Learning Unary Automata[*]

Presented at the Seventh Workshop on Descriptional Complexity of Formal Systems

DCFS, Como, Italy, June 30 - July 2, 2005

Gregor Gramlich[†]
gramlich@thi.informatik.uni-frankfurt.de

Ralf Herrmann
ralf.herrmann@internet.de

Institut für Informatik
Johann Wolfgang Goethe–Universität Frankfurt
Robert-Mayer-Straße 11-15
60054 Frankfurt am Main, Germany

Abstract

We determine the complexity of learning problems for unary regular languages. We begin by investigating the minimum consistent dfa (resp. nfa) problem which is known not to be approximable within any polynomial, unless $P = NP$. For the case of unary dfa's, we exhibit an efficient algorithm. On the other hand, we show the intractability of the unary minimum consistent nfa problem but provide an efficient quadratic approximation for its optimization version.

The VC dimension for the class of languages accepted by unary dfa's with at most n states is computed as $n + \log n \pm \Theta(\log\log n)$, which (together with the efficient solution for the consistency problem) yields an efficient PAC learning algorithm for this class. We also show that there are no efficient PAC learning algorithms for the class of languages accepted by unary nfa's with at most n states, unless every problem in NP is solvable by a quasipolynomial time Monte-Carlo algorithm. Here we assume that nfa's with few states serve as hypotheses.

In the context of learning with equivalence queries, we consider the number of counterexamples required to learn a unary regular language that is accepted by a dfa with at most n states. When submitting dfa's with at most n^d states ($d \leq n$) as queries, we show the upper bound $O(\frac{n^2}{d})$ and the lower bound $\Omega(\frac{n^2 \cdot \ln d}{d \cdot (\ln n)^2})$. If only prime cycle lengths $\leq n$ are allowed as queries, we prove that $\Theta(\frac{n^2}{\ln n})$ counterexamples are necessary and sufficient.

1 Introduction

We investigate the learnability of unary regular languages, i.e. regular languages defined over the alphabet $\Sigma = \{a\}$. In particular we consider *PAC learning* [7], where an unknown

[*]partially supported by DFG project SCHN503/2-1.

[†]Corresponding author

concept has to be learned probably approximately correctly from randomly generated positive and negative examples, and learning with *equivalence queries*, where the learning algorithm submits an hypothesis and the teacher either confirms its correctness or replies with a counterexample.

A unary deterministic finite automaton (dfa) has a simple structure: it consists of a possibly empty initial path followed by a cycle. Chrobak [2] shows that every unary nondeterministic finite automaton (nfa) with n states can be brought into the following normal form. The resulting nfa has an initial deterministic path with at most n^2+n states and the last state of the path leads to several deterministic cycles which have a total of at most n states. So every computation of an nfa in Chrobak normal form on an arbitrary word takes at most one nondeterministic step.

These simple structures raise the hope that learning unary dfa's or nfa's in the framework of algorithmic learning theory is easier to accomplish than for non-unary languages.

In the minimum consistent dfa (resp. nfa) problem positive and negative examples (words) are given and the size of a minimum dfa (resp. nfa) that accepts all positive examples and rejects all negative examples is to be computed. Already the minimum consistent dfa problem does not allow any reasonable approximation whatsoever [8, 4]. However the minimum consistent dfa problem, restricted to unary examples, is efficiently solvable (see Theorem 1) and we can also efficiently compute an nfa with $O(opt^2)$ states, provided consistent nfa's with *opt* states exist (see Theorem 3). We show in Theorem 2 that efficient exact algorithms for the unary minimum consistent nfa problem do not exist, unless $NP \subseteq \mathrm{DTIME}(n^{O(\log n)})$. Theorem 2 also implies that languages, represented by unary nfa's, are not efficiently PAC learnable with small nfa's as hypotheses, unless every problem in NP has a quasipolynomial time Monte-Carlo algorithm (see Corollary 2).

The Vapnik-Chervonenkis (VC) dimension plays a crucial role when determining the number of examples for successful PAC learning algorithms. In Theorem 4 we determine the VC dimension of the class of unary languages accepted by dfa's with at most n states to be $n+\log n \pm \Theta(\log\log n)$. We conclude in Corollary 1 that the class of unary dfa's with at most n states is efficiently PAC learnable. Corollary 2 shows that the class of unary nfa's with at most n states is efficiently PAC learnable with hypotheses of quadratic size.

The major goal in the model of learning with equivalence queries is to submit as few hypotheses as possible, thus minimizing the number of required counterexamples. We consider cyclic dfa's with p states, where $p \leq n$ is some prime number. If we only allow dfa's as hypotheses with $q \leq n$ states for some prime number q, then we show in Theorem 5 that $\Theta(\frac{n^2}{\ln n})$ counterexamples are necessary and sufficient. On the other hand Theorem 6 shows that if we allow to submit dfa's with n^d states ($d \leq n$), then there is a learning algorithm that only needs $O(\frac{n^2}{d})$ counterexamples and there are cases, where $\Omega(\frac{n^2 \cdot \ln d}{d \cdot (\ln n)^2})$ counterexamples are necessary.

The paper is organized as follows: Section 2 describes the complexity of the problems of minimum consistent dfa's and nfa's in the unary case. We calculate the VC dimension for unary dfa's and apply the results to the model of PAC learning in Section 3 and determine the number of counterexamples required to learn unary cyclic dfa's in the model of learning with equivalence queries in Section 4. We compare results for PAC learning and learning with equivalence queries in the unary case with results for the general case in Section 5.

2 Unary minimum consistent Automata

We first need some more insight into the structure of unary regular languages and introduce some terms.

Since a unary regular language $L \subseteq \{a\}^*$ is described by a dfa, we derive notions of periodicity of L from the structure of a dfa accepting L. If a dfa accepts L with z states in its cycle, we say that L is ultimately z-cyclic or has ultimate period z. Algebraicly this means that for all words a^m, a^n with $m \equiv n \pmod z$ and m, n sufficiently large (greater than the path length of the dfa), $a^m \in L \Leftrightarrow a^n \in L$ holds. If z is the number of states in the cycle of the *minimal* dfa accepting L, we say that L has the minimal ultimate period z. Every ultimate period is a multiple of the minimal ultimate period. If L can be accepted by a (minimal) dfa with z states that consists of a cycle only, then we say that L is (minimally) z-cyclic or that L has (minimal) period z.

If L is accepted by an nfa in Chrobak normal form with cycle lengths $c_1, \ldots, c_m$, then the least common multiple $\mathrm{lcm}(c_1, \ldots, c_m)$ is an ultimate period of L.

In this section, we assume that P (resp. N) denotes a set of positive (resp. negative) examples. We say that an automaton A is consistent with (P, N), if $P \subseteq L(A)$ and $N \cap L(A) = \emptyset$, where $L(A)$ denotes the language accepted by the automaton A.

For unary consistency problems, we assume that $P, N \subseteq \{a\}^*$. In a realistic setting an example $w \in P \cup N$ is represented concisely by denoting its length $|w|$ with roughly $\log |w|$ bits. Thus the input length for our minimum consistent automata problems is about $\ell(P, N) = \sum_{w \in P \cup N} \log |w|$ which is at least $|P| + |N|$ for non-trivial example sets.

In the (general) minimum consistent dfa problem, we have to determine the minimum size of a dfa which is consistent with given P and N. This problem is known to be NP-complete and any reasonable approximation is intractable[8, 4]. We present an efficient constructive algorithm for the *unary* minimum consistent dfa problem.

For an input word w of sufficient length, we know that the residue of $|w|$ modulo the cycle length of the dfa determines whether the word is accepted or rejected. The following definition partitions the words due to their residues modulo a period z and the definition of λ allows us to describe the longest word in a set.

Definition 1. *For a finite set $S \subseteq \{a\}^*$, let $S_{r,z} = \{w : w \in S \wedge |w| \equiv r \bmod z\}$ be the residue subset for residue $0 \leq r < z$ and let $\lambda(S) = max\{|w| : w \in S\}$, resp. $\lambda(\emptyset) = -1$.*

If we require a certain cycle length z, then it is easy to describe the size of a minimum consistent dfa with cycle length z.

Lemma 1. *The size of a minimum dfa with fixed given cycle length $z \in \mathbb{N}$ that is consistent with the sets $P, N \subseteq \{a\}^*$ is*

$$s_z(P, N) := z + 1 + \max\left\{\min\{\lambda(P_{r,z}), \lambda(N_{r,z})\} : 0 \leq r < z\right\}.$$

Proof. We first prove that for every fixed $0 \leq r < z$ a minimum dfa with cycle length z consistent with $P_{r,z}, N_{r,z}$ needs exactly $1 + \min\{\lambda(P_{r,z}), \lambda(N_{r,z})\}$ states in its path (and z states in its cycle). If one of the sets $P_{r,z}, N_{r,z}$ is empty, then a minimum dfa is just a cycle of length z with empty path, so w.l.o.g. we assume that $0 \leq \lambda(P_{r,z}) < \lambda(N_{r,z})$ and for the sake of contradiction suppose that the path had less states than $1 + \lambda(P_{r,z})$. Then the longest examples $w_P \in P_{r,z}$ and $w_N \in N_{r,z}$ both reach the cycle and end up in the same state. So either both w_P and w_N have to be accepted, or both have to be rejected.

On the other hand $1 + \lambda(P_{r,z})$ states in the path are enough, since no positive example reaches the cycle, and thus final and non-final states can be chosen consistently.

A dfa consistent with (P, N) has to be consistent with all residue subsets and thus needs the longest path computed for some residue r. Moreover the minimum dfa consistent with (P, N) does not need more than the described maximum, since the residue subsets are pairwise disjoint and lead to pairwise disjoint sets of reachable states in a dfa with $1 + \max\{\min\{\lambda(P_{r,z}), \lambda(N_{r,z})\} : 0 \le r < z\}$ states in its path and z states in its cycle. □

We conclude that the size of a minimum consistent dfa can be described easily and show that this size cannot grow exponentially in the input size $\ell(P, N)$.

Lemma 2. *The size of a minimum dfa consistent with* (P, N) *– denoted* $s(P, N)$ *– is* $\min\{s_z(P, N) : z \in \mathbb{N}\}$ *and* $s(P, N) < \ell^3$ *holds for input length* $\ell = \ell(P, N)$ *large enough.*

Proof. The first claim is obvious. To prove the second claim, we set $n := s(P, N)$ and observe that for every cycle length $z < n$, a consistent dfa must have a non-empty path (otherwise $z < n$ states were sufficient), thus

$$\forall z < n \; \exists (a^x, a^y) \in P \times N : x \equiv y \bmod z.$$

For reasonable inputs, we assume that $|P| + |N| \le \ell$ and thus the number of pairs $|P \times N|$ is at most $\ell^2/4$. To arrive at a contradiction we now suppose that $n \ge \ell^3$. Thus for each $z < \ell^3$, there is a pair $(a^x, a^y) \in P \times N$, such that $x \equiv y \bmod z$. We restrict ourselves to prime cycle lengths $z < \ell^3$ and let $q = \pi(\ell^3 - 1) \approx \frac{\ell^3}{3 \ln \ell}$ be the number of these primes. Since there are at most $\ell^2/4$ pairs, the pigeon hole principle implies that there must be some pair $(a^x, a^y) \in P \times N$, such that there are $4q/\ell^2$ distinct primes $z_1, \dots, z_{4q/\ell^2}$ with $x \equiv y \bmod z_i$ for each $1 \le i \le 4q/\ell^2$. By the chinese remainder theorem, we get $x \equiv y \bmod \prod z_i$ and thus $|x - y| \ge \prod z_i > (4q/\ell^2)!$. This implies, that there must be some example $a^x \in P \cup N$ with $\log x > \log((4q/\ell^2)!)$. We apply Stirling's approximation to $4q/\ell^2 \approx \frac{4\ell}{3 \ln \ell}$ and receive $\log x > \ell$ for ℓ large enough. But this contradicts the definition of $\ell = \sum_{w \in P \cup N} \log |w|$. Thus an automaton size $n \ge \ell^3$ is impossible. □

With the knowledge of Lemma 1 and 2 it is easy to derive the efficient Algorithm 1 that computes a minimum consistent dfa.

Algorithm 1

```
Input: example sets $P, N$.
$z := 1$; $n := \infty$;
while $n > z$ do
  if $s_z(P, N) < n$ then
    $n := s_z(P, N)$; $z' := z$;
  end if
  $z := z + 1$;
end while
Output dfa with cycle length $z'$ and path length $n - z'$ consistent with $P$ and $N$.
```

Theorem 1. *Let $\ell = \ell(P, N)$ be the input size for example sets $P, N \subseteq \{a\}^*$, then Algorithm 1 computes a minimum consistent dfa of size n for P, N in time $O(n \cdot \ell) = O(\ell^4)$.*

Proof. The algorithm calculates the correct size, since it calculates the correct size for every cycle length $z = 1, 2, \dots$ until some dfa with $n \leq z$ states is found. Larger cycles are not considered, since they result in dfa's larger than the one already found.

Obviously we need n iterations of the while loop. Every loop needs mainly the time to determine $s_z(P, N)$. This can be achieved in $O(\ell)$ steps as a result of Lemma 1, by storing the length of the longest example for each residue. $\square$

Next we consider the unary minimum consistent nfa problem. The following lemma shows that for a minimally d-cyclic language L, specifying all examples from $\{a\}^{<2d^2+d} := \bigcup_{i<2d^2+d}\{a^i\}$ ensures, that the minimum consistent nfa accepts exactly L. This will lead us to a hardness result for the unary minimum consistent nfa problem derived from the hardness of determining the size of a minimum nfa equivalent to a given cyclic dfa.

Lemma 3. *Let $L \subseteq \{a\}^*$ be a minimally d-cyclic language. If M is a minimum nfa consistent with $P = L \cap \{a\}^{<2d^2+d}$ and $N = \{a\}^{<2d^2+d} \setminus P$, then $L(M) = L$.*

Proof. Let A be a minimum nfa with $L(A) = L$ and $|Q_A|$ states, then A is obviously consistent with (P, N) and $|Q_A| \leq d$ holds. So for the sake of contradiction, assume that there is some nfa M with $|Q_M| < |Q_A|$ states consistent with (P, N), but $L(M) \neq L$.

M can be converted into an equivalent nfa M' in Chrobak normal form. M' has a path of length at most $|Q_M|^2 + |Q_M| < d^2 + d$. The last state of the path has transitions to disjoint deterministic cycles of cycle lengths $c_1, \dots, c_m$. The language $L(M')$ is ultimately $c := \mathrm{lcm}(c_1, \dots, c_m)$-cyclic and $\sum_{i=1}^m c_i \leq |Q_M| < d$. We thus have for every $j \geq d^2 + d$: $a^j \in L(M') \Leftrightarrow a^{j+c} \in L(M')$.

We first show that $c < d$ is impossible. Since L is *minimally* d-cyclic, $\forall c < d\ \exists i$: $a^i \in L \Leftrightarrow a^{i+c} \notin L$. This is also true for some i' with $d^2 + d \leq i' < d^2 + 2d$: let $a^i \in L \Leftrightarrow a^{i+c} \notin L$, then for every $x \in \mathbb{Z}, i + xd \geq 0 : a^{i+xd} \in L \Leftrightarrow a^{i+xd+c} \notin L$. Obviously $i' = i + xd$ can be chosen accordingly. So M' cannot be consistent with (P, N), if $c < d$.

If $c = d$, then $L = L(M') = L(M)$, but this contradicts our assumption that $L(M) \neq L$.

Now assume $c > d$. We show how to replace M' by an nfa M'' with shorter cycles, still consistent with (P, N) and ultimately d-cyclic, and thus d-cyclic. ($L(M'') = L(M')$ does not necessarily hold.) There must be some cycle C_k in M' with length $c_k < d$, $c_k \nmid d$, since $c = \mathrm{lcm}(c_1, \dots, c_m) > d$. We may assume that there is some final state in C_k (otherwise we may drop C_k) and therefore, there is some $d^2 + d \leq i < d^2 + d + c_k$, such that for every $x \in \mathbb{N}_0$: $a^{i+xc_k} \in L(M')$. Because L is d-cyclic and M' is consistent with (P, N), $\forall x \in \mathbb{N}_0, y \in \mathbb{Z}, d^2 + d \leq i + xc_k + yd < 2d^2 + d : a^{i+xc_k+yd} \in L$. From Fact 1 below follows for every $x \in \mathbb{N}_0, i + x\gcd(c_k, d) < 2d^2 + d : a^{i+x\gcd(c_k,d)} \in L$. Thus for every a^i that reaches some final state in C_k, the words a^j with $j \equiv i (\mathrm{mod}\ \gcd(c_k, d))$ and $d^2 + d \leq j < 2d^2 + d$ are also accepted by M'. Thus C_k can be replaced by a cycle of length $\gcd(c_k, d)$ and the new nfa remains consistent with (P, N). We observe that we can replace every cycle with a length not dividing d by a cycle with length dividing d and the resulting nfa M'' remains consistent with (P, N). Since M'' is ultimately d-cyclic and behaves d-cyclic on its path, we can remove the path. This contradicts the assumption that M is minimal, since we can construct an nfa of size $1 + \sum_{k=1}^m \gcd(c_k, d) < \sum_{k=1}^m c_k \leq |Q_M|$ with disjoint cycles and a single initial state that is consistent with (P, N) (and accepts L). $\square$

Fact 1. *For $a, b \in \mathbb{N}$, the set $\{ma+nb | m, n \in \mathbb{N}_0 \wedge ma+nb \geq a \cdot b\}$ equals $\{m \cdot \gcd(a,b) | m \in \mathbb{N} \wedge m \cdot \gcd(a,b) \geq a \cdot b\}$.*

We now derive a hardness result for the unary minimum consistent nfa problem. Let MinNFA-UCL (minimum nfa for a unary cyclic language) be the problem: is there a k-state nfa accepting $L(M)$ for given unary cyclic dfa M and an integer k (in binary) [6].

Theorem 2. *(a) MinNFA-UCL is polynomial time reducible to the unary minimum consistent nfa problem.*

(b) The unary minimum consistent nfa problem is in NP, but not in P, unless $NP \subseteq \mathrm{DTIME}(n^{O(\log n)})$.

Proof. (a) Given the unary cyclic dfa M with d states, we construct $P = L(M) \cap \{a\}^{<2d^2+d}$ and $N = \{a\}^{<2d^2+d} \setminus P$ in polynomial time. According to Lemma 3, there is an nfa with k states consistent with (P, N) iff there is an nfa with k states accepting $L(M)$.

(b) The problem is in NP, since we can efficiently check, if a given unary nfa is consistent with (P, N). Jiang, McDowell and Ravikumar [6] show that minNFA-UCL is not in P, unless $NP \subseteq \mathrm{DTIME}(n^{O(\log n)})$. We continue their reduction from vertex cover to minNFA-UCL for the unary minimum consistent nfa problem with part (a). $\square$

As we have seen that the exact solution of the unary minimum consistent nfa problem is intractable, we are satisfied with a good efficient approximation. The general minimum consistent nfa problem cannot be efficiently approximated within any polynomial [8] and even weaker approximations are intractable [4], whereas we next exhibit an algorithm which efficiently and constructively approximates a unary minimum consistent nfa within a quadratic function of the optimal size.

For $k = 1, 2, \ldots$ the algorithm constructs an nfa in Chrobak normal form with k^2 states in its path and k cycles with all possible cycle lengths $1, \ldots, k$. The algorithm marks every state that cannot be reached by an example from N as accepting and afterwards checks if every example of P is covered by some accepting state. If this is the case the algorithm outputs the automaton with $O(k^2)$ states, otherwise the next iteration starts with $k := k + 1$. Theorem 3 is proved by showing that no automaton with less than $k - 1$ states can be consistent with (P, N).

Theorem 3. *Let opt be the number of states of a minimum nfa consistent with $P, N \subseteq \{a\}^*$ and let $\ell = \ell(P, N)$ be the input size. There is an algorithm which computes an nfa M with $O(opt^2)$ states that is consistent with (P, N) in time $O(opt^3 \cdot (|P| + |N|)) = O(\ell^{10})$.*

Proof. Let M be constructed by the algorithm described above. M is consistent with (P, N), we make sure that no example from N is accepted and every example from P reaches some accepting state.

We show that the algorithm stops after $k \leq opt + 1$ iterations. Thus the size $O(opt^2)$ of M and the running time $O(opt^3 \cdot (|P| + |N|))$ is obvious, since we can check for each state q and each example w, whether w reaches q. Lemma 2 implies $O(opt^3 \cdot (|P| + |N|)) = O(\ell^{10})$.

At the latest in iteration $k = opt + 1$, we construct an nfa M with $(opt + 1)^2$ states which behaves as a minimum consistent nfa M_{opt} with opt states on (P, N). Just assume M_{opt} is brought into Chrobak normal form, then M can covers all of its cycles. $\square$

3 VC Dimension and PAC Learning

The Vapnik Chervonenkis dimension yields an upper bound for the number of examples that are required for successful PAC learning. For a definition see [7].

We can determine the VC dimension of the class of unary regular languages that are accepted by dfa's with n states almost exactly.

Theorem 4. *Let $\mathcal{L}_n$ be the class of unary regular languages that can be accepted by dfa's with at most n states and let $\pi(n)$ be the number of primes less than or equal to n. Then $n - 1 + \lfloor \log(\pi(n) + 1) \rfloor \leq \mathrm{VC}(\mathcal{L}_n) \leq n + \log(n)$ holds for $n \in \mathbb{N}$.*

Proof. We assume that every $L \in \mathcal{L}_n$ is accepted by a dfa with exactly n states and that states are numbered $1, 2, \ldots, n$. There are at most $n \cdot 2^n$ distinct languages in $\mathcal{L}_n$, because there are n possible path lengths and 2^n choices for the final states. The upper bound holds, since $\mathrm{VC}(\mathcal{L}_n) \leq \log(|\mathcal{L}_n|)$ is shown for arbitrary concept classes in [7].

For $n \geq 5$ we exhibit a set $S = S_1 \cup S_2$ with $n - 1 + \lfloor \log(\pi(n) + 1) \rfloor$ elements that can be shattered by $\mathcal{L}_n$, i.e. for any $s \subseteq S$, there is some $L \in \mathcal{L}_n$, such that $s = L \cap S$. S_1 is simply $\{a^0, \ldots, a^{n-3}\}$. Let $q = \lfloor 1 + \log(\pi(n) + 1) \rfloor$, let $r = 2^{q-1} - 1$ and let $S_2 = \{w_1, \ldots, w_q\}$ with words w_i chosen carefully with the help of the chinese remainder theorem. Let $s_1, \ldots, s_r$ be all the vectors from $(\{0\} \times \{0,1\}^{q-1}) \setminus \{0\}^q$, i.e. $\{0,1\}^q$-vectors with a leading 0 except for the 0^q-vector. There are $r = 2^{q-1} - 1 =\leq \pi(n)$ many of them. We use the first r primes $p_1, \ldots, p_r$ as potential cycle lengths and define a code $c(k) = (k - n + 2 \bmod p_1, \ldots, k - n + 2 \bmod p_r)$, so for $k \geq n$ with $c(k) \in \{0,1\}^r$ the word a^k reaches either the last (n-th) or the $(n-1)$st state of a dfa with n states and cycle length p_i for any $1 \leq i \leq r$.

Figure 1 shows a family of dfa's with 5 states and prime cycle lengths $2, 3, 5$. The figure also shows in which states the words of $S = \{a^0, a^1, a^2, a^{19}, a^{24}, a^{33}\}$ end. The corresponding $c(\cdot)$ vectors are $c(33) = (0,0,0)$, $c(19) = (0,1,1)$, $c(24) = (1,0,1)$. Now assume that the column vectors $s_1, \ldots, s_r$ are placed into a matrix $(m_{j,i})_{1 \leq j \leq q, 1 \leq i \leq r}$. We use those q words a^k with codes $c(k)$ that appear as rows in the matrix $(m_{j,i})$ to build the set S_2, i.e. $w_j = a^k$ with $n \leq k < n + \prod_i p_i \ \wedge \ c(k) = (m_{j,1}, m_{j,2}, \ldots, m_{j,r})$ for $1 \leq j \leq q$.

To shatter an arbitrary subset $\emptyset \neq s \subset S_2$, we look at the incidence vector $v(s)$ of s. (i) If $w_1 \notin s$, then $v(s) = s_{i^*}$, and (ii) if $w_1 \in s$, then $v(S_2 \setminus s) = s_{i^*}$ for some $1 \leq i^* \leq r$. Choose M as a unary n-state dfa with cycle length p_{i^*} and (i) state n final, state $n-1$ non-final, resp. (ii) state n non-final, state $n-1$ final. Then $L(M) \cap S_2 = s$.

For an arbitrary subset $s \subseteq S = S_1 \cup S_2$, we choose a dfa M with n states, such that $s = L(M) \cap S$ as shown above, by setting the acceptance on the path accordingly.

The size of $S = S_1 \cup S_2$ is $n - 2 + q = n - 1 + \lfloor \log(\pi(n) + 1) \rfloor$ and gives the lower bound for $\mathrm{VC}(\mathcal{L}_n)$. For $n < 5$, $S = \{0, 1, \ldots, n-1\}$ is shattered by the path already. □

Since $\pi(n) \approx n / \ln n$, the lower bound and the upper bound for the VC dimension only differ by an additive term $O(\log \ln n)$.

The connection between the VC dimension and the number of examples for PAC algorithms is applied to receive the following corollary for unary dfa's.

Corollary 1. *Let $\mathcal{C}_n$ be the class of all regular languages represented by unary dfa's with at most n states. Then $(\mathcal{C}_n | n \in \mathbb{N})$ is efficiently PAC learnable.*

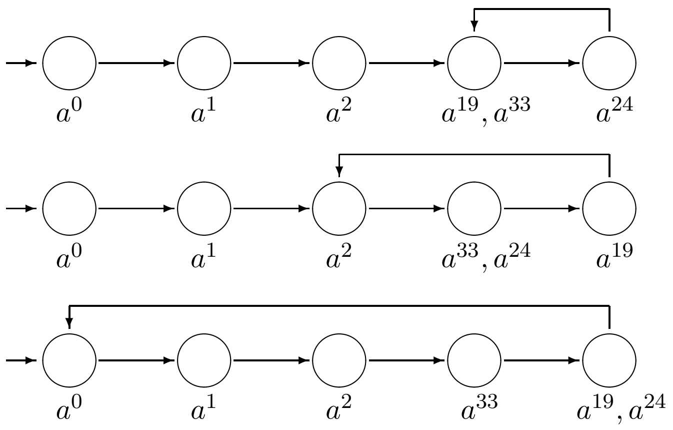

Figure 1: A family of dfa's with 5 states that shatter $S = \{a^0, a^1, a^2, a^{19}, a^{24}, a^{33}\}$.

Proof. Algorithm 1 efficiently produces a minimum consistent hypothesis for given examples. The VC dimension of $\mathcal{C}_n$ is at most $n + \log n$. Thus by Theorem 3.3 of [7], Algorithm 1 is an efficient PAC algorithm, if it demands $\Theta(\frac{1}{\epsilon}\log\frac{1}{\delta} + \frac{n+\log n}{\epsilon}\log\frac{1}{\epsilon})$ examples. $\square$

On the other hand, due to the intractability of the unary minimum consistent nfa problem, we show next that efficient PAC learning algorithms for unary nfa's probably do not exist, but approximative learning is possible. Let QRP be the class of all languages recognizable by Monte-Carlo Turing machines running in quasipolynomial time $n^{O(\log n)}$.

Corollary 2. *Let $\mathcal{N}_n$ be the class of regular languages represented by unary nfa's with at most n states. Then $(\mathcal{N}_n | n \in \mathbb{N})$ is not efficiently PAC learnable, if hypotheses from $\mathcal{N}_n$ are used for concepts in $\mathcal{N}_n$ and if $NP \not\subseteq QRP$, whereas $(\mathcal{N}_n | n \in \mathbb{N})$ is efficiently PAC learnable with hypotheses from $\mathcal{N}_{O(n^2)}$ for concepts from $\mathcal{N}_n$.*

Proof. By Theorem 2 the consistency problem does not belong to QRP, unless $NP \subseteq QRP$, and it is well known that $\mathcal{N} = (\mathcal{N}_n | n \in \mathbb{N})$ is not efficiently PAC learnable by hypotheses from $\mathcal{N}$ in quasipolynomial time, unless the consistency problem belongs to QRP. The positive result is derived as in Corollary 1 with Theorem 3 and $\mathrm{VC}(\mathcal{N}_n) \leq \log|\mathcal{N}_n| \leq n \log n$ (see [3]). $\square$

4 Learning with equivalence queries

We consider the problem of learning a concept from the concept class of unary cyclic dfa's within the model of learning with equivalence queries. I.e., the learner submits a hypothesis from the hypothesis class and the oracle will either confirm the correctness of the hypothesis or give a counterexample.

In the case of unary cyclic dfa's, we show that restricting the concept class and the hypothesis class to cycles with $p \leq n$ states for prime p implies that $\Theta(\frac{n^2}{\ln n})$ counterexamples are necessary and sufficient.

As we will see in Theorem 6, the number of questions decreases, if we allow larger dfa's in the hypothesis class.

Theorem 5. *Let $\mathcal{C}_n = \mathcal{H}_n$ be the set of unary cyclic dfa's with $p \le n$ states for prime p.*

(a) There is a learning algorithm that submits dfa's from $\mathcal{H}_n$ and needs at most $O(\frac{n^2}{\ln n})$ counterexamples to learn a concept $L(C)$ with $C \in \mathcal{C}_n$.

(b) There is an oracle, such that each learning algorithm that submits dfa's from $\mathcal{H}_n$ needs at least $\Omega(\frac{n^2}{\ln n})$ counterexamples to learn a concept $L(C)$ with $C \in \mathcal{C}_n$.

Proof. (a) The algorithm enumerates all the prime cycle lengths $p \le n$. For each p the algorithm constructs cyclic dfa's with p states and needs at most p counterexamples to conclude that no dfa with cycle length p can accept the concept $L(C)$: The algorithm first submits a dfa for the empty language. If the oracle gives a counterexample $a^x \in L(C)$, then the algorithm adjusts the dfa to accept all a^y with $y \equiv x (\mathrm{mod}\ p)$, i.e., it adds the according state to the set of final states and submits the new dfa. Again, when the algorithm receives a counterexample $a^{x'} \in L(C)$ from the oracle, it adds a final state. The algorithm may continue like this, until it receives a counterexample $a^{\bar{x}} \notin L(C)$ or the oracle confirms that the hypothesis is equivalent to C. The former implies, that the cycle length of C cannot be p, since the algorithm added the final state that accepts $a^{\bar{x}} \notin L(C)$, because there was a counterexample $a^x \in L(C)$ with $x \equiv \bar{x} (\mathrm{mod}\ p)$ given before.

So the total number of counterexamples is limited by $\sum_{p \le n, p \,\mathrm{prime}} p$. To estimate this sum, let p_i be the i-th prime number, then $i \ln i \le p_i \le 2i \ln i$ holds for $i \ge 3$. The number of primes of size at most n is approximately $\frac{n}{\ln n}$.

$$\sum_{i=1}^{\frac{n}{\ln n}} p_i \le 2 \sum_{i=1}^{\frac{n}{\ln n}} i \ln i \approx 2 \int_1^{\frac{n}{\ln n}} x \ln x \,\mathrm{d}x = O\left(\frac{n^2}{\ln n}\right).$$

(b) We construct the oracle that forces every learning algorithm to use $\Omega(\frac{n^2}{\ln n})$ equivalence queries. For each prime $p \le n$ the oracle keeps a list of residues (mod p) that are commited. We say, that a residue y is commited for each prime p, if the oracle gave a counterexample a^x with $x \equiv y (\mathrm{mod}\ p)$. We say, that a prime is fully commited, if each of its residues is commited. We assume that the oracle freely gives examples $a^0 \notin L(C)$ and $a^1 \in L(C)$ in advance and thus commits the residues 0 and 1 for each prime.

The oracle generally answers with an old counterexample if a hypothesis inconsistent with previous examples is submitted. Thus we may assume, that only consistent hypotheses are submitted. For a hypothesis with prime cycle length p^*, the chinese remainder theorem allows the oracle, to only commit one new residue modulo p^* and stick to the already commited residues 0 and 1 for the other primes as follows. For a submitted dfa D, with cycle length p^* which is not fully commited, the oracle chooses a non-commited residue $r (\mathrm{mod}\ p^*)$. If $a^r \in L(D)$, then the oracle answers $a^{r'} \notin L(C)$ with $r' \equiv r (\mathrm{mod}\ p^*)$ and $r' \equiv 0 (\mathrm{mod}\ p)$ for every prime $p \ne p^*$, $p \le n$. If $a^r \notin L(D)$, then the oracle answers accordingly with $a^{r'} \in L(C)$, where $r' \equiv r (\mathrm{mod}\ p^*)$ and $r' \equiv 1 (\mathrm{mod}\ p)$ for every prime $p \ne p^*$. If p^* is fully commited (and D is consistent), then the oracle gives $a^r \in L(C)$ with $r \equiv 0 (\mathrm{mod}\ p^*)$ and $r \equiv 1 (\mathrm{mod}\ p)$ for every prime $p \ne p^*$. Thus the learning algorithm will not submit dfa's with cycle length p^* any more, since they cannot be consistent.

The oracle will confirm the correctness of D, if every prime is fully committed and D is consistent. (D can be chosen with cycle length p^*, where p^* is the last prime that has been fully commited, and the final states of D can be chosen according to the commitments stored for p^*. These commitments have never been contradictory.) So the learning algorithm will have to receive $\sum (p_i - 1) = \Omega(\frac{n^2}{\ln n})$ counterexamples. $\square$

The situation changes, if we allow hypotheses with cycle-lengths that are composite numbers larger than n. Especially, if we allow cyclic dfa's with $n!$ states, then we only need $O(n)$ counterexamples. We can allow arbitrary cycle lengths $\leq n$ as concepts for the upper bound, whereas the oracle only needs prime cycle lengths for the lower bound.

Theorem 6. *Let $\mathcal{C}_n$ be the set of unary cyclic dfa's with at most n states and let $\mathcal{C}_n^* \subset \mathcal{C}_n$ only contain prime cycle lengths. Let $\mathcal{H}_{n,d}$ be the set of unary cyclic dfa's with at most n^d states for $d \leq n$.*

(a) There is a learning algorithm that submits dfa's from $\mathcal{H}_{n,d}$ and needs at most $O(\frac{n^2}{d})$ counterexamples to learn a concept $L(C)$ with $C \in \mathcal{C}_n$.

(b) There is an oracle, such that each learning algorithm that submits dfa's from $\mathcal{H}_{n,d}$ needs at least $\Omega(\frac{n^2 \cdot \ln d}{d \cdot (\ln n)^2})$ counterexamples to learn a concept $L(C)$ with $C \in \mathcal{C}_n^*$.

Proof. The algorithm breaks the factorial $n!$ into $\lceil \frac{n}{d} \rceil$ blocks $n_1, \ldots, n_{\lceil \frac{n}{d} \rceil} \leq n^d$ with $n_i = \frac{(d \cdot i)!}{(d \cdot (i-1))!}$ for $i < \lceil \frac{n}{d} \rceil$ and $n_{\lceil \frac{n}{d} \rceil} = \frac{n!}{(d \cdot (\lceil \frac{n}{d} \rceil - 1))!}$. Thus $n_i \leq n^d$ obviously holds.[1] For simplicity we call $d \cdot (i-1)+1, \ldots, d \cdot i$ the factors of n_i (resp. $d \cdot (\lceil \frac{n}{d} \rceil)+1, \ldots, n$ for $n_{\lceil \frac{n}{d} \rceil}$).

The algorithm considers each block n_i separately. We show, how the algorithm can rule out each factor of n_i as a potential cycle length of the concept C with $k_i + d$ counterexamples, where k_i is the largest factor of n_i.

For each factor z of n_i the algorithm stores for each residue modulo z whether the according state in a cyclic dfa with z states should be final or is unresolved. If the learning algorithm receives a positive counterexample a^x, then for each factor z of n_i the algorithm marks the residue $x (\text{mod } z)$ as final. If the algorithm receives a negative counterexample $x \notin L(C)$, then it concludes, that those factors z for which $x (\text{mod } z)$ was marked final, cannot be the cycle length of C and marks z as excluded.

The first dfa submitted accepts the empty language. From then on the algorithm will submit cyclic dfa's with n_i states. A state $x' (\text{mod } n_i)$ in this dfa is final, iff there is a non-excluded factor z of n_i, such that the residue $x' (\text{mod } z)$ is marked as final.

For each submitted dfa, the algorithm either gets a positive example, which marks exactly one residue for each factor as final, or the algorithm gets a negative example, which excludes at least one factor as a potential cycle length. So for each block n_i the algorithm needs at most $k_i + d$ counterexamples, where k_i is the largest factor of n_i. Thus the algorithm needs at most $\sum_{i=1}^{\lceil \frac{n}{d} \rceil} (k_i + d) \leq \lceil \frac{n}{d} \rceil \cdot (n+d) = O(\frac{n^2}{d})$ counterexamples.

(b) As in the proof of Theorem 5 (b) keeps a list of commited residues for each prime $p \leq n$. Commitment is defined exactly as in that proof.

Presently we assume, that the learning algorithm may only submit dfa's with a cycle length that is a product of s primes. We will see later, how we can relate s to d and how we can handle composite numbers with prime powers as factors.

The oracle commits 0 and 1 for every prime. Now it commits at most s new residues for each dfa submitted by the learning algorithm. Assume, that D is a cyclic dfa consistent with the previous examples, and that its cycle length z factors to $p_{i_1} \cdots p_{i_s}$.

Let $F \subseteq \{i_j | j = 1, \ldots, s\}$ be the index set of the fully commited primes and $N = \{i_j | j = 1, \ldots, s\} \setminus F$ be the index set of primes factoring z with non-commited residues. (Keep in mind, that F might be empty.) Then the oracle chooses a non-commited residue

[1]Packing the factors of $n!$ can be done more efficiently, but won't change the asymptotic behaviour.

$r_{i_j}(\mathrm{mod}\ p_{i_j})$ for each $i_j \in N$ and computes x with $x \equiv r_{i_j}(\mathrm{mod}\ p_{i_j})$ for each $i_j \in N$ and $x \equiv 0(\mathrm{mod}\ p_{i_j})$ for each $i_j \in F$. If $a^x \in L(D)$, then the oracle computes x' with $x' \equiv x(\mathrm{mod}\ z)$ and $x' \equiv 0(\mathrm{mod}\ p)$ for every prime $p \leq n$, $p \neq p_{i_j}$ for each j. The oracle gives the counterexample $a^{x'} \notin L(C)$. Accordingly, the oracle computes $x' \equiv x(\mathrm{mod}\ z)$ and $x' \equiv 1(\mathrm{mod}\ p)$ for every $p \neq p_{i_j}$ for each j, if $a^x \notin L(D)$ and answers $a^{x'} \in L(C)$.

Again, if *all* primes are fully committed and the learning algorithm submits a dfa $D \in \mathcal{C}$ consistent with the given examples, then the oracle confirms D's correctness. D always exists, since its cycle length can be chosen as one of the primes p, that became fully committed in the last step and D accepts according to the commitments of the residues modulo p.

To estimate the number of counterexamples needed, we first observe, that the total number of non-committed residues at the beginning is $\Omega(\frac{n^2}{\ln n})$ and that the oracle commits at most s residues for each submitted dfa. So $\Omega(\frac{n^2}{s \cdot \ln n})$ counterexamples are needed.

Now we estimate the number s of distinct prime factors for numbers of size at most n^d. The product $\prod_{i=1}^{s} p_i$ is the smallest number with s distinct prime factors. We can bound it by

$$\prod_{i=1}^{s} p_i \geq \prod_{i=1}^{s} i \ln i \geq \prod_{i=\frac{s}{2}}^{s} \frac{s}{2} \ln \frac{s}{2} \geq \left(\frac{s}{2}\right)^{\frac{s}{2}}.$$

This implies $s \leq \frac{2d \ln n}{\ln d}$, if the largest dfa allowed consists of n^d states. To prove this, assume that $s > \frac{2d \ln n}{\ln d}$. Then

$$\frac{s}{2} \ln \frac{s}{2} > \frac{d \ln n}{\ln d} \cdot \ln\left(\frac{d \ln n}{\ln d}\right) \geq \frac{d \ln n}{\ln d} \cdot \left(\ln d + \ln\left(\frac{\ln n}{\ln d}\right)\right) \geq d \ln n$$

is true for $d \leq n$ and thus $\prod_{i=1}^{s} p_i \geq \left(\frac{s}{2}\right)^{\frac{s}{2}} = e^{\frac{s}{2} \ln \frac{s}{2}} > n^d$, which contradicts the upper bound for the cycle length.

So the number of questions is at least $\Omega\left(\frac{n^2}{s \cdot \ln n}\right) \geq \Omega\left(\frac{n^2 \ln d}{d (\ln n)^2}\right)$.

If the learning algorithm submits a dfa with a cycle length that factors to a prime power p^α for $1 < \alpha \in \mathbb{N}$, we can just treat this dfa, as if the prime factor was only p^1. $\square$

Remark 1. *It is obvious, that the algorithm in the proof of (a) uses no more than* $2n$ *counterexamples for hypotheses of cycle length* $n!$*, since it needs only one block. It is well known that the VC dimension is a lower bound for the number of counterexamples required for successful learning with equivalence queries. And hence, since the VC dimension for the class of languages accepted by* cyclic *dfa's with at most* n *states is at least* n*, our algorithm is almost optimal.*

Remark 2. *If* $n \geq d \geq n^{\Omega(1)}$*, then the number of counterexamples needed in part (b) is bounded by* $\Omega\left(\frac{n^2 \ln d}{d (\ln n)^2}\right) \geq \Omega\left(\frac{n^2}{d \ln n}\right)$.

5 Conclusions and open problems

Our results show that problems of learning unary dfa's and nfa's in the context of PAC learning and learning with equivalence queries have reduced complexity compared to the non-unary case.

Unary dfa's are efficiently PAC learnable (see Corollary 1), whereas general dfa's are not efficiently PAC learnable with small dfa's as hypotheses, if $NP \neq RP$ [7]. PAC learning of unary nfa's with small nfa's as hypotheses remains intractable, but we have to use the stronger, yet plausible, assumption $NP \not\subseteq QRP$ (see Corollary 2). If we allow nfa's with n^2 states as hypotheses, unary nfa's with n states become efficiently PAC learnable.

Angluin [1] (unconditionally) shows that no polynomial time learning algorithm is able to learn dfa's in the model of learning with equivalence queries. The restriction to unary cyclic dfa's with prime cycle length yields a polynomial number of counterexamples and each hypothesis can efficiently be constructed (see Theorem 5).

It is not clear, if the upper bound for the number of counterexamples in Theorem 6 yields any efficient learning result, since the learning algorithm proposed here must construct exponentially large hypotheses.

We have not addressed the question of learning unary nfa's with equivalence queries yet. We think that a solution for learning non-cyclic unary dfa's with equivalence queries is possible by adapting the methods of Theorem 5, but we have not looked at it yet.

We have not considered learnability for unary probabilistic finite automata. The restriction to a given constant isolation might yield positive results.

We would like to thank an anonymous referee for many helpful suggestions.

References

[1] D. Angluin. Negative Results for Equivalence Queries. Machine Learning, 6(2), 1990, pp. 121–150.

[2] M. Chrobak. Finite automata and unary languages. Theoretical Computer Science, 47, 1986, pp. 149–158.

[3] M. Domaratzki, D. Kisman and J. Shallit. On the Number of Distinct Languages Accepted by Finite Automata with n States. Journal of Automata, Languages and Combinatorics, 7(4), 2002, pp. 469–486

[4] G. Gramlich, G. Schnitger. Minimizing NFA's and Regular Expressions. Proc. of Syposium on Theoretical Aspects of Computer Science 2005, Springer-Verlag, LNCS 3404, 2005, pp. 399–411.

[5] R. Herrmann. Lernen regulärer unärer Sprachen. Masters Thesis, 2004.

[6] T. Jiang, E. McDowell and B. Ravikumar. The structure and complexity of minimal NFA's over a unary alphabet. Int. J. Found. of Comp. Sci., 2 (1991), pp. 163–182.

[7] M. Kearns and U. V. Vazirani. An Introduction to Computational Learning Theory. The MIT Press, Cambridge Massachusetts, 1994.

[8] L. Pitt and M. K. Warmuth. The Minimum Consistent DFA Problem Cannot be Approximated within any Polynomial. Journal of the ACM, 40(1), 1993, pp. 95–142.