Information Processing Letters 183 (2024) 106427

Contents lists available at ScienceDirect

Information Processing Letters

journal homepage: www.elsevier.com/locate/ipl

Learning from positive and negative examples: New proof for binary alphabets

Jonas Lingg [a], Mateus de Oliveira Oliveira [b,c,*], Petra Wolf [b,d,*]

[a] Eberhard Karls Universität Tübingen, Germany
[b] University of Bergen, Norway
[c] Stockholm University, Sweden
[d] Universität Trier, Germany

ARTICLE INFO

Article history:
Received 19 June 2022
Received in revised form 18 June 2023
Accepted 27 June 2023
Available online 3 July 2023

Keywords:
Learning automata
Computational complexity
Finite samples

ABSTRACT

One of the most fundamental problems in computational learning theory is the problem of learning a finite automaton A consistent with a finite set P of positive examples and with a finite set N of negative examples. By consistency, we mean that A accepts all strings in P and rejects all strings in N. It is well known that this problem is NP-complete. In the literature, it is stated that NP-hardness holds even in the case of a binary alphabet. As a standard reference for this theorem, the work of Gold from 1978 is either cited or adapted. Nevertheless, the results in Gold's work are stated in terms of Mealy machines, and not in terms of deterministic finite automata (DFAs) as most commonly defined. As Mealy machines are equipped with an output function, they can be more compact than DFAs which accept the same language. We show that the adaptations of Gold's construction for Mealy machines stated in the literature have some issues, and provide a correct proof for the fact that the *DFA-consistency problem* for binary alphabets is NP-complete.

1. Introduction

In the *DFA-consistency problem* (DFA-Con) we are given a pair of disjoint sets of strings $P, N \subseteq \Sigma^*$ and a positive integer k. The goal is to determine whether there is a deterministic finite automaton (DFA) A with at most k states that accepts all strings in P and rejects all strings in N. Even though [1] showed that the problem cannot be approximated within any polynomial factor, DFA-Con has become one of the most central problems in computational learning theory (see [2–6]) with applications that span several subfields of artificial intelligence and related areas, such as automated synthesis of controllers [7], model checking [8,9], optimization [10–13], neural networks [14–17], multi-agent systems [18], and others [19–21].

The DFA-Con problem has been studied for at least five decades, including parameterized analysis by [22], and it was stated multiple times that the NP-completeness of the DFA-Con problem also holds in the case of binary alphabets, see [3, 21,22]. In [3], the work of [4] is cited for this fact and in [21] and [22] adaptations of the construction from [4] are given. The issue is that NP-hardness results for variants of DFA-Con considered in [4] are actually stated in terms of Mealy machines, and do not translate directly to the most usual notion of DFAs in the case of binary alphabets. By taking a closer look on the preliminaries, [3] is also considering Mealy machines. For the adaptations in [21] and [22], both of them follow the

* Corresponding authors.
E-mail addresses: j.lingg@mailbox.org (J. Lingg), oliveira@dsv.su.se (M. de Oliveira Oliveira), mail@wolfp.net (P. Wolf).

https://doi.org/10.1016/j.ipl.2023.106427

same approach, and both contain inaccuracies that invalidate the proofs. They have in common that they are adaptations of the construction by Gold for the consistency problem of Mealy machines [4], but as the Mealy machines considered in [4] (mapping $\Sigma^* \to \Gamma$, $|\Gamma| = 2$) can be more compact (when interpreted as language acceptors) than DFAs recognizing the same language, the difference in number of states causes the adaptations to fail.

We solve this issue by giving a new construction for the claim that the DFA-Con problem is NP-complete when restricted to binary alphabets.

This work is structured as follows. After giving necessary definitions, we present our main result, a new construction for the NP-hardness of DFA-Con restricted to binary alphabets. Then, we discuss in more detail why the reductions in [21,22] fail and why the construction of Gold does not directly transfer to the setting of DFAs.

We thank an anonymous reviewer of LearnAut 2022 for suggesting a simpler construction than we had initially.

2. Preliminaries

For a finite alphabet Σ, we call Σ^* the set of all words over Σ. We denote the empty word with ϵ, $|\epsilon| = 0$. For a regular expression r and a fixed number k, we denote k concatenations of r with r^k. A *deterministic finite automaton* (DFA) is a tuple $A = (Q, \Sigma, \delta, s_0, F)$ where Q is a finite set of states, Σ a finite alphabet, $\delta : Q \times \Sigma \to Q$ a total transition function, s_0 the initial state and $F \subseteq Q$ the set of final states. We call A a *complete* DFA if we want to highlight that δ is a *total* function. If δ is partial, we call A a partial DFA. We generalize δ to words by setting $\delta(q, \epsilon) = q$ for $q \in Q$, and $\delta(q, aw) = \delta(\delta(q, a), w)$ for $q \in Q, a \in \Sigma, w \in \Sigma^*$. We further generalize δ to sets of input letters $\Gamma \subseteq \Sigma$ by $\delta(q, \Gamma) = \bigcup_{\gamma \in \Gamma} \{\delta(q, \gamma)\}$. A DFA A accepts a word $w \in \Sigma^*$ if and only if $\delta(s_0, w) \in F$. We let $\mathcal{L}(A)$ denote the *language of* A, i.e., the set of all words accepted by A. The DFA-Con problem is formally defined as follows.

Definition 1 *(*DFA-Con*).*

Input: Finite sets of words $P, N \subseteq \Sigma^*$ with $P \cap N = \emptyset$, and integer k.
Question: Is there a DFA $A = (Q, \Sigma, \delta, s_0, F)$ with $|Q| \leq k$, $P \subseteq \mathcal{L}(A)$, and $\mathcal{L}(A) \cap N = \emptyset$?

3. Main result

As our main result, we present an adaptation for DFAs of the reduction by Gold.

Theorem 2. *Restricted to binary alphabets,* DFA-Con *is NP-complete.*

DFA-Con is clearly in NP. The proof of NP-hardness will use the following variant of SAT, which is known to be NP-complete (see page 314 of [4] for a proof).

Definition 3 *(*All-pos-neg *SAT). Given a Boolean formula in conjunctive normal form where all literals in a clause are either all positive or all negative. Is there a variable assignment satisfying the formula?*

We show that there is a reduction from All-pos-neg SAT to DFA-Con mapping each instance of All-pos-neg SAT with $n \geq 1$ variables and $m \geq 1$ clauses to an instance of DFA-Con with $k = n + m$, $|\Sigma| = 2$, $|P| = \mathcal{O}(m)$, and $|N| = \mathcal{O}(m^2 + nm)$ in time $n^{\mathcal{O}(1)}$.

Let ϕ be a Boolean formula in 3CNF with $n \geq 1$ variables

$$V = \{x_0, x_1, \ldots, x_{n-1}\}$$

and $m \geq 1$ clauses $C = \{C_0, C_1, \ldots, C_{m-1}\}$, where each clause contains either only positive literals or only negative literals. Given ϕ, we let $k = n + m$ and

$$\begin{aligned} P =& \{\epsilon, a^k\} \cup \{a^i bb \mid i \in [0, \ldots, m-1], C_i \text{ positive}\} \\ N =& \{a^i \mid 0 < i < k\} \\ & \cup \{a^i bb \mid i \in [0, \ldots, m-1], C_i \text{ negative}\} \\ & \cup \{a^i b a^{k-r} \mid (0 \leq i \leq m-1) \wedge (0 \leq r < k) \wedge (r < m \vee x_{r-m} \notin C_i)\} \end{aligned}$$

be the corresponding instance of DFA-Con.

We show that there exists a satisfying variable assignment

$$\beta : V \to \{\texttt{false}, \texttt{true}\}$$

if and only if there exists a DFA with at most k states that is consistent with P and N.

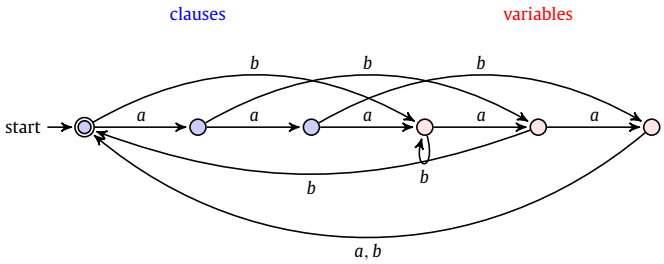

Fig. 1. Automaton for $\phi = \neg x_0 \wedge x_1 \wedge x_2$.

We start with a remark on the structure of a DFA A with at most k states accepting $\{\epsilon, a^k\}$ and rejecting $\{a^i \mid 0 < i < k\}$. Let s_i be the state of A reached after reading a^i. For each two distinct i and j in $\{0, 1, \ldots, k-1\}$, A accepts after reading a^{k-i} from s_i, but rejects after reading a^{k-j} from s_i. Therefore, the states $s_0, \ldots, s_{k-1}$ are pairwise distinct. This implies that $s_k = s_0$, and that A consists of a loop of k states $s_0, s_1, \ldots, s_{k-1}$ where s_0 is the sole initial and the sole accepting state, and for each $i \in \{0, 1, \ldots, k-1\}$, s_i transitions to $s_{i+1 \mod k}$ when reading letter a (see Fig. 1).

The intuition is that the first m states, s_0, s_1, s_{m-1}, correspond to the clauses $C_0, C_2, \ldots, C_{m-1}$ and the n last states, $s_m, s_{m+1}, \ldots, s_{m+n-1}$, correspond to the variables $x_0, x_1, \ldots, x_{n-1}$.

We start by showing that if ϕ is satisfiable, then there exists a DFA A with k states consistent with P and N. Let $\beta\colon V \to \{\texttt{false}, \texttt{true}\}$ be a satisfying variable assignment. We let $A = (\{s_0, s_1, \ldots, s_{k-1}\}, \{a, b\}, \delta, s_0, \{s_0\})$. For each $i \in \{0, 1, \ldots, k-1\}$, we let $\delta(s_i, a) = s_{i+1 \mod k}$. The transitions corresponding to letter b depend on the satisfying assignment β. For each $i \in \{0, 1, \ldots, m-1\}$, let x_j be a variable in C_i such that its assignment under β satisfies C_i. Note that since β is a satisfying assignment, such variable always exists. We set $\delta(s_i, b) = s_{m+j}$. Intuitively, if the automaton is at the state s_i corresponding to clause C_i, then after reading symbol b it transitions to the state s_{m+j} corresponding to variable x_j. For each $j \in \{0, 1, \ldots, n-1\}$, if $\beta(x_j) = \texttt{true}$, then we set $\delta(s_{m+j}, b) = s_0$. Otherwise, we set $\delta(s_{m+j}, b) = s_{m+j}$. Intuitively, if the automaton is at the state s_{m+j} corresponding to variable x_j, then after reading symbol b, it transitions to the accepting state s_0 if x_j is set to $\texttt{true}$ and to the state s_{m+j} otherwise (note that s_{m+j} is a rejecting state). See Fig. 1 for an example.

Now, we show that A accepts all words in P and rejects all words in N. By construction, A accepts $\{\epsilon, a^k\}$ and rejects $\{a^i \mid 0 < i < k\}$. Let $a^i bb \in \{a^i bb \mid i \in [0, \ldots, m-1], C_i \text{ positive}\}$. After reading a^i, A transitions to state s_i corresponding to the positive clause C_i. By construction, reading b from s_i leads to the state s_{m+j} corresponding to a variable x_j contained in C_i such that $\beta(x_j) = \texttt{true}$. Therefore, $\delta(\delta(s_i, b), b) = s_0$ and A accepts $a^i bb$. Similarly, A rejects all the words in $\{a^i bb \mid i \in [0, \ldots, m-1], C_i \text{ negative}\}$. Finally, A rejects all the words in the set

$$\{a^i b a^{k-r} \mid (0 \leq i \leq m-1) \wedge (0 \leq r < k) \wedge (r < m \vee x_{r-m} \notin C_i)\}. \tag{1}$$

To see this, note that for each $i \in \{0, \ldots, m-1\}$, the state reached by the string $a^i b$ is the state s_{m+j} corresponding to some variable x_j in C_i. Therefore, given $r \in \{0, 1, \ldots, k-1\}$, the string $a^i b a^{k-r}$ is accepted if and only if $r = m + j$. So, none of the word in (1) are accepted by A since the words $a^i b a^{k-r}$ in (1) satisfy $r < m$, or $m \leq r < k$ and $x_{r-m} \notin C_i$.

For the converse, suppose that there exists a DFA A with at most k states consistent with P and N. As we saw already, A has exactly k states $s_0, s_1, \ldots, s_{k-1}$ that form a loop with the letter a, and s_0 is both the initial state and the unique accepting state. For each $i \in \{0, 1, \ldots, m-1\}$, the negative samples in (1) ensure that the transition with b from state s_i points to the state s_{m+j} corresponding to a variable x_j contained in C_i, since if this were not the case, some string in the set (1) would have been accepted by A. Now, we define an assignment β of variables depending on the transitions labeled with b. For each $j \in \{0, 1, \ldots, n-1\}$, if the transition labeled by b going from state s_{m+j} points to the accepting state s_0, then we set $\beta(x_j) = \texttt{true}$. Otherwise, we set $\beta(x_j) = \texttt{false}$. Let $i \in \{0, 1, \ldots, m-1\}$, assume C_i is a positive clause in ϕ. By consistency, A accepts the word $a^i bb$. Note that after reading the string a^i, A reaches the state s_i corresponding to C_i, and then, by reading b the automaton transitions to the state s_{m+j} corresponding to some variable x_j contained in C_i. Finally, after reading the last b the automaton transitions to the unique accepting state s_0. Therefore, by construction, $\beta(x_j) = \texttt{true}$ and the assignment of x_j satisfies C_i. A similar argument works if C_i is a negative clause. Therefore β satisfies ϕ. □

4. Counterexamples regarding DFA-Con with $|\Sigma| = 2$

In the literature, as far as we know, two proofs for the NP-hardness of DFA-Con for fixed alphabet size of $|\Sigma| = 2$ have appeared. We give in the following a counterexample which contradicts the proofs in [22] and in [21]. We further discuss in this section why the claim does not follow directly from the construction in [4].

Counterexample ([22, Lemma 15]) We first give the construction from Lemma 15 in [22] adapted to our notation: The reduction is from All-pos-neg SAT. Let ϕ be a Boolean formula in CNF with $n \geq 1$ variables $V = \{x_1, x_2, \ldots, x_n\}$ and $m \geq 1$ clauses $C = \{C_1, C_2, \ldots, C_m\}$.

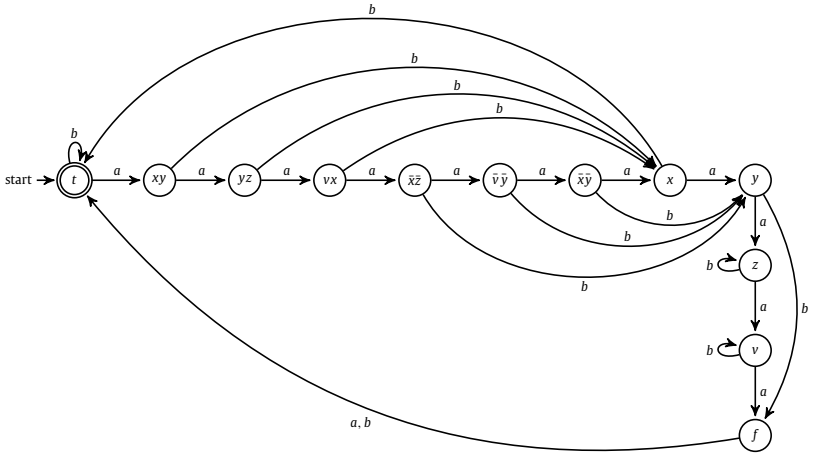

Fig. 2. Automaton corresponding to ϕ.

Let C_P denote the set of clauses containing only positive literals and let C_N denote the rest of the clauses, with $C = C_P \cup C_N$. A DFA-Con instance with alphabet $\Sigma = \{a, b\}$, $k = m + n + 2$, and the following sets P and N is constructed, starting with $P = N = \emptyset$.

1. Add $\epsilon, b, a^{m+n+2}, a^{m+n+1}b$ to P.
2. Add a^j to N, for $1 \leq j \leq m + n + 1$
3. Add $a^i b, a^i ba$ to N and $a^i bbb$ to P, for $1 \leq i \leq m$.
4. Add $a^i bb$ to P, for every $C_i \in C_P$.
5. Add $a^i bb$ to N and $a^i bba$ to P, for $C_i \in C_N$.

This completes the construction of the DFA-Con instance. Now we continue with a counterexample.

We give a Boolean formula that is unsatisfiable and shows that the construction used in the proof of Lemma 15 in [22] allows us to construct an automaton that is consistent with P and N. Consider the unsatisfiable formula

$$\phi = (x \vee y) \wedge (y \vee z) \wedge (v \vee x) \wedge (\neg x \vee \neg z) \wedge (\neg v \vee \neg y) \wedge (\neg x \vee \neg y)$$

with $n = 4$ variables and $m = 6$ clauses. We index the clauses from left to right in the formula above, starting with $i = 1$. We construct the DFA-Con instance P, $N \subseteq \{a, b\}^*$, $k = m + n + 2 = 12$ according to the construction presented in the proof of Lemma 15 in [22]:

$$\begin{aligned} P = {} & \{\epsilon, b, a^{12}, a^{11}b\} \cup \{a^i bbb \mid 1 \leq i \leq 6\} \cup \{a^i bb \mid 1 \leq i \leq 3\} \\ & \cup \{a^i bba \mid 4 \leq i \leq 6\}, \\ N = {} & \{a^k \mid 1 \leq k \leq 11\} \cup \{a^i b, a^i ba \mid 1 \leq i \leq 6\} \cup \{a^i bb \mid 4 \leq i \leq 6\}. \end{aligned}$$

Fig. 2 shows a DFA with 12 states over $\{a, b\}$ that is consistent with P and N despite the formula ϕ being unsatisfiable. This refutes Lemma 15 in [22].

Counterexample ([21]) Next, we give the construction from Theorem 6.2.1 in [21], page 120, adapted to our notation: The reduction is from a constrained version of SAT where in each instance, the number of variables equals the number of clauses and where each clause is either purely positive (all literals are positive) or purely negative (all literals are negative). Let $V = \{x_1, x_2, \ldots, x_n\}$ be a set of variables and $C = \{C_1, C_2, \ldots, C_n\}$ be a set of pure clauses. Let C_P denote the set of clauses containing only positive literals and let C_N, with $C = C_P \cup C_N$. A DFA-Con instance with alphabet $\Sigma = \{a, b\}$ and the sets P and N is constructed as follows.

- $a^n \in P$,
- $\forall \ell \colon 1 \leq \ell < n$, $a^\ell \in N, a^{n+\ell} \in N$,
- $\forall C_i \in C_P, a^{i-1}ba^n b \in P$ and $\forall x_j \notin C_i, a^{i-1}ba^{n-j+1} \in N$,
- $\forall C_i \in C_N, a^{i-1}ba^n b \in N$ and $\forall \neg x_j \notin C_i, a^{i-1}ba^{n-j+1} \in N$.

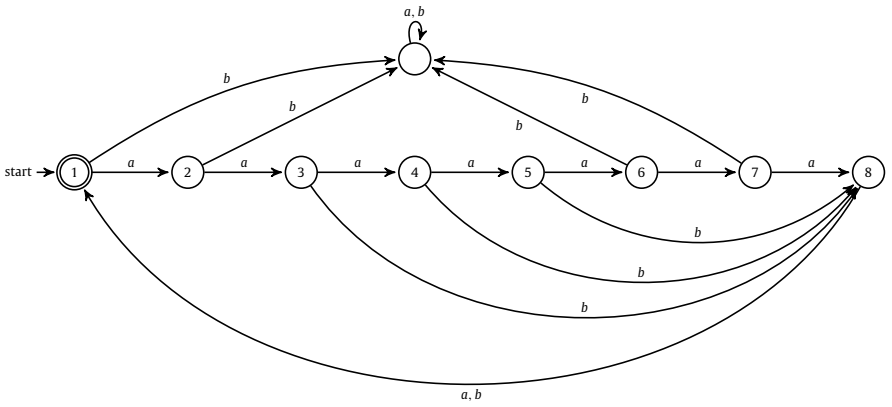

Fig. 3. Automaton with $n+1=9$ states corresponding to the set containing clauses $\{\neg x_2, \neg x_3, \neg x_5\}$, $\{\neg x_8\}$, $\{x_1, x_4, x_8\}$, $\{x_1, x_6, x_8\}$, $\{x_6, x_7, x_8\}$, $\{\neg x_4, \neg x_6\}$, $\{\neg x_1, \neg x_6\}$, $\{\neg x_1, \neg x_7\}$.

We continue by discussing a counterexample. The set of clauses

$$C = \{\{\neg x_2, \neg x_3, \neg x_5\}_1, \{\neg x_8\}_2, \{x_1, x_4, x_8\}_3, \{x_1, x_6, x_8\}_4, \{x_6, x_7, x_8\}_5, \{\neg x_4, \neg x_6\}_6, \{\neg x_1, \neg x_6\}_7, \{\neg x_1, \neg x_7\}_8\}$$

represents an *unsatisfiable* Boolean formula ϕ with $n=8$ variables and $m=8$ where each clause is either purely positive or purely negative. The index imposes an ordering on the clauses. The variable-set

$$V = \{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8\}$$

is also ordered. In accordance with the construction in the proof of Theorem 6.2.1 in [21], page 120, we define the sets

$$\begin{aligned} P =& \{a^8\} \cup \{a^{i-1}ba^8b \mid i \in \{3,4,5\}\}, \\ N =& \{a^k, a^{8+k} \mid 1 \le k < 8\} \cup \{a^{i-1}ba^8b \mid i \in \{1,2,6,7,8\}\} \\ & \cup \{a^{i-1}ba^{8-j+1} \mid i \in \{3,4,5\}, x_j \notin C_i\} \\ & \cup \{a^{i-1}ba^{8-j+1} \mid i \in \{1,2,6,7,8\}, \neg x_j \notin C_i\}, \end{aligned}$$

using the two orderings that are implicit in the proof for Theorem 6.2.1 in the textbook. Claim 1 in the proof of Theorem 6.2.1 claims that every DFA consistent with P and N has at least $n+1$ states. In contrast, Claim 3 speaks about the existence of an n-state DFA. The precise state-bound of the constructed instance in the proof of Theorem 6.2.1 is not explicitly given. We show that the construction with the state-bound given in Claim 1 cannot be correct by giving an $(n+1)$-state DFA in Fig. 3, which is consistent with the sets P and N obtained from an unsatisfiable formula. Also for a bound of exact n states, this proof is not correct, as the example discussed with respect to Gold's construction below in this section is also a counterexample for the construction of Theorem 6.2.1 in [21]. Consider the formula $\varphi = \neg x_1 \wedge x_2 \wedge x_3$ with clause set $\{\{\neg x_1\}, \{x_2\}, \{x_3\}\}$ and variable set $\{x_1, x_2, x_3\}$ implicitly ordered from left to right. Here, $m=3$ and $n=3$. This formula is clearly satisfiable and has pure clauses, but there is no DFA with 3 states that can separate the two sets of words of the de la Higuera construction properly. Let's call the three states q_1, q_2, q_3. Since $a^3 \in P$ and $\{a^1, a^2, a^4, a^5\} \subseteq N$ we get for a: $\delta(q_1, a) = q_2$, $\delta(q_2, a) = q_3$, $\delta(q_3, a) = q_1$ and q_1 must be an accepting state. Now, consider the transition with b for the state q_2. Since the second clause $\{x_2\}$ $(i=2)$ is positive we have by the third item of the construction $a^{i-1}ba^nb \in P$, hence $a^1ba^3b \in P$. By the second condition of this item we have for $x_3 \notin \{x_2\}$: $a^{i-1}ba^{n-j+1} \in N$, hence $a^1ba^1 \in N$. From this point we also get for $x_1 \notin \{x_2\}$: $a^1ba^3 \in N$. These three words now lead to a contradiction concerning $\delta(q_2, b)$. We cannot have $\delta(q_2, b) = q_3$ since then $\delta(q_1, aba) = q_1$, which contradicts $aba \in N$. We cannot have $\delta(q_2, b) = q_1$ since then $\delta(q_1, abaaa) = q_1$, which contradicts $abaaa \in N$. Hence, we must have $\delta(q_2, b) = q_2$. But then, $\delta(q_1, a^1ba^3b) = q_2$, which contradicts $abaaab \in P$. Hence, we cannot find a three-state DFA consistent with the construction.

General issue Both of the constructions ([22] and [21]) rely on the construction in the proof of Theorem 2 in [4] where a reduction from the 'Satisfiability Question' to the problem 'Is there a finite automaton with states reachable by T which agrees with D?' is given. Here, T is a finite set of words representing states, i.e., the words in T are elements of the equivalence classes representing states of the sought automaton; moreover no two elements of T are in the same class. The finite set D (set of data) consists of pairs of words over the input alphabet and single output letters. The key point is that in [4] the term automaton refers to 'Mealy model finite state automaton' and hence, the automata do not include a set of

final states, instead they are enhanced with an output function, which can be interpreted as indicating the acceptance or rejection of a word. For a binary output alphabet, these two models are equivalent considering the class of recognizable languages, but the number of states needed to recognize a language may differ. For instance, the language $\{0,1\}^*0$ can be accepted with a one-state Mealy automaton but for DFAs it can only be accepted by a two-state DFA. Coming back to the consistency problem for DFAs, one is tempted to think that a proof for the NP-hardness for DFA-Con is hidden in the corollary following Theorem 2 in [4] ('$Q_{\min}(D,n)$ is NP-complete for Card(U) = Card(Y) = 2'; U is the input alphabet and Y the output alphabet and $Q_{\min}(D,n)$ asks, given data D and positive integer n, is there a finite automaton with n states which agrees with D?) but with the arguments above, this is not the case. For instance, we can give a counterexample to this claim. Consider the satisfiable formula $\varphi_1 = \neg x_1 \wedge x_2 \wedge x_3$ with $n = 3$ and implicit variable order $x_1 < x_2 < x_3$ as well as clause order $\{\neg x_1\} < \{x_2\} < \{x_3\}$. Following the construction in Theorem 2 in [4], one can verify that there is a three-state Mealy machine recognizing the words in the state characterization matrix correctly, but there is no three-state DFA consistent with the data. Another counterexample is the satisfiable formula $\varphi_2 = x_1 \wedge x_3 \wedge \neg x_2$ with variable order $x_1 < x_3 < x_2$ and clause order $\{x_1\} < \{x_3\} < \{\neg x_2\}$.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Mateus de Oliveira Oliveira reports financial support was provided by Research Council of Norway, Petra Wolf reports financial support was provided by German Research Foundation.

Data availability

No data was used for the research described in the article.

Acknowledgements

Petra Wolf was supported by DFG project FE 560/9-1, and Mateus de Oliveira Oliveira by the RCN projects 288761 and 326537. We thank an anonymous reviewer of LearnAut 2022 for suggesting a simpler construction than we had initially.

References

[1] L. Pitt, M.K. Warmuth, The minimum consistent DFA problem cannot be approximated within any polynomial, J. ACM 40 (1) (1993) 95–142.
[2] E.M. Gold, Language identification in the limit, Inf. Control 10 (5) (1967) 447–474.
[3] D. Angluin, On the complexity of minimum inference of regular sets, Inf. Control 39 (3) (1978) 337–350.
[4] E.M. Gold, Complexity of automaton identification from given data, Inf. Control 37 (3) (1978) 302–320.
[5] L. Pitt, Inductive inference, DFAs, and computational complexity, in: K.P. Jantke (Ed.), Analogical and Inductive Inference, International Workshop AII 1989, in: Lecture Notes in Computer Science, vol. 397, Springer, 1989, pp. 18–44.
[6] R. Parekh, V. Honavar, Learning DFA from simple examples, Mach. Learn. 44 (1) (2001) 9–35.
[7] P.J. Ramadge, W.M. Wonham, Supervisory control of a class of discrete event processes, SIAM J. Control Optim. 25 (1) (1987) 206–230.
[8] A. Groce, D.A. Peled, M. Yannakakis, Adaptive model checking, in: J. Katoen, P. Stevens (Eds.), Tools and Algorithms for the Construction and Analysis of Systems, 8th International Conference, TACAS 2002, Held as Part of the Joint European Conference on Theory and Practice of Software, ETAPS 2002, Proceedings, in: Lecture Notes in Computer Science, vol. 2280, Springer, 2002, pp. 357–370.
[9] H. Mao, Y. Chen, M. Jaeger, T.D. Nielsen, K.G. Larsen, B. Nielsen, Learning deterministic probabilistic automata from a model checking perspective, Mach. Learn. 105 (2) (2016) 255–299.
[10] K. Najim, L. Pibouleau, M. Le Lann, Optimization technique based on learning automata, J. Optim. Theory Appl. 64 (2) (1990) 331–347.
[11] A. Yazidi, N. Bouhmala, M. Goodwin, A team of pursuit learning automata for solving deterministic optimization problems, Appl. Intell. 50 (9) (2020) 2916–2931.
[12] N. Bouhmala, A multilevel learning automata for MAX-SAT, Int. J. Mach. Learn. Cybern. 6 (6) (2015) 911–921.
[13] F. Coste, G. Kerbellec, Learning automata on protein sequences, in: JOBIM, 2006, pp. 199–210.
[14] M.R. Meybodi, H. Beigy, New learning automata based algorithms for adaptation of backpropagation algorithm parameters, Int. J. Neural Syst. 12 (01) (2002) 45–67.
[15] M. Hasanzadeh-Mofrad, A. Rezvanian, Learning automata clustering, J. Comput. Sci. 24 (2018) 379–388.
[16] F. Mayr, S. Yovine, Regular inference on artificial neural networks, in: A. Holzinger, P. Kieseberg, A.M. Tjoa, E.R. Weippl (Eds.), Machine Learning and Knowledge Extraction - Second IFIP TC 5, TC 8/WG 8.4, 8.9, TC 12/WG 12.9 International Cross-Domain Conference, CD-MAKE 2018, Proceedings, in: Lecture Notes in Computer Science, vol. 11015, Springer, 2018, pp. 350–369.
[17] H. Guo, S. Wang, J. Fan, S. Li, Learning automata based incremental learning method for deep neural networks, IEEE Access 7 (2019) 41164–41171.
[18] A. Nowé, K. Verbeeck, M. Peeters, Learning automata as a basis for multi agent reinforcement learning, in: K. Tuyls, P.J. Hoen, K. Verbeeck, S. Sen (Eds.), Learning and Adaption in Multi-Agent Systems, First International Workshop, LAMAS 2005, in: Lecture Notes in Computer Science, vol. 3898, Springer, 2005, pp. 71–85.
[19] A. Rezvanian, A.M. Saghiri, S.M. Vahidipour, M. Esnaashari, M.R. Meybodi, Recent Advances in Learning Automata, Studies in Computational Intelligence, vol. 754, Springer, 2018.
[20] K. Najim, A.S. Poznyak, Learning Automata: Theory and Applications, Elsevier, 2014.
[21] C. de la Higuera, Grammatical Inference: Learning Automata and Grammars, Cambridge University Press, 2010.
[22] H. Fernau, P. Heggernes, Y. Villanger, A multi-parameter analysis of hard problems on deterministic finite automata, J. Comput. Syst. Sci. 81 (4) (2015) 747–765.