq m(e, T')$ and $|m(e, T_0)| = |m(e, T_1)| \leq 2n - 1$, by Lemma 23 the s-th segment of y_1 is in F_n. We now choose another environment e_2 from E''' and iterate with T_1.

Since no element of E''' has a major segment impacted by any other element of E''', this process can be iterated for each element of E''' yielding a final tree T_m such that $un(T_m)$ is a legal parse tree for G whose yield is a string

$$x_1 x_2 \cdots x_n$$

where each $x_i \in \Sigma^{3n}$, and for at least m values of i, $x_i \in F_n$. Since $m = |E'''| \geq 3n/8C$, Lemma 22 is finally proved, with $c_1 = 3/8C = 3/25344$. Q.E.D.

9. Comments

Theorem 8 shows that the class of DNF formulas has the approximate fingerprint property. By duality, the class of CNF formulas also has the approximate fingerprint property. The proof in fact shows that a polynomial time algorithm for CNF formulas using only equivalence queries can be correct for all k-CNF formulas (CNF formulas with at most k literals per clause) for only a finite number of values of k. This is the behavior achieved by the equivalence query algorithm for k-CNF formulas (Angluin, 1988a), adapted from the corresponding algorithm given by Valiant (1984).

Since the target classes of formulas $M(n, T, S)$ are monotone, the proof also shows that there is no polynomial time algorithm using only equivalence queries (querying general DNF formulas) that identifies all the monotone DNF formulas. In contrast, there is a polynomial time algorithm that exactly identifies monotone DNF formulas using equivalence queries and membership queries (Angluin, 1988a). Hence membership queries are essential to that result.

Hancock (1989) has shown that there is no polynomial time algorithm that exactly identifies all decision trees (a strict subclass of DNF formulas) using only equivalence queries, even if the inputs to the queries may be DNF formulas.

Theorem 14 shows that the class of dfas has the approximate fingerprint property. The target classes are finite, fixed-length, and zero-reversible (Angluin, 1982a). Theorem 16 shows that the class of nfas has the approximate fingerprint property. Since the target classes are the same in both proofs, this shows that there is no polynomial time algorithm that exactly identifies dfas for finite, fixed-length, zero-reversible languages using only equivalence queries, even if we allow nfas as input to the queries. See also Ibarra and Jiang's reduction of the general case to the finite and fixed-length case for dfas (Ibarra & Jiang, 1988).

Theorem 20 shows that the class of context free languages has the approximate fingerprint property. Since in this case also the target classes are finite and fixed-length, this implies that there is no polynomial time algorithm that exactly identifies cfgs for finite, fixed-length languages using only equivalence queries.

These arguments do NOT show that these classes are not polynomial time learnable in Valiant's model of *pac*-identification (Valiant, 1984) or in a prediction model (Haussler, Littlestone & Warmuth, 1988; Littlestone, 1988; Pitt & Warmuth, 1988). Polynomial time identification using only equivalence queries implies polynomial time *pac*-identification, but not conversely. In particular, with an oracle for NP each of these classes become *pac*-learnable, by the Occam's razor technique (Blumer, Ehrenfeucht, Haussler & Warmuth, 1987), so proving one of these classes cannot be *pac*-identified in polynomial time involves proving or assuming $P \neq NP$, whereas our results do neither. Kearns and Valiant (1989) have shown that dfas, nfas, cfgs, and boolean formulas are as hard to predict modulo a polynomial time transformation as certain cryptographic predicates are to compute.

It is interesting to note that identification of dfas with equivalence queries seems to be *harder* than successful prediction. If efficient prediction of dfas in the *pac*-identification sense is required, an oracle for an NP-complete problem is sufficient. With an oracle for a #P-complete problem we can implement an efficient majority-vote prediction strategy for dfas which has a polynomial bound on the worst case number of errors of prediction. However, not even an oracle for a PSPACE-complete problem permits polynomial time identification of dfas with equivalence queries.

10. Acknowledgments

This research was funded by the National Science Foundation, under grant number IRI-8718975. Earlier versions of this material have appeared in technical reports and the COLT '89 proceedings (Angluin, 1988b; Angluin, 1988c; Angluin, 1989).

References

Angluin, D. (1982a). Inference of reversible languages. *J. ACM*, 29, 741–765.

Angluin, D. (1982b). A note on the number of queries needed to identify regular languages. *Information and Control*, *51*, 76–87.

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

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

Angluin, D. (1988b). *Negative results for equivalence queries* (Technical Report YALE/DCS/RR-648). Yale University, Department of Computer Science.

Angluin, D. (1988c). *Equivalence queries and DNF formulas* (Technical Report YALE/DCS/RR-659). Yale University, Department of Computer Science.

Angluin, D. (1989). Equivalence queries and approximate fingerprints. *Proceedings of the Second Annual Workshop on Computational Learning Theory* (pp. 134–145). Palo Alto, CA: Morgan Kaufmann.

Angluin, D., Hellerstein, L., & Karpinski (1989). *Learning read-once formulas with queries* (Technical Report UCB/CSD 89/528). University of California at Berkeley, Computer Science Division. (Also, Technical Report TR-89-050, International Computer Science Institute, Berkeley, California.)

Blumer, A., Ehrenfeucht, A., Haussler, D., & Warmuth, M. (1987). Occam's razor. *Information Processing Letters*, *24*, 377–380.

Hancock, T. (1989). Identifying decision trees with equivalence queries. Manuscript, Harvard University.

Haussler, D., Kearns, M., Littlestone, N., & Warmuth, M. (1988). Equivalence of models for polynomial learnability. *Proceedings of the 1988 Workshop on Computational Learning Theory* (pp. 42–55). Palo Alto, CA: Morgan Kaufmann.

Haussler, D., Littlestone, N., & Warmuth, M. (1988). Predicting {0, 1}-functions on randomly drawn points. *Proceedings of the 29th Symposium on Foundations of Computer Science* (pp. 100–109). Washington, DC: The Computer Society Press of the IEEE.

Ibarra, O. & Jiang, T. (1988). Learning regular languages from counterexamples. *Proceedings of the 1988 Workshop on Computational Learning Theory* (pp. 371–385). Palo Alto, CA: Morgan Kaufmann.

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

Littlestone, N., (1988). Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning, 2,* 285–318.

Pitt, L., & Warmuth, M., (1988). Reductions among prediction problems: On the difficulty of predicting automata. *Proceedings of the Third Annual Structure in Complexity Theory Conference* (pp. 60–69). Washington, DC: The Computer Society Press of the IEEE.

Porat, S., & Feldman, J., (1988). Learning automata from ordered examples. *Proceedings of the 1988 Workshop on Computational Learning Theory* (pp. 386–396). Palo Alto, CA: Morgan Kaufmann.

Sakakibara, Y., (1988). Learning context-free grammars from structural data in polynomial time. *Proceedings of the 1988 Workshop on Computational Learning Theory* (pp. 330–344). Palo Alto, CA: Morgan Kaufmann.

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