Learning Deterministic Finite Automata from Smallest Counterexamples

Andreas Birkendorf, Andreas Böker and Hans Ulrich Simon

Lehrstuhl Informatik II, Universität Dortmund, 44221 Dortmund, Germany

E-mail: `birkendo,simon@cs.uni-dortmund.de`

Abstract

We show that deterministic finite automata (DFAs) with n states and input alphabet Σ can efficiently be learned from less than $|\Sigma|n^2$ smallest counterexamples. This improves on an earlier result of Ibarra and Jiang who required $|\Sigma|n^3$ smallest counterexamples. We show furthermore, that the special DFAs operating on input words of an arbitrary but fixed length (called leveled DFAs) are efficiently learnable from $|\Sigma|n\log(n)(1+o(1))$ smallest counterexamples. This improves on an earlier result of Ibarra and Jiang who required $|\Sigma|n^2$ smallest counterexamples. Finally, we show that our algorithm for leveled DFAs cannot be substantially improved. For this purpose, we present a general lower bound on the number of smallest counterexamples (required by any learning algorithm). This bound can be stated in terms of a (new) combinatorial dimension associated with the target class. A computation of this dimension for leveled DFAs leads to a lower bound of the form $\frac{1}{4}|\Sigma|n\log n(1-o(1))$. This coincides with our upper bound modulo a factor of approximately 4.

1 Introduction

Let $\mathcal{F}$ be a class of functions of the form $f : X \rightarrow \{0,1\}$, i.e., f maps an element of domain X to either 0 or 1. In this paper, we are concerned with the problem of "learning" an unknown target function $f_* \in \mathcal{F}$. The learning model we consider is related to the query-learning model of Angluin (see [3]). In this model, the learning algorithm A gathers information about f_* by asking queries to a teacher. The most popular queries are EQ (equivalence-queries) and MQ (membership-queries). An EQ is issued by A along with a hypothesis f. If f is correct, the teacher answers "YES". Otherwise, the teacher presents a counterexample, that is, a point $x \in X$ on which f and f_* disagree. An MQ is issued by A along with a query point $x \in X$ and answered with $f_*(x)$. If A has written a description of hypothesis f or query point x on a special query tape, the corresponding answer is given in one time step. In the query learning model, a successful learning algorithm A must find a description of f_* in polynomial time, where the polynomial may depend on parameters[1] describing the complexity of f_* and on the length of the longest counterexample ever presented by the teacher[2]. If only EQs are used, we speak of *learning from counterexamples*. If both, EQs and MQs, are used, we speak of *learning from a minimum adequate teacher*.

A particular class $\mathcal{F}$, that was intensively studied in the past, is the class of functions representable by deterministic finite automata (DFAs), i.e., the class of characteristic functions of regular

[1] These parameters will be formally specified for the concrete learning problem that we consider.

[2] If the teacher always presents a counterexample of shortest length, this additional parameter is not needed.

languages. The complexity of the target DFA is measured by two parameters: the number n of its states and the size of input alphabet Σ. It is easy to see that DFAs cannot be learned in polynomial time from MQs alone. It was shown in [4] (using the technique of "approximate fingerprints") that also EQs alone are not sufficient. In contrast to these results, Angluin has presented an efficient algorithm that learns DFAs with a minimum adequate teacher (see [2]). This algorithm was later improved by Rivest and Schapire whose algorithm is simpler and needs less MQs (see [11]).

Ibarra and Jiang have shown that MQs are not needed for learning DFAs if we put some additional information in the answers to EQs. More precisely, they have shown in [8] that DFAs are learnable from smallest counterexamples. In this model, the teacher presents the "smallest" word $c \in \Sigma^\star$ on which the current hypothesis DFA M and the target DFA M_* disagree. (The word "smallest" refers to the so-called "canonical" ordering, which is formally defined below.)

The results, presented in this paper, improve on the results of Ibarra and Jiang as follows: We present a new algorithm `REDIRECT` which efficiently learns DFAs with n states and input alphabet Σ from less than $|\Sigma|n^2$ smallest counterexamples. Ibarra and Jiang suggested an algorithm which requires $|\Sigma|n^3$ smallest counterexamples. Furthermore, we present a new algorithm `LEVELED-REDIRECT` for learning so-called *leveled* DFAs[3], which operate on input words of an arbitrary but fixed length. `LEVELED-REDIRECT` requires $|\Sigma|n\log(n)(1+o(1))$ smallest counterexamples. This improves on the corresponding algorithm of Ibarra and Jiang which requires $|\Sigma|n^2$ smallest counterexamples. Finally, we show that `LEVELED-REDIRECT` cannot be substantially improved. For this purpose, we present a general lower bound on the number of smallest counterexamples (required by any learning algorithm). This bound can be stated in terms of a (new) combinatorial dimension associated with the target class. A computation of this dimension for leveled DFAs leads to a lower bound of the form $\frac{1}{4}|\Sigma|n\log n(1-o(1))$. This coincides with the bound achieved by `LEVELED-REDIRECT` modulo a factor of approximately 4. The new general lower bound may be of interest beyond its application on DFAs.

In order to design `REDIRECT`, we invent a new data structure for regular languages: *Nerode diagrams* and *partially correct subdiagrams*. It is a very efficient and useful tool when dealing with smallest counterexamples. `REDIRECT` will "grow" an (initially trivial) partially correct subdiagram until it is isomorphic to the full transition diagram of the target DFA. The correct edges in the diagram are found by a sort of exhaustive search in the case of arbitrary DFAs, and by a sort of binary search in the case of leveled DFAs. In order to prove the correctness of our algorithms, we present a series of structural results concerning Nerode diagrams and their partially correct subdiagrams. The new data structure and the structural insights, gained by its analysis, may be of independent interest.

In order to make the notion of learning from smallest counterexamples well-defined, we proceed with the formal definition of the lexicographical and canonical ordering on $\Sigma^\star$. For notational convenience, we assume that $\Sigma = \{0, \ldots, |\Sigma|-1\}$. The letters (digits) have their natural ordering. We denote the empty word by λ. Word $x \in \Sigma^\star$ is called a *prefix* of word $w \in \Sigma^\star$ if w can be written as xz for some $z \in \Sigma^\star$. If additionally $z \neq \lambda$, x is called a *proper prefix* of w. Two words v, w can always be written as $v = xv'$ and $w = xw'$, where $x \in \Sigma^\star$ is the greatest common prefix of v and w. We say that v is *lexicographically smaller* than w (denoted as $v <_{\mathrm{LEX}} w$) if either v is a proper prefix of w or v' starts with a smaller letter than w' in the above decomposition. (That's the ordering used in a lexicon.) We say that v is *canonically smaller* than w (denoted as $v < w$) if v is shorter than w or, in case of equal length, v is lexicographically smaller than w. We note that

[3]Leveled DFAs, as defined in Section 4, are basically equivalent to a data structure known as "Ordered Binary Decision Diagrams" (or OBDDs) in the literature (see [10, 1, 6]). In this paper, it is more convenient to use the same terminology as for DFAs.

the canonical ordering is more natural for learning applications, because short counterexamples are likely to be more helpful than long-ones.

We close the introduction by mentioning a relation between query-learning and combinatorial optimization, which was has been discovered recently (see [5]). The computation of the smallest DFA that is consistent with a given set S of labeled examples is an important and computationally intractable optimization problem. Query learning algorithms for DFAs can be converted into promising heuristics for this problem, as explained in detail in [5] for algorithms using EQs and MQs. The basic idea is to simulate the teacher, to guide its answers by a preoptimized (suboptimal) DFA M', and to stop when hypothesis M is (perfectly or almost) consistent with S, but before M is equivalent to M'. Often M has considerably less states than M'. Since the computation of a canonically smallest counterexample can be done efficiently, one can similarly convert `REDIRECT` into an algorithm for the optimization problem. We leave the analysis of this convertion as a problem for future research.

2 Nerode Diagrams and Partially Correct Subdiagrams

We assume some familiarity with basic concepts in formal language theory as given by any standard book (e.g., [7]). In the sequel, we recall some classical notions concerning DFAs and define a new data structure for their representation.

Let $M = (Q, q_0, \delta, Q_+)$ be a DFA, where Q denotes the set of states, q_0 the initial state, δ the transition function, and Q_+ the set of accepting states. We will represent M by its state diagram which visualizes the states as nodes and the transitions as edges. An example is shown in Figure 1. As usual, we say that M *accepts* a word w if the path $P_M(w)$, which starts in the initial state and follows the edges labeled with the letters of w, ends at an accepting state of M. Let $L(M)$ denote the language of words accepted by M. We associate with M the indicator function $M(w)$ with value 1 if $w \in L(M)$, and value 0 otherwise.

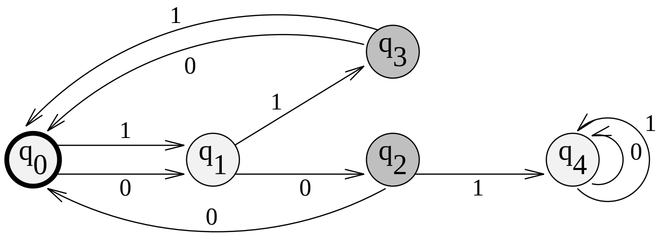

Figure 1: State diagram of a DFA $M = (\{q_0, q_1, \ldots, q_4\},\ q_0,\ \delta, \{q_2, q_3\})$ accepting language $L = (\{0,1\}\cdot\{00, 1\cdot\{0,1\}\})^*\cdot\{0,1\}\cdot\{0,1\}$. Accepting states are drawn darker than non-accepting states.

Two DFAs are called *equivalent* if they accept the same language. Let M be a minimum DFA for L, i.e., $L = L(M)$ an no other DFA with this property has less states than M. Then M is uniquely determined (up to isomophism), and its states are in a one-to-one correspondence to the classes given by the following equivalence relation $\equiv_L$, called *Nerode relation*:

$$\forall s, t \in \Sigma^\star : \quad s \equiv_L t \quad :\Longleftrightarrow \quad \forall\, z \in \Sigma^\star : sz \in L \iff tz \in L.$$

We call a word $r \in \Sigma^\star$ the *minimum representant* of its Nerode class $[r]_L = \{s \in \Sigma^\star \mid s \equiv_L r\}$ if r is the smallest word in $[r]_L$. The set of all minimum representants is denoted by $R(L)$.

It is easy to see that $ra \in R(L)$, where $r \in \Sigma^\star$ and $a \in \Sigma$, implies that $r \in R(L)$. Inductively we get the following

Lemma 2.1 *$R(L)$ is prefix-closed, i.e., if $r \in R(L)$ then $r' \in R(L)$ for all prefices r' of r.*

The *Nerode diagram* D_L of L is defined as follows. Its nodes are in one-to-one correspondence to the words in $R(L)$ and denoted by q_r for $r \in R(L)$. We call q_r "accepting" if $r \in L$ and "non-accepting", otherwise. The edge starting from q_r and being labeled by a points to q_s, where s is the minimum representant of $[ra]_L$. It is denoted by $q_r \stackrel{a}{\to} q_s$ in the sequel. An example is shown in Figure 2(a).

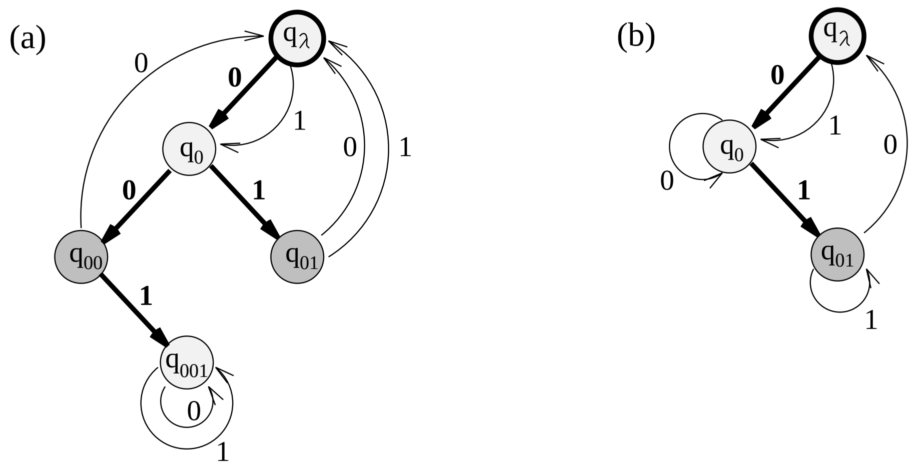

Figure 2: (a) The Nerode diagram of language L from Fig. 1. $R(L) = \{\lambda, 0, 00, 01, 001\}$. F-edges are drawn thick, B-edges are drawn thin.
(b) Partially correct subdiagram of the Nerode diagram in (a).

Obviously, D_L represents the minimum DFA for L. The edges of D_L decompose according to the following definition. Edge $q_r \stackrel{a}{\to} q_s$ is called *backward edge* (or B-edge) if $s < ra$. Otherwise, it is called *forward edge* (or F-edge). In the latter case, $s = ra$. Since $R(L)$ is prefix-closed, it follows that the F-edges form the prefix-tree of $R(L)$ (as illustrated in Figure 2(a)).

Given a prefix-closed subset $R \subseteq R(L)$, a *partially correct subdiagram* of D_L w.r.t. R consists of the prefix-tree of R and additional B-edges, not necessarily identical to the B-edges of D_L. The partitioning of nodes q_r into "accepting" and "non-accepting" ones is done as for D_L. Figure 2(b) shows an example.

A DFA M, represented by a partially correct subdiagram of D_L w.r.t. R, is called *partially correct* for L w.r.t. R. [4] (In particular, M is correct for each $r \in R$, i.e., $M(r) = 1$ iff $r \in L$.)

If path $P_M(w)$ ends in state q_r, we say that *q_r is visited by w* and call w a *visitor* of q_r. We close this section with the following obvious result.

Lemma 2.2 (Visitor Lemma) *All visitors w of q_r satisfy $r \leq w$ with equality if and only if $P_M(w)$ contains F-edges exclusively.*

[4] We will sometimes omit the expression "for L w.r.t R" when the sets L and R are clear from context.

3 The DFA learning algorithm

Let M_* denote the unknown target DFA, n the number of its states, and $L_* = L(M_*)$. Throughout the remainder of this paper, let M be a DFA, partially correct for L_* w.r.t to a prefix-closed subset $R \subseteq R(L_*)$, but $L(M) \neq L_*$. We denote the smallest counterexample for M by $\mathrm{mincex}(M, M_*)$. In order to convert the state diagram of M into the correct Nerode diagram of L_*, one has to overcome two difficulties:

1. The prefix-tree of R has to be extended to the complete prefix-tree of $R(L_*)$.
2. *Wrong B-edges*, i.e., B-edges of the form $q_r \xrightarrow{a} q_s, r, s \in R, s < ra, s \not\equiv_{L_*} ra$ must be redirected such as to point to the correct node $q_{s'}$ with $s' \equiv_{L_*} ra$.

The following lemma describes how to detect wrong B-edges.

Lemma 3.1 *Let $c = mincex(M, M_*)$. Then the following holds.*

1. *$P_M(c)$ contains at least one B-edge.*
2. *The first B-edge $q_r \xrightarrow{a} q_s$ on $P_M(c)$ is wrong, i.e., $s \not\equiv_{L_*} ra$.*

Proof.

1. If $P_M(r)$ contains F-edges exclusively, then $r \in R$. Because M is correct on R, r cannot be a counterexample. Thus, $P_M(c)$ must contain at least one B-edge.
2. If $q_r \xrightarrow{a} q_s$ is the first B-edge on $P_M(c)$, c can be written as $c = raz$ for some $z \in \Sigma^\star$. Using $sz < raz = mincex(M, M_*)$, we get $M_*(sz) = M(sz) = M(raz) \neq M_*(raz)$. Thus, $ra \not\equiv_{L_*} s$.

•

Lemma 3.1 blames the first B-edge $q_r \xrightarrow{a} q_s$ on $P_M(c)$ for receiving counterexample c. It is a straightforward idea to redirect (by way of trial-and-error) the B-edge starting from r and being labeled by a to another state $q_{s'}$, where $s' \in R, s' < ra$. We call the resulting DFA the TEST-DFA for B-edge $q_r \xrightarrow{a} q_{s'}$ and denote it by $M(r, a, s')$. Note that the partial correctness of M carries over to $M(r, a, s')$. The following result blames $q_r \xrightarrow{a} q_{s'}$ for receiving counterexample $c' = \mathrm{mincex}(M(r, a, s'), M_*)$ unless $c' > c$.

Lemma 3.2 *Let $c = mincex(M, M_*)$, $q_r \xrightarrow{a} q_s$ the first B-edge on $P_M(c)$, $q_{s'}$ a state of M with $s' < ra$, $M' = M(r, a, s')$ the TEST-DFA for B-edge $q_r \xrightarrow{a} q_{s'}$, and $c' = mincex(M', M_*)$.*
If $c' \leq c$, then $q_r \xrightarrow{a} q_{s'}$ is the first B-edge on $P_{M'}(c')$ and $s' \not\equiv_{L_*} ra$.

Proof. If $c' = c$, then $q_r \xrightarrow{a} q_{s'}$ is certainly the first B-edge on $P_{M'}(c')$.
If $c' < c = \mathrm{mincex}(M, M_*)$, then $M(c') = M_*(c') \neq M'(c')$. $P_{M'}(c')$ must therefore contain B-edge $q_r \xrightarrow{a} q_{s'}$. (Otherwise, M and M' would coincide on c'.) The following considerations are illustrated in Figure 3.

Let us assume for the sake of contradiction that $q_r \xrightarrow{a} q_{s'}$ is not the first B-edge on $P_{M'}(c')$. Hence, c' can be written as $c' = r'az'$, where $r' > r$ is a visitor of q_r. Let $c = raz$ be the corresponding decomposition of c along the first B-edge $q_r \xrightarrow{a} q_s$ on $P_M(c)$. From $r' > r$ and $r'az' = c' < c = raz$, we obtain $z' < z$. We consider the word raz'. Since $raz' < r'az' < raz$, M and M' are correct on raz', and M is correct on $r'az'$. On the other hand, we obtain

$$M(raz') = M(r'az') = M_*(r'az') \neq M'(r'az') = M'(raz'),$$

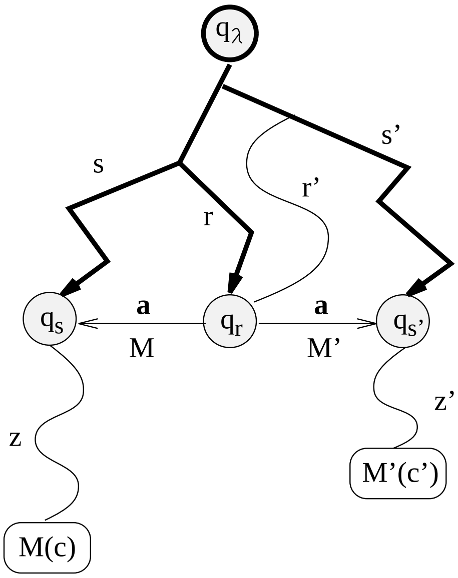

Figure 3: Illustration for the proof of Lemma 3.2.

because neither M nor M' can distinguish between r and r'. It follows that either M or M' must be wrong on raz' — a contradiction. An application of Lemma 3.1 yields $s' \not\equiv_{L_*} ra$. •

We are now prepared to describe the algorithm REDIRECT[5] which learns an unknown DFA M_* from smallest counterexamples. REDIRECT proceeds in rounds. Round k is entered with a partially correct DFA M_k for L_* w.r.t. a prefix-closed subset $R_k \subseteq R(L_*)$ and its smallest counterexample c_k. During round k, REDIRECT either stops with a DFA equivalent to M_* or enters the next round with a partially correct DFA M_{k+1} and its smallest counterexample c_{k+1}. In order to measure the progress made in each round, REDIRECT keeps track of R_k and the candidate sets $C(r,a)$ containing all $s \in R_k$ such that $s < ra$ and B-edge $q_r \xrightarrow{a} q_s$ has not been blamed for one of the counterexamples received so far. In the following description, we focus on a round not leading to the stopping condition. (Compare with Table 1.) According to Lemma 3.1, REDIRECT will blame at least one B-edge, say $q_r \xrightarrow{a} q_s$, for getting counterexample $c_k = \mathrm{mincex}(M_k, M_*)$, and may therefore safely eliminate s from $C(r,a)$. Afterwards, it skims through $C(r,a)$ and inspects the corresponding TEST-DFAs $M_k' = M_k(r,a,s')$. According to Lemma 3.2, this either leads to a counterexample $c_k' = \mathrm{mincex}(M_k', M_*) > c_k$ or to the safe elimination of s' from $C(r,a)$. In the former case, REDIRECT enters round $k+1$ with M_{k+1} equal to M_k'. In the latter case, it proceeds with the next candidate from $C(r,a)$. If all elements of $C(r,a)$ are eliminated without receiving a counterexample $c_k' > c_k$, REDIRECT decides to extend R_k to $R_{k+1} = R_k \cup \{ra\}$. In order to define M_{k+1} appropriately, REDIRECT calls procedure EXTEND(M_k, ra). The task of this procedure is to include the new state q_{ra} in the state diagram of M_k. Afterwards the state diagram contains the prefix-tree of $R_k \cup \{ra\}$. The modifications are performed carefully in order to guarantee correct accepting behaviour on ra (which is necessary to achieve partial correctness), and to modify $L(M_k)$ as less as possible. Finally, round $k+1$ is entered.

To get started, REDIRECT needs a first partially correct DFA M_1. Let M_1^0 and M_1^1 be the DFAs shown in Figure 4.

[5]The redirection of B-edges is one of its central features.

Figure 4: Initial DFAs M_1^0 and M_1^1 for $\Sigma = \{0, 1\}$.

Step 1 Find the first B-edge on $P_{M_k}(c_k)$, say $q_r \xrightarrow{a} q_s$.

Step 2 Eliminate s from $C(r, a)$. (* cf. Lemma 3.1 *)

Step 3 SAMEROUND←TRUE.

Step 4 **while** $C(r, a) \neq \emptyset$ and SAMEROUND **do**

- **Step 4.1** $s' \leftarrow$next element in $C(r, a)$
- **Step 4.2** $M'_k \leftarrow M_k(r, a, s')$.
- **Step 4.3** $c'_k \leftarrow \text{mincex}(M'_k, M_*)$.
- **Step 4.4** **if** $c'_k \leq c_k$ **then** eliminate s' from $C(r, a)$. (* cf. Lemma 3.2 *)
  else SAMEROUND←FALSE.

Step 5 **if** SAMEROUND **then**

- **Step 5.1** $M_{k+1} \leftarrow \texttt{EXTEND}(M_k, ra)$.
- **Step 5.2** $R_{k+1} \leftarrow R_k \cup \{ra\}$.
- **Step 5.3** $c_{k+1} = \text{mincex}(M_{k+1}, M_*)$.
- **Step 5.4** $\forall b \in \Sigma: \; C(ra, b) \leftarrow \{t \in R_{k+1} | \; t < rab\}$.
- **Step 5.5** $\forall t \in R_k, \forall b \in \Sigma:$ **if** $ra < tb$ **then** insert ra into $C(t, b)$.
- **else** $M_{k+1} \leftarrow M'_k$, $R_{k+1} \leftarrow R_k$, $c_{k+1} = c'_k$.

Procedure $\texttt{EXTEND}(M_k, ra)$:

Step E1 Insert the new state q_{ra} into the state diagram of M_{k+1}.

Step E2 Replace B-edge $q_r \xrightarrow{a} q_s$ by F-edge $q_r \xrightarrow{a} q_{ra}$.

Step E3 For each $b \in \Sigma$, create the B-edge starting from q_{ra} and being labeled by b.
Let it point to the same node as the edge starting from q_s and being labeled by b.

Step E4 **if** $c_k = ra$ **then** declare q_{ra} as "accepting" **iff** q_s is "non-accepting"
else declare q_{ra} as "accepting" **iff** q_s is "accepting".

Table 1: Round k of algorithm $\texttt{REDIRECT}$

Their state diagrams consist of a single state q_λ and the B-edges $q_\lambda \xrightarrow{a} q_\lambda$ for all $a \in \Sigma$. State q_λ is accepting in M_1^1 and non-accepting in M_1^0. Thus, M_1^0 and M_1^1 are acceptors of $\emptyset$ and $\Sigma^\star$, respectively. Note that M_1^0 is partially correct if $M_*(\lambda) = 0$. Otherwise, M_1^1 is partially correct. Thus, M_1 can be set to $M_0^{M_*(\lambda)}$ at the expense of one counterexample used to determine $M_*(\lambda)$. Round 1 can be entered with M_1, $c_1 = \text{mincex}(M_1, M_*)$, $R_1 = \{\lambda\}$, and $C(\lambda, a) = \{\lambda\}$ for all $a \in \Sigma$.

We will prove later that the DFAs M_k, $k \geq 1$, are partially correct. This implies that $\texttt{REDIRECT}$ creates only the n states q_u for $u \in R(L_*)$, and we can easily show the following

Theorem 3.3 $\texttt{REDIRECT}$ *learns M_* in polynomial time from at most $1 + |\Sigma|n^2 - |\Sigma|n$ smallest counterexamples.*

Proof. The number of counterexamples is bounded by 1 plus the number of eliminated candidates from sets of the form $C(r, a)$, because the first counterexample is used to define M_1 properly, and

each new counterexample leads to the elimination of a candidate. The total size of all candidate sets equals the number of candidate B-edges of the form $q_r \stackrel{a}{\rightarrow} q_s$, where $a \in \Sigma$, $r, s \in R(L_*)$, and $s < ra$. This number is obviously bounded by $|\Sigma|n^2$. Since the $|\Sigma|n$ candidates for the correct B-edges are not eliminated, the total number of eliminated candidates is at most $|\Sigma|n^2 - |\Sigma|n$. This leads to the desired bound on the number of counterexamples.
Since the run time per round is certainly polynomially bounded, it follows that the total run time is polynomially bounded. •

The following observations show that `REDIRECT` has time bound $O(|\Sigma|n^3)$.

The central data structure in the implementation of `REDIRECT` is the prefix tree of the actual set R_k. It forms the skeleton of the actual hypothesis DFA M_k. The edges are labeled with letters from Σ. Each node in the tree represents a state q_r, where $r \in R_k$ is given implicitly by the path from the root to q_r. Each state q_r is correctly labeled as either "accepting" or "non-accepting". In addition, we use pointers associated with q_r to store the B-edges starting from q_r and the $|\Sigma|$ candidate lists $C(r, a)$, $a \in \Sigma$.

Let us first consider Steps 1 to 4 of `REDIRECT`. According to Theorem 3.3, each of these steps is executed less than $|\Sigma|n^2$ times during one run of `REDIRECT`. Furthermore, it is easy to see that each step can be performed in time $O(n)$ using the data structure described above. In particular, note that each word in $R(L_*)$ has length at most n (the number of states of M_*). Each shortest counterexample, presented to a hypothesis DFA, has length smaller than $2n$ (the total number of states in M_* and the current hypothesis DFA).[6] It follows that the total time spent for Steps 1 to 4 during one run of `REDIRECT` is bounded by $O(|\Sigma|n^3)$.

Step 5 is executed exactly $n - 1$ times by `REDIRECT`, because the final hypothesis consists of n states and each call of procedure `EXTEND` leads to the creation of a new state. The most expansive Substeps are 5.4 and 5.5, which can be performed during $|\Sigma|$ appropriate walks through the prefix tree. In Substep 5.4, `REDIRECT` directs the walk to states q_t with $t < rab$. In substep 5.5, the walk is directed to states q_t with $ra < tb$. Since each walk takes time $O(n)$, the total time spent for Step 5 during one run of `REDIRECT` is bounded by $O(|\Sigma|n^2)$.

We proceed with the proof of the partial correctness of the M_k. We start with the following easy observations that hold for all $k \geq 1$:

1. M_1 is partially correct for L_* w.r.t $R_1 = \{\lambda\}$.
2. The state diagram of M_k contains exactly the nodes q_u for $u \in R_k$, and decomposes into the prefix-tree of R_k and additional B-edges.
3. For all $u \in R_k$: $M_{k+1}(u) = M_k(u)$.
4. If $M_{k+1} = M_k(r, a, s')$, then $R_{k+1} = R_k$ and $c_{k+1} > c_k$.
5. If $M_{k+1} = M_k(r, a, s')$ and M_k is partially correct for L_* w.r.t. R_k, then M_{k+1} is partially correct for L_* w.r.t. R_{k+1}.
6. If $M_{k+1} = \texttt{EXTEND}(M_k, ra)$, then $R_{k+1} = R_k \cup \{ra\}$.

These observations fall short of proving the partial correctness of all M_k by induction: the inductive step is only performed for $M_{k+1} = M_k(r, a, s')$ — see Observation 5— but not yet for $M_{k+1} = \texttt{EXTEND}(M_k, ra)$. Note that the partial correctness of M_k carries over to $M_{k+1} = \texttt{EXTEND}(M_k, ra)$ if we can show that $ra \in R(L_*)$ and $M_{k+1}(ra) = M_*(ra)$.

[6] This bound on the length of shortest counterexamples is well known, although we do not know its origin. A proof is contained in [5], for instance.

We start with the proof of the latter assertion. (For technical reasons, we prove also that $c_{k+1} \geq c_k$.)

Lemma 3.4 *If* $M_{k+1} = \texttt{EXTEND}(M_k, ra)$*, then* $M_{k+1}(ra) = M_*(ra)$ *and* $c_{k+1} \geq c_k$*.*

Proof. Remember that c_k decomposes into $c_k = raz$ for some $z \in \Sigma^\star$. Thus, $c_k \geq ra$. We prove the lemma separately for the cases $c_k > ra$ and $c_k = ra$.
If $c_k > ra$, then $\texttt{EXTEND}(M, ra)$ was defined in such a way that states q_s and q_{ra} are equivalent.[7] It follows that $L(M_k) = L(M_{k+1})$. M_k is correct on all words smaller than $c_k = \mathrm{mincex}(M_k, M_*)$, in particular on ra. Thus, the same must hold for M_{k+1} (implying $c_{k+1} \geq c_k$).
If $c_k = ra$, then $\texttt{EXTEND}(M, ra)$ was defined in such a way that $M_{k+1}(ra) \neq M_k(ra)$. Since M_k is wrong on $ra = c_k = \mathrm{mincex}(M_k, M_*)$, M_{k+1} is correct on ra. For any word $w < c_k = ra$, the path $P_{M_k}(w)$ does not reach the part of the diagram that has been modified by procedure `EXTEND`. Thus, $M_{k+1}(w) = M_k(w) = M_*(w)$, where the latter equality follows from $w < c_k = \mathrm{mincex}(M_k, M_*)$. Hence $c_{k+1} > c_k$. •

All that remains to be done is showing $ra \in R(L_*)$. The proof for this requires the following monotonicity property of the sequence c_k, given by Observation 4 and Lemma 3.4.

Corollary 3.5 *The sequence* c_k*,* $k \geq 1$*, is nondecreasing.*

Using this monotonicity, we are able to show

Lemma 3.6 *If* $M_{k+1} = \texttt{EXTEND}(M_k, ra)$*, then* $ra \in R(L_*)$*.*

Proof. Let us assume for the sake of contradiction that $ra \notin R(L_*)$, i.e., there exists a word $t \in \Sigma^\star$ satisfying $t < ra$ and $t \equiv_{L_*} ra$.

Let q_w denote the state visited by t in M_k. According to the visitor lemma, $w \leq t < ra$, so $q_r \xrightarrow{a} q_w$ was a candidate B-edge. This B-edge was eliminated during some round $j \leq k$, because all candidates from $C(r, a)$ are eliminated after round k. Let M' denote the DFA in round j leading to the elimination of w in Step 2 or Step 4.4 due to counterexample c'. Hence, M' is either M_j (and $c' = c_j$) or M' is the TEST-DFA M'_j for B-edge $q_r \xrightarrow{a} q_w$ (and $c' = c'_j$). We obtain $c' \leq c_k$ since $c_j \leq c_k$ and $c'_j \leq c_j \leq c_k$ by monotonicity. According to Lemma 3.2, c' can be written as $c' = raz'$ for some $z' \in \Sigma^\star$ and we get

$$wz' \leq tz' < raz' = c' = \mathrm{mincex}(M', M_*) \leq c_k = \mathrm{mincex}(M_k, M_*). \tag{1}$$

Hence, M_k is correct on wz' and tz'. Since M_k cannot distinguish between t and w, we obtain

$$M_*(tz') = M_k(tz') = M_k(wz') = M_*(wz'). \tag{2}$$

According to (1), M' is correct on wz', but wrong on raz'. Since M' cannot distinguish between ra and w and $t \equiv_{L_*} ra$, we obtain

$$M_*(tz') = M_*(raz') \neq M'(raz') = M'(wz) = M_*(wz). \tag{3}$$

We arrived at a contradiction between (2) and (3). •

Lemmas 3.4 and 3.6 imply that all hypotheses M_k, $k \geq 1$, are partially correct for L_* w.r.t. R_k, which concludes the analysis of `REDIRECT`.

[7] As usual, states q, q' of a DFA M are called equivalent, if the following holds for each $w \in \Sigma^\star$: M started in q accepts w iff M started in q' accepts w.

4 The Learning Algorithm for Leveled DFAs

An m-leveled DFA M is a DFA with the following special properties:

- State set Q is partitioned into $Q_0, \ldots, Q_m$, where Q_j denotes the set of states on level j.
- $Q_0 = \{q_0\}$ contains the initial state.
- If the DFA is in state $q \in Q_{j-1}$ and reads the j-th input symbol $x_j = a \in \Sigma$, it passes to a unique state $\delta(q, a) \in Q_j$ for $j = 1, \ldots, m$.
- Q_m is a non-empty subset of $\{q_+, q_-\}$, where q_+ is accepting and q_- is non-accepting.

The corresponding state diagrams are leveled acyclic graphs with nodes on levels $0, \ldots, m$. A word $w \in \Sigma^m$ is *accepted* if the computation path $P_M(w)$ ends in q_+. All other words are rejected (in particular all words of a length differing from m). Thus, $L(M)$ is a subset of Σ^m. Let DFA_m denote the class of all m-leveled DFAs.

All concepts and results considered so far for DFAs can be transferred to m-leveled DFAs in the obvious way. This transfer is used in the sequel without explicit mentioning.

As outlined in the preceding section, general DFAs are efficiently learnable from less than $|\Sigma|n^2$ smallest counterexamples. In this section, we will show that $\theta(|\Sigma| n \log n)$ smallest counterexamples are sufficient and necessary (even with unlimited computational resources) for learning m-leveled DFAs. To demonstrate sufficiency, we design an efficient algorithm called `LEVELED-REDIRECT` that is similar to `REDIRECT`. The main changes are as follows:

1. The hypotheses DFAs will be m-leveled. Consequently, only counterexamples of length m are received.
2. Each round will be concerned with exactly one of the candidate sets $C(r, a)$. $C(r, a)$ will shrink to either the empty set (in which case `EXTEND`(M, ra) is called) or a singleton set $\{s'\}$ (in which case the B-edge starting from q_r and being labeled by a will point to $q_{s'}$).
3. Each counterexample concerned with $C(r, a)$ will at least halve the size of $C(r, a)$.

The next result reveals the reason why the candidate sets can be treated one after the other. The notations are as in Lemma 3.2, but here they refer to m-leveled DFAs.

Lemma 4.1 *Let $c = mincex(M, M_*)$, $q_r \xrightarrow{a} q_s$ the first B-edge on $P_M(c)$, $q_{s'}$ a state of M located at the same level as q_s, $M' = M(r, a, s')$ the TEST-DFA for B-edge $q_r \xrightarrow{a} q_{s'}$, $c' = mincex(M', M_*)$, and $q_{r'} \xrightarrow{a'} q_t$ the first B-edge on $P_{M'}(c')$.*
If ra is not a prefix of c' (i.e., if $ra \neq r'a'$), then $ra <_{LEX} r'a'$ and $s' \equiv_{L_*} ra$.

Proof. Lemma 3.2 implies that $c' > c$. (Otherwise, ra must be a prefix of c'.) Since c and c' both have length m, $c < c'$ implies that $ra <_{\mathrm{LEX}} r'a'$.
Assume for the sake of contradiction that $s' \not\equiv_{L_*} ra$. Then there exists a word $z \in \Sigma^\star$ such that $M_*(s'z) \neq M_*(raz)$. Since M' cannot distinguish between s' and ra, it is wrong on either $s'z$ or raz. This contradicts to $s'z < raz < c' = \mathrm{mincex}(M', M_*)$. •

`LEVELED-REDIRECT` can use this lemma as follows. Whenever a TEST-DFA for B-edge $q_r \xrightarrow{a} q_{s'}$ leads to a counterexample referring to a candidate set $C(r', a')$ different from $C(r, a)$, B-edge $q_r \xrightarrow{a} q_{s'}$ is correct, and $C(r, a)$ can safely be set to singleton $\{s'\}$.

Assume that M_k, R_k, c_k, r, a (as defined within `REDIRECT`) are given. Next, we want to explain how $C(r, a)$ can be halved per counterexample. `REDIRECT` used Lemma 3.1 and 3.2 to perform a sort of "exhaustive search" for the correct candidate in $C(r, a)$. Each counterexample led to the elimination of one candidate only. It is a straightforward idea to use "binary search" instead. For this purpose, we define in the sequel a carefully selected candidate $s(C(r,a))$ which is either correct or leads to the simultaneous removal of all elements from a set $E(C(r,a))$ containing at least half of the former candidates. Since the operators for selection and elimination are defined (and computed) recursively in the sequel, the following definitions refer to an arbitrary set $C \subseteq C(r,a)$:

1. $D(C) := \{d \in \Sigma^\star |\ \exists s_1, s_2 \in C : M(r,a,s_1)(d) \neq M(r,a,s_2)(d)\}$.
2. C is called *homogenous* if $D(C) = \emptyset$, and *heterogenous* otherwise.
3. For heterogenous C, we define $d(C) := \min D(C)$, i.e., $d(C)$ is the smallest word on which at least two of the TEST-DFAs disagree. For homogenous C, we define $d(C) := \infty$ by default (where ∞ represents an "infinite word" greater than any word from $\Sigma^\star$).
4. Word $d(C)$ leads to the following partition of C into C_0 and C_1.

   $$C_i := \{s' \in C|\ M(r,a,s')(d(C)) = i\},\ i = 0, 1.$$

   Denote by C_{MAJ} the bigger-one of the two sets (say C_0 if there is a tie), and by C_{MIN} the other-one.
5. The *selection operator* is given recursively by

   $$s(C) := \begin{cases} \min C, & \text{if } C \text{ is homogenous;} \\ s(C_{\mathrm{MAJ}}), & \text{otherwise.} \end{cases}$$

6. Let $M' = M(r,a,s(C))$ and $c' = \mathrm{mincex}(M', M_*)$. Assume that c' has prefix ra. Then the *elimination operator* is given recursively by

   $$E(C) := \begin{cases} C, & \text{if } c' < d(C); \\ C_{\mathrm{MAJ}}, & \text{if } c' = d(C); \\ C_{\mathrm{MIN}} \cup E(C_{\mathrm{MAJ}}), & \text{if } c' > d(C). \end{cases}$$

It follows inductively from the recursive definition that $|E(C)| \geq \frac{1}{2}|C|$. (In particular, $E(C) = C$ if C is homogenous.) The "binary search"paradigm is therefore established by the following result stating that all elements in $E(C)$ may be blamed for getting counterexample c'.

Lemma 4.2 *1. If C is heterogenous, then ra is a prefix of $d(C)$.*

2. If ra is prefix of c', then $t \not\equiv_{L_*} ra$ for all $t \in E(C)$.

Proof.

1. Pick s_1 and s_2 from C such that $d(C) = \mathrm{mincex}(M_1, M_2)$. where $M_1 = M(r,a,s_1)$ and $M_2 = M(r,a,s_2)$. Obviously, the computation path $P_{M_1}(d(C))$ contains B-edge $q_r \xrightarrow{a} q_{s_1}$. It remains to show that this B-edge is the first B-edge on the path. Since M_1 is partially correct for $L(M_2)$, we may apply Lemma 3.1 identifying M by M_1 and M_* by M_2. We get that the first B-edge on $P_{M_1}(d(C))$ is wrong, i.e., this B-edge is not contained in M_2. The only B-edge not contained in M_2 is $q_r \xrightarrow{a} q_{s_1}$, yielding the claim.
2. If $c' < d(C)$, then $c' \notin D(C)$. Thus c' is a smallest counterexample for each $M(r,a,t)$ with $t \in C$. Since ra is prefix of c', Lemma 3.1 yields $t \not\equiv_{L_*} ra$ for all $t \in C$.
   If $c' = d(C)$, then M' votes on c' as the majority of all TEST-DFAs. Thus c' is a smallest counterexample not only for M', but for all $M(r,a,t)$ with $t \in C_{\mathrm{MAJ}}$. Again, Lemma 3.1 applies. This time, it yields $t \not\equiv_{L_*} ra$ for all $t \in C_{\mathrm{MAJ}}$.

If $c' > d(C)$, then M' is correct on $d(C)$. Thus $d(C)$ is a smallest counterexample for each $M(r, a, t)$ with $t \in C_{\mathrm{MIN}}$. Since $d(C)$ has prefix ra, Lemma 3.1 applies again. Now, it yields $t \not\equiv_{L_*} ra$ for all $t \in C_{\mathrm{MIN}}$. Furthermore, $t \not\equiv_{L_*} ra$ for all $t \in E(C_{\mathrm{MAJ}})$ follows inductively. •

The initialization of `LEVELED-REDIRECT` is performed analogously to the starting phase of `REDIRECT`. This time we find a first partially correct m-leveled DFA M_1 using the m-leveled DFA M_1^0 and M_1^1 shown in Figure 5.

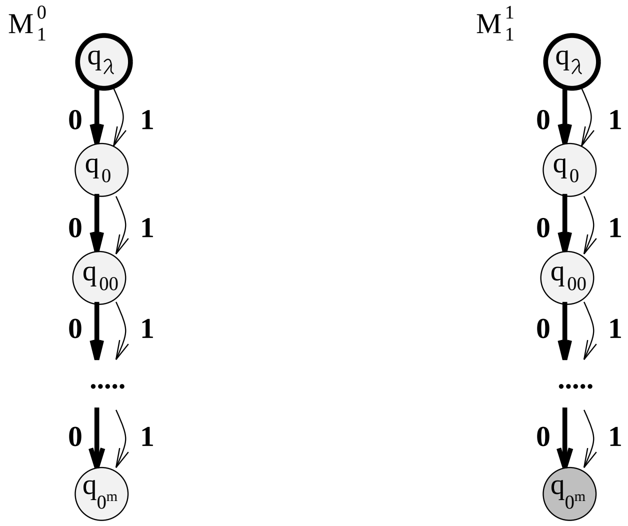

Figure 5: Initial m-leveled DFAs M_1^0 and M_1^1 for $\Sigma = \{0, 1\}$.

An arbitrary round k of algorithm `LEVELED-REDIRECT` is similarly structured as the corresponding round in `REDIRECT`. For this reason, we restrict ourselves to stress the main differences:

- The former Step 4.1 is replaced by $s' \leftarrow s(C(r, a))$, i.e., the next candidate B-edge is determined by the selection operator.
- The former Step 4.4 is replaced by

  if ra is prefix of c'_k **then** eliminate all elements of $E(C(r, a))$ from $C(r, a)$
  else SAMEROUND←FALSE.

  Hence, we treat $C(r, a)$ completely in one round, and halve it per counterexample.
- The updates in Step 5 are adapted to m-leveled DFAs. In the **else**-part (i.e., if ra is not prefix of c'), we add the command $C(r, a) \leftarrow \{s'\}$. This reflects that B-edge $q_r \xrightarrow{a} q_{s'}$ is known to be correct in this case.

We note that the selection and elimination operator allow, even in the case of unleveled DFAs, to halve one of the candidate sets per smallest counterexample. The problem with unleveled DFAs is that Lemma 4.1 does not hold. Thus, iterated halving of a set $C(r, a)$ and the insertion of up to n elements in $C(r, a)$ are possibly interleaved. This may cause linear in n many halvings. (For this reason, REDIRECT uses the simpler exhaustive search paradigm.) Because of Lemma 4.1, REDIRECT is not faced with this problem. When candidate set $C(r, a)$ is shrunken to the empty or a singleton set during one round, no later round will cause insertions into $C(r, a)$. From these considerations, we obtain

Theorem 4.3 `LEVELED-REDIRECT` *learns M_* in polynomial time from at most $|\Sigma| n \log(n)(1+o(1))$ smallest counterexamples.*

We briefly describe how LEVELED-REDIRECT can be implemented such as to run in time proportional to $|\Sigma|^2 mn^3$. We use the same data structure as for REDIRECT to represent the actual hypothesis M_k. Obviously, it is sufficient to show that the total run time, spent for the recursive evaluations of the selection and elimination operator, is bounded by $O(|\Sigma|^2 mn^3)$. Since there are $|\Sigma|n$ rounds, each round being concerned with a specific candidate set $C(r,a)$, the problem boils down to showing that the recursive evaluations during one round have time bound $O(|\Sigma|mn^2)$.

Towards this end, we consider the call of selection operator s on $C \subseteq C(r,a)$. (The analysis of call $E(C)$ is similar.) C is stored as list of pointers (pointing to states q_s for $s \in C$). Let K denote the cardinality of C and $N(K)$ the number of steps needed to compute $s(C)$. It is easy to show that, for each $s \in C$, the following holds:

$$d(C) = \min_{t \in C}\{\text{mincex}(M_k(r,a,s), M_k(r,a,t))\}.$$

Hence, we have to find the minimum of K smallest counterexamples between TEST-DFAs in order to compute $d(C)$. If we adapt a procedure, presented in [5] for the corresponding problem on OBDDs, it is possible to compute each counterexample in time proportional to $|\Sigma|m$. Hence after at most $O(|\Sigma|mK)$ steps, we either know that C is homogenous (which stops the recursion) or we can use $d(C)$ to compute C_{MAJ} of cardinality at most $K-1$ and enter the next level of recursion. Since C_{MAJ} is easy to compute from $d(C)$, we get the recursion

$$N(K) \le \alpha_1|\Sigma|mK + N(K-1),\ N(1) = \alpha_2$$

for constants α_1, α_2. Obviously, $N(K) = O(|\Sigma|mK^2)$. We note without proof that the recursive evaluation of $E(C)$ has the same time bound.

During one round, candidate set $C(r,a)$ is iteratively halved. Before each halving, the operators s and E are applied to $C(r,a)$. In the beginning of a round, the cardinality of $C(r,a)$ is at most n. The recursive calls are therefore initiated on sets of sizes $n, \frac{n}{2}, \frac{n}{4}, \ldots$. It follows that the total time spent in one round is bounded by

$$O(|\Sigma|mn^2) \cdot \left(1^2 + \left(\frac{1}{2}\right)^2 + \left(\frac{1}{4}\right)^2 + \cdots\right),$$

which is proportional to $|\Sigma|mn^2$ (as desired).

We close this section by showing that any learning algorithm for leveled DFAs requires at least $\Omega(|\Sigma|n\log n)$ smallest counterexamples. (Thus, LEVELED-REDIRECT is optimal in this respect.) On the way to this result, we prove a general lower bound in terms of a combinatorial dimension associated with the target class. Then we apply this result to the class of DFAs with at most n states and input alphabet Σ.

Let $\mathcal{F}$ be a class of functions of the form $f : X \to \{0,1\}$, where X is a linearly ordered domain. We say that a finite set $S \subseteq X$ is SC-shattered[8] by $\mathcal{F}$, if $\mathcal{F}$ contains $2^{|S|}$ functions that are pairwise different on S, but coincide on $X^{<S} := \{x \in X \setminus S \mid x < \max S\}$. Note that these functions can realize each binary output pattern on S. The *SC-dimension*[9] of $\mathcal{F}$ is defined as follows:

$$\text{SCdim}(\mathcal{F}) := \sup\{\, d \mid \exists S \subseteq X :\ |S| = d \text{ and } S \text{ is SC-shattered by } \mathcal{F}\}.$$

We show the following general lower bound:

[8] SC stands for "Smallest Counterexamples".

[9] The reader, familiar with the theory of Vapnik and Chervonenkis (see [12]), will notice the similarity between the SC-dimension and the VC-dimension. In general, $\text{SCdim}(\mathcal{F}) \le \text{VCdim}(\mathcal{F})$.

Lemma 4.4 *Each learning algorithm for $\mathcal{F}$ requires at least SCdim($\mathcal{F}$) smallest counterexamples.*

Proof. Let S be an SC-shattered set of size d, $\mathcal{F}_0 \subseteq \mathcal{F}$ the set containing the 2^d SC-shattering functions for S, and $v(x) \in \{0,1\}$ their common value on a point $x \in X^{<S}$. The lower bound is obvious from the following "adversary-argument":
Let $s_1 < s_2 < \cdots < s_d$ denote the ordered sequence of elements from S. The strategy of the adversary is to present $s_1, s_2, \ldots, s_d$ (in this order) as smallest counterexamples. When s_i is presented, the value of the target function on s_i is revealed, but still all values on $s_{i+1}, \ldots, s_d$ are conceivable. The mistakes on $s_1, \ldots, s_d$ are therefore not avoidable. This strategy can be slightly disturbed by a hypothesis h with an "additional" mistake on an s_j, $j \leq i$, already presented, or on an element $s < s_{i+1}, s \notin S$. Now the adversary presents the smallest point s_{min} on which h makes an additional mistake. In case of $s_{min} = s_j$, the learner does not gain information. In case of $s_{min} \notin S$, the learner only comes to know value $v(s_{min})$ being common for all functions in $\mathcal{F}_0$. Thus, the additional mistakes do not restrict the variability of the adversary on $s_{i+1}, \ldots, s_d$. •

Lemma 4.5 *The SC-dimension $d(n,\Sigma)$ of the class of DFAs (leveled or unleveled) with at most n states and input alphabet Σ has a lower and an upper bound of the following form:*

$$\frac{1}{4}|\Sigma| n \log(n)(1 - o(1)) \leq d(n,\Sigma) \leq |\Sigma| n \log(n)(1 + o(1)).$$

Proof. The upper bound $|\Sigma| n \log(n)(1+o(1))$ follows from the fact that SC-dim$(\mathcal{F}) \leq \log|\mathcal{F}|$ and from counting the number

$$N(n,\Sigma) = n 2^n n^{n|\Sigma|}$$

of unleveled DFAs (which are more powerful than the leveled-ones, and represent a larger function class therefore).
The lower bound follows from showing that the set

$$S = \{0,1\}^{\log n} \Sigma \{0,1\}^{\log(\log n - \log\log n)}$$

is SC-shattered by leveled DFAs of less than $4n$ states. A word w from S has the form $w = sat$, where s is a binary word of length $\log n$, a is an arbitrary letter, and t is a binary word of length $\log(\log n - \log\log n)$. It is convenient to represent a binary output pattern on S as a collection of $|\Sigma|$ binary matrices with n rows and $m = \log n - \log\log n$ columns, respectively. Given $w = sat \in S$, we obtain the corresponding output bit from entry (s,t) of the matrix $\mathcal{M}_a$ associated with letter a. We now show that an arbitrary collection $(\mathcal{M}_a)_{a\in\Sigma}$ of such matrices can be realized by a leveled DFA M with less than $4n$ states. Figure 6 shows the structure of the transition diagram D.

We describe the resulting computation on a correctly formed input $w = sat \in S$:
Diagram D starts with a complete binary tree of depth $\log n$ with $2n-1$ nodes and n leaves. The leaves are in one-to-one correspondence to the possible prefices s of w. In other words: when reaching a leaf, s is stored in the finite control. Now M reads letter a. At this moment, it perfectly knows the relevant matrix $\mathcal{M}_a$, and the relevant row s. The remainder of the computation does only depend on the binary pattern in row s of matrix $\mathcal{M}_a$. Instead of storing (s,a) in the finite control, it is more economical to store the binary pattern of the row. Consequently, the next level in D contains $2^m = n/\log n$ nodes, which is precisely the number of binary patterns of length $m = \log n - \log\log n$. Each of these $n/\log n$ nodes is the root of a complete binary tree of depth $\log m$. When M reaches a leaf in one of these trees, it has stored suffix t (and still the relevant

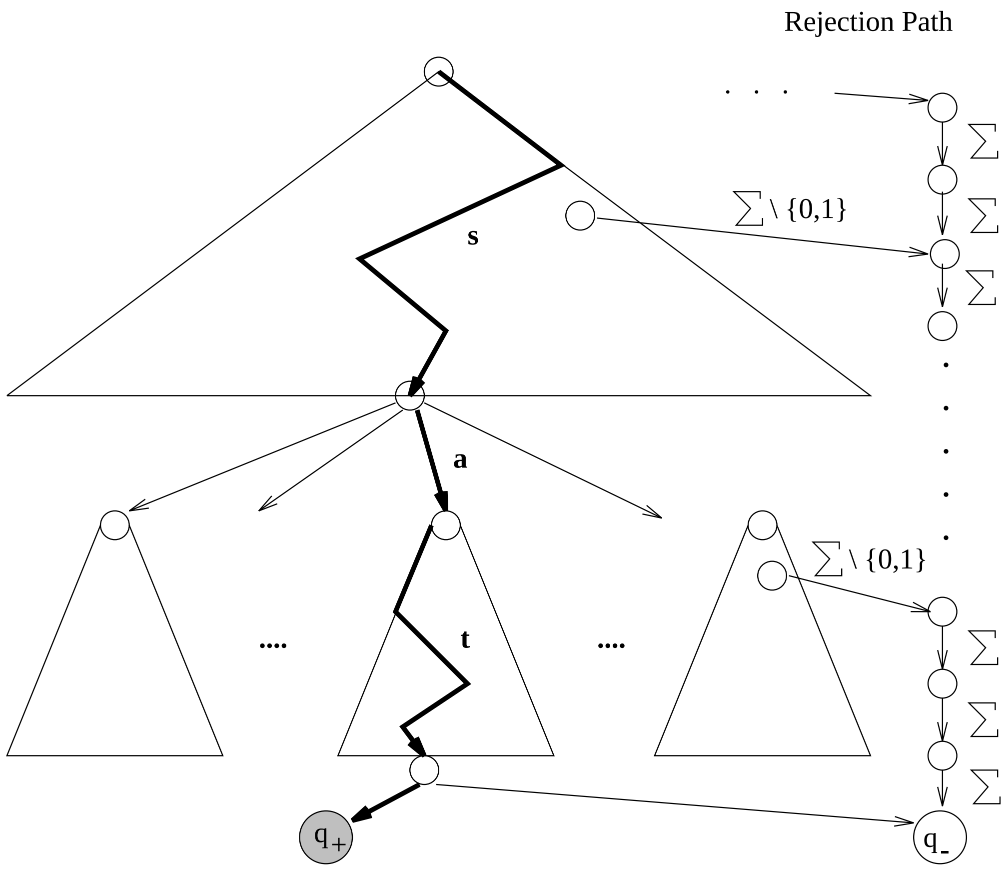

Figure 6: State diagram D for the proof of Lemma 4.5. The path for an input word $w = sat \in S$ through D is displayed accentuated.

binary pattern $p \in \{0,1\}^m$) in its finite control, and can select and output the correct bit.
If the input word w has not the correct form, i.e., $w = sat \notin S$, either s or t contains a digit different from 0 and 1. In this case, diagram D feds the computation path into a "rejection path", which assigns default value 0 to each $w \notin S$. The number of nodes in diagram D is easily seen to be less than $4n$. This shows that S is SC-shattered by the class of leveled DFAs with less than $4n$ states. •

A similar result was shown recently in [9] for the VC-dimension of the class of arbitrary DFAs with at most n states and input alphabet Σ. Our lower bound is stronger for two reasons. First, it applies to leveled DFAs. Second, it applies to the SC-dimension which is smaller, in general, than the VC-dimension. Besides that, our proof is much simpler.

We have shown that $\theta(|\Sigma| n \log n)$ smallest counterexamples are necessary and sufficient to learn leveled DFAs. For unleveled DFAs, we leave the gap between the lower bound $\frac{1}{4}|\Sigma| n \log(n)(1-o(1))$ and the upper bound $|\Sigma| n^2$ (achieved by `REDIRECT`) as an object of future research.

References

[1] S. B. Akers. Binary decision diagrams. *IEEE Transactions on Computers*, 27:509–516, 1978.

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

[3] Dana Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, 1988.

[4] Dana Angluin. Negative results for equivalence queries. *Machine Learning*, 5:121–150, 1990.

[5] Andreas Birkendorf and Hans Ulrich Simon. Using computational learning strategies as a tool in combinatorial optimization. To appear in Annals of Mathematics and Artificial Intelligence and presented at the 4th International Symposium on Artificial Intelligence and Mathematics, 1996.

[6] R. E. Bryant. Symbolic boolean manipulation with ordered binary-decision diagrams. *ACM Computing Surveys*, 24(3):293–318, 1992.

[7] John E. Hopcroft and Jeffrey D. Ullman. *Formal Languages and their Relation to Automata.* Addison Wesley, 1969.

[8] Oscar H. Ibarra and Tao Jiang. Learning regular languages from counterexamples. *Journal of Computer and System Sciences*, 43:299–316, 1991.

[9] Yoshiyasu Ishigami and Sei'ichi Tani. VC-dimensions of finite automata and commutative finite automata with k letters and n states. *Discrete Applied Mathematics*, 74:123–134, 1997.

[10] C. Y. Lee. Representation of switching circuits by binary-decision programs. *Bell Systems Technical Journal*, 38:985–999, 1959.

[11] Ronald L. Rivest and Robert E. Schapire. Inference of finite automata using homing sequences. *Information and Computation*, 103(2):299–347, 1993.

[12] V. N. Vapnik and A. Ya. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theor. Probability and Appl.*, 16(2):264–280, 1971.