The Query Complexity of Learning DFA *

José L. Balcázar *Josep Díaz* *Ricard Gavaldà*
Dept. Llenguatges i Sistemes Informàtics
Universitat Politècnica Catalunya
Pau Gargallo 5, 08028 Barcelona, Spain

Osamu Watanabe
Department of Computer Science
Tokyo Institute of Technology
Meguro-ku, Tokyo 152, Japan

Abstract. It is known that the class of deterministic finite automata is polynomial time learnable by using membership and equivalence queries. We investigate the query complexity of learning deterministic finite automata, i.e., the number of membership and equivalence queries made during the process of learning. We extend a known lower bound on membership queries to the case of randomized learning algorithms, and prove lower bounds on the number of alternations between membership and equivalence queries. We also show that a trade-off exists, allowing us to reduce the number of equivalence queries at the price of increasing the number of membership queries.

1. Introduction

Query learning was introduced by Angluin [1] and is currently one of the most important models in computational learning theory. It differs from other models, such as inductive inference [5] or certain PAC-learning algorithms [10], in that the learning process, the

*This research was partially supported by the ESPRIT II Basic Research Actions Program of the EC under contract No. 3075 (project ALCOM). While this research was done, the third author was visiting the University of California at Santa Barbara, supported in part by the National Science Foundation under grant CCR89-13584. The fourth author was supported in part by Takayanagi Foundation of Electronics and Science Technology. E-mail addresses: balqui@lsi.upc.es, diaz@siva.upc.es, gavalda@lsi.upc.es, watanabe@cs.titech.ac.jp

learner, obtains information about the concept to learn by making *queries* to some *teacher*, instead of passively receiving examples.

Variants of the formalization of learning via queries have been proposed in [11, 12]. We are interested here in the notion of "bounded learning" described there. In bounded learning, the learning algorithm is given a number as an input parameter. The goal of the algorithm is to output some hypothesis that only needs to be correct up to the length indicated by the input parameter. This learning notion is somewhat different from the original notion studied in Angluin's papers, but it allows us to avoid tedious and minor problems in the original notion. (See [12] for the justification of the bounded learning notion.)

It must be mentioned that all the concept classes used in this paper are finite and have a fixed length, so that the negative results also hold under Angluin's learning notion. Additionally, our learning algorithms for positive results also achieve exact learning.

The formalization of the concept of learning is particularly useful since a substantial gain of understanding comes from the possibility of relating it to various concepts from Computational Complexity. In particular, there are negative results for learning that rely on widely believed complexity-theoretic hypothesis, such as R $\neq$ NP or the existence of cryptographic one-way functions [7, 4]. Additionally, the idea of considering queries as a resource allows one to prove absolute negative results, whose proofs are independent of the learner's computational power: they are based instead on bounding the number of queries asked, and do not rely on any assumption. This contrasts with the negative results that depend on additional hypothesis. All our results here are absolute in this sense.

One of the successful fields in query learning is the problem of constructing a deterministic finite automaton (henceforth dfa) from information about the set it accepts. This is the problem we study in this paper. Pitt [8] surveys the status of this important problem in several learning models. For the case of query learning, Angluin proved an important positive result:

Proposition 1.1. [1] There exists a polynomial time algorithm that constructs a dfa using membership and equivalence queries.

Algorithms using these two kinds of queries will be called here *(Mem,Equ)-learners*. When only membership or only equivalence queries are allowed to be asked by the algorithm, we will call it a (Mem)-learner, respectively (Equ)-learner. Angluin showed that

neither membership queries alone nor equivalence queries alone are good enough to learn dfa in polynomial time.

Proposition 1.2.
(1) [2] No polynomial time (Mem)-learner exists for dfa.
(2) [3] No polynomial time (Equ)-learner exists for dfa.

Actually, it is easy to see that Angluin's dfa learning algorithm (witnessing Proposition 1.1) needs at most n equivalence queries for learning n state dfa. In this paper we investigate the *query complexity* of learning algorithms for dfa, i.e. the number of membership and equivalence queries used or needed to learn a fixed dfa.

First, we extend the negative result on (Mem)-learners to the case in which the learning algorithm has access to a source of random bits and is required to operate within a certain error probability bound. We study the number of membership queries necessary for such *randomized* learners to learn dfa by using only membership queries. We show that randomized computation, the ability to "toss coins" during the computation, cannot improve over the deterministic lower bound.

Second, we study the number of times that a learner must alternate between membership and equivalence queries. We impose limitations on this "alternations" resource, as a natural generalization of the learners that only use one of the two sorts of queries —which alternate 0 times—, and show that this restriction implies an increase of the total number of queries. For instance, as a consequence of this result, we show that in order to learn an n-state dfa in polynomial time, membership and equivalence queries must alternate at least $\Omega(n/\log^2 n)$ times.

Finally, we study whether it is possible to reduce the number of one type of queries, maybe at the expense of the other. The learning formalism does not take into account how the queries are answered; but it is intuitively clear that, for many representation classes, answering a membership query can be substantially easier than answering an equivalence query. For instance, in the dfa case, evaluating a dfa on a word is one of the simplest problems in complexity theory, while deciding the equivalence of two dfa is complete for nondeterministic logspace. We prove that the number of such expensive queries can be reduced to some extent. More precisely, when only a bound on the total number of membership and equivalence is given, a certain type of "trade-off" between the number of membership and equivalence queries occurs. We show that it is possible to reduce the number of equivalence of queries, say, to $n/f(n)$ while increasing that of membership queries by a factor of $2^{f(n)}$. On the other hand, we also prove that in order

to reduce equivalence queries to $n/f(n)$ one has to increase the number of membership queries by a factor of $2^{\Omega(f(n))}$. Thus, the above $2^{f(n)}$ increase is essential. For example, we can construct a polynomial time dfa learning algorithm that asks $n/c\log n$ equivalence queries, but it is impossible to reduce equivalence queries more than a factor of $O(\log n)$ without using a superpolynomial number of membership queries.

2. Preliminaries

In this paper we follow standard definitions and notations in formal language theory and computational complexity theory; in particular, those for finite automata are used without definition. The reader will find them in standard textbooks such as [6].

Let Σ denote $\{0,1\}$, and throughout this paper, we use Σ as our alphabet. For any set A of strings, let $\overline{A}$ to denote the complement of A, i.e., $\Sigma^* - A$. For any sets A and B of strings, let $A \triangle B$ denote the set $(A - B) \cup (B - A)$. The length of a string x is denoted by $|x|$. The cardinality of a finite set A is written as $\|A\|$. Symbols $A^{\leq m}$ and $A^{=m}$ are used to denote the sets $\{x \in A : |x| \leq m\}$ and $\{x \in A : |x| = m\}$ respectively.

Notions and Notations for Query Learning

We briefly explain the notions and notations for discussing query learning formally. We basically follow the style established in [11, 12].

A learning problem is specified as a "representation class" [9]. A *representation class* is a triple (R, Φ, ρ), where $R \subseteq \Sigma^*$ is a *representation language*, $\Phi : R \to 2^{\Sigma^*}$ is a *semantic function* or *concept mapping*, and $\rho : R \to \mathbf{N}$ is a *size function*. For example, a representation class for dfa[1] is formally defined as follows: $DFA = (R_{\rm dfa}, \Phi_{\rm dfa}, \rho_{\rm dfa})$, where $R_{\rm dfa}$ is the set of dfa that are encoded in Σ^*, and for any $r \in R_{\rm dfa}$, $\Phi_{\rm dfa}(r)$ and $\rho_{\rm dfa}(r)$ are respectively the regular language accepted by the dfa (represented by) r and the number of states in the dfa (represented by) r. Following common convention, we write $\Phi_{\rm dfa}(r)$ as $L(r)$ and $\rho_{\rm dfa}(r)$ as $|r|$.

The encoding $R_{\rm dfa}$ is assumed to be honest, i.e. not much longer than necessary; in particular, we assume that the encoding of a dfa is polynomially long in the number of states.

Our computation model for learning is the "learning system". A *learning system* $\langle S, T \rangle$ is formed by a *learner* S and a *teacher* T that are organized as in Figure 1.

[1]By "deterministic finite automaton" we mean a "complete" deterministic finite automaton over Σ^*. That is, we assume that the transition function is total.

Figure 1: Learning System

The tapes except the communication tape and the target tape are used in a ordinary way. The communication tape is a read-write tape and used for the communication between S and T. That is, the queries from S and the answers from T are written on it. The target tape is a read-only tape, and its role is to let the teacher T know a *target concept*, a set to be learned. That is, a representation r of a target concept is written on the target tape; this situation intuitively means that T knows the concept that is represented by r. A teacher T who knows r (or, more precisely, T with r on the target tape) is written as "$T(r)$". Prior to the execution, the input ω and a target representation r are given respectively on the input tape and the target tape. Then the computation of $\langle S, T\rangle$ (which is written as $\langle S, T(r)\rangle(\omega)$) starts from S, executes S and T in turn, and finally halts at S. If S outputs y on its output tape and halts normally, then we say that $\langle S, T(r)\rangle(\omega)$ *outputs* y (and write $\langle S, T(r)\rangle(\omega) = y$).

In our framework, T is regarded as a function while S is regarded as some algorithm, or a Turing machine. That is, we omit considering T's computation and assume that T can *somehow* answer to queries. Note that S could be a randomized algorithm; but unless it is explicitly stated, S is considered as a deterministic algorithm.

For query types, we consider membership query (Mem) and equivalence query (Equ). When learning dfa, a *membership query* is to ask whether a queried string is accepted by the target machine, and an *equivalence query* is to ask whether a queried dfa is equivalent to the target machine. Thus, for each membership query, some string w is queried to the teacher, and the teacher is supposed to answer "yes" if $w \in L(r)$, and "no" otherwise. For each equivalence query, some dfa r' is queried to the teacher, and the teacher is supposed to answer "yes" if $L(r') = L(r)$; on the other hand, in the case $L(r') \neq L(r)$, the teacher must provide a *counterexample*, some string in the difference $L(r') \triangle L(r)$, to the query. A learner is called, e.g., *(Mem,Equ)-learner* if it asks membership and equivalence queries, and a teacher is called, e.g., *(Equ)-teacher* if it answers only to equivalence queries. A

tuple such as (Mem,Equ) is called a *query-answer* type[2].

Now we are ready to define our "learnability" notion. To simplify our discussion, we explain and define notions by using (Mem,Equ) for a typical query-answer type. However, these notions are defined similarly for other query-answer types.

In this paper, we consider only "bounded learning", which has been introduced in [11, 12] as one reasonable query learning notion. Intuitively, in the bounded learning, for a given parameter $m \geq 0$, the goal of a learner is to obtain a representation that denotes a target set up to length m. The parameter m is called a *length bound*. On the other hand, though we assume that teachers provide correct answers up to a given length bound, answers may not be correct if they are out of the length bound. By considering length bounds, we can avoid many tedious difficulties that come with the original and more general learning notion. Furthermore, bounded learning is well-motivated, and it is not just an artificial notion. Thus we use this learning notion throughout this paper. (Hence "bounded" is often omitted.) It should be noted, however, that every proof in this paper works even in the original query learning notion.

Let us define "bounded learning" more precisely. For any target representation r and, for a given equivalence query r', we say that $T(r)$ answers r' correctly up to length m if T gives a counterexample if it exists in $\Sigma^{\leq m}$ and answers "yes" otherwise. A teacher T is called a (*consistent*) *bounded (Mem,Equ)-teacher* for *DFA* if for given target representation r and length bound m, $T(r)$ answers each membership query correctly w.r.t. r, and $T(r)$ answers each equivalence query correctly up to length m. By considering a bounded teacher, we can avoid the case where a learner is given unnecessarily long counterexamples and the case where a learner abuses the teacher's power of searching through an infinite number of strings.

The value of m will be provided to the learning system as a part of the common input. Another part of the common input will be a value n, which is understood as a bound on the size of the output description to be written by the learner. This convention allows us to measure the time bound in terms of the input, as is customary in complexity theory.

Definition 2.1. [12] A (Mem,Equ)-learner S *learns* $C = (R, \Phi, \rho)$ (or C is *learned* by S) *in the bounded learning sense* if for every bounded (Mem,Equ)-teacher T for C, every $r \in R$, every $n \geq \rho(r)$, and every $m \geq 0$,

[2]The notation for query-answer types used in [11, 12] is more complicated in order to denote a finer query-answer type classification, including other query types. However, such classification is not necessary here; thus, we use this simpler notation.

$$\langle S, T(r)\rangle(n,m) = r' \text{ such that } \Phi(r')^{\leq m} = \Phi(r)^{\leq m}.$$

Remark.

(1) Notice that the definition does not include the case where $n < \rho(r)$ is given as input for learning r. In other words, a learner can output anything in such a case. Thus, for specifying a learning algorithm, it is enough to consider the case that $n \geq \rho(r)$.

(2) In the later discussion, we assume that some additional parameter is given as an input. In such a case, the above and following definitions are extended naturally.

In this paper we consider sometimes randomized learners. These are algorithms that have access to a source of random bits. Each possible sequence of outcomes of the random bits may lead to a different computation path of the algorithm, and therefore we define the result of the algorithm in terms of the probability that the random bits lead to a successful computation path. For randomized learners we use the following definition.

Definition 2.2. A randomized (Mem,Equ)-learner S *learns* $C = (R, \Phi, \rho)$ *with success probability* $\geq \delta$ if for every bounded (Mem,Equ)-teacher T for C, every $r \in R$, every $n \geq \rho(r)$, and every $m \geq 0$,

$$\Pr\{\ \langle S, T(r)\rangle(n,m) = r' \text{ such that } \Phi(r')^{\leq m} = \Phi(r)^{\leq m}\ \} \ \geq\ \delta.$$

Now define the polynomial time learnability in the bounded learning sense. A learner is *polynomial time* if for some polynomial p and for all inputs $\langle n, m\rangle$, it halts within $p(n+m)$ steps. A representation class C is *polynomial time (Mem,Equ)-learnable in the bounded learning sense* if C is learnable by some polynomial time (Mem,Equ)-learner.

Finally, we define "query complexity". Intuitively, the "query complexity" is the number of queries asked by S in the worst case. More precisely, for any learner S for *DFA*, the query complexity $\#query_S$ is defined as follows: Let $\mathcal{T}$ be the family of bounded teachers for *DFA* of S's query-answer type. For any $T \in \mathcal{T}$, any $r \in R_{\rm dfa}$, and any $n, m \geq 0$, let $\#query_{\langle S,T(r)\rangle}(n,m)$ be the number of queries asked during the computation $\langle S, T(r)\rangle(n,m)$. Now for any $n, m \geq 0$,

$$\#query_S(n,m) \ =\ \max\{\ \#query_{\langle S,T(r)\rangle}(n,m)\ :\ T \in \mathcal{T}, r \in R_{\rm dfa}\ \}.$$

For a randomized learner S, a teacher T, and a representation r, $\#query_{\langle S,T(r)\rangle}(n,m)$ is defined as the average number of queries taken over all possible randomized computations

of $\langle S, T(r)\rangle(n,m)$. Then $\#query_S(n,m)$ is defined exactly as above. Membership query complexity $\#mem\text{-}query_S$ and equivalence query complexity $\#equ\text{-}query_S$ are defined similarly.

We are also interested in the alternation complexity of a (Mem,Equ)-learner S. Let $\mathcal{T}$ be the family of bounded (Mem,Equ)-teachers. For any $T \in \mathcal{T}$, any $r \in R_{\rm dfa}$, and any $n, m \geq 0$, let $\#alt_{\langle S,T(r)\rangle}(n,m)$ be the number of times that S changes from membership to equivalence queries or vice-versa during the computation $\langle S, T(r)\rangle(n,m)$. Now for any $n, m \geq 0$,

$$\#alt_S(n,m) \;=\; \max\{\; \#alt_{\langle S,T(r)\rangle}(n,m) \;:\; T \in \mathcal{T}, r \in R_{\rm dfa} \;\}.$$

3. Randomized Learners

We investigate the number of membership queries necessary by a randomized (Mem)-learner for *DFA*.

Theorem 3.1. For any δ, $0 < \delta \leq 1$, let S be any randomized (Mem)-learner S that learns *DFA* with success probability $\geq \delta$. Then for any $k > 0$, we have the following bound:

$$\#mem\text{-}query_S(k+2,k) \;\geq\; \delta 2^k - 1.$$

Remark. Proposition 1.2 (1) is a special case, i.e., $\delta = 1$, of this theorem.

Proof. Consider any δ, $0 < \delta \leq 1$. We show that any learner with query complexity better than the above lower bound cannot learn *DFA* with success probability $\geq \delta$.

Let T_0 be a (Mem)-teacher for *DFA*. Let S be any (Mem)-learner, and suppose that for some $k > 0$, $\#mem\text{-}query_S(k+2,k) < \delta 2^k - 1$. We consider the problem of learning the empty set or singleton sets $\{w\}$, where $|w| = k$. Note that there is some dfa with $k+2$ states for the empty set. Also for every $w \in \Sigma^k$, there is some dfa with $k+2$ states that accepts $\{w\}$. (Recall that we are considering complete dfa; thus an "error state" is necessary.) The representation of a dfa for the empty set is denoted as $r_\emptyset$, and the one for $\{w\}$ is denoted as r_w. We show that for some $w \in \Sigma^k$, the probability that $\langle S, T_0(r_w)\rangle(k+2,k)$ outputs a correct answer (i.e., some dfa representation of $\{w\}$) is less than δ.

Now consider the execution $\langle S, T_0(r_\emptyset)\rangle$ on input $(k+2,k)$. For any string $w \in \Sigma^k$, we say that w is *δ-well treated* if

$$\Pr\left\{(*)\quad \begin{array}{l}\text{either } \langle S, T_0(r_\emptyset)\rangle(k+2,k) \text{ queries } w,\\ \text{or } \langle S, T_0(r_\emptyset)\rangle(k+2,k) \text{ outputs some } r \text{ s.t. } L(r)=\{w\}\end{array}\right\} \geq \delta.$$

Where the probability is taken over the learner's randomized computation. That is, the probability is the proportion of randomized computations of S on which $(*)$ holds. We say that a randomized computation of S *treats* w if $(*)$ holds for w on the computation.

Let w be any string that is not δ-well treated. That is, more than $1-\delta$ of randomized computations of S do not treat w. Notice that on such a computation, S cannot distinguish which of $r_\emptyset$ and r_w is given as a target because w is not queried; furthermore, S yields some description r such that $L(r) \neq L(r_w)$. That is, $\langle S, T_0(r_w)\rangle(k+2,k)$ outputs a wrong answer on the computation. Hence, $\Pr\{\ \langle S, T_0(r_w)\rangle(k+2,k)$ outputs a correct answer $\} < \delta$. On the other hand, the following claim states that the number of δ-well treated strings is not large enough.

Claim. The number of δ-well treated strings is at most $\dfrac{\#mem\text{-}query_S(k+2,k)+1}{\delta}$.

Proof. Let N be the number of possible random computation paths of $\langle S, T_0(r_\emptyset)\rangle$ on input $(k+2,k)$. Recall that $\#mem\text{-}query_S(k+2,k)$ is the average number of queries of S on input $\langle k+2,k\rangle$, taken over all randomized computations; this implies that the total number of queries among all computations is at most $\#mem\text{-}query_S(k+2,k) \times N$. Also, each randomized computation can treat at most the number of queries it asks plus 1. Hence, the total number of strings treated on all computation paths is at most $(\#mem\text{-}query_S(k+2,k)+1) \times N$. On the other hand, a δ-well treated string must be treated by at least $\delta \times N$ paths. Therefore, the number of δ-well treated strings is at most $(\#mem\text{-}query_S(k+2,k)+1)/\delta$. □ Claim

Now from the assumption $\#mem\text{-}query_S(k+2,k) < \delta 2^k - 1$, it is clear that some $w \in \Sigma^k$ is not δ-well treated. Therefore, we have some w such that S cannot learn r_w with success probability $\geq \delta$. □

From this theorem, we immediately have the following negative result.

Corollary 3.2. For any δ, $0 < \delta \leq 1$, no polynomial time randomized (Mem)-learner S exists that learns *DFA* with success probability $\geq \delta$.

4. Query Alternations

Here we consider the case where the number of alternations between membership and equivalence queries is limited.

Theorem 4.1. Let $n_{a,k}$ and $m_{a,k}$ denote $(3k^2+2)(a+1)$ and $2k^2(a+1)$ respectively. For every constant $c > 1$ there is some constant $c' > 0$ such that for every (Mem,Equ)-learner S for *DFA*, every a, and every sufficiently large k,

- either S on input $\langle n_{a,k}, m_{a,k}\rangle$ asks some equivalence query with a dfa of size greater than $2^{c'k}$,
- or $\#alt_S(n_{a,k}, m_{a,k}) \geq a+1$,
- or $\#equ\text{-}query_S(n_{a,k}, m_{a,k}) \geq c^k$,
- or $\#mem\text{-}query_S(n_{a,k}, m_{a,k}) \geq 2^k - 1$.

Remark. Proposition 1.2 (2) is proved as a special case of this theorem.

It is shown that if we cannot use both membership and equivalence queries, and only one type of queries are allowed, then exponential number of queries are necessary to learn dfa. (The case where membership queries are allowed is discussed in [2], and the case where equivalence queries are allowed is discussed in [3].) Our theorem shows a similar lower bound when the number of alternations between membership and equivalence queries are bounded by constant, and thus an extension of these previous results. In fact, we will prove the theorem by merging two proofs in [2, 3].

First we recall some definitions and facts from [3]. For any $k > 0$, and any i, $1 \leq i \leq n$, define $L(i,k)$ to be the set of strings of length $2k$ whose ith bit is equal to the $(k+i)$th bit. Consider any set $L(i_1,k)L(i_2,k)\cdots L(i_k,k)$, where $1 \leq i_1, ..., i_k \leq k$. The words in this set have length $m_{0,k} = 2k^2$. It is easy to show that the set is accepted by some dfa with $n_{0,k} = 3k^2+2$ states. Let R_k denote the set of dfa representations r such that $|r| = n_{0,k}$ and $L(r) = L(i_1,k)L(i_2,k)\cdots L(i_k,k)$ for some $1 \leq i_1, ..., i_k \leq k$. From the above discussion, any set of the form $L(i_1,k)L(i_2,k)\cdots L(i_k,k)$ has a dfa representation in R_k; thus, $\|R_k\| = k^k$.

The following lemma, which states the lower bound of the number of equivalence queries for learning $r \in R_k$, plays a key role for proving our theorem.

Lemma 4.2. For any constant $c > 1$ there is a constant $c' > 0$ with the following property. Let S be any (Equ)-learner for *DFA* such that S on input $\langle n_{0,k}, m_{0,k}\rangle$ never asks a dfa (as an equivalence query) with more than $2^{c'k}$ states. Then $\#equ\text{-}query_S(n_{0,k}, m_{0,k}) \geq c^k$, for all sufficiently large k.

Proof. The proof is immediate from the argument in [3]. Here we review some important facts and state the proof outline.

Let S be defined as in the lemma, and let $c > 1$ be any constant. For any $k > 0$, define the following sets:

$$A_k = \{\ x_1x_1x_2x_2\cdots x_kx_k\ :\ \forall i, 1 \le i \le k\ [\, x_i \in \Sigma^k\,]\ \},$$
$$B_{c,k} = \{\ x_1y_1x_2y_2\cdots x_ky_k\ :\ \forall i, 1 \le i \le k\ [\, x_i, y_i \in \Sigma^k \wedge d(x_i, y_i) > (1-1/c)k\,]\ \},$$

where $d(x, y)$ is the Hamming distance of x and y.

The following facts can be shown as in [3].

Fact 1.

(1) $A_k = \bigcap\{\ L(r)\ :\ r \in R_k\ \}$.

(2) For any $w \in B_{c,k}$, $\|\{\, r \in R_k\ :\ w \in L(r)\,\}\| \le \left(\frac{k}{c}\right)^k$.

Fact 2. For every c there is some $c' > 0$ such that for any sufficiently large k, and for any dfa M with at most $2^{c'k}$ states, if M accepts all strings in A_k, then it accepts some string in $B_{c,k}$.

We define a teacher T_1 that answers an equivalence query $r \in R_{\mathrm{dfa}}$ in the following way:

- So long as there are any "positive" counterexamples from A_k (i.e., strings in $A_k - L(r)$), T_1 returns one of them as a counterexample.
- Else if there are any "negative" counterexamples from $B_{c,k}$ (i.e., strings in $L(r) - B_{c,k}$), then T_1 returns one of them as a counterexample.
- Otherwise, T_1 returns some counterexample within the length bound, or returns "yes" if no counterexample exists.

Let k be any sufficiently large integer for which Fact 2 holds. We show that S needs to ask at least c^k equivalence queries to learn *some* $r_* \in R_k$ from T_1, where r_* will be determined through our discussion. Now consider the execution of $\langle S, T_1(r_*)\rangle(n_{0,k}, m_{0,k})$. This process is regarded as identifying r_* among the potential candidates. Clearly, at the beginning, every $r \in R_k$ is candidate, and for getting a correct answer for r_*, it is necessary to reduce the number of candidates to 1. We show that in order to achieve this goal, the execution needs at least c^k equivalence queries for some r_*. (In the following discussion, we assume that a candidate set is a subset of R_k. A real candidate set may contain other representations, but this only increases the number of queries.)

Let r_1 be S's first query in the execution. Suppose (the dfa represented by) r_1 does not accept some strings in A_k. Then T_1 returns one of them w as a positive counterexample. (I.e., w witnesses $L(r_*) - L(r_1) \neq \emptyset$.) But every $r \in R_k$ accepts w, so we cannot reduce the number of candidates by this counterexample. On the other hand, suppose that r_1 accepts every string in A_k. Then from Fact 2, it must accept some string u in $B_{c,k}$. Here we can assume that r_* is chosen so that it does not accept u. (Because the number of $r \in R_k$ that accepts u is at most $(k/c)^k << \|R_k\|$.) That is, $(L(r_1) - L(r_*)) \cap B_{c,k}$ contains at least one element, i.e., u. Then T_1 answers one of them w as a negative counterexample. (I.e., w witnesses $L(r_1) - L(r_*) \neq \emptyset$.) But by this counterexample, we can reduce the number of candidates by at most $(k/c)^k$. For, w is a negative counterexample to at most $(k/c)^k$ representations in R_k. By a similar argument, the second, third, ... counterexamples kill at most $(k/c)^k$ candidates each (if r_* is chosen appropriately). Thus, after q queries, at least $k^k - q(k/c)^k$ candidates are left. Hence, in order to have $k^k - q(k/c)^k \leq 1$, q must satisfy $q \geq c^k - (c/k)^k > c^k - 1$. That is, $q \geq c^k$. $\square$

Proof of Theorem 4.1. Assume for contradiction that there is an integer a such that for some (Mem,Equ)-learner S and for infinitely many k, we have

- S on input $\langle n_{a,k}, m_{a,k} \rangle$ never asks a dfa with more than $2^{c'k}$ states,
- $\#alt_S(n_{a,k}, m_{a,k}) < a + 1$,
- $\#equ\text{-}query_S(n_{a,k}, m_{a,k}) < c^k$, and
- $\#mem\text{-}query_S(n_{a,k}, m_{a,k}) < 2^k - 1$.

Select such a to be the minimum with this property. We will contradict this minimality.

First observe that a is not 0. For $a = 0$, no alternation occurs. Thus, the learner is either a (Mem)-learner, and the lower bound essentially follows from Theorem 3.1 (where $\delta = 1$), or is an (Equ)-learner, and then the lower bound follows from Lemma 4.2. More precisely, for (Mem)-learners, considering the class of dfa accepting a set of the form $\{w\}$ for some string w of length $k+1$, at least $2^k - 1$ queries are necessary. On the other hand, for (Equ)-learners, considering the class R_k, at least c^k queries are necessary. Notice that these dfa have at most $n_{0,k}$ states and accept only strings of length $m_{0,k}$. Thus, both lower bound results hold for any input $\langle n_{0,k}, m_{0,k} \rangle$ if k is sufficiently large.

Now we have that $a \geq 1$. Let $R_{a,k}$ be the set of all dfa representations with at most $n_{a,k}$ states accepting only strings of length at most $m_{a,k}$. We distinguish two cases depending on the type of the first query of S on input $\langle n_{a,k}, m_{a,k} \rangle$.

Case 1: membership queries are asked first.

Consider sets of representations for languages of the form $L = wL(r)$, where $r \in R_{a-1,k}$ and $|w| = k$. These representations have size at most $k+2+n_{a-1,k} \leq n_{a,k}$, and the length of the strings they accept is bounded by $k + m_{a-1,k} \leq m_{a,k}$. Hence, they are in $R_{a,k}$.

Simulate the initial membership query phase of S answering always "no". The number of queries is less than the total number of membership queries; hence, at the end of the phase there is still some w_0 of length k that has never appeared as a prefix of any query in the phase. Now from our assumption, S can learn the representations for $w_0L(r)$ where $r \in R_{a-1,k}$. Then we can modify S to a learner S_0 that learns $R_{a-1,k}$ with $a-1$ query alternations, which contradicts the minimality of a. (S_0 learns $L(r)$ just like S learns $w_0L(r)$. Notice that all membership queries of S before its first equivalence queries can be answered "no"; thus, S_0's first query corresponds to S's first equivalence query.)

Case 2: equivalence queries are asked first.

We now consider representations for sets of the form $0L(r) \cup 1L(r')$, where $r \in R_{a-1,k}$ and r' is in R_k, the class used in Lemma 4.2. The representations for $0L(r) \cup 1L(r')$ have size[3] at most $n_{a-1,k}+(3k^2+2) = n_{a,k}$, and the length of the strings they accept is bounded by $m_{a-1,k} + 1 \leq m_{a,k}$. Hence, they are in $R_{a,k}$ again.

For the first phase of equivalence queries of S, use a teacher that answers while possible with counterexamples for the $1L(r')$ part. By the bound on the number of equivalence queries, we know that after this phase there remain at least two representations in R_k that S cannot distinguish. Moreover, during the process, S has obtained only counterexamples beginning with 1. Now from our assumption again, S must be able to learn the part $0L(r)$ correctly after this phase. Thus a trivial modification of S learns $R_{a-1,k}$ in $a-1$ alternations, contradicting again the minimality of a. □

The following negative result is easy to obtain from the theorem.

Theorem 4.3. There is no polynomial time (Mem,Equ)-learner for *DFA* that alternates $o(\frac{n}{\log^2 n})$ times between membership and equivalence queries.

Proof. Consider any infinite sequence $\{n_i\}_{i\geq 0}$ of natural numbers such that for each n_i there are a_i and k_i with the properties:

- $n_i = (3k_i^2 + 1) \cdot (a_i + 1)$,

[3]It seems that we need $1 + n_{a-1,k} + (3k^2 + 2)$ states. But we can merge the final states of r and r', thereby reducing one state.

- $a_i = o(n_i/\log^2 n_i)$ (as a function of i).

Note that $k_i = \omega(\log n_i)$. We will show that no polynomial time (Mem,Equ)-learner for DFA can alternate less than a_i times to learn dfa with n_i states. The theorem follows if we can prove this for any sequence $\{n_i\}_{i\geq 0}$ with these properties.

Take any learner S that runs in polynomial time, and for each n_i in the sequence consider the behavior of S with input $\langle n_i, 2k_i(a_i+1)\rangle$. By Theorem 4.1, one of the following facts holds for some target dfa with n_i states:

1. either S asks an equivalence query of size at least $2^{c'k_i}$,
2. or S alternates more than a_i times,
3. or S asks at least c^{k_i} equivalence queries,
4. or S asks at least $2^{k_i}-1$ membership queries,

where $c > 1$ and $c' > 0$ are constants. If cases 1, 3, or 4 hold for infinitely many i, then the running time of S is $d^{k_i} = d^{\omega(\log n_i)}$, for infinitely many i and the constant $d = \min\{2^{c'}, c, 2\} > 1$. This contradicts the assumption that S runs in polynomial time. Hence, case 2 must hold for all but finitely many i and we are done. □

5. Trade-off Between the Number of Membership and Equivalence Queries

In this section, we consider the general case; that is, no restriction (except the number of queries) is assumed on the way of asking membership and equivalence queries. We show some trade-off relation between the number of membership and equivalence queries.

Let us consider the performance of Angluin's query learning algorithm [1] for *DFA*. Suppose that the algorithm is to learn a n state dfa within a length bound m. Then it is easy to see that the algorithm asks at most n equivalence queries and a polynomial number of membership queries. Here we improve the equivalence query complexity while spending some more membership queries. More specifically, our improved algorithm takes $\langle n, m, h\rangle$ as input and learns a target dfa in the bounded learning sense, while asking n/h equivalence queries and $2^h \cdot p_1(n+m)$ membership queries, where p_1 is some fixed polynomial. Furthermore, the algorithm runs in polynomial time w.r.t. the number of queries.

Theorem 5.1. There is a (Mem,Equ)-learner S_0 for *DFA* with the following complexity: for every $n, m, h > 0$,

(a) $\#\mathit{equ\text{-}query}_{S_0}(n,m,h) \leq \frac{n}{h}$,
(b) $\#\mathit{mem\text{-}query}_{S_0}(n,m,h) \leq 2^h \cdot p_1(n+m)$, and
(c) S_0 on input $\langle n,m,h\rangle$ halts within time $p_2(\#\mathit{query}_{S_0}(n,m,h))$,
where p_1 and p_2 are polynomials depending on S_0.

Remark. The upper bound for the membership query complexity depends on the choice of our alphabet, i.e., $\Sigma = \{0,1\}$. More in general, we have $\#\mathit{mem\text{-}query}_{S_0}(n,m,h) \leq \|\Sigma\|^h \cdot p_1'(n+m+\|\Sigma\|)$.

Our algorithm is a generalization of Angluin's algorithm. Let us recall some facts about Angluin's algorithm. Angluin's algorithm uses *observation table* for constructing hypothesis. An observation table is a tuple (S,E,T), where S and E are finite and prefix-closed sets, and T maps $(S \cup S\cdot\Sigma)\times E$ to $\{0,1\}$. For each $s \in S \cup S\cdot\Sigma$ and $e \in E$, $T(s,e)$ is set 0 if $s\cdot e$ is not in the target set, and 1 if $s\cdot e$ is in the target set. At certain points, the algorithm builds a dfa $M = M(S,E,T)$ (i.e., a hypothesis) from the table, and presents M to the teacher as an equivalence query. If the answer is "yes", the algorithm halts. Otherwise, it uses a received counterexample to expand S, E, and T, in a way such that the next equivalence query must have at least one more state than the previous one. Furthermore, the algorithm has the following property: If the target set is accepted by a n state dfa, then when the constructed hypothesis has n states at some point, it must accept exactly the target language. From these properties, it is clear that Angluin's algorithm needs at most n equivalence queries.

In an observation table, the part corresponding to $S\cdot\Sigma\times E$ is used for determining M's move when reading one symbol. We will extend this part to $S\cdot\Sigma^{\leq h}\times E$ so that M's move after reading up to h symbols can be determined from this part. It will be shown that if dfa is constructed from such a table, then at least h states are added between each two consecutive equivalence queries. Thus, our learning algorithm, which uses this extended observation table, needs at most n/h equivalence queries. By this expansion, however, the algorithm needs to ask more membership queries to fill in the table. This is the idea of our generalization.

Proof. We first define some notions and notations. In the following discussion, we use the same symbols as above. Here, as in [1], we use $\mathit{row}(s)$, where $s \in S\cdot\Sigma^{\leq h}$, to denote the finite function mapping each $e \in E$ to $T(s,e)$. In other words, $\mathit{row}(s)$ is the sth row of observation table (S,E,T). Our learning algorithm uses *observation table with lookahead* h. It is a tuple (S,E,T) as before, but table T maps $(S\cdot\Sigma^{\leq h})\times E$ to $\{0,1\}$. Note that Angluin's tables are tables with lookahead 1. Angluin used the notions of

"closed observation table" and "consistent observation table". These notions are extended naturally here. That is, an observation table with lookahead h is called *closed* if for every $s \in S$ and every $u \in \Sigma^{\leq h}$, there is some $s' \in S$ such that $row(s \cdot u) = row(s')$. An observation table with lookahead h is called *consistent* if for every pair of s_1 and $s_2 \in S$, if $row(s_1) = row(s_2)$, then $row(s_1 \cdot u) = row(s_2 \cdot u)$ for all $u \in \Sigma^{\leq h}$.

Now describe our learner S_0. It is almost the same as Angluin's algorithm, except that it uses an observation table with lookahead h. This requires filling $\|\Sigma^{\leq h}\|$ entries in T with membership queries each time that S increases.

Clearly, this modification does not affect the correctness of the learner; that is, like Angluin's learning algorithm, S_0 learns *DFA* correctly. Furthermore, maintaining this additional information roughly increases the number of necessary membership queries by $\|\Sigma^{\leq h}\| \cdot p_1(n+m)$ for some polynomial p_1. Thus, the entire membership query complexity satisfies the theorem with some polynomial p_1. It is also easy to show that S_0 halts in time polynomial in the total number of queries.

Now it remains to show that at least h states are added after each equivalence query since this implies the desired upper bound on the number of equivalence queries. To show this property, it is enough to prove the following stronger version of Lemma 4 in [1]. □

Lemma 5.2. Assume that (S, E, T) is a closed and consistent observation table with lookahead h. Suppose that dfa $M = M(S, E, T)$ has k states. If M' is any dfa consistent with T that has less than $k + h$ states, then M' is isomorphic to M.

Proof. In the following, let $M = (Q, q_0, F, \delta)$ and $M' = (Q', q'_0, F', \delta')$. We assume without loss of generality that M' is minimum. That is, every state of M' can be reached from q'_0 and no two states in M' are equivalent.

We show an isomorphism between M and M'. Let us first recall or give some definitions.

- Recall that $Q = \{row(s) : s \in S\}$ is the set of states of M.
- For every $q' \in Q'$, define $Row(q')$ to be a finite function from E to $\{0, 1\}$ such that $Row(q')(e) = 1$ iff $\delta'(q', e) \in F'$.
- For every $s \in S$, define $f(s) = \delta'(q'_0, s)$. Note that $Row(f(s)) = row(s)$, from the assumption that M' is consistent with T. In fact, for every $u \in \Sigma^{\leq h}$, $Row(\delta'(f(s), u)) = row(s \cdot u)$.
- For every $q \in Q$, define $\phi(q) = \{f(s) : row(s) = q\}$.

In the following sequence of claims, we show that ϕ defines a bijection between Q and the set $\{\{q'\} : q' \in Q'\}$. Clearly, we can then transform ϕ into a bijection from Q to Q', and

this turns out to be our desired isomorphism.

Claim 1. $\|Range(f)\| \geq k$.

Proof. From the above remark, it is easy to see that $f(s_1) = f(s_2)$ implies $row(s_1) = row(s_2)$. On the other hand, there are k different rows $row(s)$ in T (i.e., the states of M); hence, there must be at least k different $f(s)$. Thus, $\|Range(f)\| \geq k$. □ Claim 1

Intuitively, the next claim states that for any two states in M' there is already some string in E that proves them different. Here is where we make explicit use of the lookahead.

Claim 2. For any two different states q'_1 and q'_2 in Q', $Row(q'_1) \neq Row(q'_2)$.

Proof. By induction on the length of a string x witnessing that q'_1 and q'_2 are not equivalent.

If x is the empty string, then $Row(q'_1)$ and $Row(q'_2)$ are different in the entry corresponding to the empty string.

Consider the case where x is not empty. For the first symbol $a \in \Sigma$ of x, define $q'_3 = \delta'(q'_1, a)$ and $q'_4 = \delta'(q'_2, a)$. Then $q'_3 \neq q'_4$ (otherwise x is not a witness), and a string shorter than x witnesses that q'_3 and q'_4 are not equivalent. By induction hypothesis, $Row(q'_3)$ is different from $Row(q'_4)$.

On the other hand, because there are less than h states in $Q' - Range(f)$ (since $\|Range(f)\| \geq k$ from Claim 1), q'_1 and q'_2 must be reachable from states in $Range(f)$ with a path of length less than h. More precisely, there exist u, v in $\Sigma^{<h}$ and s_1, s_2 in S such that $q'_1 = \delta'(f(s_1), u)$ and $q'_2 = \delta'(f(s_2), v)$. Then

$$row(s_1 \cdot ua) = Row(q'_3) \neq Row(q'_4) = row(s_2 \cdot va)$$

(the equalities are true because M' is consistent with T and ua and va are in $\Sigma^{\leq k}$). By the consistency of T, it must happen that

$$row(s_1 \cdot u) \neq row(s_2 \cdot v).$$

But $row(s_1 \cdot u) = Row(q'_1)$ and $row(s_2 \cdot v) = Row(q'_2)$, again because M' is consistent with T, and u and v are in $\Sigma^{\leq p}$. Hence $Row(q'_1) \neq Row(q'_2)$. □ Claim 2

Now using Claim 2, we prove that ϕ is a bijection from Q to $\{\{q'\} : q' \in Q'\}$ in the following way.

Claim 3.

(1) For every $q \in Q$, $\|\phi(q)\| \leq 1$,

(2) $Q' \subseteq Range(f)$, and

(3) ϕ is a bijection from Q to $\{\{q'\} : q' \in Q'\}$.

Proof. Part (1): Suppose that some $\phi(q)$ has two different states q'_1 and q'_2 in Q'. By Claim 2, $Row(q'_1) \neq Row(q'_2)$. Since both q'_1 and q'_2 are in $\phi(q)$, there are strings s_1, s_2 in S such that $q'_1 = f(s_1)$, $q'_2 = f(s_2)$, and $row(s_1) = row(s_2) = q$. However, we have $row(s_1)$ $(= Row(f(s_1))) = Row(q'_1) \neq Row(q'_2) = (Row(f(s_2)) =)\ row(s_2)$. A contradiction.

Part (2): Take any q' in Q'. By an argument as in Claim 2, q' must be reachable from some state $f(s_1)$ using a string $u \in \Sigma^{<h}$, that is, $Row(q') = row(s_1 \cdot u)$. Because T is closed, there is some $s_2 \in S$ such that $row(s_1 \cdot u) = row(s_2)$. Therefore, $Row(q') = row(s_2) = Row(f(s_2))$. By Claim 2 this means $q' = f(s_2)$, and thus $q' \in Range(f)$.

Part (3): From the above part (2) and our definitions, we have

$$Q' \subseteq Range(f) \subseteq \bigcup_{q \in Q} \phi(q) \subseteq Q'.$$

Note also that $\|Range(f)\| \geq k$ (but $\|Q\| = k$) and that $\|\phi(q)\| \leq 1$ for every $q \in Q$. Thus, it must happen that $\|\phi(q)\| = 1$ for every $q \in Q$ and that all $\phi(q)$ are different. Furthermore, every $q' \in Q'$ has some $q \in Q$ such that $q' \in \phi(q)$, which is in fact $\{q'\} = \phi(q)$. □ Claim 3

Finally we must show that ϕ is not only a bijection but also an isomorphism between M and M'. That is, it carries q_0 to q'_0, it preserves δ to δ', and it carries F to F'. But having proved that Q and Q' have the same cardinality, the rest is exactly as in Angluin's proof. □

From this theorem, it is straightforward to derive the following two upper bound results.

Corollary 5.3. Let $f(n)$ be any polynomial time computable function such that $f(n) < n$. There exist a (Mem,Equ)-learner S_f for *DFA*, and a polynomial q_f such that for every $n, m > 0$,

$$\#equ\text{-}query_{S_f}(n,m) \ \leq\ \frac{n}{f(n)} \ \text{ and } \ \#mem\text{-}query_{S_f}(n,m) \ \leq\ 2^{f(n)} \cdot q_f(n+m).$$

Corollary 5.4. For any $c > 0$, there exists a polynomial time (Mem,Equ)-learner S_c for *DFA* such that $\#equ\text{-}query_{S_c}(n,m) \ \leq\ \dfrac{n}{c \log n}$ for every $n, m > 0$.

Concerning these upper bound results, a natural question is whether they can be improved. For example, we may ask whether the number of equivalence queries can be reduced by more than a $O(\log n)$ factor. However, it is shown in the following that such reduction is not possible without increasing the number of membership queries more than polynomially. That is, there is a certain type of trade-off between the number of membership and equivalence queries.

For showing such trade-off phenomena, we first prove the following somewhat general lower bound.

Theorem 5.5. For any (Mem,Equ)-learner S for *DFA*, and for any $n, m > 0$, we have the following bounds:

$$\#\mathit{equ\text{-}query}_S(n,m) \;\geq\; \left\lfloor \frac{n-2}{m} \right\rfloor \quad \text{or} \quad \#\mathit{mem\text{-}query}_S(n,m) \;\geq\; 2^m - \left\lfloor \frac{n-2}{m} \right\rfloor.$$

Remark. Proposition 1.2 (1) is a special case, i.e., $n = m + 2$, of this theorem.

Proof. We show a property stronger than the theorem: For every (Mem,Equ)-learner S that has query complexity better than the above, and *every* (Mem,Equ)-teacher T, S fails to learn some dfa from T.

To prove this, fix a (Mem,Equ)-learner S, a bounded (Mem,Equ)-teacher T for *DFA*, and any $n, m > 0$. Without loss of generality we may assume that $m + 2 \leq n \leq m2^m + 2$ (otherwise one of the inequalities in the theorem holds trivially). Let $l = \lfloor (n-2)/m \rfloor$, and assume that $\#\mathit{equ\text{-}query}_S(n,m) \;<\; l$ and $\#\mathit{mem\text{-}query}_S(n,m) \;<\; 2^m - l$. Define $R_{l,m,n}$ to be the set of dfa representations r such that $|r| \leq n \wedge L(r) \subseteq \Sigma^m \wedge \|L(r)\| \leq l$. Notice that every $L \subseteq \Sigma^m$ such that $\|L\| \leq l$ is accepted by some dfa with at most $lm + 2 \leq n$ states. Thus, any $L \subseteq \Sigma^m$ of size at most l has some dfa representation in $R_{l,m,n}$.

In the following, we exhibit some set that has a dfa representation in $R_{l,m,n}$ and for which S fails to learn from T. The set is constructed while simulating S with T. Define two sets *Pos* and *Neg* to be initially empty. Simulate S with input $\langle n, m \rangle$, answering its queries as follows:

- When S makes a membership query x, answer "yes" if $x \in \mathit{Pos}$; otherwise, answer "no" and add x to *Neg*.
- When S makes equivalence query r, one of the following three cases occurs.
  - $\mathit{Pos} \neq L(r)^{\leq m}$: Return the counterexample x given by T for $L(r)$ when the target concept is *Pos*, and add x to *Neg* if it is not in *Pos*.
  - $\mathit{Pos} = L(r)^{\leq m}$ and there is some x of length m not in $\mathit{Pos} \cup \mathit{Neg}$: Return x and add x to *Pos*.

- Otherwise: Return "yes".

Let POS and NEG be the values of Pos and Neg when S halts, and we will use POS to define the desired set. But first, we need to show that S indeed halts in the above simulation. In fact, we can prove that S makes less than l equivalence queries and $2^m - l$ membership queries in the simulation.

Let Pos_i be the value of Pos just after the ith query of S is answered in the simulation. We make the following claim, whose verification is straightforward and left to the reader.

Claim. For every i, the answers returned for the first i queries of the simulation are exactly those returned by the teacher T when the target concept is Pos_i.

With this claim we can prove that in the simulation, S makes less than l equivalence queries. Assume otherwise that S makes l (or more) equivalence queries. By the claim, it also makes l equivalence queries to the teacher T. Let Pos' be the value of Pos just before S makes its lth equivalence query. Note that strings are added to Pos only when S makes equivalence queries, so $\|Pos'\| \leq l-1$. Furthermore, we always have $Pos' \subseteq \Sigma^m$. Hence, Pos' has some dfa representation in $R_{l,m,n}$ of size at most n. That is, S makes l or more equivalence queries to T on some target concept in $R_{l,m,n}$. A contradiction with the query bound we have assumed for S.

Similarly, we can prove that S makes less than $2^m - l$ membership queries in the simulation. Thus, the total number of queries in the simulation is at most $(l-1) + (2^m - l - 1) = 2^m - 2$.

Let POS and NEG be the values of Pos and Neg when S halts. Note that each query of S adds at most one string to either Pos or Neg; hence, $\|POS \cup NEG\| \leq 2^m - 2$. Thus, there are two different strings $w_1, w_2 \in \Sigma^m$ that are not in $POS \cup NEG$. Note that $POS \cup \{w_1\}$ and $POS \cup \{w_2\}$ has at most l elements and that they are subsets of Σ^m; thus, some dfa $r_1, r_2 \in R_{l,m,n}$ recognize these two sets. Now, it follows from our discussion above that S receives the same answers during the executions of $\langle S, T(r_1)\rangle(n,m)$ and $\langle S, T(r_2)\rangle(n,m)$, namely, the answers given in the simulation. Therefore, S with teacher T outputs a wrong representation for either r_1 or r_2. □

As a corollary of this theorem, we have the following lower bound in contrast with Corollary 5.3.

Corollary 5.6. Let $f(n)$ be any function such that $f(n) < n$ and $f(n)$ becomes arbitrarily large as n increases, and let S be any (Mem,Equ)-learner for *DFA*. Then for some constants $c_1, c_2 > 0$, and for infinitely many $n, m > 0$, we have

$$\#\textit{equ-query}_S(n,m) \ \geq \ \frac{n}{f(n)} \ \text{ or } \ \#\textit{mem-query}_S(n,m) \ \geq \ 2^{c_1 f(n)} - \frac{c_2 n}{f(n)}$$

Proof. Define a nondecreasing sequence $m_1, m_2, ...$ so that $m_n \geq f(n)/2$ and $n/f(n) \leq \lfloor (n-2)/m_n \rfloor \leq 3n/f(n)$. Then the corollary follows from Theorem 5.5 for these pairs of n and m_n. (In this rough estimation, $c_1 = 1/2$ and $c_2 = 3$.) $\square$

Thus, roughly speaking, the reduction of the equivalence query complexity by $1/f(n)$ factor always costs us about $2^{O(f(n))}$ membership queries. One interesting example is the following case, which shows the limitation of Corollary 5.4.

Corollary 5.7. Let $f(n)$ be any function such that $f(n) < n$ and $f(n)/\log n$ becomes arbitrarily large as n increases. Then there exists no polynomial time (Mem,Equ)-learner S for *DFA* with the query complexity $\#\textit{equ-query}_S(n,m) \ \leq \ \frac{n}{f(n)}$.

References

1. Angluin, D., Learning regular sets from queries and counterexamples, *Information and Computation*, Vol. 75, 1987, pp.87-106.

2. Angluin, D., Queries and concept learning, *Machine Learning*, Vol. 2, 1988, pp.319–342.

3. Angluin, D., Negative results for equivalence queries, *Machine Learning*, Vol. 5, 1990, pp.121–150.

4. Angluin, D., and Kharitonov, M., When won't membership queries help?, in *Proc. 23rd ACM Sympos. on Theory of Computing*, ACM, 1991, pp.444–454.

5. Gold, E.M., Identification in the limit, *Information and Control*, Vol. 10, 1967, pp.447–474.

6. Hopcroft, J, and Ullman, J., *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, Reading, 1979.

7. Kearns, M., and Valiant, L., Cryptographic limitations on learning boolean formulae and finite automata, in *Proc. 21st ACM Symposium on Theory of Computing*, ACM, 1989, pp.433–444.

8. Pitt, L., Inductive inference, DFAs, and computational complexity, in *Proc. International Workshop on Analogical and Inductive Inference AII'89*, Lecture Notes in Artificial Intelligence 397, Springer-Verlag, 1989, pp.18–42.

9. Pitt, L., and Warmuth, M., Reductions among prediction problems: on the difficulty of prediction automata, in *Proc. 3rd Structure in Complexity Theory Conference*, IEEE, 1988, pp.60–69.

10. Valiant, L., A theory of the learnable, *Communications of the ACM*, Vol. 27, 1984, pp.1134–1142.

11. Watanabe, O., A formal study of learning via queries, in *Proc. 17th International Colloquium on Automata, Languages and Programming*, Lecture Notes in Computer Science 443, Springer-Verlag, 1990, pp.137–152.

12. Watanabe, O., A framework for polynomial time query learnability, Technical Report 92TR-0003, Dept. of Computer Science, Tokyo Institute of Technology, 1992. To appear in *Mathematical Computation Theory.*