Deterministic Weighted Automata under Partial Observability

Jakub Michaliszyn[1][0000−0002−5053−0347] and Jan Otop[1][0000−0002−8804−8011]

University of Wrocław
{jmi,jotop}@cs.uni.wroc.pl

Abstract. Weighted automata is a basic tool for specification in quantitative verification, which allows to express quantitative features of analysed systems such as resource consumption. Quantitative specification can be assisted by automata learning as there are classic results on Angluin-style learning of weighted automata. The existing work assumes perfect information about the values returned by the target weighted automaton. In assisted synthesis of a quantitative specification, knowledge of the exact values is a strong assumption and may be infeasible. In our work, we address this issue by introducing a new framework of *partially-observable* deterministic weighted automata, in which weighted automata return intervals containing the computed values of words instead of the exact values. We study the basic properties of this framework with the particular focus on the challenges of active learning partially-observable deterministic weighted automata.

1 Introduction

Finite automata is a fundamental computational model with a wide range of applications spanning from computational complexity, through AI [17] to formal methods [7]. In some applications, however, the qualitative answers returned by finite automata, i.e., each word is *accepted* or *rejected*, are insufficient. For instance, in formal verification, one can check the existence of execution trances violating a given specification, but violating traces come from a model rather than the actual system and their severity may differ from critical, which are likely to occur in the actual system to one, which are unlikely to be reproduced. Similarly, while checking whether a system has no deadlocks, one can ask whether every request is eventually fulfilled, which lacks performance guarantees involving a bound on the timeframe for fulfilment.

To address these issues, there has been proposed quantitative verification, in which the specification refers to quantitative features of the system. Quantitative verification is based on weighted automata, which return numeric values for words rather than accept/reject words. Weighted automata and their extensions have been extensively studied [9,5,6]. These models can express the severity of errors [11] and various performance metrics such as average response time [6]. The expressive power of such models entails hardness of specification.

Specifying quantitative properties may be difficult because in addition to describing events (such as a system failure) one has to come up with the associated values. This is especially difficult for properties of an approximate nature such as the aforementioned severity of a failure. Furthermore, the precise values are often not that important as we would be typically interested whether the number is within some acceptable interval, e.g., does not exceed our resources. For instance, the exact value of average response time depends on the computing environment, e.g., its cache size, which is typically not modeled precisely. For the same reason, assigning reasonable values of the average response time to traces is considerably more difficult than specifying a deadlock.

In this paper, we address the issue of construction of quantitative specifications. To ease the specification process, we propose a new framework, in which automata do not reveal the exact values. We study this framework from the specification-synthesis perspective , i.e., we ask whether it is possible to semi-automatically produce quantitative specifications using automata-learning approach. The conditions may be more involved; for example, we may want to express properties stating that the values 0-10 are good, 11-20 are satisfactory, and anything over 20 is bad.

1.1 Our framework

We introduce partially-observable deterministic weighted automata (PODWA). These automata behave as regular deterministic weighted automata over $\mathbb{Z}$, but return an interval (from a given finite set of possible intervals) that contains the computed value rather than the value itself. The choice of intervals as partial observations is natural. Other choices are possible, but can increase the complexity – even making the membership problem undecidable.

Our motivation comes from the specification-synthesis via automata learning. The idea is that we would like to be able to synthesize quantitative properties without necessarily providing exact values. For that reason, we focus on problems related to active automata learning. First, we study the equivalence problem. It is fundamental in automata learning as one needs to answer whether the learned automaton is admissible. Second, learning algorithms typically construct the structure of an automaton with no weights [2], which leads to the weight synthesis question: given a PODWA Λ_1 and an automaton structure $\mathcal{A}_2$ (a deterministic finite automaton) without weights, is there a weight assignment for $\mathcal{A}_2$, which makes it equivalent (w.r.t. partial observations) to Λ_1? Specifically, assuming that such a weight assignment does exist, is there one such that weights vales are of polynomial order w.r.t. weights from Λ_1? Finally, active automata learning algorithms construct minimal automata [1,2,14]. Thus, to assess feasibility of learning weighted automata in our framework, we study the minimization problem for PODWA.

1.2 Results

The main contribution of the paper is identifying obstacles in developing a polynomial-time active learning algorithm for the new model. We start with the basic properties of the model. We show that the class of PODWA can express more than regular languages and is closed under the complement, but not under the union or the intersection. Then, we show that:

- the equivalence problem for PODWA is CONP-complete in general, and it can be solved in polynomial time if weights are given in unary,
- there is a PODWA Λ with weights $-1, 0, 1$, such that all equivalent minimal-state automata are isomorphic and have exponential weights, and
- the minimization via state-merging for PODWA with unary weights is NP-complete.

These results highlight challenges in learning weighted automata under partial observation. In order to obtain polynomial-time algorithm for active learning of PODWA, we need to focus on automata with unary weights. However, equivalence up to partial observation is too permissive to have an active learning algorithm. One needs a more rigid equivalence notion, which would make minimization decidable in polynomial time, and prevent exponential blow-up of weights in the minimization process.

1.3 Related work

Typically, the partial observation term applies to equivalence on the set of control states, which has been used to model decisions under imperfect information in Markov decision processes (partially observable Markov decision process [18]), graph games (games with imperfect information [8]), or multi-agent system (multi-player games with imperfect information [3,10]). In contrast, in this work, the state space is intact, and partial observability refers to the returned value. This is related to games with interval objectives [13], in which one of the players objective is to make the numeric outcome of the game fall into a set being a finite union of intervals.

This work is motivated by active automata-learning algorithms, which have been developed for deterministic finite automata [1], deterministic weighted word automata [2] and deterministic weighted tree automata [14] and other types of automata. Similar algorithms have recently been developed for infinite-word automata: deterministic Büchi automata (DBA) [16] and deterministic limit-average automata [15]. These algorithms work in polynomial time even though minimization, closely related to active learning, is NP-complete for DBA. It was made possible thanks to in-depth difficulty assessment of problems related to

active learning, which indicated how to extend the learning framework to make polynomial-time learning algorithms possible [16]. We conduct such an assessment in this work to pave the way for the development of active learning algorithms.

2 Preliminaries

A *word* w is a finite sequence of letters from a finite alphabet Σ. By Σ^* we denote the set of all words over Σ. By $w[i]$ we denote the ith letter of a word w, and $w[i, j]$ stands for the subword $w[i]w[i+1]\ldots w[j]$ of w. The empty word is denoted by ϵ.

Automata and runs. A *deterministic weighted automaton* (DWA) is a tuple $\langle \Sigma, Q, q_0, \delta, \mathbf{c} \rangle$ consisting of

1. an alphabet Σ,
2. a finite set of states Q,
3. an initial state $q_0 \in Q$,
4. a transition function $\delta \colon Q \times \Sigma \to Q$, and
5. a weight function $\mathbf{c} \colon Q \times \Sigma \to \mathbb{Z}$.

The size of a DWA $\mathcal{A}$, denoted by $|\mathcal{A}|$, is its number of states plus the sum of the lengths of all the weights given in binary.

We extend δ to $\hat{\delta} \colon Q \times \Sigma^* \to Q$ inductively: for each q, we set $\hat{\delta}(q, \epsilon) = q$, and for all $w \in \Sigma^*, a \in \Sigma$, we set $\hat{\delta}(q, wa) = \delta(\hat{\delta}(q, w), a)$. The run π of a DWA $\mathcal{A}$ on a word w is the sequence of states $q_0 \hat{\delta}(q_0, w[1]) \hat{\delta}(q_0, w[1, 2]) \ldots$. We do not consider any acceptance condition here.

The semantics of a DWA $\mathcal{A}$ is a function $\mathcal{L}(\mathcal{A})$ from non-empty words $\Sigma^* \setminus \{\epsilon\}$ into integers. For a non-empty word w of length k, we define $\mathcal{L}(\mathcal{A})(w)$ as the sum of weights of transitions along the run of $\mathcal{A}$ on w:
$$\mathcal{L}(\mathcal{A})(w) = \mathbf{c}(q_0, w[1]) + \mathbf{c}(\hat{\delta}(q_0, w[1]), w[2]) + \ldots + \mathbf{c}(\hat{\delta}(q_0, w[1, k-1]), w[k]).$$

Remark 1. The tropical seminring The weighted automata model considered in this paper is an instance of a more general framrework of weighted automata over semirings [9], where the semiring is the tropical semiring restricted to integers.

3 Our framework

A *Partially-Observable DWA*, PODWA, is a pair $\Lambda = (\mathcal{A}, S)$ consisting of a DWA $\mathcal{A}$ and a set of a finite number of pairwise-disjoint intervals S covering $\mathbb{Z}$ called *observations*. We assume that intervals are enumerated by $\{0, \ldots, s\}$ according to the order on $\mathbb{Z}$. The *language* of a PODWA Λ, denoted as $\mathcal{L}(\Lambda)$, is a function from $\Sigma^* \setminus \{\epsilon\}$ to $\{0, \ldots, s\}$ such that $\mathcal{L}(\Lambda)(w)$ is the number of the interval containing $\mathcal{L}(\mathcal{A})(w)$.

A *binary PODWA* is a special case of PODWA having only two intervals: $(-\infty, 0]$ and $(0, +\infty)$. We consider words ending in the interval $(0, +\infty)$ as *accepted*. Then, the function $\mathcal{L}(\Lambda)$ is essentially a characteristic function of a set that can be seen as a classic language.

Example 1. Consider a single-state automaton $\mathcal{A}$ over $\Sigma = \{a, b, c\}$. The weights of the transitions over a, b, c are, respectively, $-1, 0, 1$. Consider the set of intervals $S = \{(-\infty, 0], (0, +\infty)\}$ and the binary PODWA $\Lambda = (\mathcal{A}, S)$. Then, $\mathcal{L}(\Lambda)(w) = 1$ if w contains more occurrences of c than a, and 0 otherwise.

Binary PODWA can define all regular languages (without the empty word) and some non-regular languages (see Example 1). All PODWA-recognizable languages are context-free and can be emulated by a deterministic one-counter automaton. On the other hand, deterministic one-counter automata define languages that cannot be expressed by binary PODWA, as the former rely on the counter value at every transition while the latter are agnostic of the counter value. For instance, a pumping argument shows that the language of words that have the same number of (occurrences of) a and b between every pair of c cannot be expressed by a binary PODWA (or any other PODWA with a reasonable language definition).

Binary PODWA can be easily complemented – it suffices to multiply all the weights by -1 and adjust the initial state (for words with value 0). We show that the class of languages recognizable by binary PODWA is not closed under union nor intersection. We will prove the former; for the latter observe that closure under intersection implies closure under union as the union operation can be expressed by the intersection and complement operations.

Let $\mathcal{L}_\cup$ be the language of words w that the number of occurrences of c is greater than the number of occurrences of b or is greater than the number of occurrences of a. Observe that $\mathcal{L}_\cup$ is the union of two PODWA-recognizable languages $\mathcal{L}_a$, $\mathcal{L}_b$, they can be defined as in Example 1. A simple pumping argument shows that $\mathcal{L}_\cup$ is not PODWA-recognizable.

Lemma 1. $\mathcal{L}_\cup$ *is not PODWA-recognizable.*

Proof. Assume a PODWA $\Lambda = (\mathcal{A}, \{(-\infty, 0], (0, +\infty)\})$ with less than N states that recognizes $\mathcal{L}_\cup$.

Consider the word $w = a^N b^N c^{N+1}$. Clearly, $w \in \mathcal{L}_\cup$ because there are more occurrences of c than a.

Since Λ has less that N states, there is $k \geq 0$ and $l > 0$ with $k + l \leq N$ such that the states $\hat{\delta}(q_0, a^k)$ and $\hat{\delta}(q_0, a^{k+l})$ are the same.

Since the automaton is deterministic, for any j the states $\hat{\delta}(q_0, a^k)$ and $\hat{\delta}(q_0, a^{k+jl})$ are the same. Notice that since the automaton is deterministic, this implies that for any j we have $\hat{\delta}(q_0, a^N) = \hat{\delta}(q_0, a^{N+j \cdot l})$.

Let $w_i = a^{k+i \cdot l}$. We argue that $\mathcal{A}(w_1) - \mathcal{A}(w_0) \geq 0$. Notice that for any j we have $\mathcal{A}(w_{j+1}) - \mathcal{A}(w_j) = \mathcal{A}(w_1) - \mathcal{A}(w_0)$. If this number was negative, for a sufficiently large j we would have

$$\mathcal{A}(a^{N+jl} b^N c^{N+1}) \leq 0$$

which contradicts the fact that this words belongs to $\mathcal{L}_\cup$.

Similarly, there is $k' \geq 0$ and $l' > 0$ with $k' + l' \leq N$ such that the states $\hat{\delta}(q_0, a^N b^{k'})$ and $\hat{\delta}(q_0, a^N b^{k'+l'})$ are the same.

Let $w'_i = a^N b^{k'+i \cdot l'}$. As before, we can show that $\mathcal{A}(w'_1) - \mathcal{A}(w'_0) \geq 0$.

Now consider $w_F = a^{N+l} b^{N+l'} c^{N+1}$. The above reasoning shows that $\mathcal{A}(w_F) \geq \mathcal{A}(w)$. However, since Λ recognizes $\mathcal{L}_\cup$, we have $\mathcal{A}(w_F) \leq 0$ and $\mathcal{A}(w) > 0$, which is a contradiction. $\qquad\square$

3.1 Sample fitting

We briefly discuss the following counterpart of the sample fitting problem, which is related to passive learning: given a set of pairs consisting of a word and an interval, called *the sample*, and a number n, is there a PODWA with n states that is consistent with the sample? The sample fitting problem is NP-complete for PODWA; it is NP-complete even for DFA. However, we discuss it here as the hardness proof is simple and robust.

For membership in NP, observe that if n is larger than the number of letters in the sample (and the sample does not contain a direct contradiction, i.e., a word with different intervals), then such a PODWA always exists (and can be a tree). Otherwise, we can nondeterministically pick a PODWA and check it in polynomial time.

For hardness, consider an instance φ of 3-SAT with variables $p_1, \ldots, p_m$. Consider $n = 1$, $\Sigma = \{q\} \cup \{p_i, \overline{p_1} \mid i \leq m\}$, and $S = \{(-\infty, 0), [0, 1], [2, +\infty)\}$. The sample consists of:

- $(q, [0, 1])$, $(qq, [2, +\infty))$
- $(p_i, [0, 1])$, $(\overline{p_i}, [0, 1])$, $(p_i \overline{p_i}, [0, 1])$ $(p_i \overline{p_i} q, [2, +\infty))$ for each i
- $(xyzq, [2, +\infty))$ for each clause $x \vee y \vee z$ of φ (we identify $\neg p_i$ with $\overline{p_i}$).

If there is a single-state automaton consistent with this sample, then each letter has a value corresponding to the only transition over this letter. The value of each letter is an integer. The first condition guarantees that the value of q is 1. The second guarantees that exactly one letter among p_i, $\overline{p_i}$ has value 1 and the other has the value 0 (we rely on the fact that the weights are over integers). Thus, the values define a valuation of variables $p_1, \ldots, p_m$ from φ. The last condition guarantees that this valuation satisfies every clause of φ, and thus it satisfies φ.

4 Towards active learning PODWA

The sample fitting problem is intractable for one-state automata, which is a strong negative result for passive learning. In this section, we now focus on active learning of automata. The classic L^*-algorithm for active learning of DFA asks membership and equivalence queries. While in the PODWA framework, answering a membership query amounts to evaluating the DWA over the input word and returning the interval containing the value, answering equivalence queries is more involved.

4.1 Equivalence

PODWA Λ_1, Λ_2 are *equivalent* if $\mathcal{L}(\Lambda_1) = \mathcal{L}(\Lambda_2)$. The sets of intervals may be different and hence PODWA equivalence is invariant to linear operations, which are consistently applied to all weights and intervals. The *equivalence problem* asks whether two given PODWAs are equivalent. We show its CONP-hardness via reduction from (the complement of) the subset sum problem [12]. Let $a_1, \ldots, a_k$ be a list of integers and T be the target value represented in binary. W.l.o.g. we assume that $a_1, \ldots, a_k$ are even. We construct two binary PODWA $\Lambda_1 = (\mathcal{A}_1, S), \Lambda_2 = (\mathcal{A}_2, S)$ (where $S = \{(-\infty, 0], (0, +\infty)\}$) such that $\mathcal{A}_1$ computes the possible values of sums of subsets of $\{a_1, \ldots, a_k\}$ minus T, and $\mathcal{A}_2$ returns the value in $\mathcal{A}_1$ plus 1, i.e., $\mathcal{L}(\mathcal{A}_2)(w) = \mathcal{L}(\mathcal{A}_1)(w) + 1$. Observe that Λ_1 and Λ_2 are not equivalent if and only if $\mathcal{A}_1$ returns 0 for some word. For such a word $\mathcal{A}_2$ returns 1, which is in a different interval than 0. Thus, the PODWAs are not equivalent if and only if the subset sum problem has a solution.

Lemma 2. *The equivalence problem for (binary) PODWA is* CONP-*hard.*

Proof. We discuss the construction of DWA $\mathcal{A}_1, \mathcal{A}_2$ such that PODWA $(\mathcal{A}_1, S)$ and $(\mathcal{A}_2, S)$ are equivalent if and only if there is no subsequence of $a_1, \ldots, a_k$, which sums up to T.

Without loss of generality, we assume that all values $a_1, \ldots, a_k$ and T are even. The automaton $\mathcal{A}_1$ works over the alphabet $\{0, 1\}$ and input words are interpreted as the characteristic sequence of picked numbers minus T, i.e., the weighted accumulated over a word $w \in \{0, 1\}$ equals the sum of a_i such that $i \in \{1, \ldots, k\}$ and $w[i] = 1$ with T subtracted. One can easily construct such an automaton with $k + 2$ states $q_0, \ldots, q_{k+1}$: it moves from q_i to q_{i+1} regardless of the letter if $i \leq k - 1$; the transition over 1 have weight a_{i+1} and the transition over 0 has weight 0. Then, from q_k it moves to q_{k+1} with both transitions of the weight $-T$. Finally, in q_{k+1} it has self-loops of the weight 0.

Next, the automaton $\mathcal{A}_2$ has the same structure as $\mathcal{A}_1$, but the last weight is $-T + 1$ rather than $-T$. Observe that if there is a word w distinguishing $\mathcal{L}((\mathcal{A}_1, S))$ and $\mathcal{L}((\mathcal{A}_2, S))$, then it has to have the value 0 in $\mathcal{A}_1$ and 1 in $\mathcal{A}_2$ — since the values of the two automata differ by 1 and the values of $\mathcal{A}_1$ are even. So the two automata are not observationally equivalent exactly when the word w encodes the solution for the considered instance of the subset sum problem. □

The subset sum problem has a pseudo-polynomial time algorithm and hence the hardness result from Lemma 2 relies on weights having exponential values w.r.t. the automata sizes. Assuming unary weights in automata and the interval endpoints leads to a polynomial-time algorithm for equivalence of PODWA. More precisely, a PODWA $(\mathcal{A}, S)$ is *unary* if weights in $\mathcal{A}$ and interval ends in S are represented in unary.

Theorem 1. *The equivalence problem is* CONP-*complete for PODWA and in* PTIME *for unary PODWA.*

Proof. The lower bound for the binary case follows from Lemma 2. For the upper bound, we show that PODWA equivalence reduces to $\mathbb{Z}$-reachability in 2-dimensional vector addition systems (VASS), i.e., reachability in which values of counters may become negative. The weights in the resulting VASS are from the weighted automata. The $\mathbb{Z}$-reachability problem for fixed-dimension VASS is NP-complete if vectors' values are represented in binary, and it is in PTIME if they are represented in unary [4].

First, consider PODWA $\Lambda_1 = (\mathcal{A}_1, S_1)$ and $\Lambda_2 = (\mathcal{A}_2, S_2)$. If they are not equivalent, then there is $i \neq j$ and a word w such that $\mathcal{A}_1(w)$ belongs to an i-th interval and $\mathcal{A}_2(w)$ belongs to a j-th interval. Without loss of generality, $i < j$ and hence there are values λ_1, λ_2 such that $\mathcal{A}_1(w) < \lambda_1$ and $\mathcal{A}_2(w) \geq \lambda_2$. There are

$|S_1| \cdot |S_2|$ candidates for pairs λ_1, λ_2 and one can verify all pairs. Therefore, we assume that λ_1, λ_2 are given and focus on finding w such that $\mathcal{A}_1(w) < \lambda_1$ and $\mathcal{A}_2(w) \geq \lambda_2$.

We construct a VASS $\mathcal{V}$ of dimension 2 such that there is a path from the initial state s_0 with counters $(0, 0)$ to the final state t with counters $(0, 0)$ if and only if there is a word w such that $\mathcal{A}_1(w) < \lambda_1$ and $\mathcal{A}_2(w) \geq \lambda_2$. The VASS $\mathcal{V}$ is as a product of automata $\mathcal{A}_1$ and $\mathcal{A}_2$, where each transition is labeled by a vector of the weights of the corresponding transitions in $\mathcal{A}_1$ and $\mathcal{A}_2$. The $\mathcal{V}$ has an additional sink state t, which is the terminal state, such that from any other state one can reach t over a transition labeled by $(-\lambda_1 + 1, -\lambda_2)$. Additionally, t has self-loops labeled by $(1, 0)$ and $(0, -1)$. Finally, the initial state s of $\mathcal{V}$ is the pair consisting of initial states of $\mathcal{A}_1$ and $\mathcal{A}_2$.

Formally, for $i = 1, 2$ let $\mathcal{A}_i = \langle \Sigma, Q_i, q_{0,i}, \delta_i, \mathbf{c}_i \rangle$. The VASS $\mathcal{V} = \langle Q, q_0, \tau \rangle$ is defined as follows: $Q = Q_1 \times Q_2 \cup \{t\}$, $q_0 = \langle q_{0,1}, q_{0,2} \rangle$, and $\tau \subseteq Q \times \mathbb{Z}^2 \times Q$ consist of three types of tuples:

- tuples $\langle (q, s), x, (q', s') \rangle$, for all $q, q' \in Q_1, s, s' \in Q_2$ such that there exists $a \in \Sigma$ satisfying $\delta_1(q, a) = q'$, $\delta_1(s, a) = s'$, and $x = \langle \mathbf{c}_1(q, a, q'), \mathbf{c}_2(s, a, s') \rangle$
- tuples $\langle (q, s), (-\lambda_1 + 1, -\lambda_2), t \rangle$, for all $q \in Q_1, s \in Q_2$, and
- tuples $\langle t, (1, 0), t \rangle$ and $\langle t, (0, -1), t \rangle$.

Now, assume that there is a word w such that $\mathcal{A}_1(w) < \lambda_1$ and $\mathcal{A}_2(w) \geq \lambda_2$. Then we construct a path in $\mathcal{V}$ corresponding to w, which leads from s with counter values $(0, 0)$ to some state with counter values (a, b), where $a < \lambda_1$ and $b \geq \lambda_2$. Since weights are integers, $a \leq \lambda_1 - 1$. Next, we take a transition to t and the counter values change to (a', b') such that $a' \leq 0$ and $b' \geq 0$. Finally, we can reach counter values $(0, 0)$ by taking self-loops over t labeled by $(1, 0)$ and $(0, -1)$. Conversely, consider a path π in $\mathcal{V}$ from s with counter values $(0, 0)$ to t with counter values $(0, 0)$. Then, let s' be the last state before reaching t and (x, y) be the counter values at that position. Observe that $x \leq \lambda_1 - 1$ and $y \geq \lambda_2$ and hence the prefix of π up to s' with (x, y) corresponds to a word w such that $\mathcal{A}_1(w) < \lambda_1$ and $\mathcal{A}_2(w) \geq \lambda_2$. □

4.2 Unary weights

Theorem 1 suggests that restricting the attention to unary PODWA can make learning feasible. However, below we show that minimization of automata with bounded weights from $\{-1, 0, 1\}$ may involve exponential-blow up weights, i.e., the decrease in the number of states is possible only through introduction of weights of exponential value:

Theorem 2. *There exists a sequence of PODWA $\Lambda_n = (\mathcal{A}_n, S)$, for $n > 1$, with weights $-1, 0, 1$ such that for all $n > 1$ every PODWA $(\mathcal{B}, S)$ equivalent to Λ_n with $\mathcal{B}$ having the minimal number of states, has exponential weights in n.*

Proof. We define, for each $n > 1$, a PODWA $\Lambda_n = (\mathcal{A}_n, \{(-\infty, 0), [0, 0], (0, +\infty)\})$ over $\Sigma = \{a, b, i\}$ with weights $\{-1, 0, 1\}$ such that the minimal equivalent PODWA to Λ_n needs weights exponential in n.

The automaton $\mathcal{A}_n$ is depicted in Figure 1 a). Intuitively, the value of the word depends on its first $n + 1$ letters. If the word starts with the prefix $i^k a$, where $0 \leq k < n$, then it has the value $+1$ unless it is followed by b^{n-k}, in which case its value is 0 (and symmetrically with $i^k b$ and -1). Words i^k have value 0.

An example of a minimal automaton equivalent to Λ_n is depicted in Figure 1 b). To show its minimality, observe that for $j, k \in \{0, \ldots, n+1\}$ s.t. $j < k$, the words i^j and i^k have to lead to different states, because $\mathcal{L}(\Lambda_n)(i^j i^{n-j} a) = 2$ and $\mathcal{L}(\Lambda_n)(i^k i^{n-j} a) = 0$.

There are infinitely many minimal automata equivalent to λ_n though. For example, one can multiply all the weights of the automaton in Figure 1 b) by 2. We can show that all automata equivalent to Λ_n with the minimal number of states are structurally isomorphic to the automaton in Figure 1 b); this proof is relegated to the appendix.

In all such automata for any $j < n$ we have $\mathbf{c}(q_j, a) = -\sum_{k=j+1}^{n} \mathbf{c}(q_k, b)$ and similarly $\mathbf{c}(q_j, b) = -\sum_{k=j+1}^{n} \mathbf{c}(q_k, a)$. Therefore, one can inductively show that for $j < n - 1$ we have $\mathbf{c}(q_j, a) = -\mathbf{c}(q_j, b) = 2^{n-j-2}(\mathbf{c}(q_{n-1}, a) + \mathbf{c}(q_n, a))$. Since $\mathbf{c}(q_{n-1}, a)$ and $\mathbf{c}(q_n, a)$ are both positive (because i^{n-1}, i^n have the value 0 and $i^{n-1}a$, $i^n a$ have positive values), we conclude that the value of $\mathbf{c}(q_0, a)$ is exponential in n. □

Fig. 1. a) The automaton Λ_n. The omitted edges lead to s with weight 0. b) A minimal automaton equivalent to Λ_n.

4.3 Minimization

The L^*-algorithm relies on the right congruence relation, which has its natural counterpart for DWA. The right congruence relation defines the structure of the minimal DWA (which is unique) and hence the active learning algorithm can be applied to minimize DWA. Observe that minimal-size PODWA need not be unique.

Example 2. Consider the two binary PODWA presented in Figure 2. They both define the language such that all word have positive values exept for the word a, which has a negative value. Both PODWA are equivalent and minimal; if there was an equivalent PODWA with the underlying DWA of a single state q, then either $\mathbf{c}(q,a) \geq 1$, which would contradict the value for a, or $\mathbf{c}(q,a) \leq 0$, which would contradict the value for aa. Clearly, the automata are non-isomorphic.

Remark 2 (The right congruence for DWA). For a function $f : \Sigma^* \setminus \{\epsilon\} \to \mathbb{Z}$, consider a relation $\equiv_f$ defined on non-empty words w, v as follows:

$$w \equiv_f v \text{ if and only if for all } u \in \Sigma^* \text{ we have } f(wu) - f(w) = f(vu) - f(v).$$

7

$$S = \{(-\infty, 0], (0, +\infty)\}$$

Fig. 2. Two binary PODWA that are equivalent and minimal but not isomorphic.

The relation $\equiv_f$ is a counterpart of the right congruence relation for DWA and one can easily show the counterpart of the Myhill-Nerode theorem: f is defined by some DWA if and only if $\equiv_f$ has finitely many equivalence classes, and the relation $\equiv_f$ defines the structure of the minimal DWA. This relation cannot be straightforwardly adapted to PODWA as the result $f(wu) - f(w)$ cannot be inferred from observations for wu and w. More generally, Example 2 implies that there is no counterpart of $\equiv_f$ for PODWA as it would imply the uniqueness of the structure of minimal PODWA.

We discuss the complexity of minimization for PODWA, assuming that the set of intervals S is fixed and weights are given in unary. We say that DWA $\mathcal{A}_2$ is *observationally equivalent* to a PODWA $(\mathcal{A}_1, S)$, if PODWA $(\mathcal{A}_1, S)$ and $(\mathcal{A}_2, S)$ are equivalent. The O-minimization problem is to find a minimal-size DWA $\mathcal{A}_2$ that is observationally equivalent to a given PODWA $(\mathcal{A}_1, S)$. We study the decision variant of the O-minimization problem obtained by stating bound k on $\mathcal{A}_2$, i.e., given a PODWA $\Lambda = (\mathcal{A}_1, S)$ and $k > 0$, is there a DWA $\mathcal{A}_2$ with at most k states, which is observationally equivalent to Λ.

Minimization by merging. A natural approach to minimization of automata is to define an equivalence relation on the set of states of the input automaton $\mathcal{A}$, corresponding to states being *semantically indistinguishable*, and construct the output automaton $\mathcal{B}$ based on the equivalence classes. In that approach, semantically indistinguishable are merged into a single state. Minimization by merging alleviates the problems arising from ambiguity of minimal automata; it guarantees that the input automaton and the minimized one are structurally related. We study minimization by merging for PODWA.

A DWA $\mathcal{B}$ is obtained from a DWA $\mathcal{A}$ by *merging* if there is a surjective (partial) function $f: Q_{\mathcal{A}} \to Q_{\mathcal{B}}$ from the set of reachable states of $\mathcal{A}$ onto the set of states $\mathcal{B}$ such that $\delta_{\mathcal{A}}(q, a) = q'$ if and only if $\delta_{\mathcal{B}}(f(q), a) = f(q')$.

The unary O-minimization by merging problem is, given an unary PODWA $(\mathcal{A}, S)$ and $k > 0$, is there a DWA $\mathcal{B}$, with at most k states and (the absolute value of) weights bounded by the weights of $\mathcal{A}$, obtained by merging from $\mathcal{A}$ that is observationally equivalent to $(\mathcal{A}, S)$.

Theorem 3. *The unary O-minimization by merging problem is* NP*-complete.*

Proof. The problem is in NP as one can non-deterministically pick a weighted automaton with unary weights $\mathcal{A}'$ along with the homomorphism witnessing that $\mathcal{A}'$ can be obtained by merging from $\mathcal{A}$. Next, we can check observational equivalence of $\mathcal{A}$ and $\mathcal{A}'$ in polynomial time (Theorem 1).

We show NP-hardness via reduction from the k-coloring problem. Let $G = (V, E)$ be a graph – for readability we assume it is a directed graph. We construct a binary PODWA $\Lambda_G = (\mathcal{A}_G, \{(-\infty, 0], (0, +\infty)\})$, which can be O-minimized to an automaton with $k + 2$ states if and only if the vertices of G can be colored with k colors such that each edge connects vertices with different colors.

Let $\Sigma = \{e^+, e^- \mid e \in E\}$ where $E = \{e_1, \dots, e_m\}$. The states of $\mathcal{A}_G$ are q_0, q_f and $\{q_v : v \in V\}$. For an edge $e_i = (v, u)$ we define $\delta(q_0, e_i^-) = v$ and $\delta(q_0, e_i^+) = u$, i.e., over e_i^-, e_i^+ the automaton reaches both ends of e. All the remaining transitions lead to q_f.

We define weights function **c** so that pairs of states q_v, q_u can be merged if and only if they correspond to vertices u, v not connected in G. For any $e \in E$ we will ensure that in $\mathcal{A}_G$ the values of words $e^- e^-, e^+ e^+$ are negative and the value of words $e^- e^+, e^+ e^-$ are positive. This guarantees that e^+ and e^- cannot lead to

the same state. Intuitively, after e^- the state in $\mathcal{A}_G$ has outgoing transitions over e^-, e^+, where the weight of e^+ is strictly greater than the weight of e^-, and for the state reachable over e^+, the order of weights is the opposite.

For every $e_i = (v, u) \in E$ we define $\mathbf{c}(q_0, e_i^-) = \mathbf{c}(q_0, e_i^+) = -3i - 1$. Then, for q_v we define $\mathbf{c}(q_v, e_i^-) = 3i$ and $\mathbf{c}(q_v, e_i^+) = 3i + 2$. For q_u we define $\mathbf{c}(q_u, e_i^-) = 3i + 2$ and $\mathbf{c}(q_u, e_i^+, q_f) = 3i$. For u that is not an endpoint of e_j we set $\mathbf{c}(q_u, e_j^-) = \mathbf{c}(q_u, e_j^+) = 3j + 1$. The weights $\mathbf{c}(q_f, *)$ are all 0.

We show that G is k-colorable if and only if Λ_G can be O-minimized to an automaton with $k + 2$ states. First, observe that the values $e_i^- e_i^-$ and $e_i^+ e_i^+$ in $\mathcal{A}_G$ are -1 and the values $e_i^- e_i^+$ and $e_i^+ e_i^-$ are 1 and hence q_u and q_v cannot be merged. Second, q_0 and q_f cannot be merged with one another or any other state; all words starting from q_0 are negative, and all word starting from q_0 retain their values. No other state has such a property. Therefore, if $\mathcal{A}_G$ is minimized by merging to an automaton with $k + 2$ states, then k is at least equal to the chromatic number of G.

Conversely, assume that $\lambda \colon V \to \{1, \ldots, k\}$ is a valid coloring of G. We construct a DWA $\mathcal{A}'_G$ with the same structure as $\mathcal{A}_G$, with the property that states corresponding to nodes of the same color have the same values of outgoing edges. Recall that for u that is not an endpoint of e_j we set $\mathbf{c}(q_u, e_j^-) = \mathbf{c}(q_u, e_j^+) = 3j + 1$. Changing any such weight to $3j$ or $3j + 2$ leads to an equivalent automaton. Indeed, the state q_u can be reached with values $-3i - 1$, where $i \neq j$ and hence the values $-3i - 1 + 3j, -3i - 1 + 3j + 1, -3i - 1 + 3j + 2$ are either all positive or all negative. With that observation, we can modify weights in $\mathcal{A}_G$ such that for u, v with the same color, the weights of all outgoing transitions from q_i, q_v are the same and hence the states can be merged. Assume that $u[1], \ldots, u[k]$ have the same color; then for every edge e at most one of these vertexes can be an endpoint of e; if there is such $u[i]$ then we fix weights of all transitions $(q_{u[1]}, e^-), \ldots, (q_{u[k]}, e^-)$ to be the same as the weight of $(q_{u[i]}, e^-)$. If there is no such vertex, we do not change the weights. We fix weights over e^+ accordingly. Observe, the in the resulting automaton states $q_{u[1]}, \ldots, q_{u[k]}$ have all the outgoing transitions to q_f, and transitions over the same letter have the same weight. Therefore, they all can be merged into the same state. □

5   Conclusions

This paper introduces partially-observable deterministic weighted automata, which address the difficulty in specification synthesis originating from the need of feeding the exact values to the specification procedure. We have studied the basic properties of the model as well as problems related to specification synthesis via automata learning: equivalence and minimization. The main contribution of the paper is identifying obstacles in developing polynomial-time active learning algorithm for the new model. While our framework is unlikely to admit such an algorithm, it is possible that restricting the equivalence notion may lead framework admitting polynomial-time active learning algorithm.

References

1. Dana Angluin. Learning regular sets from queries and counterexamples. *Information and computation*, 75(2):87–106, 1987.
2. Amos Beimel, Francesco Bergadano, Nader H. Bshouty, Eyal Kushilevitz, and Stefano Varricchio. Learning functions represented as multiplicity automata. *J. ACM*, 47(3):506–530, 2000.
3. Patrick Blackburn, Johan FAK van Benthem, and Frank Wolter. *Handbook of modal logic.* Elsevier, 2006.
4. Michael Blondin, Alain Finkel, Stefan Göller, Christoph Haase, and Pierre McKenzie. Reachability in two-dimensional vector addition systems with states is pspace-complete. In *30th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2015*, pages 32–43. IEEE Computer Society, 2015.

5. Krishnendu Chatterjee, Laurent Doyen, and Thomas A. Henzinger. Quantitative languages. *ACM Trans. Comput. Log.*, 11(4):23:1–23:38, 2010.
6. Krishnendu Chatterjee, Thomas A. Henzinger, and Jan Otop. Nested weighted automata. *ACM Trans. Comput. Log.*, 18(4):31:1–31:44, 2017.
7. Edmund M Clarke, Thomas A Henzinger, Helmut Veith, and Roderick Bloem. *Handbook of model checking*, volume 10. Springer, 2018.
8. Laurent Doyen and Jean-François Raskin. Games with imperfect information: theory and algorithms. In Krzysztof R. Apt and Erich Grädel, editors, *Lectures in Game Theory for Computer Scientists*, pages 185–212. Cambridge University Press, 2011.
9. Manfred Droste, Werner Kuich, and Heiko Vogler. *Handbook of weighted automata*. Springer Science & Business Media, 2009.
10. Dimitar P Guelev and Catalin Dima. Epistemic atl with perfect recall, past and strategy contexts. In *Computational Logic in Multi-Agent Systems: 13th International Workshop, CLIMA XIII, Proceedings 13*, pages 77–93. Springer, 2012.
11. Thomas A. Henzinger and Jan Otop. From model checking to model measuring. In Pedro R. D'Argenio and Hernán C. Melgratti, editors, *CONCUR 2013 - Concurrency Theory - 24th International Conference. Proceedings*, volume 8052 of *Lecture Notes in Computer Science*, pages 273–287. Springer, 2013.
12. John E. Hopcroft and Jefferey D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Adison-Wesley Publishing Company, Reading, Massachusets, USA, 1979.
13. Paul Hunter and Jean-François Raskin. Quantitative games with interval objectives. In Venkatesh Raman and S. P. Suresh, editors, *34th International Conference on Foundation of Software Technology and Theoretical Computer Science, FSTTCS 2014*, volume 29 of *LIPIcs*, pages 365–377. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2014.
14. Ines Marusic and James Worrell. Complexity of equivalence and learning for multiplicity tree automata. *Journal of Machine Learning Research*, 16:2465–2500, 2015.
15. Jakub Michaliszyn and Jan Otop. Minimization of limit-average automata. In Zhi-Hua Zhou, editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*, pages 2819–2825. ijcai.org, 2021.
16. Jakub Michaliszyn and Jan Otop. Learning infinite-word automata with loop-index queries. *Artif. Intell.*, 307:103710, 2022.
17. Ian Millington. *AI for Games*. CRC Press, 2019.
18. Christos H. Papadimitriou and John N. Tsitsiklis. The complexity of markov decision processes. *Math. Oper. Res.*, 12(3):441–450, 1987.

A   Appendix

Theorem 2. *There exists a sequence of PODWA $\Lambda_n = (\mathcal{A}_n, S)$, for $n > 1$, with weights $-1, 0, 1$ such that for all $n > 1$ every PODWA $(\mathcal{B}, S)$ equivalent to Λ_n with $\mathcal{B}$ having the minimal number of states, has exponential weights in n.*

Proof. Here we only fill the remaining details of the proof presented in the main body of the paper. For readability, we will say "the value $(-\infty, 0)$ / $[0, 0]$ / $(0, +\infty)$" rather than the technically correct "the value $0/1/2$".

Assume that $\mathcal{A}_w$ is an minimal automaton observationally equivalent to $\Lambda_n = (\mathcal{A}_n, S)$. We have already argued that $\mathcal{A}_w$ has $n + 2$ states: the initial state q_0, states q_j, for $j \in \{1, \ldots, n\}$, reachable over the words i^j and the state s reachable over i^{n+1}. Here we argue that the remaining transitions of $\mathcal{A}_w$ are as in Figure 1 b).

Observe that from any state q_j for $j \leq n$ there is a transition over a with a positive weight (so that the word $i^j a$ is in $(0, +\infty)$) and a transition over b with a negative weight (so that the word $i^j b$ is in $(-\infty, 0)$).

The transitions from s can only lead to s: note that all words starting with i^{n+1} have the value $[0, 0]$ in Λ_n. If the word i^{n+2} led to a state q_j for $jn + 1$, then $i^{n+2}a$ would have the value $(0, +\infty)$ by the above observation. It also follows that the weight of all edges from s (to itself) is 0.

It remains to show that the for any $j \leq n$ we have $\delta(q_j, a) = q_{j+1}$ and $\delta(q_j, b) = q_{j+1}$.

First, we show that

$$\text{for all } j < k \leq n \text{ in } \mathcal{A}_w \text{ we have } \delta(q_j, a) \neq \delta(q_k, a) \tag{1}$$

Assume towards contradiction that $\delta(q_j, a) = \delta(q_k, a)$. Consider $w_1 = i^j a$ and $w_2 = i^k a$ such that $j < k$ and $q = \hat{\delta}(q_0, w_1) = \hat{\delta}(q_0, w_2)$. We show that $\hat{\delta}(q, b^{n-k}) = s$. This is because the value of $w_1 b^{n-k}$ and $w_2 b^{n-k} a$ in Λ_n is both $[0, 0]$ and s is the only state where the weight of the transition a is 0.

On the other hand, the value of $\mathcal{A}_w$ for $w_1 b^{n-k}$ is in $(0, +\infty)$. Since $\hat{\delta}(w_1 b^{n-k})$ is s, this means that the value of $\mathcal{A}_w$ for $w_1 b^{n-k} b^{k-j}$ is in $(0, +\infty)$. But the value of $\mathcal{A}_b$ for this word is in $[0, 0]$, which contradicts the equivalence.

We now show that

$$\text{for all } j \text{ we have } \delta(q_j, a) \neq q_0 \text{ and } \delta(q_j, b) \neq q_0 \tag{2}$$

Assume w.l.o.g. that $\delta(q_j, a) = q_0$. Observe that $\hat{\delta}(q_0, i^j ab^{n-j}) = s$. Let Y be the set of states along the run over $i^j ab^{n-j}$, i.e., $Y = \{\hat{\delta}(q_0, w) \mid \exists v \in \Sigma^*.wv = i^j ab^{n-j}\}$ and

$$X = \{q_0, \ldots, q_n, s\} \setminus Y$$

Observe that $\hat{\delta}(q_0, i^j a) = q_0$, i.e., the state q_0 occurs at least twice in the run over $i^j ab^{n-j}$ and hence the set Y has at most $n + 1$ states. Therefore, X is non-empty and does not contain s as $s \in Y$. Let q_G be the state with the greatest index in X. Since some state from Y has a transition over b to s, (1) implies that for $q_G \notin Y$ we have $\delta(q_G, b) \neq s$. Thus, $\delta(q_G, b) = q_L$ for some $L \leq G$.

Now, consider the words of the form $w_c = i^j ai^G (bi^{G-L})^c$ for $c > 0$. The value of Λ_n for each w_c is either $(0, +\infty)$ or $[0, 0]$. However, since all the weights for i are 0, and the transition from q_G over b has a negative weight, for some large enough c the value of $\mathcal{A}_w$ for w_c will be negative – a contradiction.

We finally show that for each for each $k \leq n$ we have $\delta(q_k, a) = \delta(q_k, b) = q_{k+1}$ and $\delta(q_n, a) = \delta(q_n, b) = s$. From (1) and (2) it follows that the states of $\mathcal{A}_w$ after reading words $a, ia, \ldots, i^n a$ are some permutation of the states $q_1, \ldots, q_n, s$ – and the same for b.

Assume w.l.o.g. that for some $k < n$ we have $\delta(q_k, a) \neq q_{k+1}$ (resp., $\delta(q_n, a) \neq s$). It means that there is $l < n$ such that $\delta(q_l, a) = q_m$ for $m \leq l$. Also, there is a state q_s such that $\delta(q_s, b) = q_m$.

Now consider the word

$$w_c = i^s b(ai^{m-l})^c$$

Notice that for each c, we have that $\hat{\delta}(w_c) = q_l$ and $c(q_l, a) > 0$. It follows that for sufficiently large c, the value of $\mathcal{A}_w$ for w_c is in $(0, +\infty)$ – which is a contradiction with the value of $\mathcal{A}_n$.