Polynomial Distinguishability of Timed Automata

Sicco Verwer, Mathijs de Weerdt, and Cees Witteveen

Delft University of Technology
{S.E.Verwer,M.M.deWeerdt,C.Witteveen}@tudelft.nl

Abstract. We study the complexity of identifying (learning) timed automata in the limit from data. Timed automata are finite state models that model time explicitly, i.e., using numbers. Because timed automata use numbers to represent time, they can be exponentially more compact than models that model time implicitly, i.e., using states.

We show three results that are essential in order to exactly determine when timed automata are efficiently identifiable in the limit. First, we show that polynomial distinguishability is a necessary condition for efficient identifiability in the limit. Second, we prove that deterministic time automata with two or more clocks are not polynomially distinguishable. As a consequence, they are not efficiently identifiable. Last but not least, we prove that deterministic timed automata with one clock are polynomially distinguishable, which makes them very likely to be efficiently identifiable in the limit.

1 Introduction

Timed automata [1] (TAs) are finite state models that model time *explicitly*, i.e., using numbers. They can be used to model and reason about real-time systems, see e.g. [2]. In practice, it can be very difficult to construct such an automaton by hand, for instance because expert knowledge is unavailable or hard to obtain. That is why we are interested in automatically identifying such models from data. In this paper, we prove several theorems regarding the complexity of identifying TAs from data.

Often this data is obtained using sensors. This results in a time series of system states: every millisecond the state of (or event occurring in) the system is measured and recorded. From such timed data, we could have opted to identify a model that models time *implicitly*, i.e., using states. Examples of such models are the *deterministic finite state automaton* (DFA), see e.g. [3], and the *hidden Markov model* (HMM) [4]. Our main reason for modeling time explicitly is that modeling time implicitly results in an exponential blow-up of the model size: numbers use a binary representation of time while states use a unary representation of time. Because of this, we believe that if the data contains timed properties, i.e., it can be modeled efficiently using a timed automaton, then it is less efficient and much more difficult to identify an untimed model correctly from this data. In previous work [5], we have experimentally compared a state merging and transition splitting algorithm for identifying a simple timed automaton with the evidence driven state merging method (EDSM) [6] for identifying DFAs on such data. While EDSM has been the most successful method for identifying DFAs from untimed data, on timed data it performed worse than our algorithm.

In contrast to DFAs and HMMs, the identification problem for TAs has not been well-studied. We are only aware of studies on the related problem of the identification of event-recording automata (ERAs) [7]. For example, it has been shown that ERAs are identifiable in the query learning framework [8]. However, the used query learning algorithm requires an exponential amount of queries, and is hence inefficient in the amount of data it requires. Naturally, we would like our identification process to be efficient. This is difficult due to the fact the identification problem for DFAs is NP-complete [9]. This property easily generalizes to the problem of identifying a TA (by setting all time values to 0). Thus, unless $P = NP$, a TA cannot be identified efficiently. Even more troublesome is the fact that the DFA identification problem cannot even be approximated within any polynomial [10]. Hence (since this also generalizes), the TA identification problem is also inapproximable.

These two facts make the prospects of finding an efficient identification process for TAs look very bleak. However, both of these results rely on there being a fixed input for the identification problem (encoding a hard problem). While in normal decision problems this is very natural, in an identification problem the amount of input data is somewhat arbitrary: more data can be sampled if necessary. Therefore, it makes sense to study the behavior of an identification process when is it given more and more data (no longer encoding the hard problem). The framework that studies this behavior is called *identification in the limit* [11]. This framework can be summarized as follows. Let C be a class of languages (for example the regular languages, modeled by DFAs). When given an increasing amount of examples from some language $L \in C$, a limit identification algorithm for C should at some point converge to L. If there exists such an algorithm A, the class C is said to be *identifiable in the limit*. If a polynomial amount of examples in the size of the smallest model for L is sufficient for convergence of A, C is said to be *identifiable in the limit from polynomial data*. If A requires time polynomial in the size of the examples, C is said to be *identifiable in polynomial time*. If both these statements hold, then C is *identifiable from polynomial time and data*, i.e *efficiently identifiable in the limit*.

DFAs have been shown to be efficiently identifiable in the limit using a state merging method [12]. Also, it has been shown that *non-deterministic finite automata* (NFAs) are not efficiently identifiable in the limit [13]. This again generalizes to the problem of identifying a non-deterministic TA. Therefore, we only consider the identification problem for *deterministic timed automata* (DTAs). Our goal is to determine exactly when DTAs are efficiently identifiable in the limit and data. In this paper, we show the following results in order to achieve this goal:

– Polynomial distinguishability (Definition 6) is a necessary condition for efficient identifiability in the limit (Lemma 1).
– DTAs with two or more clocks are not polynomially distinguishable (Proposition 2). Hence, they are not efficiently identifiable (Corollary 1).
– DTAs with one clock (1-DTAs) are polynomially distinguishable (Theorem 5).

The fact that 1-DTAs are polynomially distinguishable is based on a central lemma regarding their modeling power (Lemma 2). These results tell us that 1-DTAs seem to be a good model for identifying a timed system.

The paper is organized as follows. In order to prove our results, we start with a brief introduction to DTAs (Section 2), and a formal explanation of efficient identifiability in the limit (Section 3). We then prove that DTAs are not (Section 4), and that 1-DTAs are (Section 5) polynomially distinguishable. We end our paper with a discussion regarding the obtained results (Section 6).

2 Timed Automata

An *timed automaton* (TA) [1] is an automaton that accepts (or generates) strings with event-time value pairs, called timed strings. A finite *timed string* τ over a finite set of symbols Σ is a sequence $(a_1, t_1)(a_2, t_2) \dots (a_n, t_n)$ of symbol-time value pairs $(a_i, t_i) \in \Sigma \times \mathbb{N}$.[1] We use τ_i to denote the prefix of length i of τ, i.e., $\tau_i = (a_1, t_1) \dots (a_i, t_i)$. Every time value t_i in a timed string represents the time until the occurrence of symbol a_i since the occurrence of the previous symbol a_{i-1}. We define the length of a timed string τ, denoted $|\tau|$, as the number of symbol-time value pairs in τ, i.e., $|\tau| = n$.

In TAs, timing conditions are added using a finite set X of *clocks* and one *clock guard* on every transition. These clocks may have different valuations, but all move at the same speed. A *valuation* v is a mapping from X to $\mathbb{N}$, returning the value of a clock $x \in X$. We can add or subtract constants or other valuations to or from a valuation: if $v = v' + t$ then $\forall x \in X : v(x) = v'(x) + t$, and if $v = v' + v''$ then $\forall x \in X : v(x) = v'(x) + v''(x)$. Every transition δ in a TA is associated with a set of clocks R. When a transition δ occurs (or fires), the values of all the clocks in R are set to 0, i.e., $\forall x \in R : v(x) := 0$. The values of all other clocks remain the same. We say that δ *resets* x if $x \in R$. In this way, clocks are used to record the time since the occurrence of some specific event. Clock guards are then used to change the behavior of the TA depending on the value of clocks. A clock guard g is a boolean constraint defined by the grammar: $g := x \leq c \mid x \geq c \mid g \wedge g$, where $x \in X$ is a clock, and $c \in \mathbb{N}$ is a constant.[2] A valuation v is said to *satisfy* a clock guard g, denoted $v \in g$, if for each clock $x \in X$, whenever each occurrence of x in g is replaced by $v(x)$ the resulting statement is true. A timed automaton is defined as follows:

[1] Sometimes $\mathbb{R}$ is used as a time domain for TAs. However, for identification of TAs $\mathbb{N}$ is sufficient since in practice we always measure time using finite precision.

[2] Since we use the natural numbers to represent time open $(x < c)$ and closed $(x \leq c)$ timed automata are equivalent.

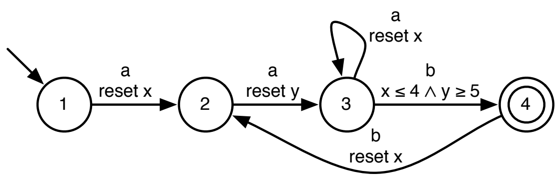

Fig. 1. A timed automaton. The start state (state 1) is denoted by an arrow pointing to it from nowhere. The final state (state 4) has two circles instead of one. The arrows represent transitions. The labels, clock guards, and clock resets are specified for every transition. When no guard is specified it means that the guard is always satisfied.

Definition 1. *A* timed automaton *(TA) is a tuple $\mathcal{A} = \langle Q, X, \Sigma, \Delta, q_0, F\rangle$, where Q is a finite set of states, X is a finite set of clocks, Σ is a finite set of symbols, Δ is a finite set of transitions, q_0 is the start state, and $F \subseteq Q$ is a set of final states.*

A transition $\delta \in \Delta$ is a tuple $\langle q, q', a, g, R\rangle$, where $q, q' \in Q$ are the source and target states, $a \in \Sigma$ is a symbol called the transition label, g is a clock guard, and $R \subseteq X$ is the set of clock resets.

The final states are also known as *accepting states.* The non-final states ($q \in Q \setminus F$) are known as *rejecting states.* Figure 1 shows an example of a TA. The behavior of a TA and the way it depends on time values is defined by what is called a *run* of a TA:

Definition 2. *A finite* run *of a TA $\mathcal{A} = \langle Q, X, \Sigma, \Delta, q_0, F\rangle$ over a (finite) timed string $\tau = (a_1, t_1) \ldots (a_n, t_n)$ is a finite sequence*

$$(q_0, v_0) \xrightarrow{t_1} (q_0, v_0 + t_1) \xrightarrow{a_1} (q_1, v_1) \xrightarrow{t_2} (q_1, v_1 + t_2) \xrightarrow{a_2} (q_2, v_2) \ldots$$
$$\ldots (q_{n-1}, v_{n-1}) \xrightarrow{t_n} (q_{n-1}, v_{n-1} + t_n) \xrightarrow{a_n} (q_n, v_n)$$

such that for all $1 \leq i \leq n : q_i \in Q$, there exists a transition $\delta = \langle q_{i-1}, q_i, a_i, g, R\rangle \in \Delta$ such that $v_{i-1} + t_i \in g$, and for all $x \in X : v_0(x) = 0$, and $v_i(x) := 0$ if $x \in R$, $v_i(x) := v_{i-1}(x) + t_i$ otherwise.

We call a pair (q, v) of a state and a valuation a *timed state.* In a run the subsequence $(q_i, v_i + t) \xrightarrow{a_{i+1}} (q_{i+1}, v_{i+1})$ represents a state transition like in a finite automaton without time. In addition to these, a TA makes time transitions represented by $(q_i, v_i) \xrightarrow{t_{i+1}} (q_i, v_i + t_{i+1})$. A *time transition* of t time units increases the value of all clocks of the TA by t. One can view such a transition as moving from one timed state (q, v) to another timed state $(q, v+t)$ while remaining in the same untimed state q. We say that a timed string τ *reaches* a timed state (q, v) in a TA $\mathcal{A}$ if there exist two time values $t \leq t'$ such that $(q, v') \xrightarrow{t'} (q, v' + t')$ occurs somewhere in the run of $\mathcal{A}$ over τ and $v = v' + t$. If a timed string reaches a timed state (q, v) in $\mathcal{A}$ for some valuation v, it also reaches the untimed state q in $\mathcal{A}$. A timed string *ends* in the last (timed) state it reaches, i.e., (q_n, v_n)

(or q_n). A timed string τ is *accepted* by a TA $\mathcal{A}$ if τ ends in a final state $q_f \in F$. The set of all strings τ that are accepted by $\mathcal{A}$ is called the *language* $L(\mathcal{A})$ of $\mathcal{A}$.

Example 1. Consider the TA $\mathcal{A}$ of Fig. 1. The run of $\mathcal{A}$ over the timed string $\tau = (a,5)(a,6)(a,2)(b,3)$ is given by: $(1,(0,0)) \xrightarrow{5} (1,(5,5)) \xrightarrow{a} (2,(0,5)) \xrightarrow{6} (2,(3,8)) \xrightarrow{a} (3,(3,0)) \xrightarrow{2} (3,(5,2)) \xrightarrow{a} (3,(0,2)) \xrightarrow{3} (3,(3,5)) \xrightarrow{b} (4,(3,5))$, where a timed state (q,v) is written as $(i,(j,k))$ meaning that: q is the state labeled with i, $v(x) = i$, and $v(y) = j$. Since state 4 is a final state, it holds that $\tau \in L(\mathcal{A})$. Note that a run cannot reach state 4 directly after reaching state 3 from state 2: the value of x is greater or equal to the value of y and the guard of the transition to state 4 requires it to be less then the value of y.

In this paper we only consider deterministic timed automata. A TA $\mathcal{A}$ is called *deterministic* (DTA) if for each possible timed string τ there exists at most one run of $\mathcal{A}$ over τ. We only consider DTAs because non-deterministic TAs cannot be identified efficiently in the limit due to the fact that untimed non-deterministic automata are not efficiently identifiable in the limit [13].

3 Efficient Identification in the Limit

An identification process tries to find (learn) a model that explains a set of observations (data). The ultimate goal of such a process is to find a model equivalent to the actual concept that was responsible for producing the observations, called the *target concept*. In our case, we try to find a DTA model $\mathcal{A}$ that is equivalent to a target language L_t, i.e., $L(\mathcal{A}) = L_t$. If this is the case, we say that L_t is identified correctly. We try to find this model using *labeled data*: an *input sample* S is a pair of finite sets of positive examples $S_+ \subseteq L_t$ and negative examples $S_- \subseteq L_t^{\mathrm{c}} = \{\tau \mid \tau \notin L_t\}$. We modify the non-strict set inclusion operators for input samples such that they operate on the positive and negative examples separately, for example if $S = (S_+, S_-)$ and $S' = (S'_+, S'_-)$ then $S \subseteq S'$ means $S_+ \subseteq S'_+$ and $S_- \subseteq S'_-$.

An identification process is called *efficient in the limit* (from polynomial time and data) if the time and data it needs to converge to the target concept are both polynomial in the size of the target concept. Efficient identifiability in the limit can be proved by showing the existence of polynomial *characteristic sets* [13].

Definition 3. *A* characteristic set S_{cs} *of a target language* L_t *for an identification algorithm* A *is an input sample* $\{S_+ \in L_t, S_- \in L_t^c\}$ *such that:*

- *given* S_{cs} *as input, algorithm* A *identifies* L_t *correctly, i.e.,* A *returns an automaton* $\mathcal{A}$ *such that* $L(\mathcal{A}) = L_t$,
- *and given any input sample* $S' \supseteq S_{cs}$ *as input, algorithm* A *still identifies* L_t *correctly.*[3]

[3] This requirement is necessary to avoid collusion: otherwise it is possible to encode L_t in S' making identification a trivial task.

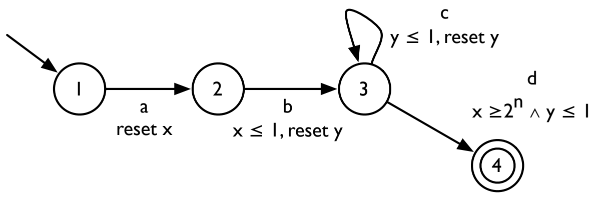

Fig. 2. In order to reach state 4, we require a string of exponential length (2^n). However, due to the binary encoding of clock guards, the DTA is of size polynomial in n.

Definition 4. *A class of automata C is* efficiently identifiable in the limit *if there exist two polynomials p and q and an algorithm A such that:*

- *given an input sample of size n, A runs in time bounded by $p(n)$,*
- *and for every target language $L_t = L(\mathcal{A})$, $\mathcal{A} \in C$, there exists a characteristic set S_{cs} of L_t for A of size bounded by $q(|\mathcal{A}|)$.*

4 Timed Automata Are Not Efficiently Identifiable

DTAs are not efficiently identifiable in the limit. The reason is that in order to reach some parts of a DTA, one may need a timed string of exponential length. We give an example of this in Fig. 2. Formally, this example can be used to show that in general DTAs are not *polynomially reachable*:

Definition 5. *We call a class of automata C* polynomially reachable *if there exists a polynomial function p, such that for any reachable state q from any $\mathcal{A} \in C$, there exists a string τ, with $|\tau| \leq p(|\mathcal{A}|)$, such that τ reaches q in $\mathcal{A}$.*

Proposition 1. *The class of DTAs is not polynomially reachable.*

Proof. Let $C^* = \{\mathcal{A}_n \mid n \geq 1\}$ denote the (infinite) class of DTAs defined by Fig. 2. In any DTA $\mathcal{A}_n \in C^*$, state $q = 4$ can be reached only if both $x \geq 2^n$ and $y \leq 1$ are satisfied. Moreover, $x \leq 1$ is satisfied when y is reset for the first time, and later y can be reset only if $y \leq 1$ is satisfied. Therefore, in order to satisfy both $y \leq 1$ and $x \geq 2^n$, y has to be reset 2^n times. Hence, the shortest string τ that reaches state $q = 4$ is of length 2^n. However, since the clock guards are encoded in binary, the size of $\mathcal{A}_n$ is only polynomial in n. Thus, there exists no polynomial function p such that $\tau \leq p(|\mathcal{A}_n|)$. Since every $\mathcal{A}_n \in C^*$ is a DTA, DTAs are not polynomially reachable.

The non-polynomial reachability of DTAs implies non-polynomial distinguishability of DTAs:

Definition 6. *We call a class of automata C* polynomially distinguishable *if there exists a polynomial function p, such that for any two automata $\mathcal{A}, \mathcal{A}' \in C$ such that $L(\mathcal{A}) \neq L(\mathcal{A}')$, there exists a string $\tau \in L(\mathcal{A}) \triangle L(\mathcal{A}')$, such that $|\tau| \leq p(|\mathcal{A}| + |\mathcal{A}'|)$.*

Proposition 2. *The class of DTAs is not polynomially distinguishable.*

Proof. DTAs are not polynomially reachable, hence there exists no polynomial function p such that for every state q of any DTA $\mathcal{A}$, the length of the shortest timed string τ that reaches q in $\mathcal{A}$ is bounded by $p(|\mathcal{A}|)$. So, there is a DTA $\mathcal{A}$ with a state q for which the length of τ cannot be polynomially bounded by $p(|\mathcal{A}|)$. Given this $\mathcal{A} = \langle Q, X, \Sigma, \Delta, q_0, F \rangle$, construct two DTAs $\mathcal{A}_1 = \langle Q, X, \Sigma, \Delta, q_0, \{q\} \rangle$ and $\mathcal{A}_2 = \langle Q, X, \Sigma, \Delta, q_0, \emptyset \rangle$. By definition of $\mathcal{A}_1$ and $\mathcal{A}_2$, τ is the shortest string such that $\tau \in L(\mathcal{A}_1) \triangle L(\mathcal{A}_2)$. Since $|\mathcal{A}_1| + |\mathcal{A}_2| \leq 2 \times |\mathcal{A}|$, there exists no polynomial function p such that the length of τ is bounded by $p(|\mathcal{A}_1| + |\mathcal{A}_2|)$. Hence the class of DTAs is not polynomially distinguishable.

It is fairly straightforward to show that polynomial distinguishability is a necessary requirement for efficient identifiability:

Lemma 1. *If a class of automata C is efficiently identifiable, then C is polynomially distinguishable.*

Proof. Suppose a class of automata C is efficiently identifiable, but not polynomially distinguishable. Thus, there exists no polynomial function p such that for any two automata $\mathcal{A}, \mathcal{A}' \in C$ (with $L(\mathcal{A}) \neq L(\mathcal{A}')$) the length of the shortest timed string $\tau \in L(\mathcal{A}) \triangle L(\mathcal{A}')$ is bounded by $p(|\mathcal{A}| + |\mathcal{A}'|)$. Let $\mathcal{A}$ and $\mathcal{A}'$ be two automata for which such a function p does not exist and let S_{cs} and S'_{cs} be their polynomial characteristic sets. Let $S = S_{cs} \cup S'_{cs}$ be the input sample for the identification algorithm A for C from Definition 4. Since C is not polynomially distinguishable, neither S_{cs} or S'_{cs} contains an example τ such that $\tau \in L(\mathcal{A})$ and $\tau \notin L(\mathcal{A}')$, or vice versa (because no distinguishing string is of polynomial length). Hence, $S = (S_+, S_-)$ is such that $S_+ \subseteq L(\mathcal{A})$, $S_+ \subseteq L(\mathcal{A}')$, $S_- \subseteq L(\mathcal{A})^{\mathrm{C}}$, and $S_- \subseteq L(\mathcal{A})^{\mathrm{C}}$. The second requirement of Definition 3 now requires that A returns both $\mathcal{A}$ and $\mathcal{A}'$, a contradiction.

This leads to the main result of this section:

Theorem 1. *DTAs cannot be identified efficiently.*

Proof. By Proposition 2 and Lemma 1.

Or more specifically:

Corollary 1. *DTAs with two or more clocks cannot be identified efficiently.*

Proof. The corollary follows from the fact that the argument of Proposition 1 only requires a DTA with two clocks.

This result seems to shatter all hope of ever finding an efficient algorithm for identifying DTAs. Instead of identifying general DTAs, we therefore would like to focus on subclasses of DTAs that are efficiently identifiable.

5 Polynomially Distinguishable Timed Automata

In the previous section we showed DTAs not to be efficiently identifiable in general. The proof for this result was based on the fact that DTAs are not polynomially distinguishable. Since polynomial distinguishability is a necessary requirement for efficient identifiability, we are interested in classes of DTAs that are polynomially distinguishable. In this section, we show that DTAs with a single clock are polynomially distinguishable.

A one-clock DTA (1-DTA) is a DTA that contains exactly one clock, i.e., $|X| = 1$. Our proof that 1-DTAs are polynomially distinguishable is based on the following observation:

– If a timed string τ reaches some timed state (q, v) in a 1-DTA $\mathcal{A}$, then all timed states (q, v') with $v'(x) \geq v(x)$ can be reached in $\mathcal{A}$.

This holds because when a timed string reaches (q, v) it could have made a bigger time transition to reach all bigger valuations. This property is specific to 1-DTAs: a DTA with multiple clocks can wait in q, but only those bigger valuations can be reached where the difference between the clocks remains the same. It is this property of 1-DTAs that allows us to polynomially bound the length of a timed string that distinguishes between two 1-DTAs. We first use this property to show that 1-DTAs are polynomially reachable. We then use a similar argument to show the polynomial distinguishability of 1-DTAs.

Proposition 3. *1-DTAs are polynomially reachable.*

Proof. Given a 1-DTA $\mathcal{A} = \langle Q, \{x\}, \Sigma, \Delta, q_0, F\rangle$, let $\tau = (a_1, t_1) \dots (a_n, t_n)$ be a shortest timed string such that τ reaches some state $q_n \in Q$. Suppose that some prefix $\tau_i = (a_1, t_1) \dots (a_i, t_i)$ of τ ends in some timed state (q, v). Then for any $j > i$, τ_j cannot end in (q, v') if $v(x) \leq v'(x)$. If this were the case, τ_i instead of τ_j could be used to reach (q, v'), and hence a shorter timed string could be used to reach q_n, resulting in a contradiction. Thus, for some index $j > i$, if τ_j also ends in q, then x has to be reset between index i and j in τ. In other words, there exists some index $i < k \leq j$ and a state $q' \neq q$ such that τ_k ends in (q', v_0), where $v_0(x) = 0$. It follows that, if x is reset at index i (τ_i ends in (q, v_0)), there cannot exist any index $j > i$ such that τ_j ends in q. Hence:

– For every state $q \in Q$, the number of prefixes of τ that end in q is bounded by the number of times x is reset by τ.
– For every state $q' \in Q$, there exists at most one index i such that τ_i ends in (q', v_0). In other words, x is reset by τ at most $|Q|$ times.

Consequently, each state is visited at most $|Q|$ times by the run of $\mathcal{A}$ on τ. Thus, the length of a shortest timed string τ that reaches q_n is bounded by $|Q| * |Q|$, which is polynomial in the size of $\mathcal{A}$.

Given that 1-DTAs are polynomially reachable, one would guess that it should be easy to prove the polynomial distinguishability of 1-DTAs. However, this is not the case. The main problem is that when considering the difference between two 1-DTAs, we effectively have access to two clocks instead of one. Note that, although we have access to two clocks, there are no clock guards that bound both clock values. Because of this, we cannot construct DTAs such as the one in Fig. 2. Our proof for the polynomial distinguishability of 1-DTAs follows the same line of reasoning as our proof of Proposition 3, although it is much more complicated to bound the amount of times x is reset. We have split the proof of this bound into several proofs of smaller propositions and lemmas. The main theorem follows from combining these propositions and lemmas.

For the remainder of this section, let $\mathcal{A}_1 = \langle Q_1, \{x_1\}, \Sigma_1, \Delta_1, q_{1,0}, F_1\rangle$ and $\mathcal{A}_2 = \langle Q_2, \{x_2\}, \Sigma_2, \Delta_2, q_{2,0}, F_2\rangle$ be two 1-DTAs. Let $\tau = (a_1, t_1)\dots(a_n, t_n)$ be a shortest string that distinguishes between these 1-DTAs, i.e., $\tau \in L(\mathcal{A}_1)$, $\tau \notin L(\mathcal{A}_2)$, or vice versa, and the size of τ is minimal amongst all such timed strings. The *combined run* of $\mathcal{A}_1$ and $\mathcal{A}_2$ over τ is the sequence:

$$\langle q_{1,0}, v_{1,0}, q_{2,0}, v_{2,0}\rangle \xrightarrow{t_1} \langle q_{1,0}, v_{1,0}+t_1, q_{2,0}, v_{2,0}+t_1\rangle \dots$$
$$\dots \langle q_{1,n-1}, v_{1,n-1}+t_n, q_{2,n-1}, v_{2,n-1}+t_n\rangle \xrightarrow{a_n} \langle q_{1,n}, v_{1,n}, q_{2,n}, v_{2,n}\rangle$$

where $(q_{1,0}, v_{1,0}) \xrightarrow{t_1} (q_{1,0}, v_{1,0}+t_1)\dots(q_{1,n-1}, v_{1,n-1}+t_n) \xrightarrow{a_n} (q_{1,n}, v_{1,n})$ is the run of $\mathcal{A}_1$ over τ and $(q_{2,0}, v_{2,0}) \xrightarrow{t_1} (q_{2,0}, v_{2,0}+t_1)\dots(q_{2,n-1}, v_{2,n-1}+t_n) \xrightarrow{a_n}$ $(q_{2,n}, v_{2,n})$ is the run of $\mathcal{A}_2$ over τ. All the definitions of properties of runs are easily adapted to properties of combined runs. We now use the notion of a combined run to show the following:

Proposition 4. *The length of τ is bounded by a polynomial in the size of $\mathcal{A}_1$, the size of $\mathcal{A}_2$, and the number of times x_1 or x_2 is reset by τ.*

Proof. Suppose that for some index $1 \le i \le n$, τ_i ends in $\langle q_1, v_1, q_2, v_2\rangle$. Using the same argument used in the proof of Proposition 3, one can show that for every $j > i$ and for some v_1' and v_2', if τ_j ends in $\langle q_1, v_1', q_2, v_2'\rangle$, then there exists an index $i < k \le j$ such that τ_k ends in $(q_1', v_{1,0})$ in $\mathcal{A}_1$ for some $q_1' \in Q_1$, or in $(q_2', v_{2,0})$ in $\mathcal{A}_2$ for some $q_2' \in Q_2$. Thus, for every combined state $(q_1, q_2) \in Q_1 \times Q_2$, the number of prefixes of τ that end in (q_1, q_2) is bounded by the number of times r that τ resets either x_1 or x_2. Hence the length of τ is bounded by $|Q_1| * |Q_2| * r$, which is polynomial in r and in the sizes of $\mathcal{A}_1$ and $\mathcal{A}_2$.

We want to bound the number of clock resets in the combined run of a shortest distinguishing string τ. In order to do so, we first prove a restriction on the possible clock valuations in a combined state (q_1, q_2) that is reached directly after one clock x_1 has been reset. In Proposition 3, there was exactly one possible valuation, namely $v(x) = 0$. But since we now have an additional clock x_2, this restriction no longer holds. We can show, however, that after the second time (q_1, q_2) is reached by τ directly after resetting x_1, the valuation of x_2 has to be decreasing with respect to the previous times τ reached (q_1, q_2):

Lemma 2. *If there exists (at least) three indexes $1 \le i < j < k \le n$ such that τ_i ends in $\langle q_1, v_{1,0}, q_2, v_{2,i}\rangle$, τ_j ends in $\langle q_1, v_{1,0}, q_2, v_{2,j}\rangle$, and τ_k ends in $\langle q_1, v_{1,0}, q_2, v_{2,k}\rangle$, then the valuation of x_2 has to be decreasing, i.e., it has to be the case that $v_{2,i}(x_2) > v_{2,k}(x_2)$ and $v_{2,j}(x_2) > v_{2,k}(x_2)$.*

Proof. Without loss of generality we assume that $\tau \in L(\mathcal{A}_1)$, and consequently $\tau \notin L(\mathcal{A}_2)$. Let $l = k + 1$, and let $\tau_{-l} = (a_{l+1}, t_{l+1}) \ldots (a_n, t_n)$ denote the suffix of τ starting at index $l + 1$, i.e., $\tau = \tau_k(a_l, t_l)\tau_{-l}$. Assume for the sake of contradiction that the valuations $v_{2,i}$, $v_{2,j}$, and $v_{2,k}$ are such that $v_{2,i}(x_2) < v_{2,j}(x_2) < v_{2,k}(x_2)$ (the argument below can be repeated for the case when $v_{2,j}(x_2) < v_{2,i}(x_2) < v_{2,k}(x_2)$). Let d_1 and d_2 denote the differences in clock values of x_2 between the first and second, and second and third time (q_1, q_2) is reached by τ, i.e., $d_1 = v_{2,j}(x_2) - v_{2,i}(x_2)$ and $d_2 = v_{2,k}(x_2) - v_{2,j}(x_2)$.

We are now going to make some observations about the acceptance property of the runs of $\mathcal{A}_1$ and $\mathcal{A}_2$ over τ. But, instead of following the path specified by τ, we are going to make a time transition in (q_1, q_2) and then only run the final part of τ. Under our assumption, this is possible because we assume that (q_1, q_2) is reached (at least) three times, and each time the valuation of x_2 is increasing. Hence, in $\mathcal{A}_2$ we can reach the timed state that is reached the last time (q_1, q_2) is visited (at index k) by making a time transition. Because we reach the same timed state and the subsequent run is identical, the acceptance property has to remain the same. We know that $\tau = \tau_k(a_l, t_l)\tau_{-l} \notin L(\mathcal{A}_2)$. Hence, it has to hold that $\tau_j(a_l, t_l + d_2)\tau_{-l} \notin L(\mathcal{A}_2)$ and that $\tau_i(a_l, t_l + d_1 + d_2)\tau_{-l} \notin L(\mathcal{A}_2)$. Similarly, since $\tau \in L(\mathcal{A}_1)$, and since τ_i, τ_j, and τ_k all end in the same timed state $(q_1, v_{1,0})$ in $\mathcal{A}_1$, it holds that $\tau_i(a_l, t_l)\tau_{-l} \in L(\mathcal{A}_1)$ and that $\tau_j(a_l, t_l)\tau_{-l} \in L(\mathcal{A}_1)$. Lets put this type of information in a table (+ denotes true, and − denotes false):

value of t	0	d_1	d_2	$(d_1 + d_2)$
$\tau_i(a_l, t_l + t)\tau_{-l} \in L(\mathcal{A}_1)$	+			
$\tau_i(a_l, t_l + t)\tau_{-l} \in L(\mathcal{A}_2)$				−
$\tau_j(a_l, t_l + t)\tau_{-l} \in L(\mathcal{A}_1)$	+			
$\tau_j(a_l, t_l + t)\tau_{-l} \in L(\mathcal{A}_2)$			−	

Since τ is a shortest distinguishing string, it cannot be that both $\tau_i(a_l, t_l + t)\tau_{-l} \in L(\mathcal{A}_1)$ and $\tau_i(a_l, t_l + t)\tau_{-l} \notin L(\mathcal{A}_2)$ (or vice versa) hold for for some $t \in \mathbb{N}$. Otherwise, $\tau_i(a_l, t_l + t)\tau_{-l}$ would be a shorter distinguishing string for $\mathcal{A}_1$ and $\mathcal{A}_2$. This also holds if we replace i by j. Furthermore, since τ_i ends in the same timed state as τ_j in $\mathcal{A}_2$, it holds that for all $t \in \mathbb{N}$: $\tau_i(a_l, t_l + t)\tau_{-l} \in L(\mathcal{A}_2)$ if and only if $\tau_j(a_l, t_l + t)\tau_{-l} \in L(\mathcal{A}_2)$. The table thus becomes:

value of t	0	d_1	d_2	$(d_1 + d_2)$
$\tau_i(a_l, t_l + t)\tau_{-l} \in L(\mathcal{A}_1)$	+		−	−
$\tau_i(a_l, t_l + t)\tau_{-l} \in L(\mathcal{A}_2)$	+		−	−
$\tau_j(a_l, t_l + t)\tau_{-l} \in L(\mathcal{A}_1)$	+		−	−
$\tau_j(a_l, t_l + t)\tau_{-l} \in L(\mathcal{A}_2)$	+		−	−

Now, since $\tau_i(a_l, t_l + d_1)$ ends in the same timed state as $\tau_j(a_l, t_l)$ in $\mathcal{A}_1$, it holds that $\tau_i(a_l, t_l + d_1)\tau_{-l} \in L(\mathcal{A}_1)$ if and only if $\tau_j(a_l, t_l)\tau_{-l} \in L(\mathcal{A}_1)$ (they reach the same timed state and then their subsequent runs are identical). More generally, for any time value $t \in \mathbb{N}$, $\tau_i(a_l, t_l + d_1 + t)\tau_{-l} \in L(\mathcal{A}_1)$ if and only if $\tau_j(a_l, t_l + t)\tau_{-l} \in L(\mathcal{A}_1)$. Hence we can extend the table in the following way:

value of t	0	d_1	d_2	(d_1+d_2)	$2d_1$	$(2d_1+d_2)$	$3d_1$	$(3d_1+d_2)$
$\tau_i(a_l, t_l+t)\tau_{-l} \in L(\mathcal{A}_1)$	$+$	$+$	$-$	$-$	$+$	$-$	$+$	$-$
$\tau_i(a_l, t_l+t)\tau_{-l} \in L(\mathcal{A}_2)$	$+$	$+$	$-$	$-$	$+$	$-$	$+$	$-$
$\tau_j(a_l, t_l+t)\tau_{-l} \in L(\mathcal{A}_1)$	$+$	$+$	$-$	$-$	$+$	$-$	$+$	$-$
$\tau_j(a_l, t_l+t)\tau_{-l} \in L(\mathcal{A}_2)$	$+$	$+$	$-$	$-$	$+$	$-$	$+$	$-$

This extention can be continued infinitely. Thus, for any value $n \in \mathbb{N}$ it holds that $\tau_i(a_l, t_l + n * d_1)\tau_{-l} \in L(\mathcal{A}_1)$ and $\tau_i(a_l, t_l + n * d_1 + d_2)\tau_{-l} \notin L(\mathcal{A}_1)$. This can only be the case if a different transition is fired for each of these $|\mathbb{N}|$ different values for x. Consequently, $\mathcal{A}$ should contain an infinite amount of transitions, and hence $\mathcal{A}$ is not a 1-DTA, a contradiction.

We have just shown that if τ reaches some combined state (q_1, q_2) twice directly after a reset of x_1 (i.e., $\langle q_1, v_{1,0}, q_2, v\rangle$ is reached at least twice for some v), then the valuation of x_2 has to be *decreasing* with respect to the previous time it reached (q_1, q_2). Without loss of generality we assume that x_1 has already been reset at least twice at previous indexes just before reaching (q_1, q_2). This can be used to show that if τ reaches $(q1, q_2)$ again then:

- x_2 is reset before again reaching (q_1, q_2) and resetting x_1, and
- on the path from (q_1, q_2) to (q_1, q_2), there has to exist at least one transition that cannot be satisfied by a valuation smaller than the one reached by τ.

The first statement follows from the observation that the valuation of x_2 has to be decreasing. The second statement holds because if there is no such transition, then a timed string τ' exists that reaches a smaller valuation of x_2 than τ when it reaches (q_1, q_2) again. By the argument of Lemma 2, it cannot be the case that a non-shortest distinguishing reaches a smaller valuation than τ. Hence this either leads to a contradiction or τ' is a shortest distinguishing string. In this case the statement holds for the shortest distinguishing string τ'. Formally:

Corollary 2. *If for some index i, τ_i ends in (q_1, q_2) directly after a reset of x_1, and if there exists an index $j > i$ such that τ_j ends in (q_1, q_2) directly after a reset of x_1, then there exists an index $i < k \leq j$ such that τ_k ends in $(q_{2,k}, v_{2,0})$.*

Proof. By Lemma 2, it holds that $v_{2,i}(x_2) > v_{2,j}(x_2)$. The value of x_2 can only decrease if it is reset. Hence, there exists an index $i < k \leq j$ at which x_2 is reset.

Corollary 3. *If for some index i, τ_i ends in (q_1, q_2) directly after a reset of x_1, and if there exists another index $j > i$ such that τ_j ends in (q_1, q_2) directly after a reset of x_1, then there exists an index $i < l \leq j$ such that τ_l ends in $\langle q_{1,l}, v_{1,l}, q_{2,l}, v_{2,l}\rangle$, where either $v_{1,l}$ or $v_{2,l}$ is the minimum clock valuation satisfying the last transition fired by τ_l in $\mathcal{A}_1$ or $\mathcal{A}_2$, respectively.*

Proof. Suppose there exists no such index l. In this case, we can subtract 1 from a time value occurring in τ_j to create a timed string τ'_j that follows the same path as τ_j, but ends in $\langle q_1, v_{1,0}, q_2, v_{2,j} - 1\rangle$ instead of $\langle q_1, v_{1,0}, q_2, v_{2,j}\rangle$. Without loss of generality τ'_j is not a shortest distinguishing string (otherwise the corollary also holds). We know that τ'_j reaches a smaller valuation than τ_l, i.e., it holds that $v_{2,j}(x_2) < v_{2,j}(x_2)$. Hence τ'_j can be used to reach a contradiction using the argument of Lemma 2.

We now use these these two properties of τ to polynomially bound the number of different ways in which x_2 can be reset by τ before reaching (q_1, q_2). By Corollary 2, this also polynomially bounds the amount of resets of x_1. In combination with Proposition 4 this proves that 1-DTAs are polynomially distinguishable.

Lemma 3. *The number of times x_2 is reset by τ before reaching a combined state (q_1, q_2) directly after a reset of x_1 is bounded by a polynomial in the sizes of $\mathcal{A}_1$ and $\mathcal{A}_2$*

Proof. Suppose x_1 is reset at index $1 \leq i \leq n$ just before reaching (q_1, q_2), i.e., τ_i ends in $\langle q_1, v_{1,0}, q_2, v_{2,i}\rangle$. Thus, by Lemma 2, $v_{2,i}$ is decreasing with respect to previous indexes. By Corollary 2, we know that x_2 has reset before index i. Let $k < i$ be the largest index before i where x_2 is reset. By Lemma 2, we know that $v_{1,k}$ is decreasing with respect to previous indexes. We also know, by Corollary 3, that there exists an index l such that the last transition that is fired τ_l in either $\mathcal{A}_1$ or $\mathcal{A}_2$ has a clock guard g_1 or g_2 such that the minimal valuation that satisfies g_1 or g_2 is $v_{1,l}$ or $v_{2,l}$, respectively. Let us consider these two cases.

Suppose $v_{1,l}$ is the minimal valuation that satisfies g_1. Since $v_{1,k}$ is decreasing, and x_1 is not reset between index k and l, it has to be the case that $v_{1,l}$ is decreasing. Hence, if at later indexes $i < m < o$ it again occurs that: x_1 is reset at index o just before reaching (q_1, q_2), m is the largest index before o where x_2 is reset, and τ_m ends in the same combined state as τ_k, then there can be no index p such that $v_{1,p}$ satisfies g_1. Thus, if the same combined state (q_1', q_2') is used to reset x_2 before reaching (q_1, q_2) and resetting x_1, there exists at least one transition in $\mathcal{A}_1$ that can no longer be fired on the path from (q_1', q_2') to (q_1, q_2). Hence, there are at most $|Q_1| * |Q_2| * |\Delta_1|$ ways in which this can occur in τ.

Suppose $v_{2,l}$ is the minimal valuation that satisfies g_2. Since $v_{2,i}$ is decreasing, and x_2 is not reset between index l and i, it has to be the case that $v_{2,l}$ is decreasing. Hence if at some later index $i < o$ it occurs again that x_1 is reset at index o just before reaching (q_1, q_2), then there can be no index p such that $v_{2,p}$ satisfies g_2. Hence, there are at most $|\Delta_2|$ ways in which this can occur.

In conclusion, the number of times that x_2 can be reset by τ before reaching (q_1, q_2) directly after a reset of x_1 is bounded by $(|Q_1| * |Q_2| * |\Delta_1| + |\Delta_2|)$, which is polynomial in the sizes of $\mathcal{A}_1$ and $\mathcal{A}_2$.

We are now ready to show the main result of this section:

Theorem 2. *1-DTAs are polynomially distinguishable.*

Proof. By Lemma 2, after the second time a combined state (q_1, q_2) is reached by τ after resetting x_1, it can only be reached again if x_2 is reset. By Lemma 3, the total number of different ways in which x_2 can be reset before reaching (q_1, q_2) and resetting x_1 is bounded by a polynomial p in $|\mathcal{A}_1| + |\mathcal{A}_2|$. Hence the total number of times a combined state (q_1, q_2) can be reached by τ directly after resetting x_1 is bounded by $|Q_1| * |Q_2| * p(|\mathcal{A}_1| + |\mathcal{A}_2|)$. This is polynomial in $|\mathcal{A}_1|$ and $|\mathcal{A}_2|$. By symmetry, this also holds for combined states that are reached directly after resetting x_2. Hence, the total numer of resets of either x_1 or x_2 by τ is bounded by a polynomial in $|\mathcal{A}_1| + |\mathcal{A}_2|$. Since, by Proposition 4, the length of τ is bounded by this number, 1-DTAs are polynomially distinguishable.

6 Discussion and Conclusions

In this paper we have shown that deterministic timed automata (DTAs) cannot be identified efficiently in the limit (Theorem 1). Moreover, this even holds if the class of DTAs is only allowed access to two clocks (Corollary 1). Furthermore, we have shown that DTAs with a single clock (1-DTAs) are polynomially distinguishable (see Definition 6 and Theorem 5). Polynomial distinguishability is a necessary condition for efficient identifiability in the limit (Lemma 1). Therefore, it is an important step for proving the efficient identifiability of 1-DTAs.

It is possible to construct for every DTA a DFA that accepts the same language. However, this DFA is exponentially larger than the original DTA. Therefore, it is not unexpected that DTAs cannot be identified efficiently. For 1-DTAs, the standard method of creating a DFA that accepts the same language (i.e., the region construction [1]) still results in an exponential blowup of the amount of states. Therefore, one may guess that 1-DTAs can not be identified efficiently. Surprisingly, however, in this paper we have shown that 1-DTAs are polynomially distinguishable, which makes them very likely to be efficiently identifiable.

Currently, we are writing an algorithm that we intend to use to prove efficient identifiability based on the results in this paper. The idea is to write a state merging and transition splitting algorithm like our algorithm for identifying simple TAs (see [5]) that uses polynomial distinguishing strings to ensure the correct identification of 1-DTAs. This is similar to the way state merging was used to show the efficient identifiability of DFAs (see [12]).

Besides allowing us to write such an algorithm, the results in this paper have several other important consequences and/or possible applications. We now give a few examples of consequences and applications.

The fact that 1-DTAs are polynomially distinguishable relies on an important lemma regarding their modeling power (Lemma 2). We believe this lemma has consequences beyond the scope of the 1-DTA identification problem. For example, when model checking a system of two 1-DTAs, the search space may be reduced by restricting the search to smaller valuations in combined states.

The efficiency results have important consequences for anyone interested in identifying timed systems (and TAs in particular). Most importantly, they tell us that 1-DTAs seem to be a good model for identifying a timed system. Furthermore, they show that anyone who needs to identify a DTA with two or more clocks should either be satisfied with sometimes requiring an exponential amount of data, or he or she has to find some other method to deal with this problem. This also holds for other learning frameworks.

For instance, in related work, a query learning algorithm is described for identifying event recording automata (ERAs) [8]. An important property of ERAs is that they are determinizable [7]. This property ensures that the language inclusion problem is decidable for determinizable TAs. Since language inclusion is relevant for identification, this seems to indicate that ERAs are a class of automata that are well-suited for identification. However, a class of automata can only be identified efficiently from queries if it is also efficiently identifiable in the limit from data [14]. Thus, our results show that ERAs can never be identified efficiently since an ERA has access to multiple clocks. We believe it would be interesting, and very valuable for real-world applications, to adapt the timed query learning algorithm to the class of 1-DTAs. Our results indicate that this may result in an efficient query learning algorithm for timed systems.

Acknowledgement. We would like to thank Pieter Adriaans for the helpful comments on an early draft of this paper. This research has been supported and funded by the Dutch Ministry of Economic Affairs under the SENTER program, project IS041022 Real-Time Optimalisatie Motor Management.

References

1. Alur, R., Dill, D.L.: A theory of timed automata. Theoretical Computer Science 126, 183–235 (1994)
2. Larsen, K.G., Petterson, P., Yi, W.: Uppaal in a nutschell. International journal on software tools for technology transfer 1(1-2), 134–152 (1997)
3. Sipser, M.: Introduction to the Theory of Computation. PWS Publishing (1997)
4. Rabiner, L.R.: A tutorial on hidden markov models and selected applications in speech recognition. Proceedings of the IEEE 77 (1989)
5. Verwer, S., de Weerdt, M., Witteveen, C.: An algorithm for learning real-time automata. In: Benelearn, pp. 128–135 (2007)
6. Lang, K.J., Pearlmutter, B.A., Price, R.A.: Results of the Abbadingo One DFA learning competition and a new evidence-driven state merging algorithm. In: Honavar, V.G., Slutzki, G. (eds.) ICGI 1998. LNCS (LNAI), vol. 1433. Springer, Heidelberg (1998)
7. Alur, R., Fix, L., Henzinger, T.A.: Event-clock automata: a determinizable class of timed automata. Theoretical Computer Science 211(1), 253–273 (1999)
8. Grinchtein, O., Jonsson, B., Petterson, P.: Inference of event-recording automata using timed decision trees. In: Baier, C., Hermanns, H. (eds.) CONCUR 2006. LNCS, vol. 4137, pp. 435–449. Springer, Heidelberg (2006)
9. Gold, E.M.: Complexity of automaton identification from given data. Information and Control 37(3), 302–320 (1978)
10. Pitt, L., Warmuth, M.K.: The minimum consistent DFA problem cannot be approximated within any polynomial. Journal of the ACM 40(1), 95–142 (1993)
11. Gold, E.M.: Language identification in the limit. Information and Control 10(5), 447–474 (1967)
12. Oncina, J., Garcia, P.: Inferring regular languages in polynomial update time. In: Pattern Recognition and Image Analysis. Series in Machine Perception and Artificial Intelligence, vol. 1, pp. 49–61. World Scientific, Singapore (1992)
13. de la Higuera, C.: Characteristic sets for polynomial grammatical inference. Machine Learning 27 (1997)
14. Parekh, R., Hanovar, V.G.: On the relationship between models for learning in helpful environments. In: Oliveira, A.L. (ed.) ICGI 2000. LNCS (LNAI), vol. 1891, pp. 207–220. Springer, Heidelberg (2000)