Negative Results for Equivalence Queries

Dana Angluin*, Yale University

YALEU/DCS/RR-648
September 1988

*Supported by the National Science Foundation, IRI-8718975

Negative Results for Equivalence Queries

Dana Angluin [*]
Yale University

September 1988

Abstract

In [1] we posed the question of whether deterministic finite state machines can be learned in polynomial time using only equivalence queries. In this paper we exhibit and correct a loophole in the definition of polynomial-time learnability used in [1]. We then give a negative answer to this question. We also show that equivalence queries do not suffice for polynomial-time learnability of nondeterministic finite state machines, bottom-up tree automata, or context-free grammars.

1 A Loophole

We consider deterministic finite-state acceptors (dfas) over the fixed alphabet $\Sigma = \{0,1\}$. The *size* of such an acceptor is the number of states in it.

In the dfa learning problem we assume that there is some unknown regular set L_*, and the goal of a learning algorithm is to halt and output a dfa M such that $L(M) = L_*$. The algorithm may gather information about L_* by making equivalence queries.

The input to an *equivalence query* is a description in some standard form of a dfa M over Σ, and the reply is "yes" if $L(M) = L_*$, and "no" otherwise. If the reply is "no", a *counterexample* is also supplied, that is, a string in the symmetric difference of L_* and $L(M)$. The choice of counterexample is assumed to be arbitrary.

The following definition is implicit in [1].

Definition 1 *An algorithm A is a polynomial-time learning algorithm for dfas using only equivalence queries 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, the time used by A 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 in the run.*

With respect to this definition, there is a simple polynomial-time learning algorithm for dfas that uses only equivalence queries. The algorithm is as follows: enumerate descriptions of dfas in some standard lexicographic order and test each one with an equivalence query until a correct one is found, say M. If the length of the string describing M is l, then the

[*]Supported by NSF grant IRI-8718975

time used by the algorithm to this point is polynomial in 2^l. Construct a machine M' that has the same behavior as M except on one string of length 2^l, and make an equivalence query with M'. The counterexample to this query must have length 2^l. Now output M and halt. The length of the longest counterexample in this run is at least 2^l, and the total time used is bounded by a polynomial in 2^l.

This algorithm can be made a little more subtle, so that it doesn't get the "right" answer until after a "long counterexample", but in any case it is clear that there is a problem with Definition 1. Hence, we propose the following correction.

Definition 2 *An algorithm A is a polynomial-time learning algorithm for dfas using only equivalence queries 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.*

It is clear that the algorithm described above fails under this definition of polynomial-time learnability. Moreover, with respect to Definition 2 we have the following.

Theorem 3 *There is no polynomial-time algorithm to learn dfas using only equivalence queries.*

This contrasts with the results in [1], which show that there is a polynomial-time algorithm (in the sense of Definition 2) to learn dfas using equivalence and membership queries. It also contrasts with the results of Ibarra and Jiang [4], which show that if equivalence queries are restricted to return the lexicographically least counterexample, then there is a polynomial time algorithm to learn dfas using only equivalence queries. The next section is devoted to a proof of Theorem 3.

2 Equivalence Queries and DFAs

We prove Theorem 3 by exhibiting a family of languages that cannot be identified by any polynomially-bounded query strategy, computable or not. That is, if a strategy is limited to a polynomially bounded number of polynomially bounded equivalence queries, it cannot gather information about this family of languages quickly enough to do correct identification of every member of the family. In this sense, the result is analogous to the simple proof that the singleton languages cannot be identified using polynomially many equivalence queries in [2]. In that proof, the absence of a hypothesis for the empty set in the restricted hypothesis space was crucial for the exponential bound. In this case, our proof shows that the full hypothesis space of dfas has an analogous kind of limitation. In particular, Theorem 3 shows that the "majority vote" strategy described in [2] cannot be efficiently implemented in the domain of dfas: the required majority vote machine is sometimes just too large.

Recall that $\Sigma = \{0,1\}$. Let Σ^n denote all the strings over Σ of length n. If w is a string, let $w[i]$ denote the i^{th} symbol of w. If x and y are two strings of length n, let $E(x,y)$ denote the complement of the Hamming distance between x and y, that is, the number of positions i such that $x[i] = y[i]$.

Let i and n be positive integers, with $i \leq n$. Define

$$L(i,n) = \{w \in \Sigma^{2n} : w[i] = w[i+n]\}.$$

Thus, $L(i,n)$ is the set of all strings of length $2n$ such that the i^{th} symbol is the same as the $(i+n)^{th}$.

Let m be a positive integer, $m \leq n$. Define

$$K(m,n) = \{xy : x, y \in \Sigma^n \text{ and } E(x,y) \geq m\}.$$

Thus, $K(m,n)$ is the set of all strings $w \in \Sigma^{2n}$ such that $w \in L(i,n)$ for at least m different values of i. Clearly,

$$K(n,n) = \{xx : x \in \Sigma^n\}.$$

Note that each $L(i,n)$ can be accepted by a dfa with $3n+2$ states, but the smallest dfa to accept $K(n,n)$ has size exponential in n.

For each positive integer n we define the class H_n to consist of all languages L such that for some positive integers $1 \leq i_j \leq n$ for $j = 1, \ldots, n$,

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

There are n^n distinct languages in the class H_n. Each one can be accepted by a dfa of $3n^2+2$ states. The intersection of all of them is the language $K(n,n)^n$, the set of all strings of the form

$$w_1 w_1 w_2 w_2 \cdots w_n w_n$$

such that each $w_i \in \Sigma^n$.

We prove that the classes H_n are hard to learn using equivalence queries. We first establish some lemmas.

Lemma 4 *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 5 *For the constant c_0 of Lemma 4 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 $E(x,y) \geq 3n/4$.*

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 $E(x,y) \geq 3n/4$ 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 4. Q.E.D.

Lemma 6 *Let $p(n)$ be any increasing polynomial in n. For all sufficiently large n, if M is any dfa of size at most $p(n)$ and q is any state of M, then there exist two strings x and y of length n such that $E(x,y) < 3n/4$ and $\delta(q,x) = \delta(q,y)$, where δ is the transition function of M.*

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

Choose any $x \in \Sigma^n$ such that $\delta(q,x) = q'$. By Lemma 5, there are fewer than $2^{c_0 n}$ strings $x' \in \Sigma^n$ such that $E(x,x') \geq 3n/4$.

Since for all sufficiently large n,

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

there must be at least one string $y \in \Sigma^n$ such that $\delta(q,y) = q'$ and $E(x,y) < 3n/4$. Q.E.D.

Let

$$I_n = K(n,n)$$

and

$$F_n = (\Sigma^{2n} - K(\lceil 3n/4 \rceil, n)).$$

Lemma 7 *Let $p(n)$ be any increasing polynomial in n. Then for all sufficiently large n, if M is any dfa of size at most $p(n)$ that accepts all the strings in $(I_n)^n$, then M accepts some string in $(F_n)^n$.*

Proof. Let n be sufficiently large that the conclusion of Lemma 6 holds for $p(n)$ and n. Assume that M is any dfa of size at most $p(n)$ and $(I_n)^n \subseteq L(M)$. Let δ denote the transition function of M and let q_0 denote the initial state of M. We construct a string in $(F_n)^n$ accepted by M by induction.

By Lemma 6 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 $E(x_1,y_1) < 3n/4$. Let $q_1 = \delta(q_0, x_1x_1)$. Note that

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

and $x_1x_1 \in I_n$ while $y_1x_1 \in F_n$.

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 $E(x_i,y_i) < 3n/4$. Then, by Lemma 6, 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 $E(x_{k+1}, y_{k+1}) < 3n/4$. Let

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

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

$$\delta(q_{i-1}, x_i x_i) = \delta(q_{i-1}, y_i x_i) = q_i$$

and $E(x_i, y_i) < 3n/4$. Thus,

$$x_1 x_1 x_2 x_2 \cdots x_n x_n \in (I_n)^n$$

and

$$y_1 x_1 y_2 x_2 \cdots y_n x_n \in (F_n)^n.$$

Moreover,

$$\delta(q_0, x_1 x_1 x_2 x_2 \cdots x_n x_n) = \delta(q_0, y_1 x_1 y_2 x_2 \cdots y_n x_n) = q_n.$$

Since M accepts every string in $(I_n)^n$ by hypothesis, the state q_n must be accepting, so M accepts the string

$$y_1 x_1 y_2 x_2 \cdots y_n x_n \in (F_n)^n.$$

This conclude the proof of Lemma 7. Q.E.D.

Proof of Theorem 3.

Assume that A is a polynomial time algorithm to learn dfas using equivalence queries only, and let $p(n, m)$ be a polynomial bounding the running time of A according to Definition 2. Without loss of generality, we may assume that the polynomial $p(n, m)$ is increasing in both of its arguments. Let $q(n) = p(3n^2 + 2, 2n^2)$ for all positive integers n.

Let n be sufficiently large that if M is any dfa of size at most $q(n)$ that accepts all the strings in $(I_n)^n$, then M accepts some string in $(F_n)^n$, by Lemma 7. Also let n be sufficiently large that

$$(4/3)^n > q(n) + (4/3)^n / n^n,$$

which implies that

$$n^n - q(n)(3/4)^n n^n > 1.$$

A must correctly identify every element of H_n and the time used at any point must be bounded by $p(3n^2 + 2, m)$, where m is the length of the longest counterexample seen in the run to that point. Consider the following adversary strategy, which returns counterexamples of length $2n^2$ consistent with some element of H_n. Run algorithm A until it makes an equivalence query with some dfa M. Since the running time of A must be bounded by $p(3n^2 + 2, 2n^2) = q(n)$, the size of M is bounded by $q(n)$.

If $(I_n)^n$ is not a subset of $L(M)$, then the adversary returns any element of $(I_n)^n - L(M)$ as a counterexample. Otherwise, since M accepts all the strings in $(I_n)^n$, it must accept some string in $(F_n)^n$, so any element of $(F_n)^n \cap L(M)$ is returned as a counterexample.

Consider the class of hypotheses H_n. A counterexample from $(I_n)^n - L(M)$ is consistent with all the hypotheses in H_n, and does not eliminate any of them as possible values of the unknown language.

A counterexample $u \in (F_n)^n \cap L(M)$ eliminates every $L \in H_n$ that contains u. Suppose

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

where for each i, y_i and x_i are in Σ^n, and $E(x_i, y_i) < 3n/4$. If

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

then L is eliminated by u if and only if for each $j = 1, \ldots, n$,

$$x_j[i_j] = y_j[i_j].$$

Since for each j, there are fewer than $3n/4$ values of i_j for which $x_j[i_j] = y_j[i_j]$, the total number of elements of H_n that are eliminated by the counterexample u is bounded above by $(3n/4)^n$.

Every hypothesis in H_n that has not been eliminated by some counterexample remains a possible value for the unknown language. As long as A has made $t \leq q(n)$ equivalence queries, there must remain at least

$$n^n - t \cdot (3n/4)^n \geq n^n - q(n)(3/4)^n n^n > 1$$

hypotheses in H_n that are consistent with all the replies to queries so far, by our choice of n. Since at least two elements of H_n are still consistent with all the counterexamples, if A halts and outputs an answer after making $q(n)$ or fewer queries, it is incorrect on some $L \in H_n$. On the other hand, since the counterexamples to all the queries up through the $q(n)^{th}$ are of length $2n^2$, if A makes more than $q(n)$ queries, it will exceed its time bound of $q(n) = p(3n^2 + 2, 2n^2)$ on some element of H_n. Thus we have a contradiction, since A cannot correctly terminate within its time bound for every element of H_n. Hence no such A can exist, which proves Theorem 3. Q.E.D.

3 Equivalence Queries and NFAS

Our goal in this section is to prove the same theorem for nfas. The proof technique is a little more complex, and serves as an introduction to the decidedly complicated machinery for the cfg case.

Theorem 8 *There is no polynomial-time algorithm to learn nondeterministic finite state acceptors using only equivalence queries.*

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. The definitions of equivalence queries and polynomial time learning algorithms are the same, substituting nfas for dfas. The definitions of H_n, I_n, and F_n are the same.

The analog of Lemma 7 that we use is the following.

Lemma 9 *Let $p(n)$ be any increasing polynomial in n. Then for all sufficiently large n, if M is any nfa of size at most $p(n)$ that accepts all the strings in $(I_n)^n$, then M accepts some string $v_1 v_2 \cdots v_n$ such that each $v_i \in \Sigma^{2n}$ and for at least $n/2$ values of i, $v_i \in F_n$.*

Proof. Let $p(n)$ be any increasing polynomial in n, and let n be sufficiently large that the conclusion of Lemma 4 holds, and

$$2^{n^2(1-c_0)/2} > p(n)^{2n},$$

where c_0 is the constant in Lemma 4. There exists such an n because $c_0 < 1$, so the lefthand 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 $p(n)$ that accepts all the strings in $(I_n)^n$. 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 $x_1x_1x_2x_2\cdots x_nx_n$ is in $(I_n)^n$, and therefore 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

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

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 $x_1x_1\cdots x_nx_n$.

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 $p(n)^{2n}$, so there must be at least one point in the range that is the image of at least $2^{n^2}/p(n)^{2n}$ 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

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

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 4. 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

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

that is,

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

which contradicts our choice of n.

Now choose any string $u = x_1x_1 \cdots x_nx_n$ 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 5 there must be a string $y_i \in S[i]$ such that $E(x_i, y_i) < 3n/4$, that is, $x_iy_i \in F_n$. 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, note that each string $v_i \in \Sigma^{2n}$. Also, since $|D| \geq n/2$, for at least $n/2$ indices, $v_i \in F_n$. 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 13. Q.E.D.

Proof of Theorem 11.

Assume that A is a polynomial time algorithm to learn nfas using equivalence queries only, and let $p(n, m)$ be a polynomial bounding the running time of A according to Definition 2. Without loss of generality, we may assume that the polynomial $p(n, m)$ is increasing in both of its arguments. Let $q(n) = p(3n^2 + 2, 2n^2)$ for all positive integers n.

Let n be sufficiently large that if M is any nfa of size at most $q(n)$ that accepts all the strings in $(I_n)^n$, then M accepts some string $v_1 \cdots v_n$ such that each $v_i \in \Sigma^{2n}$ and for at least $n/2$ values of i, $v_i \in F_n$, by Lemma 9. Also let n be sufficiently large that

$$(4/3)^{n/2} > q(n) + (4/3)^{n/2}/n^n,$$

which implies that

$$n^n - q(n) \cdot (3/4)^{n/2} n^n > 1.$$

A must correctly identify every element of H_n and the time used at any point must be bounded by $p(3n^2 + 2, m)$, where m is the length of the longest counterexample seen in the run to that point. Consider the following adversary strategy, which returns counterexamples of length $2n^2$ consistent with some element of H_n. Run algorithm A until it makes an equivalence query with some nfa M. Since the running time of A must be bounded by $p(3n^2 + 2, 2n^2) = q(n)$, the size of M is bounded by $q(n)$.

If $(I_n)^n$ is not a subset of $L(M)$, then the adversary returns any element of $(I_n)^n - L(M)$ as a counterexample. Otherwise, since M accepts all the strings in $(I_n)^n$, it must accept some string $v_1 \cdots v_n$ such that each $v_i \in \Sigma^{2n}$ and for at least $n/2$ values of i, $v_i \in F_n$, by Lemma 9 and our choice of n. In this case, the adversary returns any such string $v_1 \cdots v_n$ as the counterexample.

Consider the class of hypotheses H_n. A counterexample from $(I_n)^n - L(M)$ is consistent with all the hypotheses in H_n, and does not eliminate any of them as possible values of the unknown language.

The second type of counterexample, $v_1 \cdots v_n \in L(M)$, eliminates every $L \in H_n$ that contains $v_1 \cdots v_n$. How many $L \in H_n$ contain $v_1 \cdots v_n$? Let $v_i = x_i y_i$ for $x_i, y_i \in \Sigma^n$. If

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

then L is eliminated by u if and only if for each $j = 1, \ldots, n$,

$$x_j[i_j] = y_j[i_j].$$

There are at least $n/2$ values of j for which $v_j \in F_n$, which means that $E(x_j, y_j) < 3n/4$, so there are fewer than $3n/4$ values of i_j for which $x_j[i_j] = y_j[i_j]$. Thus, the total number of elements of H_n that are eliminated by the counterexample $v_1 \cdots v_n$ is bounded above by

$$(3n/4)^{n/2} n^{n/2} = (3/4)^{n/2} n^n.$$

Every hypothesis in H_n that has not been eliminated by some counterexample remains a possible value for the unknown language. As long as A has made $t \le q(n)$ equivalence queries, there must remain at least

$$n^n - t \cdot (3n/4)^{n/2} n^n \ge n^n - q(n) \cdot (3/4)^{n/2} n^n > 1$$

hypotheses in H_n that are consistent with all the replies to queries so far, by our choice of n. Thus, if A halts and outputs an answer after making $q(n)$ or fewer queries, it is incorrect on some $L \in H_n$. On the other hand, since the counterexamples to all the queries up through the $q(n)^{th}$ are of length $2n^2$, if A makes more than $q(n)$ queries, it will exceed its time bound of $q(n) = p(3n^2 + 2, 2n^2)$ on some element of H_n. Thus we have a contradiction, since A cannot correctly terminate within its time bound for every element of H_n. Hence no such A can exist, which proves Theorem 8. Q.E.D.

By simply tacking a linear tree structure onto the strings used in the proofs above, we have the following.

Corollary 10 *Deterministic and nondeterministic bottom-up tree automata cannot be learned in polynomial time using only equivalence queries.*

4 Equivalence Queries and CFGs

In this section we extend the technique used in the proofs of Theorems 3 and 8 to show the following.

Theorem 11 *The class of context-free grammars cannot be learned in polynomial time using only equivalence queries.*

Some extra technical machinery is required, to prove an analog of Lemmas 7 and 9. We consider context-free grammars (cfgs) over the terminal alphabet $\Sigma = \{0, 1\}$. We assume that all grammars are in Chomsky normal form (CNF), 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. The *size* of a CNF cfg G is the number of nonterminal symbols it contains.

Let n be any positive integer. Let

$$L_1(n) = \{xx^r y : x, y \in \Sigma^n\}$$

and let

$$L_2(n) = \{yx^r x : x, y \in \Sigma^n\},$$

where x^r denotes the reverse of the string x. $L_1(n)$ and $L_2(n)$ can each be generated by a CNF cfg of size at most $4n+1$. However, the intersection language,

$$I_n = L_1(n) \cap L_2(n) = \{xx^r x : x \in \Sigma^n\},$$

requires a cfg of size exponential in n. Define

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

Any string in F_n is in at most one of $L_1(n)$ and $L_2(n)$.

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

Lemma 12 *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 $xyz = w_1 w_1^r w_1$. 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 $w_2 \in \Sigma^n$, $xy'z = w_2 w_2^r w_2$. 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.

We now define the crucial hypothesis classes. For each n we define H_n to be the class of all languages L of the form

$$L = L_{i_1}(n) \cdot L_{i_2}(n) \cdots L_{i_n}(n),$$

where each i_j is either 1 or 2. There are 2^n languages in the class H_n. Each language in H_n contains only strings of length $3n^2$ and can be generated by a CNF cfg of size $6n$.

The analog of Lemmas 7 and 9 that we use in this case is the following.

Lemma 13 *Let $p(n)$ be any increasing 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 $p(n)$ such that $(I_n)^n \subseteq L(G)$, then there exists a string*

$$u = x_1 x_2 \cdots x_n$$

such that each $x_i \in \Sigma^{3n}$ and for at least $c_1 n$ values of i, $x_i \in F_n$.

The proof of this lemma is given in the next two subsections. Here we show that it suffices to prove Theorem 11.

Assume that A is an algorithm to learn cfgs in polynomial time using only equivalence queries. We may assume without loss of generality that A makes queries only with context-free grammars in Chomsky normal form. Let $p(n, m)$ be a polynomial that bounds the running time of A. We may assume without loss of generality that $p(n, m)$ is increasing in both arguments.

For any Chomsky normal form context-free grammar G, A with equivalence queries for $L(G)$ halts and outputs a grammar G' such that $L(G') = L(G)$. Moreover, if the size of G is n, then at any point in a legal run, the time used by A is bounded by $p(n, m)$, where m is the maximum length of any counterexample seen to that point.

We describe an adversary that forces A either to be incorrect or to exceed its time bound. Let $q(n) = p(6n, 3n^2)$. This is clearly an increasing polynomial of n. Let n be sufficiently large that the conclusion of Lemma 13 holds of $q(n)$. Also, let n be sufficiently large that

$$q(n) < 2^{c_1 n} - 1/2^{n - c_1 n},$$

which implies that

$$2^n - q(n) 2^{n - c_1 n} > 1.$$

The adversary runs the algorithm A until A makes an equivalence query. The adversary then returns a counterexample and continues running A. Assume that A has just made an equivalence query with the grammar G. Also assume that to this point all the counterexamples returned by the adversary have been of length $3n^2$, and moreover, there is some element L of H_n that is consistent with all the counterexamples returned so far. In order successfully to identify L, the time used by A must be bounded by $p(6n, 3n^2) = q(n)$, and so the size of G must be also be bounded by $q(n)$.

If $(I_n)^n$ is not a subset of $L(G)$, then the adversary returns as a counterexample any element of $(I_n)^n$ that is not contained in $L(G)$. Otherwise, $(I_n)^n \subseteq L(G)$, so by Lemma 13 there exists a string $u \in L(G)$ such that

$$u = x_1 x_2 \cdots x_n$$

where each $x_i \in \Sigma^{3n}$ and for at least $c_1 n$ values of i, $x_i \in F_n$. The adversary returns any such string as the counterexample to the equivalence query.

After $t \leq q(n)$ equivalence queries, any language in H_n that is consistent with all the counterexamples is a possible value for the unknown language. A counterexample from $(I_n)^n$ does not eliminate any language in H_n. Consider the second type of counterexample, a string of the form

$$u = x_1 x_2 \cdots x_n$$

such that each $x_i \in \Sigma^{3n}$ and for at least $c_1 n$ values of i, $x_i \in F_n$. A language $L \in H_n$ is of the form

$$L = L_{i_1}(n) \cdots L_{i_n}(n)$$

where each i_j is 1 or 2. This language is eliminated by the counterexample u if and only if for each j such that $x_j \in F_n$, $x_j \in L_{i_j}(n)$.

Thus, there are at most $2^{n-c_1 n}$ languages in H_n that are eliminated by the counterexample u. Hence, after $t \leq q(n)$ equivalence queries, there remain at least

$$2^n - t2^{n-c_1 n} \geq 2^n - q(n)2^{n-c_1 n} > 1$$

elements of H_n that are consistent with all the counterexamples so far. Hence A must either make more than $q(n) = p(6n, 3n^2)$ equivalence queries, or must not correctly identify some element of H_n, a contradiction. Thus no such A exists, which proves Theorem 11. Q.E.D.

We now turn to the proof of Lemma 13.

4.1 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 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 14 *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 15 *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 14, 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 15. Q.E.D.

4.2 Parse tree surgery

The goal of this subsection is to prove Lemma 13 and thus to conclude the proof of Theorem 11. 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 any 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 $L(G)$. Let $T(w_1, \ldots, w_n)$ be any parse tree for this string with respect to the grammar G. 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 15 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 16 *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 $3n(s-1)$ 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 13.

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

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

where $C = 3168$, $q(n) = 81n^8p(n)^2$, 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 $2^{K \log n + c}$ 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 17 *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 of E_{96} that have as there major segment 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 18 $|E'''| \geq 3n/8C$.

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 12 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 13 is finally proved, with $c_1 = 3/8C = 3/25344$. Q.E.D.

5 Comments

The proofs of Theorems 3, 8, and 11 depend only on the fact that the number and size of the hypotheses conjectured by algorithm A are bounded by $p(n, m)$. Thus, even an oracle for a PSPACE-complete set would not help A.

Since the "hard classes" H_n are finite and fixed-length, our proofs also show that dfas, nfas, and cfgs for finite and fixed-length languages cannot be learned in polynomial time using only equivalence queries. See also the reduction of the general case to the finite and fixed-length case for dfas in [4].

These arguments do NOT show that these classes are not polynomial-time learnable in Valiant's model of *pac*-identification [7] or in the equivalent model of prediction [5]. 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 becomes *pac*-learnable, by the Occam's razor technique [3], 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.

What do these results mean in practice? Interpreted in the domain of "passive" prediction, that is, with no access to experiments, these results suggest that in some interesting domains it is imprudent to use a single hypothesis as the basis for predictions, that a mixed strategy based on a version-space approach is possibly preferable. (Formally, with an oracle for a #P-complete problem we can implement an efficient majority-vote prediction strategy for dfas, but even a PSPACE oracle doesn't help in the case of equivalence queries.)

6 Acknowledgements

This research was funded by the National Science Foundation, under grant number IRI-8718975. The papers [4,6] and conversations with their authors contributed greatly to the work herein.

References

[1] D. Angluin. Learning regular sets from queries and counterexamples. *Information and Computation*, 75:87–106, 1987. Preliminary version appeared as YALEU/DCS/RR-464.

[2] D. Angluin. Queries and concept learning. *Machine Learning*, 2:319–342, 1987. Preliminary version appeared as YALEU/DCS/RR-479.

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

[4] O. Ibarra and T. Jiang. *Learning regular languages from counterexamples.* Technical Report, University of Minnesota Computer Science Dept., TR 88-33, 1988.

[5] N. Littlestone. Learning quickly when irrelevant attributes abound: a new linear-threshold algorithm. *Machine Learning*, 2:285–318, 1988.

[6] S. Porat and J. Feldman. *Learning automata from ordered examples.* Technical Report, University of Rochester, Computer Science Dept., TR 241, 1988.

[7] L. G. Valiant. A theory of the learnable. *C. ACM*, 27:1134–1142, 1984.