The class DBW$\cap$DCW of ω-languages is identifiable in the limit from positive data and membership queries with polynomial time and data

Dana Angluin
Department of Computer Science, Yale University

YALEU/DCS/TR-1528
August 2016

The class $\mathrm{DBW} \cap \mathrm{DCW}$ of ω-languages is identifiable in the limit from positive data and membership queries with polynomial time and data

Dana Angluin
Department of Computer Science, Yale University

1 Introduction

Saoudi and Yokomori defined local and recognizable ω-languages, as well as monadic logic programs [11]. The class of recognizable ω-languages is the same as the class of safety ω-languages [1]. In their paper, Saoudi and Yokomori presented an algorithm, denoted Algorithm(3), and claimed in Theorem 16 that Algorithm(3) efficiently learns in the limit any unknown recognizable ω-language from positive data and restricted subset queries. Unfortunately, this claim about Algorithm(3) is incorrect. In Appendix A we discuss the problems with this claim and other papers affected by these issues.

In the main body of this paper, we prove that the class $\mathrm{DBW} \cap \mathrm{DCW}$ of ω-languages is identifiable in the limit from positive data and membership queries with polynomial time and data. A corollary shows that this also holds with restricted subset queries instead of membership queries. Because $\mathrm{DBW} \cap \mathrm{DCW}$ properly contains the class of safety ω-languages, this proves the existence of algorithms for identification in the limit from positive data and membership queries (or restricted subset queries) with polynomial time and data for safety ω-languages and for monadic logic programs. [1]

2 Preliminaries

Σ denotes a finite alphabet of symbols. The set of all finite strings over Σ is denoted Σ^* and the set of all ω-words over Σ is denoted Σ^ω. The empty string is denoted by ε. The length of a finite string w is denoted by $|w|$.

[1] The subjects of Saoudi and Yokomori's Theorems 16 and 19.

The size of a pair (u, v) of finite strings is $|u| + |v|$. If S is a set of pairs of finite strings, the size of S, denoted $\mathit{size}(S)$, is the sum of the sizes of its elements.

We define deterministic Büchi and co-Büchi ω-automata (denoted by DBW and DCW, respectively.) Each is given by a structure $M = (Q, q_0, \delta, F)$, where Q is a finite set of states, $q_0 \in Q$ is the initial state, $\delta : Q \times \Sigma \to Q$ is the transition function, and $F \subseteq Q$ specifies the acceptance condition. For a DBW automaton, F is the set of accepting states, and for a DCW automaton, F is the set of rejecting states. The size of M, denoted $\mathit{size}(M)$, is the number of states of M.

Given an ω-word w

$$w = \sigma_1\sigma_2\sigma_3 \cdots ,$$

where each $\sigma_n \in \Sigma$, we define the **run** of M on w as the sequence of states

$$q_1, q_2, q_3, \ldots ,$$

where q_1 is the initial state of M and for each index n, $q_{n+1} = \delta(q_n, \sigma_n)$. The run is uniquely determined because we consider deterministic automata.

For a DBW automaton M, the word w is accepted if and only if some element of F occurs infinitely often in the run of M on w. For a DCW automaton M, the word w is accepted if and only if every element of F occurs finitely often in the run of M on w. The notation $L(M)$ denotes the language **recognized** by M, the set of all ω-words accepted by M. We also use DBW and DCW to denote the classes of languages recognized by automata of these types. The size of a language L in DBW, denoted $\mathit{size}(L)$, is the minimum size of a DBW automaton M such that $L = L(M)$.

For a DBW automaton M, a **strongly connected component** is any nonempty set $R \subseteq Q$ of states such that for every $q_1, q_2 \in R$, there is a nonempty finite string x such that $\delta(q_1, x) = q_2$. The strongly connected components that are reachable from the start state are precisely those that may be the set of states visited infinitely often by some ω-word. States not in some strongly connected component are termed **transient**. No transient state can be visited more than once in a run. Inclusion or exclusion of transient states from F does not affect the language recognized.

2.1 The class $\mathrm{DBW} \cap \mathrm{DCW}$

The class of languages $\mathrm{DBW} \cap \mathrm{DCW}$ consists of those L that can be recognized both by a DBW automaton and by a DCW automaton. The class is properly contained in DBW and in DCW, and properly contains the class of safety ω-languages. Löding and Thomas have shown that it is also exactly the class of languages accepted by deterministic weak parity automata [8].

Maler and Pnueli have given a polynomial time algorithm to learn languages in $\mathrm{DBW} \cap \mathrm{DCW}$ from membership and equivalence queries [9]. Their paper includes a number of useful properties of the class. Consider the following condition on a DBW automaton M.

(SCC) For every strongly connected component of M, either all the states in the component are accepting, or all the states in the component are non-accepting.

If M is a DBW automaton recognizing some $L \in \mathrm{DBW} \cap \mathrm{DCW}$, the set of accepting states of M may be changed to satisfy the condition (SCC) while preserving $L(M) = L$.

Moreover, a language in $L \in \mathrm{DBW} \cap \mathrm{DCW}$ has a canonical recognizer, M_L, defined as follows. Given an ω-language L, the **syntactic right congruence** of L is the binary relation on Σ^* defined by $u \equiv v$ if and only if for every ω-word w, $uw \in L$ if and only if $vw \in L$. For languages $L \in \mathrm{DBW} \cap \mathrm{DCW}$, there exists a DBW automaton recoginizing L whose transition graph is isomorphic to the syntactic right congruence of L. [2]

We define the canonical recognizer M_L for L to be the DBW whose transition graph is isomorphic to the syntactic right congruence of L, satisfies the condition (SCC), and has no transient states in F. No DBW automaton recognizing L can have fewer states than M_L, and therefore $\mathit{size}(L) = \mathit{size}(M_L)$.

For learning ω-automata, we consider a particular countable subset of the set of all ω-words, namely the set of all ultimately periodic ω-words. The ω-word w is ultimately periodic if there exist finite strings u and v such that $w = u(v)^\omega$. Regular ω-languages that agree on all ultimately periodic words must agree on all ω-words, so behavior on the subset of ultimately periodic words suffices to learn automata that recognize regular ω-languages. A pair of finite strings (u, v) is used to represent the ω-word $u(v)^\omega$.

The output of Maler and Pnueli's polynomial time algorithm to learn $\mathrm{DBW} \cap \mathrm{DCW}$ using membership and equivalence queries is the canonical recognizer M_L.

2.2 Positive data and queries

We consider learning algorithms that receive positive examples of a target language L and may make queries of a specified type about L. [3]

[2] This property does not hold of DBW languages in general.

[3] For a discussion of types of queries, see [3].

Each example is a pair (u, v) of finite strings such that v is nonempty, representing the ultimately periodic ω-word $u(v)^{\omega}$. A sequence $S_1, S_2, S_3, \ldots$ is a **positive presentation** of a language L if and only if each S_i is a finite set of pairs of finite strings representing elements of L such that

$$S_1 \subseteq S_2 \subseteq S_3 \subseteq \ldots,$$

and for every (u, v) that represents an element of L, there exists an n such that $(u, v) \in S_n$. That is, each pair representing an element of L must eventually be included.

The types of queries we consider are membership queries and restricted subset queries. In a **membership query** about L, the learning algorithm specifies a pair (u, v) of finite strings such that v is nonempty, and receives the answer "yes" if the ω-word $u(v)^{\omega}$ is in L, and "no" otherwise. In a **restricted subset query** about L, the learning algorithm specifies a DBW automaton M and is answered "yes" if $L(M) \subseteq L$ and "no" otherwise.

For identification in the limit from positive data and queries of a specified type τ, we consider a learning algorithm A that takes as input a finite set S of pairs of finite strings (u, v) representing ω-words in the target language L, and may also ask queries of type τ about L. A computes and asks queries, and must eventually halt with an output specifying a DBW automaton M. We denote this output by $A(S)$.

The algorithm A successfully **identifies** the language L **in the limit from positive data and queries of type** τ if and only if for any positive presentation $S_1, S_2, S_3, \ldots$ of L, the sequence

$$A(S_1), A(S_2), A(S_3), \ldots,$$

where A may make queries of type τ to L, is equal to some fixed M for all but finitely many n, and $L(M) = L$. That is, A receives a nested sequence of finite subsets of L, such that each representation of an element of L is eventually included, and makes a sequence of conjectures that eventually stops changing, and is such that the final value of the conjectures is a correct representation of L. If the algorithm A succeeds for every $L \in \mathcal{C}$ for some class $\mathcal{C}$ of ω-languages, then we say the A identifies $\mathcal{C}$ in the limit from positive data and queries of type τ.

We follow de la Higuera's generalization of Gold's definition of what it means for an algorithm A to identify a class $\mathcal{C}$ of languages in the limit with polynomial time and data [4, 6], adapting it to the protocol of positive data and queries of type τ. The adaptations are that the algorithm may return a hypothesis $A(S)$ that is not compatible with S, and in condition (1) below the running time of A is bounded by a polynomial in $size(L)$ as well as $size(S)$, to accomodate the availability of queries.

First, A must identify $\mathcal{C}$ in the limit from positive data and queries of type τ. In addition, A must satisfy the following two conditions on resource use.

1. For any $L \in \mathcal{C}$, given a finite set S of representations of elements of L as input and access to queries of type τ about L, the algorithm must return its conjecture $A(S)$ in time bounded by a polynomial in $size(S)$ and $size(L)$.

2. For any $L \in \mathcal{C}$, there exists a set S_L of representations of elements of L such that $size(S_L)$ is bounded by a polynomial in $size(L)$, and for any set S of representations of examples of L such that $S_L \subseteq S$ we have $A(S) = A(S_L)$, and if M is their common value, then $L(M) = L$.

A must return its conjecture M in time polynomial in the sizes of the input set S and the target concept L. The sample S_L is called a **characteristic sample of L with respect to A**. When the input of A is S_L, it must output a correct conjecture M representing L, and this must also be the output of A for any set S of representations of elements of L that contains S_L as a subset.

3 Identifiability of $\mathrm{DBW} \cap \mathrm{DCW}$

In this section we prove the following main result.

Theorem 1. *There is an algorithm to identify* $\mathrm{DBW} \cap \mathrm{DCW}$ *in the limit from positive data and membership queries with polynomial time and data.*

Given a representation (u, v) of an ultimately periodic ω-word, there is a DBW automaton $M_{(u,v)}$ that accepts only the word $u(v)^\omega$, and is constructable in polynomial time in the size of (u, v). A membership query with (u, v) has the same answer as a restricted subset query with $M_{(u,v)}$. This allows a membership query to be simulated in polynomial time using a restricted subset query, and proves the following corollary of Theorem 1.

Corollary 2. *There is an algorithm to identify* $\mathrm{DBW} \cap \mathrm{DCW}$ *in the limit from positive data and restricted subset queries with polynomial time and data.*

To prove Theorem 1, we follow the approach used to prove that the class of DFA languages can be learned in polynomial time from a representative sample and membership queries [2], using the results of Maler and Pnueli [9] about the class $\mathrm{DBW} \cap \mathrm{DCW}$.

If $M = (Q, q_0, \delta, F)$ is a DBW automaton and $q \in Q$, then M_q is the DBW automaton (Q, q, δ, F), equal to M with q replacing the start state. An ω-word w **distinguishes** q_1 from q_2 in M provided $w \in L(M_{q_1}) \oplus L(M_{q_2})$, that is, w is accepted starting at one of q_1 or q_2, and rejected starting at the other. The following is from Maler and Pnueli [9].

Lemma 3. *Let $L \in \mathrm{DBW} \cap \mathrm{DCW}$ and let M_L be its canonical recognizer, with $n = \mathit{size}(M_L)$. For distinct states q_1 and q_2 of M_L, there exists a pair of finite strings (u, v) of size at most n^2 such that $u(v)^\omega$ distinguishes q_1 from q_2 in M_L.*

Given $L \in \mathrm{DBW} \cap \mathrm{DCW}$, let M_L be its canonical recognizer. If S is a finite set of pairs of finite strings (u, v) then S is a **representative sample** of L provided that the following conditions are satisfied.

1. For every $(u, v) \in S$, $u(v)^\omega \in L$.
2. For every pair of distinct states q_1 and q_2 of M_L, there exists a pair $(u, v) \in S$ and a suffix u_2 of u such that $u_2(v)^\omega$ distinguishes q_1 from q_2 in M_L.

Lemma 4. *Let $L \in \mathrm{DBW} \cap \mathrm{DCW}$ and $n = \mathit{size}(L)$. Then there exists a representative sample R_L of L of size $O(n^3)$.*

Proof. Consider the canonical recognizer M_L for L. Then M_L has n states. There exists a set D of at most $n - 1$ pairs of finite strings, each of size at most n^2, such that for each pair q_1 and q_2 of states of M_L, there exists $(u, v) \in D$ such that $u(v)^\omega$ distinguishes q_1 from q_2 in M_L.

Such a set D may be constructed by repeatedly refining a partition of the states of M_L, starting with all the states in one block, until each state is in its own block. For the refinement step, consider a block that has at least two distinct states q_1 and q_2. By Lemma 3, there exists a pair (u, v) of size at most n^2 that distinguishes q_1 from q_2 in M_L. Add the element (u, v) to D and split the block into two blocks: those states from which $u(v)^\omega$ is accepted, and those from which it is rejected. After $n - 1$ such refinements, all the blocks of the partition will be singletons.

The set R_L is defined as follows. For each pair $(u_2, v) \in D$ there is a state q of M_L such that $u_2(v)^\omega$ is accepted from q. Let u_1 be a shortest string such that $\delta(q_0, u_1) = q$. Add the element (u_1u_2, v) to R_L. Then R_L contains $O(n)$ pairs, each of size at most $O(n^2)$.

To see that R_L is a representative sample for L, if $(u, v) \in R_L$ then there exists $(u_2, v) \in D$ and a state q of M_L from which $u_2(v)^\omega$ is accepted, and a string u_1 such that $\delta(q_0, u_1) = q$ such that $u = u_1u_2$. Thus, $u(v)^\omega \in L$ and condition (1) is satisfied.

For every pair q_1 and q_2 of distinct states of M_L, there is a pair $(u_2, v) \in D$ such that $u_2(v)^\omega$ distinguishes q_1 from q_2 in M_L. For the pair $(u_2, v) \in D$ there is a state q in M_L from which $u_2(v)^\omega$ is accepted and a string u_1 such that $\delta(q_0, u_1) = q$ and $(u_1u_2, v) \in D$, so condition (2) is satisfied, and R_L is a representative sample. □

We now proceed to the proof of Theorem 1.

Proof. We describe an algorithm ID^ω that takes as input a finite set S of pairs (u, v) of finite strings representing elements of L, may ask membership queries about L, outputs a DBW automaton M and halts.

If S is the empty set, ID^ω outputs a canonical recognizer of the empty language and halts. Otherwise, ID^ω constructs the set E containing all pairs (u_2, v) such that for some $(u, v) \in S$, u_2 is a suffix of u. The set E is nonempty and can be constructed in time polynomial in $\mathit{size}(S)$.

For any finite string $x \in \Sigma^*$, we define $\mathit{row}(x)$ to be the finite function with domain E such that for $(u, v) \in E$ the value of $\mathit{row}(x)(u, v) = 1$ if $xu(v)^\omega \in L$ and $\mathit{row}(x)(u, v) = 0$ otherwise. The values of $\mathit{row}(x)$ can be determined by making $|E|$ membership queries about L.

ID^ω now constructs a DBW automaton as follows. We assume a fixed length-lex ordering of Σ^*. The set Q of states consists initially of just the string ε, marked as unprocessed. If there are any states in Q marked as unprocessed, select the least one, say s, and consider the one-symbol extensions $s\sigma$ of s, in order. If $\mathit{row}(s\sigma)$ is different from $\mathit{row}(t)$ for all strings t currently in Q, then add the string $s\sigma$ to Q, marked as unprocessed. Once all the one-symbol extensions of s have been considered, s is marked as processed. This continues until there are no states in Q marked as unprocessed. The total number of strings added to Q is at most the number of states of M_L, and each one requires processing each of its $|\Sigma|$ one-symbol extensions.

The initial state is the string ε, and the transition function is defined by

$$\delta(s, \sigma) = t,$$

where t is the unique element of Q such that $\mathit{row}(t) = \mathit{row}(s\sigma)$.

The accepting states F are determined as follows. For each state $s \in Q$, if it is transient, it is not included in F. If it is not transient, then there is a shortest nonempty string v such that $\delta(s, v) = s$. The state s is included in F if a membership query to L with $s(v)^\omega$ is answered "yes". Then ID^ω outputs the DBW automaton

$$M = (Q, \varepsilon, \delta, F)$$

and halts.

To see that ID^ω identifies $\mathrm{DBW} \cap \mathrm{DCW}$ in the limit from positive data and membership queries, we argue as follows. The running time of ID^ω is polynomial in $\mathit{size}(M_L) = \mathit{size}(L)$ and $\mathit{size}(S)$.

For any $L \in \mathrm{DBW} \cap \mathrm{DCW}$ with $n = \mathit{size}(L)$, let R_L be a representative sample of L of size $O(n^3)$, by Lemma 4. We show that R_L is a characteristic sample of L with respect to ID^ω, which will conclude the proof.

Suppose S is any finite set of pairs (u, v) representing elements of L such that $R_L \subseteq S$. Consider the algorithm ID^ω run with input S and membership query access to L. Because R_L is a representative sample, for every pair of distinct states q_1 and q_2 of M_L, there exists a pair $(u_2, v) \in E$ that distinguishes q_1 from q_2 in M_L. This guarantees that the strings in the state set Q will consist of the length-lex least string reaching each state q of M_L. The state transitions, based on the values of $\mathit{row}(s)$ and $\mathit{row}(s\sigma)$ for each $s \in Q$ and $\sigma \in \Sigma$, will agree with those of M_L. The choice of states to include in F will likewise agree with the accepting states of M_L.

Thus, M is isomorphic to M_L and $L(M) = L$. This same M will be the output for any finite set S of pairs representing elements of L such that $R_L \subseteq S$, so R_L is a characteristic sample of polynomial size for L with respect to ID^ω, and ID^ω identifies $\mathrm{DBW} \cap \mathrm{DCW}$ in the limit from positive data and membership queries with polynomial time and data. $\square$

4 Acknowledgements

The author thanks Dana Fisman and Takashi Yokomori for their helpful comments on earlier drafts of this paper.

References

[1] Bowen Alpern and Fred B. Schneider. Recognizing safety and liveness. *Distributed Computing*, 2(3):117–126, 1987.

[2] Dana Angluin. A note on the number of queries needed to identify regular languages. *Information and Control*, 51(1):76–87, 1981.

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

[4] Colin de la Higuera. Characteristic sets for polynomial grammatical inference. *Mach. Learn.*, 27(2):125–138, May 1997.

[5] E. Mark Gold. Language identification in the limit. *Information and Control*, 10(5):447–474, 1967.

[6] E Mark Gold. Complexity of automaton identification from given data. *Information and Control*, 37(3):302–320, 1978.

[7] M. Jayasrirani, M. Humrosia Begam, and D. Gnanaraj Thomas. Learning of regular omega-tree languages. In *Grammatical Inference: Algorithms and Applications, 9th International Colloquium, ICGI 2008, Saint-Malo, France, September 22-24, 2008, Proceedings*, pages 295–297, 2008.

[8] Christof Löding and Wolfgang Thomas. *Alternating Automata and Logics over Infinite Words*, pages 521–535. Springer Berlin Heidelberg, 2000.

[9] Oded Maler and Amir Pnueli. On the learnability of infinitary regular sets. *Inf. Comput.*, 118(2):316–326, May 1995.

[10] Dominique Perrin and Jean-Eric Pin. *Infinite words: automata, semigroups, logic and games*. Elsevier, Amsterdam, 2004. Pure and Applied Mathematics Series, vol. 141.

[11] Ahmed Saoudi and Takashi Yokomori. Learning local and recognizable ω-languages and monadic logic programs. In *Proceedings of the First European Conference on Computational Learning Theory*, Euro-COLT '93, pages 157–169, New York, NY, USA, 1994. Oxford University Press, Inc.

[12] D. G. Thomas, M. Humrosia Begam, K. G. Subramanian, and S. Gnanasekaran. Learning of regular bi-omega languages. In *Proceedings of the 6th International Colloquium on Grammatical Inference: Algorithms and Applications*, ICGI '02, pages 283–292, London, UK, UK, 2002. Springer-Verlag.

A Appendix: Problems with Theorem 16 in [11]

It appears that Theorem 16 in the paper of Saoudi and Yokomori [11] is incorrect. The theorem claims that Algorithm(3) in the paper learns an arbitrary recognizable ω-language in the limit from positive data and restricted subset queries, and does so efficiently. Both the correctness and efficiency claims for Algorithm(3) appear to be erroneous. The class of recognizable ω-languages defined by Saoudi and Yokomori is the same as the subclass of safety languages of the regular ω-languages, so the claim in Theorem 16 also concerns the learnability of safety languages.

A.1 Efficiency

Saoudi and Yokomori do not define "efficiently", but applying our modification of de la Higuera's definition [4] of identifiable in the limit from polynomial time and data, we have the conditions: (1) the learning algorithm should return a hypothesis in time bounded by a polynomial in the sum of the lengths of the examples it has seen and the size of the target automaton, and (2) for each target concept there should exist a characteristic sample of size polynomial in the size of a minimum automaton recognizing the concept.

Algorithm(3) does not meet the first criterion. Consider the algorithm run with some target language U over the alphabet Σ and parameter n representing the number of states of a minimum B-type deterministic acceptor for U. The alphabet Γ contains all symbols of the form (i, σ, j) such that $i, j \in \{1, 2, ..., n\}$ and $\sigma \in \Sigma$.

In response to the first positive example

$$w = a_1 a_2 \cdots a_t,$$

Algorithm(3) computes the set of "good words" for w. This set contains every word in Γ^t of the form

$$(1, a_1, i_1)(i_1, a_2, i_2) \ldots (i_{t-1}, a_t, i_t),$$

where each $i_j \in \{1, 2, \ldots, n\}$ and we assume that the initial state is labeled 1. The number of such words is n^t, which is not bounded by a polynomial in n and t.

A.2 Correctness

The more serious issue is that Algorithm(3) does not necessarily identify every recognizable ω-language in the limit. An example will illustrate this

	a	b
1	2	1
2	2	$-$

Figure 1: Transition function for a B-type automaton for U.

issue. Let the alphabet be $\Sigma = \{a, b\}$ and define

$$U = b^\omega + b^* a^\omega.$$

This is a recognizable ω-language. A minimum B-type automaton for it has states $\{1, 2\}$, with initial state 1, both states accepting, and the transition function given in Figure 1.

Note that $a^\omega \in U$. Consider what happens when Algorithm(3) is run with U and $n = 2$, and the first positive example it processes is a^ω. The canonical representation of the example is (λ, a), which is presented to the algorithm as the word $w = aa$.

The algorithm constructs all the "good words" for aa and processes them one at a time. Assume that the first good word processed is

$$\alpha_1 = (1, a, 1)(1, a, 1).$$

This has the 1-prefix

$$P_1(\alpha_1) = (1, a, 1),$$

and the set of 2-factors

$$F_2(\alpha_1) = \{(1, a, 1)(1, a, 1)\}.$$

These are used to modify the initial local system $S = (\emptyset, \Gamma^2)$ to

$$S' = (\{(1, a, 1)\}, \Gamma^2 \setminus \{(1, a, 1)(1, a, 1)\}).$$

Then a restricted subset query is used to determine whether $h(L^\omega(S'))$ is a subset of U, where h is the strictly alphabetic morphism that maps the symbol $(i, \sigma, k) \in \Gamma$ to $\sigma \in \Sigma$.

The answer to the subset query is "yes" because $h(L^\omega(S'))$ is just $\{a^\omega\}$, which is a subset of U. This means that for the subsequent $S_i = (I_i, C_i)$, $(1, a, 1)$ is permanently added to I_i and $(1, a, 1)(1, a, 1)$ is permanently removed from C_i. Once these commitments are made, no further additions to I_i or removals from C_i can create a local system $S_i = (I_i, C_i)$ such that $h(L^\omega(S_i)) = U$.

This is because $b^\omega \in U$, but if transitions are added to allow acceptance of b^ω from state 1, then they will also allow acceptance of ab^ω, which is not in U. Thus in this case, Algorithm(3) will never output a correct hypothesis for U.

A.3 Other consequences

Subsequent work based on the approach of Algorithm(3) is also compromised by these issues. In the paper of Thomas et al. [12], Algorithm(4) is directly based on Algorithm(3) of Saoudi and Yokomori and suffers from the same problems. Theorem 4.4 claims that Algorithm(4) learns an arbitrary regular bi-ω language in the limit from positive data and restricted superset queries, and is not correct.

The paper of Jayasrirani, Begam and Thomas [7] on learning regular ω-tree languages says that it is based on the approach of the Saoudi and Yokomori paper. However, at under three pages, the paper does not provide sufficient details to enable a reader to check its claims. It seems likely that Theorem 6, claiming the existence of an algorithm to learn an arbitrary regular ω-tree language in terms of a Büchi local system, is also compromised.

A.4 Limit identification without resource bounds

If we ignore resource bounds, any class of languages represented by a recursively enumerable class of automata with a decidable membership problem can be identified in the limit using just restricted subset queries (resp., restricted superset queries), with no positive data, using a variant of Gold's method of identification by enumeration [5]. We describe an algorithm A using restricted subset queries.

A runs in stages $n = 1, 2, \ldots$. Let L denote the target language and let $[n]$ denote the set $\{1, 2, \ldots, n\}$. In stage n, A enumerates the first n automata $M_1, M_2, \ldots, M_n$ and the first n examples $x_1, x_2, \ldots, x_n$, and computes the finite sets

$$S_i = L(M_i) \cap \{x_1, x_2, \ldots, x_n\}$$

for $i \in [n]$, using the decidability of membership.

For each $i \in [n]$, A makes a restricted subset query with M_i. If none of the subset queries is answered "yes", then A outputs an arbitrary hypothesis, say M_1, and goes to the next stage. Otherwise, A outputs M_i for the least $i \in [n]$ such that $L(M_i) \subseteq L$ and for no $j \in [n]$ do we have both $L(M_j) \subseteq L$ and $S_i \subsetneq S_j$, and goes to the next stage.

It is not difficult to see that the sequence of conjectures of A finitely converges to M_m for the least m for which $L(M_m) = L$. The algorithm for restricted superset queries is analogous, with subset replaced by superset.

The regular ω-word languages are recognized by nondeterministic Büchi ω-word automata, the regular bi-ω-word languages are recognized by nondeterministic Büchi bi-ω-word automata, and the regular ω-tree languages are recognized by nondeterministic Muller ω-tree automata [10]. These classes of automata are recursively enumerable and their respective membership problems (for ultimately periodic ω-words, ultimately periodic bi-ω words, and regular ω-trees) are decidable, which implies the following.

Corollary 5. *The classes of regular ω-word languages, bi-ω-word languages and regular ω-tree languages are identifiable in the limit from restricted subset (resp., restricted superset) queries.*