Determinisability of one-clock timed automata

Lorenzo Clemente
University of Warsaw, Poland
clementelorenzo@gmail.com

Sławomir Lasota
University of Warsaw, Poland
sl@mimuw.edu.pl

Radosław Piórkowski
University of Warsaw, Poland
r.piorkowski@mimuw.edu.pl

arXiv:2007.09340v1 [cs.FL] 18 Jul 2020

Abstract

The deterministic membership problem for timed automata asks whether the timed language recognised by a nondeterministic timed automaton can be recognised by a deterministic timed automaton. We show that the problem is decidable when the input automaton is a one-clock nondeterministic timed automaton without epsilon transitions and the number of clocks of the deterministic timed automaton is fixed. We show that the problem in all the other cases is undecidable, i.e., when either 1) the input nondeterministic timed automaton has two clocks or more, or 2) it uses epsilon transitions, or 3) the number of clocks of the output deterministic automaton is not fixed.

2012 ACM Subject Classification Theory of computation - Automata over infinite objects; Theory of computation - Quantitative automata; Theory of computation - Timed and hybrid models.

Keywords and phrases Timed automata, determinisation, deterministic membership problem

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2020.38

Funding *Lorenzo Clemente*: Partially supported by the Polish NCN grant 2017/26/D/ST6/00201.
Sławomir Lasota: Partially supported by the Polish NCN grant 2019/35/B/ST6/02322 and by the ERC grant LIPA, agreement no. 683080.
Radosław Piórkowski: Partially supported by the Polish NCN grant 2017/27/B/ST6/02093.

Acknowledgements We thank S. Krishna for fruitful discussions and the anonymous reviewers for their constructive comments.

© Lorenzo Clemente and Sławomir Lasota and Radosław Piórkowski;
licensed under Creative Commons License CC-BY

31st International Conference on Concurrency Theory (CONCUR 2020).
Editors: Igor Konnov and Laura Kovács; Article No. 38; pp. 38:1–38:21
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

Nondeterministic timed automata (NTA) are one of the most widespread model of real-time reactive systems. They are an extension of finite automata with real-valued clocks which can be reset and compared by inequality constraints. The nonemptiness problem for NTA is decidable and in fact PSPACE-complete, as shown by Alur and Dill in their landmark paper [3]. As a testimony to the importance of the model, the authors received the 2016 Church Award [1] for the invention of timed automata. This paved the way to the automatic verification of timed systems, leading to mature tools such as UPPAAL [9], UPPAAL Tiga (timed games) [16], and PRISM (probabilistic timed automata) [33]. The reachability problem is still a very active research area to these days [22, 30, 2, 26, 27, 29], as well as expressive generalisations thereof, such as the binary reachability problem [20, 21, 32, 24].

Deterministic timed automata (DTA) form a strict subclass of NTA where the next configuration is uniquely determined from the current one and the timed input symbol. The class of DTA enjoys stronger properties than NTA, such as decidable universality and inclusion problems and closure under complementation [3]. Moreover, the more restrictive nature of DTA is necessary in several applications of timed automata, such as test generation [37], fault diagnosis [13], and learning [46, 42], winning conditions in timed games [5, 31, 14], and in a notion of recognisability of timed languages [35]. For these reasons, and for the more general quest of understanding the nature of the expressive power of nondeterminism in timed automata, many researchers have focused on defining determinisable classes of timed automata, such as strongly non-zeno NTA [6], event-clock NTA [4], and NTA with integer-resets [41]. The classes above are not exhaustive, in the sense that there are NTA recognising deterministic timed languages not falling into any of the classes above.

Another remarkable subclass of NTA is obtained by requiring the presence of just one clock (without epsilon transitions). The resulting class of NTA_1 is incomparable with DTA: For instance, NTA_1 are not closed under complement (unlike DTA) and there are very simple DTA languages which are not recognisable by any NTA_1. Nonetheless, NTA_1, like DTA, have decidable inclusion, equivalence, and universality problems [38, 34], albeit the complexity is non-primitive recursive [34, Corollary 4.2] (see also [39, Theorem 7.2] for an analogous lower bound for the satisfiability problem of metric temporal logic). Moreover, the non-emptiness problem for NTA_1 is NLOGSPACE-complete (vs. PSPACE-complete for unrestricted NTA and DTA, already with two clocks [22]), and computing the binary reachability relation is simpler when there is only one clock than in the general case [18].

The deterministic membership problem. The DTA *membership problem* asks, given an NTA, whether there exists a DTA recognising the same language. There are two natural variants of this problem, which are obtained by restricting the resources available to the sought DTA. Let $k \in \mathbb{N}$ be a bound on the number of clocks, and let $m \in \mathbb{N}$ be a bound on the maximal absolute value of numerical constants. The DTA_k and $\text{DTA}_{k,m}$ *membership problems* are the restriction of the problem above where the DTA is required to have at most k clocks, resp., at most k clocks and absolute value of maximal constant bounded by m. Notice that we do not bound the number of control locations of the DTA, which makes the problem non-trivial.

Since regular languages are deterministic, the DTA_k membership problem can be seen as a quantitative generalisation of the regularity problem. For instance, the DTA_0 membership problem is exactly the regularity problem since a timed automaton with no clocks is the same as a finite automaton. We remark that the regularity problem is usually undecidable for nondeterministic models of computation generalising finite automata, e.g., context-free grammars/pushdown automata [40, Theorem 6.6.6], labelled Petri nets under reachability semantics [45], Parikh automata [15], etc. One way to obtain decidability is to either restrict the input model to be deterministic (e.g., [44, 45, 8]), or to consider finer notions of equivalence, such as bisimulation (e.g., [28]).

This negative situation is generally confirmed for timed automata. For every number of clocks $k \in \mathbb{N}$ and maximal constant m, the DTA, DTA_k, and $\text{DTA}_{k,m}$ membership problems are known to be undecidable when the input NTA has ≥ 2 clocks, and for 1-clock NTA with epsilon transitions [23, 43]. To the best of our knowledge, the deterministic membership problem was not studied before when the input automaton is NTA_1 without epsilon transitions.

Contributions. We complete the study of the decidability border for the deterministic membership problem initiated in [23, 43]. Our main result is the following.

▶ **Theorem 1.1.** *The* DTA_k *membership and the* $\text{DTA}_{k,m}$ *membership problems are decidable for* NTA_1 *languages.*

Our decidability result contrasts starkly with the abdundance of undecidability results for the regularity problem. We establish decidability by showing that if a $\text{NTA}_{k,m}$ recognises a DTA_k language, then in fact it recognises a $\text{DTA}_{k,m}$ language and moreover there is a computable bound on the number of control locations of the deterministic acceptor (c.f. Lemma 4.1). This provides a decision procedure since there are finitely many DTA once the number of clocks, the maximal constant, and the number of control locations are fixed.

In our technical analysis we find it convenient to introduce the so called *always resetting* subclass of NTA_k. These automata are required to reset at least one clock at every transition and are thus of expressive power intermediate between NTA_{k-1} and NTA_k. Always resetting NTA_2 are strictly more expressive than NTA_1: For instance, the language of timed words of the form $(a, t_0)(a, t_1)(a, t_2)$ s.t. $t_2 - t_0 > 2$ and $t_2 - t_1 < 1$ can be recognised by an always resetting NTA_2 but by no NTA_1. Despite their increased expressive power, always resetting NTA_2 still have a decidable universality problem (the well-quasi order approach of [38] goes through), which is not the case for NTA_2. Thanks to this restricted form, we are able to provide in Lemma 4.1 an elegant characterisation of those NTA_1 languages which are recognised by an always resetting DTA_k.

We complement the decidability result above by showing that the problem becomes undecidable if we do not restrict the number of clocks of the DTA.

▶ **Theorem 1.2.** *The* DTA *and* $\text{DTA}_{\_,m}$ *($m > 0$) membership problems are undecidable for* NTA_1 *without epsilon transitions.*

Finally, by refining the analysis of [23], we show that the DTA_k and $\text{DTA}_{k,m}$ membership problems for NTA_1 are non-primitive recursive.

▶ **Theorem 1.3.** *The* DTA_k *and* $\text{DTA}_{k,m}$ *membership problems are* HyperAckermann*-hard for* NTA_1.

Related research. Many works addressed the construction of a DTA equivalent to a given NTA (see [10] and references therein), however since the general problem is undecidable, one has to either sacrifice termination, or consider deterministic under/over-approximations. In a related line of work, we have shown that the *deterministic separability problem* is decidable for the full class of NTA, when the number of clocks of the separator is given in the input [19]. This contrasts with undecidability of the corresponding membership problem. Decidability of the deterministic separability problem when the number of clocks of the separator is not provided remains a challenging open problem.

2 Preliminaries

Timed words and languages. Fix a finite alphabet Σ. Let $\mathbb{R}$ and $\mathbb{R}_{\geq 0}$ denote reals and nonnegative reals[1], respectively. A *timed word* over Σ is any sequence of the form

$$w \;=\; (a_1, t_1) \ldots (a_n, t_n) \;\in\; (\Sigma \times \mathbb{R}_{\geq 0})^* \tag{1}$$

which is *monotonic*, in the sense that the timestamps t_i's satisfy $0 \leq t_1 \leq t_2 \leq \cdots \leq t_n$. Let $\mathbb{T}(\Sigma)$ be the set of all timed words over Σ, and let $\mathbb{T}_{\geq t}(\Sigma)$ be, for $t \in \mathbb{R}_{\geq 0}$, the set of timed words with $t_1 \geq t$. A *timed language* is a subset of $\mathbb{T}(\Sigma)$.

The concatenation $w \cdot v$ of two timed words w and v is defined only when the first time-stamp of v is greater or equal than the last timestamp of w. Using this partial operation, we define, for a timed word $w \in \mathbb{T}(\Sigma)$ and a timed language $L \subseteq \mathbb{T}(\Sigma)$, the left quotient $w^{-1}L := \{v \in \mathbb{T}(\Sigma) \mid w \cdot v \in L\}$. Clearly $w^{-1}L \subseteq \mathbb{T}_{\geq t_n}(\Sigma)$.

Clock constraints and regions. Let $\mathtt{X} = \{\mathtt{x}_1, \ldots, \mathtt{x}_k\}$ be a finite set of clocks. A *clock valuation* is a function $\mu \in \mathbb{R}^{\mathtt{X}}_{\geq 0}$ assigning a non-negative real number $\mu(\mathtt{x})$ to every clock $\mathtt{x} \in \mathtt{X}$. A *clock constraint* is a quantifier-free formula of the form

$$\varphi, \psi \;::\equiv\; \textbf{true} \mid \textbf{false} \mid \mathtt{x}_i - \mathtt{x}_j \sim z \mid \mathtt{x}_i \sim z \mid \neg\varphi \mid \varphi \wedge \psi \mid \varphi \vee \psi,$$

where "$\sim$" is a comparison operator in $\{=, <, \leq, >, \geq\}$ and $z \in \mathbb{Z}$. A clock valuation μ satisfies a constraint φ, written $\mu \models \varphi$, if interpreting each clock $\mathtt{x}_i$ by $\mu(\mathtt{x}_i)$ makes φ a tautology. An k, m*-region* is a non-empty set of valuations $[\![\varphi]\!]$ satisfied by a constraint φ with k clocks and absolute value of maximal constant bounded by m, which is minimal w.r.t. set inclusion. For instance, the clock constraint $1 < \mathtt{x}_1 < 2 \wedge 4 < \mathtt{x}_2 < 5 \wedge \mathtt{x}_2 - \mathtt{x}_1 < 3$ defines a $2, 5$-region consisting of an open triangle with nodes $(1, 4)$, $(2, 4)$ and $(2, 5)$.

Timed automata. A (nondeterministic) *timed automaton* is a tuple $A = (\Sigma, \mathtt{L}, \mathtt{X}, \mathtt{I}, \mathtt{F}, \Delta)$, where Σ is a finite input alphabet, $\mathtt{L}$ is a finite set of control locations, $\mathtt{X}$ is a finite set of clocks, $\mathtt{I}, \mathtt{F} \subseteq \mathtt{L}$ are the subsets of initial, resp., final, control locations, and Δ is a finite set of transition rules of the form

$$(p, a, \varphi, \mathtt{Y}, q) \tag{2}$$

with $p, q \in \mathtt{L}$ control locations, $a \in \Sigma$, φ a clock constraint to be tested, and $\mathtt{Y} \subseteq \mathtt{X}$ the set of clocks to be reset. We write NTA for the class of all nondeterministic timed automata, NTA_k when the number k of clocks is fixed, $\text{NTA}_{\_,m}$ when the bound m on constants is fixed, and $\text{NTA}_{k,m}$ when both k and m are fixed.

An $\text{NTA}_{\_,m}$ A is *always resetting* if every transition rule as in (2) resets some clock $\mathtt{Y} \neq \emptyset$, and *greedily resetting* if, for every clock $\mathtt{x}$, whenever φ implies that $\mathtt{x}$ belongs to $\{0, \ldots, m\} \cup (m, \infty)$, then $\mathtt{x} \in \mathtt{Y}$.

[1] Equivalently, nonnegative rationals may be considered in place of reals.

Reset-point semantics. A *configuration* of an NTA A is a tuple (p, μ, t_0) consisting of a control location $p \in \mathtt{L}$, a reset-point assignment $\mu \in \mathbb{R}_{\geq 0}^{\mathtt{X}}$, and a "now" timestamp $t_0 \in \mathbb{R}_{\geq 0}$ satisfying $\mu(\mathtt{x}) \leq t_0$ for all clocks $\mathtt{x} \in \mathtt{X}$. Intuitively, t_0 is the last timestamp seen in the input and, for every clock $\mathtt{x}$, $\mu(\mathtt{x})$ stores the timestamp of the last reset of $\mathtt{x}$. A configuration is *initial* if p is so, $t_0 = 0$, and $\mu(\mathtt{x}) = 0$ for all clocks $\mathtt{x}$, and it is *final* if p is so (without any further restriction on μ or t_0). For a set of clocks $\mathtt{Y} \subseteq \mathtt{X}$ and a timestamp $u \in \mathbb{R}_{\geq 0}$, let $\mu[\mathtt{Y} \mapsto u]$ be the assignment which is u on $\mathtt{Y}$ and agrees with μ on $\mathtt{X} \setminus \mathtt{Y}$. An assignment μ together with t_0 induces a clock valuation $t_0 - \mu$ defined as $(t_0 - \mu)(\mathtt{x}) = t_0 - \mu(\mathtt{x})$ for all clocks $\mathtt{x} \in \mathtt{X}$. Clock assignments and valuations have the same type $\mathbb{R}_{\geq 0}^{\mathtt{X}}$, however we find it technically convenient to store assignments in configurations and use the derived valuations to interpret the clock constraints. Such reset-point semantics based on reset-point assignments has already appeared in the literature on timed automata [25] and it is the foundation of the related model of timed-register automata [12].

Every transition rule (2) induces a *transition* between configurations $(p, \mu, t_0) \xrightarrow{a,t} (q, \nu, t)$ labelled by $(a, t) \in \Sigma \times \mathbb{R}_{\geq 0}$ whenever $t \geq t_0$, $t - \mu \models \varphi$, and $\nu = \mu[\mathtt{Y} \mapsto t]$. The *timed transition system* induced by A is $(\llbracket A \rrbracket, \rightarrow, F)$, where $\llbracket A \rrbracket$ is the set of configurations, $\rightarrow \subseteq \llbracket A \rrbracket \times \Sigma \times \mathbb{R}_{\geq 0} \times \llbracket A \rrbracket$ is as defined above, and $F \subseteq \llbracket A \rrbracket$ is the set of final configurations. Since there is no danger of confusion, we use $\llbracket A \rrbracket$ to denote either the timed transition system above, or its domain. A *run* of A *over* a timed word w as in (1) *starting* in configuration (p, μ, t_0) and *ending* in configuration (q, ν, t_n) is a path ρ in $\llbracket A \rrbracket$ of the form $\rho = (p, \mu, t_0) \xrightarrow{a_1,t_1} \ldots \xrightarrow{a_n,t_n} (q, \nu, t_n)$. The run ρ is accepting if its last configuration satisfies $(q, \nu, t_n) \in F$. The language *recognised* by configuration (p, μ, t_0) is defined as:

$$L_{\llbracket A \rrbracket}(p, \mu, t_0) = \{w \in \mathbb{T}(\Sigma) \mid \llbracket A \rrbracket \text{ has an accepting run over } w \text{ starting in } (p, \mu, t_0)\}.$$

Clearly $L_{\llbracket A \rrbracket}(p, \mu, t_0) \subseteq \mathbb{T}_{\geq t_0}(\Sigma)$. We write $L_A(c)$ instead of $L_{\llbracket A \rrbracket}(c)$. The language recognised by the automaton A is $L(A) = \bigcup_{c \text{ initial}} L_A(c)$. A configuration is *reachable* if it is the ending configuration in a run starting in an initial configuration. In an always resetting $\text{NTA}_{\_,m}$, every reachable configuration (p, μ, t_0) satisfies $t_0 \in \mu(\mathtt{X})$, and in a greedily resetting one, 1) (p, μ, t_0) has *m-bounded span*, in the sense that $\mu(\mathtt{X}) \subseteq (t_0 - m, t_0]$, and moreover 2) any two clocks $\mathtt{x}, \mathtt{y}$ with integer difference $\mu(\mathtt{x}) - \mu(\mathtt{y}) \in \mathbb{Z}$ are actually equal $\mu(\mathtt{x}) = \mu(\mathtt{y})$. Condition 2) follows from the fact that if $\mathtt{x}, \mathtt{y}$ have integer difference and $\mathtt{y}$ was reset last, then $\mathtt{x}$ was itself an integer when this happened, and in fact they were both reset together in a greedily resetting automaton.

Deterministic timed automata. A timed automaton A is *deterministic* if it has exactly one initial location and, for every two rules $(p, a, \varphi, \mathtt{Y}, q), (p, a', \varphi', \mathtt{Y}', q') \in \Delta$, if $a = a'$ and $\llbracket \varphi \wedge \varphi' \rrbracket \neq \emptyset$ then $\mathtt{Y} = \mathtt{Y}'$ and $q = q'$. Hence A has at most one run over every timed word w. A DTA can be easily transformed to a *total* one, where for every location $p \in \mathtt{L}$ and $a \in \Sigma$, the sets defined by clock constraints $\{\llbracket \varphi \rrbracket \mid \exists \mathtt{Y}, q \cdot (p, a, \varphi, \mathtt{Y}, q) \in \Delta\}$ are a partition of $\mathbb{R}_{\geq 0}^{\mathtt{X}}$. Thus, a total DTA has exactly one run over every timed word w. We write DTA for the class of deterministic timed automata, and DTA_k, $\text{DTA}_{\_,m}$, and $\text{DTA}_{k,m}$ for the respective subclasses thereof. A timed language is called NTA language, DTA language, etc., if it is recognised by a timed automaton of the respective type.

▶ **Example 2.1.** Let $\Sigma = \{a\}$ be a unary alphabet. As an example of a timed language L recognised by a NTA_1, but not by any DTA, consider the set of non-negative timed words of the form $(a, t_1) \cdots (a, t_n)$ where $t_n - t_i = 1$ for some $1 \leq i < n$. The language L is recognised by the NTA_1 $A = (\Sigma, \mathtt{L}, \mathtt{X}, \mathtt{I}, \mathtt{F}, \Delta)$ with a single clock $\mathtt{X} = \{\mathtt{x}\}$ and three locations $\mathtt{L} = \{p, q, r\}$, of which $\mathtt{I} = \{p\}$ is initial and $\mathtt{F} = \{r\}$ is final, and transition rules

$$(p, a, \mathbf{true}, \emptyset, p) \qquad (p, a, \mathbf{true}, \{\mathtt{x}\}, q) \qquad (q, a, \mathtt{x} < 1, \emptyset, q) \qquad (q, a, \mathtt{x} = 1, \emptyset, r)\,.$$

Intuitively, in p the automaton waits until it guesses that the next input will be (a, t_i), at which point it moves to q by resetting the clock (and subsequently reading a). From q, the automaton can accept by going to r only if exactly one time unit elapsed since (a, t_i) was read. The language L is not recognised by any DTA since, intuitively, any deterministic acceptor needs to store unboundedly many timestamps t_i's.

Deterministic membership problems. Let $\mathcal{X}$ be a subclass of NTA. We are interested in the following decision problem.

$\mathcal{X}$ MEMBERSHIP PROBLEM.
Input: A timed automaton $A \in$ NTA.
Output: Does there exist a $B \in \mathcal{X}$ s.t. $L(A) = L(B)$?

In the rest of the paper, we study the decidability status of the $\mathcal{X}$ membership problem where $\mathcal{X}$ ranges over DTA, DTA_k (for every fixed number of clocks k), $\text{DTA}_{\_,m}$ (for every maximal constant m), and $\text{DTA}_{k,m}$ (when both clocks k and maximal constant m are fixed). Example 2.1 shows that there are NTA languages which cannot be accepted by any DTA. Moreover, there is no computable bound for the number of clocks k which suffice to recognise a NTA_1 language by a DTA_k (when such a number exists), which follows from the following three observations: 1) the DTA membership problem is undecidable for NTA_1 (Theorem 1.2), 2) the problem of deciding equivalence of a given NTA_1 to a given DTA is decidable [38], and 3) if a $\text{NTA}_{1,m}$ is equivalent to some DTA_k then it is in fact equivalent to some $\text{DTA}_{k,m}$ with computably many control locations (by Lemma 4.1).

3 Timed automorphisms and invariance

A fundamental tool in this paper is invariance properties of timed languages recognised by NTA with respect to permutations of $\mathbb{R}$ preserving integer differences. In this section we establish these properties. A *timed automorphism* is a monotone bijection $\pi : \mathbb{R} \to \mathbb{R}$ s.t. for every $x \in \mathbb{R}$, $\pi(x+1) = \pi(x) + 1$. For instance, if $\pi(3.4) = 4.5$, then necessarily $\pi(5.4) = 6.5$ and $\pi(-3.6) = -2.5$. Timed automorphisms π are extended point-wise to timed words $\pi((a_1, t_1) \dots (a_n, t_n)) = (a_0, \pi(t_1)) \dots (a_n, \pi(t_n))$, configurations $\pi(p, \mu, t_0) = (p, \pi \circ \mu, \pi(t_0))$, transitions $\pi(c \xrightarrow{a,t} c') = \pi(c) \xrightarrow{a,\pi(t)} \pi(c')$, and sets X thereof $\pi(X) = \{\pi(x) \mid x \in X\}$.

▶ Remark 3.1. A timed automorphism π can in general take a nonnegative real $t \geq 0$ to a negative one. Whenever we write $\pi(x)$, we always implicitly assume that π is defined on x.

Let $S \subseteq \mathbb{R}_{\geq 0}$. An *$S$-timed automorphism* is a timed automorphism s.t. $\pi(t) = t$ for all $t \in S$. Let Π_S denote the set of all S-timed automorphisms, and let $\Pi = \Pi_\emptyset$. A set X is *S-invariant* if $\pi(X) = X$ for every $\pi \in \Pi_S$; equivalently, for every $\pi \in \Pi_S$, $x \in X$ if, and only if $\pi(x) \in X$. A set X is *invariant* if it is S-invariant with $S = \emptyset$. The following three facts express some basic invariance properties.

▶ **Fact 3.2.** *The timed transition system $[\![A]\!]$ is invariant.*

By unrolling the definition of invariance in the previous fact, we obtain that the set of configurations is invariant, the set of transitions $\to$ is invariant, and that the set of final configurations F is invariant.

▶ **Fact 3.3** (Invariance of the language semantics). *The function $c \mapsto L_A(c)$ from $[\![A]\!]$ to languages is invariant, i.e., for all timed permutations π, $L_A(\pi(c)) = \pi(L_A(c))$.*

▶ **Fact 3.4** (Invariance of the language of a configuration). *The language $L_A(p, \mu, t_0)$ is $(\mu(\mathtt{X}) \cup \{t_0\})$-invariant. Moreover, if A is always resetting, then $L_A(p, \mu, t_0)$ is $\mu(\mathtt{X})$-invariant.*

Since timed automorphisms preserve integer differences, only the fractional parts of elements of $S \subseteq \mathbb{R}_{\geq 0}$ matter for S-invariance, and hence it makes sense to restrict to subsets of the half-open interval $[0, 1)$. Let $\mathsf{fract}(S) = \{\mathsf{fract}(x) \mid x \in S\} \subseteq [0, 1)$ stand for the set of fractional parts of elements of S. The following lemma shows that, modulo the irrelevant integer parts, there is always the least set S witnessing S-invariance.

▶ **Lemma 3.5.** *For finite subsets $S, S' \subseteq \mathbb{R}_{\geq 0}$, if a timed language L is both S-invariant and S'-invariant, then it is also S''-invariant where $S'' = \mathsf{fract}(S) \cap \mathsf{fract}(S')$.*

The S-*orbit* of an element $x \in X$ (which can be an arbitrary object on which the action of timed automorphisms is defined) is the set $\textsc{orbit}_S(x) = \{\pi(x) \in X \mid \pi \in \Pi_S\}$ of all elements $\pi(x)$ which can be obtained by applying some S-automorphism to x. The *orbit* of x is just its S-orbit with $S = \emptyset$, written $\textsc{orbit}(x)$. Clearly x and x' have the same S-orbit if, and only if, $\pi(x) = x'$ for some $\pi \in \Pi_S$. For greedily resetting NTA, orbits of single configurations are in bijective correspondence with bounded regions.

▶ **Fact 3.6.** *Assume A is a greedily resetting $\text{NTA}_{k,m}$. Two reachable configurations (p, μ, t_0) and (p, μ', t'_0) of A with the same control location p have the same orbit if, and only if, the corresponding clock valuations $t_0 - \mu$ and $t'_0 - \mu'$ belong to the same k, m-region.*

The *S-closure* of a set Y, written $\Pi_S(Y) = \bigcup_{x \in Y} \textsc{orbit}_S(x)$, is the union of the S-orbits of all its elements. The following fact characterises invariance in term of closures.

▶ **Fact 3.7.** *A set Y is S-invariant if, and only if, $\Pi_S(Y) = Y$.*

Proof. Only if direction follows by the definition of S-invariance. For the converse direction observe that $\Pi_S(X) = X$ implies $\pi(X) \subseteq X$ for every $\pi \in \Pi_S$. The opposite inclusion follows by closure of S-timed automorphisms under inverse: $\pi^{-1}(X) \subseteq X$, hence $X \subseteq \pi(X)$. ◀

4 Decidability of DTA_k and $\text{DTA}_{k,m}$ membership for NTA_1

In this section we prove Theorem 1.1 thus establishing decidability of the DTA_k and $\text{DTA}_{k,m}$ membership problems for NTA_1. Both results are shown using the following key characterisation of DTA_k languages as a subclass of NTA_1 languages. In particular, this characterisation provides a small bound on the number of control locations of a DTA_k equivalent to a given NTA_1 (if any exists).

crucial lemma

▶ **Lemma 4.1.** *Let A be a $\text{NTA}_{1,m}$ with n control locations, and let $k \in \mathbb{N}$. The following conditions are equivalent:*

1. *$L(A) = L(B)$ for some always resetting DTA_k B.*
2. *For every timed word w, there is $S \subseteq \mathbb{R}_{\geq 0}$ of size at most k s.t. the last timestamp of w is in S and $w^{-1}L(A)$ is S-invariant.*
3. *$L(A) = L(B)$ for some always resetting $\text{DTA}_{k,m}$ B with at most $f(k, m, n) = \mathsf{Reg}(k, m) \cdot 2^{n(2km+1)}$ control locations ($\mathsf{Reg}(k, m)$ stands for the number of k, m-regions).*

The proof of Theorem 1.1 builds on Lemma 4.1 and on the following fact:

▶ **Lemma 4.2.** *The* DTA_k *and* $\text{DTA}_{k,m}$ *membership problems are both decidable for* DTA *languages.*

Proof. We reduce to a deterministic separability problem. Recall that a language S *separates* two languages L, M if $L \subseteq S$ and $S \cap M = \emptyset$. It has recently been shown that the DTA_k and $\text{DTA}_{k,m}$ separability problems are decidable for NTA [19, Theorem 1.1], and thus, in particular, for DTA. To solve the membership problem, given a DTA A, the procedure computes a DTA A' recognising the complement of $L(A)$ and checks whether A and A' are DTA_k separable (resp., $\text{DTA}_{k,m}$ separable) by using the result above. It is a simple set-theoretic observation that $L(A)$ is a DTA_k language if, and only if, the languages $L(A)$ and $L(A')$ are separated by some DTA_k language, and likewise for $\text{DTA}_{k,m}$ languages. ◀

Proof of Theorem 1.1. We solve both problems in essentially the same way. Given a $\text{NTA}_{1,m}$ A, the decision procedure enumerates all always resetting $\text{DTA}_{k+1,m}$ B with at most $f(k, m, n)$ locations and checks whether $L(A) = L(B)$ (which is decidable by [38]). If no such DTA_{k+1} B is found, the $L(A)$ is not an always resetting DTA_{k+1} language, due to Lemma 4.1, and hence forcedly is not a DTA_k language either; the procedure therefore answers negatively. Otherwise, in case when such a DTA_{k+1} B is found, then DTA_k membership (resp. $\text{DTA}_{k,m}$ membership) test is performed on B, decidable due to Lemma 4.2. ◀

▶ Remark 4.3 (Complexity). The decision procedure for NTA_1 invokes the HYPERACKERMANN subroutine of [38] to check equivalence between a NTA_1 and a candidate DTA. This is in a sense unavoidable, since we show in Lemma 5.5 that the DTA_k and $\text{DTA}_{k,m}$ membership problems are HYPERACKERMANN-hard for NTA_1.

In the rest of this section we present the proof of Lemma 4.1. Let us fix a $\text{NTA}_{1,m}$ $A = (\Sigma, \mathtt{L}, \{\mathtt{x}\}, \mathtt{I}, \mathtt{F}, \Delta)$, where m is the greatest constant used in clock constraints in A, and $k \in \mathbb{N}$. We assume w.l.o.g. that A is greedily resetting: This is achieved by resetting the clock as soon as upon reading an input symbol its value becomes greater than m or is an integer $\leq m$; we can record in the control location the actual integral value if it is $\leq m$, or a special flag otherwise. Consequently, after every discrete transition the value of the clock is at most m, and if it is an integer then it equals 0.

The implication $3 \Longrightarrow 1$ follows by definition. For the implication $1 \Longrightarrow 2$ suppose, by assumption, $L(A) = L(B)$ for a total always resetting DTA_k B. Every left quotient $w^{-1}L(A)$ equals $L_B(c)$ for some configuration c, hence Point 2 follows by Fact 3.4. Here we use the fact that B is always resetting in order to apply the second part of Fact 3.4; without the assumption, we would only have S-invariance for sets S of size at most $k + 1$.

It thus remains to prove the implication $2 \Longrightarrow 3$, which will be the content of the rest of the section. Assuming Point 2, we are going to define an always resetting $\text{DTA}_{k,m}$ B' with clocks $\mathtt{X} = \{\mathtt{x}_1, \ldots, \mathtt{x}_k\}$ and with at most $f(k, m, n)$ locations such that $L(B') = L(A)$. We start from the timed transition system $\mathcal{X}$ obtained by the finite powerset construction underlying the determinisation of A, and then transform this transition system gradually, while preserving its language, until it finally becomes isomorphic to the reachable part of $[\![B']\!]$ for some $\text{DTA}_{k,m}$ B'. As the last step we extract from this deterministic timed transition system a syntactic definition of B' and prove equality of their languages. This is achievable due to the invariance properties witnessed by the transition systems in the course of the transformation.

Macro-configurations. Configurations of the NTA_1 A are of the form $c = (p, u, t_0)$ where $u, t_0 \in \mathbb{R}_{\geq 0}$ and $u \leq t_0$. A *macro-configuration* is a (not necessarily finite) set X of configurations (p, u, t_0) of A which share the same value of the current timestamp t_0, which we denote as $\textsc{now}(X) = t_0$. We use the notation $L_A(X) := \bigcup_{c \in X} L_A(c)$. Let $\textsc{succ}_{a,t}(X) := \left\{c' \in [\![A]\!] \;\middle|\; c \xrightarrow{a,t} c' \text{ for some } c \in X\right\}$ be the set of successors of configurations in X. We define a deterministic timed transition system $\mathcal{X}$ consisting of the macro-configurations reachable in the course of determinisation of A. Let $\mathcal{X}$ be the smallest set of macro-configurations and transitions such that

- $\mathcal{X}$ contains the initial macro-configuration: $X_0 = \{(p, 0, 0) \mid p \in \mathtt{I}\} \in \mathcal{X}$;
- $\mathcal{X}$ is closed under successor: for every $X \in \mathcal{X}$ and $(a, t) \in \Sigma \times \mathbb{R}_{\geq 0}$, there is a transition $X \xrightarrow{a,t} \textsc{succ}_{a,t}(X)$ in $\mathcal{X}$.

Due to the fact that $[\![A]\!]$ is finitely branching, i.e. $\textsc{succ}_{a,t}(\{c\})$ is finite for every fixed (a, t), all macro-configurations $X \in \mathcal{X}$ are finite. Let the final configurations of $\mathcal{X}$ be $F_{\mathcal{X}} = \{X \in \mathcal{X} \mid X \cap F \neq \emptyset\}$.

▷ Claim 4.4. $L_A(X) = L_{\mathcal{X}}(X)$ for every $X \in \mathcal{X}$. In particular $L(A) = L_{\mathcal{X}}(X_0)$.

For a macro-configuration X we write $\textsc{Val}(X) := \{u \mid (p, u, \textsc{now}(X)) \in X\} \cup \{\textsc{now}(X)\}$ to denote the reals appearing in X. Since A is greedily resetting, every macro-configuration $X \in \mathcal{X}$ satisfies $\textsc{Val}(X) \subseteq (\textsc{now}(X) - m, \textsc{now}(X)]$. Whenever a macro-configuration X satisfies this condition we say that *the span of X is bounded by m*.

Pre-states. By assumption (Point 2), $L_A(X)$ is S-invariant for some S of size at most k, but the macro-configuration X itself needs not be S-invariant in general. Indeed, a finite macro-configuration $X \in \mathcal{X}$ is S-invariant if, and only if, $\mathsf{fract}(\textsc{Val}(X)) \subseteq \mathsf{fract}(S)$, which is impossible in general when X is arbitrarily large, its span is bounded (by m), and size of S is bounded (by k). Intuitively, in order to assure S-invariance we will replace X by its S-closure $\Pi_S(X)$ (recall Fact 3.7). does S-closure preserve the language here?

A set $S \subseteq \mathbb{R}_{\geq 0}$ is *fraction-independent* if it contains no two reals with the same fractional part. A *pre-state* is a pair $Y = (X, S)$, where X is an S-invariant macro-state, and S is a finite fraction-independent subset of $\textsc{Val}(X)$ that contains $\textsc{now}(X)$. The intuitive rationale behind assuming the S-invariance of X is that it implies, together with the bounded span of X and bounded size of S, that there are only finitely many pre-states, up to timed automorphism. We define the deterministic timed transition system $\mathcal{Y}$ as the smallest set of pre-states and transitions between them such that:

- $\mathcal{Y}$ contains the initial pre-state: $Y_0 = (X_0, \{0\}) \in \mathcal{Y}$;
- $\mathcal{Y}$ is closed under the closure of successor: for every $(X, S) \in \mathcal{Y}$ and $(a, t) \in \Sigma \times \mathbb{R}_{\geq 0}$, there is a transition $(X, S) \xrightarrow{a,t} (X', S')$, where S' is the least, with respect to set inclusion, subset of $S \cup \{t\}$ containing t such that the language $L' = (a, t)^{-1} L_A(X) = L_A(\textsc{succ}_{a,t}(X))$ is S'-invariant, and $X' = \Pi_{S'}(\textsc{succ}_{a,t}(X))$.

▶ **Example 4.5.** Suppose $k = 3$, $m = 2$, $\textsc{succ}_{a,t}(X) = \{(p, 3.7, 5), (q, 3.9, 5), (r, 4.2, 5)\}$ and $S' = \{3.7, 4.2, 5\}$. Then $X' = \{(p, 3.7, 5)\} \cup \{(q, t, 5) \mid t \in (3.7, 4)\} \cup \{(r, 4.2, 5)\}$. $\textsc{now}(X') = 5$. A corresponding *state* is (X', μ'), where $\mu' = \{\mathtt{x}_1 \mapsto 3.7, \mathtt{x}_2 \mapsto 4.2, \mathtt{x}_3 \mapsto 5\}$.

Observe that the least such fraction-independent subset S' exists due to the following facts: as X is S-invariant, due to Fact 3.3 so is its language $L_A(X)$, and hence L' is necessarily $(S \cup \{t\})$-invariant; by assumption (Point 2), L' is R-invariant for some set $R \subseteq \mathbb{R}_{\geq 0}$ of size at most k containing t; let $T \subseteq \mathbb{R}_{\geq 0}$ be the least set given by Lemma 3.5, i.e., $\mathsf{fract}(T) \subseteq$

$\mathsf{fract}(S) \cap \mathsf{fract}(R)$; and finally let $S' \subseteq S$ be chosen so that $\mathsf{fract}(S') = \mathsf{fract}(T \cup \{t\})$. Due to fraction-independence of S the choice is unique, S' is fraction-independent, and $t \in S'$. Furthermore, the size of S' is at most k. By Fact 3.3, we deduce:

▷ Claim 4.6 (Invariance of $\mathcal{Y}$). For every two transitions $(X_1, S_1) \xrightarrow{a,t_1} (X_1', S_1')$ and $(X_2, S_2) \xrightarrow{a,t_2} (X_2', S_2')$ in $\mathcal{Y}$ and a timed permutation π, if $\pi(X_1) = X_2$ and $\pi(S_1) = S_2$ and $\pi(t_1) = t_2$, then we have $\pi(X_1') = X_2'$ and $\pi(S_1') = S_2'$.

Let the final configurations of $\mathcal{Y}$ be $F_{\mathcal{Y}} = \{(X, S) \in \mathcal{Y} \mid X \cap \mathtt{F} \neq \emptyset\}$. By induction on the length of timed words it is easy to show:

▷ Claim 4.7. $L_{\mathcal{X}}(X_0) = L_{\mathcal{Y}}(Y_0)$.

Due to the assumption that A is greedily resetting and due to Point 2, in every pre-state $(X, S) \in \mathcal{Y}$ the span of X is bounded by m and the size of S is bounded by k.

States. We now introduce *states*, which are designed to be in one-to-one correspondence with configurations of the forthcoming DTA_k B'. Intuitively, a state differs from a pre-state (X, S) only by allocating the values from S into k clocks, thus while a pre-state contains a set S, the corresponding state contains a clock assignment $\mu : \mathtt{X} \to \mathbb{R}_{\geq 0}$ with image $\mu(\mathtt{X}) = S$.

Let $\mathtt{X} = \{\mathtt{x}_1, \ldots, \mathtt{x}_k\}$ be a set of k clocks. A *state* is a pair $Z = (X, \mu)$, where X is a macro-configuration, $\mu : \mathtt{X} \to \text{VAL}(X)$ is a clock reset-point assignment, $\mu(\mathtt{X})$ is a fraction-independent set containing $\text{NOW}(X)$, and X is $\mu(\mathtt{X})$-invariant. Thus every state $Z = (X, \mu)$ determines uniquely a corresponding pre-state $\sigma(Z) = (X, S)$ with $S = \mu(\mathtt{X})$. We define the deterministic timed transition system $\mathcal{Z}$ consisting of those states Z s.t. $\sigma(Z) \in \mathcal{Y}$, and of transitions determined as follows: $(X, \mu) \xrightarrow{a,t} (X', \mu')$ if the corresponding pre-state has a transition $(X, S) \xrightarrow{a,t} (X', S')$ in $\mathcal{Y}$, where $S = \mu(\mathtt{X})$, and

$$\mu'(\mathtt{x}_i) := \begin{cases} t & \text{if } \mu(\mathtt{x}_i) \notin S' \text{ or } \mu(\mathtt{x}_i) = \mu(\mathtt{x}_j) \text{ for some } j > i \\ \mu(\mathtt{x}_i) & \text{otherwise.} \end{cases} \tag{3}$$

Intuitively, the equation (3) defines a deterministic update of the clock reset-point assignment μ that amounts to resetting ($\mu'(\mathtt{x}_i) := t$) all clocks $\mathtt{x}_i$ whose value is either no longer needed (because $\mu(\mathtt{x}_i) \notin S'$), or is shared with some other clock x_j, for $j > i$ and is thus redundant. Due to this disciplined elimination of redundancy, knowing that $t \in S'$ and the size of S' is at most k, we ensure that at least one clock is reset in every step. In consequence, $\mu'(\mathtt{X}) = S'$, and the forthcoming DTA_k B' will be always resetting. Using Claim 4.6 we derive:

▷ Claim 4.8 (Invariance of $\mathcal{Z}$). For every two transitions $(X_1, \mu_1) \xrightarrow{a,t_1} (X_1', \mu_1')$ and $(X_2, \mu_2) \xrightarrow{a,t_2} (X_2', \mu_2')$ in $\mathcal{Z}$ and a timed permutation π, if $\pi(X_1) = X_2$ and $\pi \circ \mu_1 = \mu_2$ and $\pi(t_1) = t_2$, then we have $\pi(X_1') = X_2'$ and $\pi \circ \mu_1' = \mu_2'$.

Let the initial state be $Z_0 = (X_0, \mu_0)$, where $\mu_0(\mathtt{x}_i) = 0$ for all $\mathtt{x}_i \in \mathtt{X}$, and let final states be $F_{\mathcal{Z}} = \{(X, \mu) \in \mathcal{Z} \mid X \cap F \neq \emptyset\}$. By induction on the length of timed words one proves:

▷ Claim 4.9. $L_{\mathcal{Y}}(Y_0) = L_{\mathcal{Z}}(Z_0)$.

In the sequel we restrict $\mathcal{Z}$ to states reachable from Z_0. In every state $Z = (X, \mu)$ in $\mathcal{Z}$, we have $\text{NOW}(X) \in \mu(\mathtt{X})$. This will ensure the resulting DTA_k B' to be always resetting.

Orbits of states. While a state is designed to correspond to a configuration of the forthcoming DTA_k B', its orbit is designed to play the role of control location of B'. We therefore need to prove that the set of states in $\mathcal{Z}$ is orbit-finite, i.e., the set of orbits $\{\mathrm{ORBIT}(Z) \mid Z \in \mathcal{Z}\}$ is finite and its size is bounded by $f(k, m, n)$. We start by deducing an analogue of Fact 3.6:

▷ Claim 4.10. For two states $Z = (X, \mu)$ and $Z' = (X', \mu')$ in $\mathcal{Z}$, their clock assignments are in the same orbit, i.e., $\pi \circ \mu = \mu'$ for some $\pi \in \Pi$, if, and only if, the corresponding clock valuations $\mathrm{NOW}(X) - \mu$ and $\mathrm{NOW}(X') - \mu'$ belong to the same k, m-region.

(In passing note that, since in every state (X, μ) in $\mathcal{Z}$ the span of X is bounded by m, only bounded k, m-regions can appear in the last claim. Moreover, in each of k, m-regions one of clocks equals 0.) The action of timed automorphisms on macro-configurations and clock assignments is extended to states as $\pi(X, \mu) = (\pi(X), \pi \circ \mu)$. Recall that the orbit of a state Z is defined as $\mathrm{ORBIT}(Z) = \{\pi(Z) \mid \pi \in \Pi\}$.

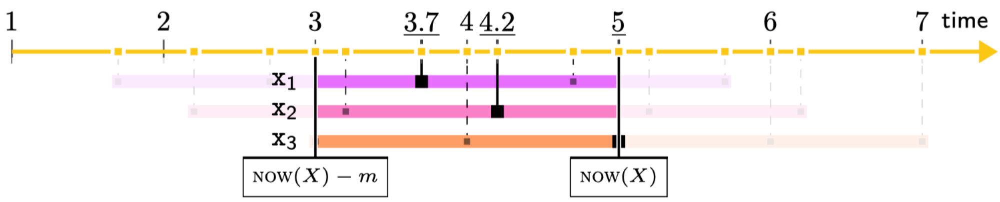
▷ Claim 4.11. The number of orbits of states in $\mathcal{Z}$ is bounded by $f(k, m, n)$.

Proof. We finitely represent a state $Z = (X, \mu)$, relying on the following general fact.

▶ **Fact 4.12.** *For every $u \in \mathbb{R}_{\geq 0}$ and $S \subseteq \mathbb{R}_{\geq 0}$, the S-orbit*[2] $\mathrm{ORBIT}_S(u)$ *is either the singleton $\{u\}$ (when $u \in S$) or an open interval with ends-points of the form $t + z$ where $t \in S$ and $z \in \mathbb{Z}$ (when $u \notin S$).*

We apply the fact above to $S = \mu(\mathtt{X})$. In our case the span of X is bounded by m, and thus the same holds for $\mu(\mathtt{X})$. Consequently, the integer z in the fact above always belongs to $\{-m, -m+1, \ldots, m\}$. In turn, X splits into disjoint $\mu(\mathtt{X})$-orbits $\mathrm{ORBIT}_{\mu(\mathtt{X})}(u)$ consisting of open intervals separated by endpoints of the form $t + z$ where $t \in \mu(\mathtt{X})$ and $z \in \{-m, -m+1, \ldots, m\}$.

▶ **Example 4.13.** Continuing Example 4.5, the endpoints are $\{3, 3.2, 3.7, 4, 4.2, 4.7, 5\}$, as shown in the illustration:



Recall that $\mu(\mathtt{X})$ is fraction-independent. Let $e_1 < e_2 < \cdots < e_{l+1}$ be all the endpoints of open-interval orbits ($l \leq km$), and let $o_1, o_2, o_3, \ldots := \{e_1\}, (e_1, e_2), \{e_2\}, \ldots$ be the consecutive S-orbits $\mathrm{ORBIT}_{\mu(\mathtt{X})}(u)$ of elements $u \in \mu(\mathtt{X})$. The number thereof is $2l + 1 \leq 2km + 1$. The finite representation of $Z = (X, \mu)$ consists of the pair (O, μ), where

$$O = \{(o_1, P_1), \ldots, (o_{2l+1}, P_{2l+1})\} \tag{4}$$

assigns to each orbit o_i the set of locations $P_i = \{p \mid (p, u, t_0) \in X \text{ for some } u \in o_i\} \subseteq \mathtt{L}$, (which is the same as $P_i = \{p \mid (p, u, t_0) \in X \text{ for all } u \in o_i\}$ since X is $\mu(\mathtt{X})$-invariant, and hence $\mu(\mathtt{X})$-closed). Thus a state $Z = (X, \mu)$ is uniquely determined by the sequence O as in (4) and the clock assignment μ.

[2] The orbits of states Z should not be confused with S-orbits of individual reals $u \in \mathbb{R}_{\geq 0}$.

We claim that the set of all the finite representations (O, μ), as defined above, is orbit-finite. Indeed, the orbit of (O, μ) is determined by the orbit of μ and the sequence

$$P_1,\ P_2,\ \ldots,\ P_{2km+1} \tag{5}$$

induced by the assignment O as in (4). Therefore, the number of orbits is bounded by the number of orbits of μ (which is bounded, due to Claim 4.10, by $\mathsf{Reg}(k, m)$) times the number of different sequences of the form (5) (which is bounded by $(2^n)^{2km+1}$). This yields the required bound $f(k, m, n) = \mathsf{Reg}(k, m) \cdot 2^{n(2km+1)}$. ◀

Construction of the DTA. As the last step we define a DTA_k $B' = (\Sigma, \mathtt{L}', \mathtt{X}, \{o_0\}, \mathtt{F}', \Delta')$ such that the reachable part of $[\![B']\!]$ is isomorphic to $\mathcal{Z}$. Let locations $\mathtt{L}' = \{\text{ORBIT}(Z) \mid Z \in \mathcal{Z}\}$ be orbits of states from $\mathcal{Z}$, the initial location be the orbit o_0 of Z_0, and final locations $\mathtt{F}' = \{\text{ORBIT}(Z) \mid Z \in F_{\mathcal{Z}}\}$ be orbits of final states. A transition $Z = (X, \mu) \xrightarrow{a,t} (X', \mu') = Z'$ in $\mathcal{Z}$ induces a transition rule in B'

$$(o, a, \psi, \mathtt{Y}, o') \in \Delta' \tag{6}$$

whenever $o = \text{ORBIT}(Z)$, $o' = \text{ORBIT}(Z')$, ψ is the unique k, m-region satisfying $t - \mu \in [\![\psi]\!]$, and $\mathtt{Y} = \{\mathtt{x}_i \in \mathtt{X} \mid \mu'(\mathtt{x}_i) = t\}$. The automaton B' is indeed a DTA since o, a and ψ uniquely determine $\mathtt{Y}$ and o':

▷ Claim 4.14. Suppose that two transitions $(X_1, \mu_1) \xrightarrow{a,t_1} (X_1', \mu_1')$ and $(X_2, \mu_2) \xrightarrow{a,t_2} (X_2', \mu_2')$ in $\mathcal{Z}$ induce transition rules $(o, a, \psi, \mathtt{Y}_1, o_1'), (o, a, \psi, \mathtt{Y}_2, o_2') \in \Delta'$ with the same source location o and constraint ψ, i.e,

$$t_1 - \mu_1 \in [\![\psi]\!] \qquad t_2 - \mu_2 \in [\![\psi]\!]\,. \tag{7}$$

Then the target locations are equal $o_1' = o_2'$, and the same for the reset sets $\mathtt{Y}_1 = \mathtt{Y}_2$.

Proof. We use the invariance of semantics of A and Claim 4.8. Let $o = \text{ORBIT}(X_1, \mu_1) = \text{ORBIT}(X_2, \mu_2)$. Thus there is a timed automorphism π such that

$$X_2 = \pi(X_1) \qquad \mu_2 = \pi \circ \mu_1. \tag{8}$$

It suffices to show that there is a (possibly different) timed permutation σ satisfying the following equalities:

$$t_2 = \sigma(t_1) \quad \{i \mid \mu_1'(\mathtt{x}_i) = t_1\} = \{i \mid \mu_2'(\mathtt{x}_i) = t_2\} \quad \mu_2' = \sigma \circ \mu_1' \quad X_2' = \sigma(X_1'). \tag{9}$$

We now rely the fact that both $t_{01} = \text{NOW}(X_1) \in \mu_1(\mathtt{X})$ and $t_{02} = \text{NOW}(X_2) \in \mu_2(\mathtt{X})$ are assigned to (the same) clock due to the second equality in (8): $t_{01} = \mu_1(\mathtt{x}_i)$ and $t_{02} = \mu_2(\mathtt{x}_i)$. We focus on the case when $t_1 - t_{01} \leq m$ (the other case is similar but easier as all clock are reset due to greedy resetting), which implies $t_2 - t_{02} \leq m$ due to (7). In this case we may assume w.l.o.g., due to (7) and the equalities (8), that π is chosen so that $\pi(t_1) = t_2$. We thus take $\sigma = \pi$ for proving the equalities (9). Being done with the first equality, we observe that the last two equalities in (9) hold due to the invariance of $\mathcal{Z}$ (cf. Claim 4.8). The remaining second equality in (9) is a consequence of the third one. ◀

▷ Claim 4.15. Let $Z = (X, \mu)$ and $Z' = (X', \mu)$ be two states in $\mathcal{Z}$ with the same clock assignment. If $\pi(X) = X'$ and $\pi \circ \mu = \mu$ for some timed automorphism π then $X = X'$.

▷ Claim 4.16. $\mathcal{Z}$ is isomorphic to the reachable part of $[\![B']\!]$.

Proof. For a state $Z = (X, \mu)$, let $c(Z) = (o, \mu, t)$, where $o = \textsc{orbit}(Z)$ and $t = \textsc{now}(X)$. By Claim 4.15, the mapping $c(\_)$ is a bijection between $\mathcal{Z}$ and its image $c(\mathcal{Z}) \subseteq [\![B']\!]$. By (6), $\mathcal{Z}$ is isomorphic to a subsystem of the reachable part of $[\![B']\!]$. The converse inclusion follows by the observation that $\mathcal{Z}$ is total: for every $(a_1, t_1) \ldots (a_n, t_n) \in \mathbb{T}(\Sigma)$, there is a sequence of transitions $(X_0, \mu_0) \xrightarrow{a_1,t_1} \cdots \xrightarrow{a_n,t_n}$ in $\mathcal{Z}$. ◀

Claims 4.4, 4.7, 4.9, and 4.16 prove $L(A) = L(B')$.

5 Undecidability and hardness

In this section we complete the decidability status of the deterministic membership problem by providing matching undecidability and hardness results. In Section 5.1 we prove undecidability of the $\textsc{dta}_m$ embership problem for $\textsc{nta}_1$ (c.f. Theorem 1.2) and in Section 5.2 we prove HyperAckermann-hardness of the $\textsc{dta}_k$ membership problem for $\textsc{nta}_1$ (c.f. Theorem 1.3).

5.1 Undecidability of DTA and $\textsc{dta}_{\_,m}$ membership for $\textsc{nta}_1$

It has been shown in [23, Theorem 1] that it is undecidable whether a $\textsc{nta}_k$ timed language can be recognised by some DTA, for any fixed $k \geq 2$. This was obtained by a reduction from the $\textsc{nta}_k$ universality problem, which is undecidable for any fixed $k \geq 2$. While the universality problem becomes decidable for $k = 1$, we show in this section that, as announced in Theorem 1.2, the DTA membership problem remains undecidable for $\textsc{nta}_1$.

Since the universality problem for $\textsc{nta}_1$ is decidable, we need to reduce from another (undecidable) problem. Our candidate is the finiteness problem of lossy counter machines, which is undecidable [36, Theorem 13]. A *k-counters lossy counter machine* (k-LCM) is a tuple $M = (C, Q, q_0, \Delta)$, where $C = \{c_1, \ldots, c_k\}$ is a set of k counters, Q is a finite set of control locations, $q_0 \in Q$ is the initial control location, and Δ is a finite set of instructions of the form $(p, \mathtt{op}, q)$, where $\mathtt{op}$ is one of $c\,$++, $c\,$--, and $c \stackrel{?}{=} 0$. A configuration of an LCM M is a pair (p, u), where $p \in Q$ is a control location, and $u \in \mathbb{N}^C$ is a counter valuation. For two counter valuations $u, v \in \mathbb{N}^C$, we write $u \leq v$ if $u(c) \leq v(c)$ for every counter $c \in C$. The semantics of an LCM M is given by a (potentially infinite) transition system over the configurations of M s.t. there is a transition $(p, u) \xrightarrow{\delta} (q, v)$, for $\delta = (p, \mathtt{op}, q) \in \Delta$, whenever 1) $\mathtt{op} = c\,$++ and $v \leq u[c \mapsto u(c) + 1]$, or 2) $\mathtt{op} = c\,$-- and $v \leq u[c \mapsto u(c) - 1]$, or 3) $\mathtt{op} = c \stackrel{?}{=} 0$ and $u(c) = 0$ and $v \leq u$. The *finiteness problem* (a.k.a. space boundedness) for an LCM M asks to decide whether the reachability set $\mathrm{Reach}(M) = \{(p, u) \mid (q_0, u_0) \to^* (p, u)\}$ is finite, where u_0 is the constantly 0 counter valuation.

▶ **Theorem 5.1** ([36, Theorem 13]). *The 4-LCM finiteness problem is undecidable.*

We use the following encoding of LCM runs as timed words over the alphabet $\Sigma = Q \cup \Delta \cup C$ (c.f. [34, Definition 4.6] for a similar encoding). We interpret a counter valuation $u \in \mathbb{N}^C$ as the word over Σ

$$u \;=\; \underbrace{c_1 c_1 \cdots c_1}_{u(c_1) \text{ letters}} \; \underbrace{c_2 c_2 \cdots c_2}_{u(c_2) \text{ letters}} \; \underbrace{c_3 c_3 \cdots c_3}_{u(c_3) \text{ letters}} \; \underbrace{c_4 c_4 \cdots c_4}_{u(c_4) \text{ letters}} .$$

With this interpretation, we encode an LCM run $\pi = (p_0, u_0) \xrightarrow{\delta_1} (p_1, u_1) \xrightarrow{\delta_2} \cdots \xrightarrow{\delta_n} (p_n, u_n)$ as the following timed word, called the *reversal-encoding* of π,

$$p_n \delta_n u_n \quad \cdots \quad p_1 \delta_1 u_1 \quad p_0 u_0,$$

s.t. p_n occurs at time 0, for every $1 \leq i < n$, p_i occurs exactly after one time unit since p_{i+1}, and if a "unit" of counter c_1 did not disappear due to lossiness when going from u_i to u_{i+1}, then the timestamps of the corresponding occurrences of letter c_1 in u_i and u_{i+1} are also at distance one (and similarly for the other counters). Under the encoding above, we can build a NTA_1 A recognising the complement of the set of reversal-encodings of the runs of M ([34] for more details about the construction of A). Intuitively, when reading the reversal-encoding of a run of M, the counters are allowed to spontaneously increase. Therefore, the only kind of error that A must verify is that some counter spontaneously decreases. This can be done by guessing an occurrence of letter (say) c_1 in the current configuration which does not have a corresponding occurrence in the next configuration after exactly one time unit. This check can be performed by an NTA with one clock.

▶ **Lemma 5.2.** *The set of reachable configurations $Reach(M)$ is finite if, and only if, $L(A)$ is a deterministic timed language.*

Since the timed automaton constructed in the proof uses only constant 1, the reduction works also for the $\text{DTA}_{\_,m}$ membership problem for every $m > 0$:

▶ **Corollary 5.3.** *For every fixed $m > 0$, the $\text{DTA}_{\_,m}$ membership problem for NTA_1 languages is undecidable.*

This result is the best possible in terms of the parameter m since the problem becomes decidable for $m = 0$. In fact, the class of $\text{DTA}_{k,0}$ languages coincides with the class of $\text{DTA}_{1,0}$ languages (one clock is sufficient; c.f. [38, Lemma 19]), and thus $\text{DTA}_{\_,0}$ membership reduces to $\text{DTA}_{1,0}$ membership, which is decidable for NTA_1 by Theorem 1.1.

▶ Remark 5.4. We observe that the reduction above uses a large alphabet Σ whose size depends on the input LCM M. In fact, an alternative encoding exists using a unary alphabet $\Sigma = \{a\}$. Let the input LCM M have control locations $Q = \{p_1, \ldots, p_m\}$ and instructions $\Delta = \{\delta_1, \ldots, \delta_n\}$. An LCM configuration $p_j \delta_k u$ is represented by the timed word consisting of 6 blocks $\underbrace{a \cdots a}_{j \text{ letters}} \; \underbrace{a \cdots a}_{k \text{ letters}} \; \underbrace{a \cdots a}_{u(c_1) \text{ letters}} \; \underbrace{a \cdots a}_{u(c_2) \text{ letters}} \; \underbrace{a \cdots a}_{u(c_3) \text{ letters}} \; \underbrace{a \cdots a}_{u(c_4) \text{ letters}}$ s.t. in each block the last a is at timed distance exactly one from the last a of the previous block. A unit of counter c_1 now repeats at distance 6 in the next configuration (instead of 1). This shows that the DTA membership problem is undecidable for NTA_1 using maximal constant $m = 6$ over a unary alphabet.

5.2 Undecidability and hardness for DTA_k and $\text{DTA}_{k,m}$ membership

All the lower bounds in this section are obtained by a reduction from the universality problem for the respective language classes (does a given language $L \subseteq \mathbb{T}(\Sigma)$ satisfy $L = \mathbb{T}(\Sigma)$?). The reduction is a suitable adaptation, generalization, and simplification of [23, Theorem 1] showing undecidability of DTA membership for NTA languages.

A timed language L is *timeless* if $L = L(A)$ for $A \in \text{NTA}_0$ a timed automaton with no clocks (hence timestamps appearing in input words are irrelevant for acceptance). For two languages $L \subseteq \mathbb{T}(\Sigma)$ and $M \subseteq \mathbb{T}(\Gamma)$, and a fresh alphabet symbol $\$ \notin \Sigma \cup \Gamma$, we define their *composition* $L \triangleright \{\$\} \triangleright M$ to be the following timed language over $\Sigma' = \Sigma \cup \{\$\} \cup \Gamma$:

$$L \triangleright \{\$\} \triangleright M \;=\; \{v(\$, t)(a_1, t_1 + t) \ldots (a_n, t_n + t) \in \mathbb{T}(\Sigma') \mid v \in L, (a_1, t_1) \ldots (a_n, t_n) \in M\}\,.$$

▶ **Lemma 5.5.** *Let $k, m \in \mathbb{N}$ and let $\mathcal{Y}$ be a class of timed languages that*

1. *contains all the timeless timed languages,*

2. *is closed under union and composition, and*
3. *contains some non-*DTA_k *(resp. non-*$\mathrm{DTA}_{k,m}$*) language.*

The universality problem for languages in $\mathcal{Y}$ *reduces in polynomial time to the* DTA_k *(resp.* $\mathrm{DTA}_{k,m}$*) membership problem for languages in* $\mathcal{Y}$*.*

We immediately obtain Theorem 1.3 as a corollary of Lemma 5.5, thanks to the following observations. First, the lemma is applicable by taking as $\mathcal{Y}$ the classes of languages recognised by NTA_1 since this class contains all timeless timed languages, is closed under union and composition, and is not included in DTA_k for any k nor in $\mathrm{DTA}_{k,m}$ for any k, m (c.f. the NTA_1 language from Example 2.1 which is not recognised by any DTA). Second, HyperAckermann-hardness of the universality problem for NTA_1 follows form the same lower bound for the reachability problem in lossy channel systems [17, Theorem 5.5], together with the reduction from this problem to universality of NTA_1 given in [34, Theorem 4.1].

Since the universality problem is undecidable for NTA_2 [3, Theorem 5.2] and $\mathrm{NTA}_1^{\varepsilon}$ (NTA_1 with epsilon steps) [34, Theorem 5.3], using the same reasoning we can apply Lemma 5.5 to observe that the DTA_k and $\mathrm{DTA}_{k,m}$ membership problems are undecidable for NTA_2 and $\mathrm{NTA}_1^{\varepsilon}$, which refines the analysis of [23, Theorem 1].

6 Conclusions

We have shown decidability and undecidability results for several variants of the deterministic membership problem for timed automata. Regarding undecidability, we have extended the previously known results [23, 43] by proving that the DTA membership problem is undecidable already for NTA_1 (Theorem 1.2), and, over a unary input alphabet, it is undecidable for $\mathrm{NTA}_{1,m}$ with $m \geq 6$ (Remark 5.4). We leave open the question of what is the minimal m guaranteeing undecidability. Regarding decidability, we have shown that when the resources available to the deterministic automaton are fixed (either just the number of clocks k, or both clocks k and maximal constant m), then the respective deterministic membership problem is decidable (Theorem 1.1) and HyperAckermann-hard (Theorem 1.3).

Our deterministic membership algorithm is based on a characterisation of NTA_1 languages which happen to be DTA_k (Lemma 4.1), which is proved using a semantic approach leveraging on notions from the theory of sets with atoms [12]. Analogous decidability results for register automata can be obtained with similar techniques. It would be interesting to compare this approach to the syntactic determinisation method of [7].

Finally, our decidability results extend to the slightly more expressive class of always resetting NTA_2, which have intermediate expressive power strictly between NTA_1 and NTA_2.

References

1 https://siglog.org/the-2016-alonzo-church-award-for-outstanding-contributions-to-logic-and-computation/, 2016.

2 S. Akshay, Paul Gastin, and Shankara Narayanan Krishna. Analyzing Timed Systems Using Tree Automata. *Logical Methods in Computer Science*, Volume 14, Issue 2, May 2018. URL: https://lmcs.episciences.org/4489, doi:10.23638/LMCS-14(2:8)2018.

3 Rajeev Alur and David L. Dill. A theory of timed automata. *Theor. Comput. Sci.*, 126:183–235, 1994.

4 Rajeev Alur, Limor Fix, and Thomas A. Henzinger. Event-clock automata: a determinizable class of timed automata. *Theor. Comput. Sci.*, 211:253–273, January 1999.

5 Eugene Asarin and Oded Maler. As soon as possible: Time optimal control for timed automata. In *Proc. of HSCC'99*, HSCC '99, pages 19–30, London, UK, UK, 1999. Springer-Verlag. URL: http://dl.acm.org/citation.cfm?id=646879.710314.

6 Eugene Asarin, Oded Maler, Amir Pnueli, and Joseph Sifakis. Controller synthesis for timed automata. In *Proc. of the 5th IFAC Conference on System Structure and Control (SSSC'98)*, volume 31, pages 447–452, 1998. URL: http://www.sciencedirect.com/science/article/pii/S1474667017420325, doi:https://doi.org/10.1016/S1474-6670(17)42032-5.

7 Christel Baier, Nathalie Bertrand, Patricia Bouyer, and Thomas Brihaye. When are timed automata determinizable? In Susanne Albers, Alberto Marchetti-Spaccamela, Yossi Matias, Sotiris Nikoletseas, and Wolfgang Thomas, editors, *Proc of ICALP'09*, pages 43–54, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

8 Vince Bárány, Christof Löding, and Olivier Serre. Regularity problems for visibly pushdown languages. In *Proc. of STACS'06*, STACS'06, pages 420–431, Berlin, Heidelberg, 2006. Springer-Verlag. URL: http://dx.doi.org/10.1007/11672142_34, doi:10.1007/11672142_34.

9 Gerd Behrmann, Alexandre David, Kim G. Larsen, John Hakansson, Paul Petterson, Wang Yi, and Martijn Hendriks. Uppaal 4.0. In *Proceedings of the 3rd International Conference on the Quantitative Evaluation of Systems*, QEST '06, pages 125–126, Washington, DC, USA, 2006. IEEE Computer Society. doi:10.1109/QEST.2006.59.

10 Nathalie Bertrand, Amélie Stainer, Thierry Jéron, and Moez Krichen. A game approach to determinize timed automata. *Formal Methods in System Design*, 46(1):42–80, 2015. doi:10.1007/s10703-014-0220-1.

11 Mikołaj Bojańczyk, Bartek Klin, and Sławomir Lasota. Automata theory in nominal sets. *Logical Methods in Computer Science*, 10(3:4):paper 4, 2014.

12 Mikolaj Bojańczyk and Sławomir Lasota. A machine-independent characterization of timed languages. In *Proc. ICALP 2012*, pages 92–103, 2012.

13 Patricia Bouyer, Fabrice Chevalier, and Deepak D'Souza. Fault diagnosis using timed automata. In *Proc. of FOSSACS'05*, pages 219–233, Berlin, Heidelberg, 2005. Springer-Verlag. doi:10.1007/978-3-540-31982-5_14.

14 Thomas Brihaye, Thomas A. Henzinger, Vinayak S. Prabhu, and Jean-François Raskin. Minimum-time reachability in timed games. In Lars Arge, Christian Cachin, Tomasz Jurdziński, and Andrzej Tarlecki, editors, *In Proc. of ICALP'07*, pages 825–837, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

15 Michaël Cadilhac, Alain Finkel, and Pierre McKenzie. On the expressiveness of Parikh automata and related models. In Rudolf Freund, Markus Holzer, Carlo Mereghetti, Friedrich Otto, and Beatrice Palano, editors, *Proc. of NCMA'11*, volume 282 of *books@ocg.at*, pages 103–119. Austrian Computer Society, 2011.

16 Franck Cassez, Alexandre David, Emmanuel Fleury, Kim G. Larsen, and Didier Lime. Efficient on-the-fly algorithms for the analysis of timed games. In Martín Abadi and Luca de Alfaro, editors, *Proc. of CONCUR'05*, pages 66–80, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

17 Pierre Chambart and Philippe Schnoebelen. The ordinal recursive complexity of lossy channel systems. In *Proc. of LICS'08*, pages 205–216, 2008.

18 Lorenzo Clemente, Piotr Hofman, and Patrick Totzke. Timed Basic Parallel Processes. In Wan Fokkink and Rob van Glabbeek, editors, *Proc. of CONCUR'19*, volume 140 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 15:1–15:16, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. URL: `http://drops.dagstuhl.de/opus/volltexte/2019/10917`, `doi:10.4230/LIPIcs.CONCUR.2019.15`.

19 Lorenzo Clemente, Sławomir Lasota, and Radosław Piórkowski. Timed games and deterministic separability. In *Proc. of ICALP 2020*, pages 121:1–121:16, 2020.

20 Hubert Comon and Yan Jurski. Timed automata and the theory of real numbers. In *Proc. of CONCUR'99*, pages 242–257, London, UK, UK, 1999. Springer-Verlag.

21 C. Dima. Computing reachability relations in timed automata. In *Proc. of LICS'02*, pages 177–186, 2002.

22 John Fearnley and Marcin Jurdziński. Reachability in two-clock timed automata is PSPACE-complete. *Information and Computation*, 243:26–36, 2015. URL: `http://www.sciencedirect.com/science/article/pii/S0890540114001564`, `doi:http://dx.doi.org/10.1016/j.ic.2014.12.004`.

23 Olivier Finkel. Undecidable problems about timed automata. In *Proc. of FORMATS'06*, pages 187–199, Berlin, Heidelberg, 2006. Springer-Verlag. URL: `http://dx.doi.org/10.1007/11867340_14`, `doi:10.1007/11867340_14`.

24 Martin Fränzle, Karin Quaas, Mahsa Shirmohammadi, and James Worrell. Effective definability of the reachability relation in timed automata. *Information Processing Letters*, 153:105871, 2020. URL: `http://www.sciencedirect.com/science/article/pii/S0020019019301541`, `doi:https://doi.org/10.1016/j.ipl.2019.105871`.

25 Laurent Fribourg. A closed-form evaluation for extended timed automata. Technical report, CNRS & ECOLE NORMALE SUPERIEURE DE CACHAN, 1998.

26 Paul Gastin, Sayan Mukherjee, and B. Srivathsan. Reachability in Timed Automata with Diagonal Constraints. In Sven Schewe and Lijun Zhang, editors, *Proc. of CONCUR'18*, volume 118 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 28:1–28:17, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. URL: `http://drops.dagstuhl.de/opus/volltexte/2018/9566`, `doi:10.4230/LIPIcs.CONCUR.2018.28`.

27 Paul Gastin, Sayan Mukherjee, and B. Srivathsan. Fast algorithms for handling diagonal constraints in timed automata. In Isil Dillig and Serdar Tasiran, editors, *Computer Aided Verification*, pages 41–59, Cham, 2019. Springer International Publishing.

28 Stefan Göller and Paweł Parys. Bisimulation finiteness of pushdown systems is elementary. In *Proc. of LICS'20*, pages 521–534, 2020.

29 R. Govind, Frédéric Herbreteau, B. Srivathsan, and Igor Walukiewicz. Revisiting Local Time Semantics for Networks of Timed Automata. In Wan Fokkink and Rob van Glabbeek, editors, *Proc. of CONCUR 2019*, volume 140 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 16:1–16:15, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. URL: `http://drops.dagstuhl.de/opus/volltexte/2019/10918`, `doi:10.4230/LIPIcs.CONCUR.2019.16`.

30 Frédéric Herbreteau, B. Srivathsan, and Igor Walukiewicz. Better abstractions for timed automata. *Information and Computation*, 251:67–90, 2016. URL: `http://www.sciencedirect.com/science/article/pii/S0890540116300438`, `doi:https://doi.org/10.1016/j.ic.2016.07.004`.

31 Marcin Jurdziński and Ashutosh Trivedi. Reachability-time games on timed automata. In *In Proc. of ICALP'07*, pages 838–849, Berlin, Heidelberg, 2007. Springer-Verlag. URL: `http://dl.acm.org/citation.cfm?id=2394539.2394637`.

32 Pavel Krčál and Radek Pelánek. On sampled semantics of timed systems. In Sundar Sarukkai and Sandeep Sen, editors, *In Proc. of FSTTCS'05*, volume 3821 of *LNCS*, pages 310–321. Springer, 2005. URL: `http://dx.doi.org/10.1007/11590156_25`.

33 M. Kwiatkowska, G. Norman, and D. Parker. PRISM 4.0: Verification of probabilistic real-time systems. In G. Gopalakrishnan and S. Qadeer, editors, *Proc. of CAV'11*, volume 6806 of *LNCS*, pages 585–591. Springer, 2011.

34 Slawomir Lasota and Igor Walukiewicz. Alternating timed automata. *ACM Trans. Comput. Logic*, 9(2):10:1–10:27, 2008. URL: `http://doi.acm.org/10.1145/1342991.1342994`, `doi:10.1145/1342991.1342994`.

35 Oded Maler and Amir Pnueli. On recognizable timed languages. In Igor Walukiewicz, editor, *Proc. of FOSSACS'04*, volume 2987 of *LNCS*, pages 348–362. Springer Berlin Heidelberg, 2004. URL: `http://dx.doi.org/10.1007/978-3-540-24727-2_25`, `doi:10.1007/978-3-540-24727-2_25`.

36 Richard Mayr. Undecidable problems in unreliable computations. *Theor. Comput. Sci.*, 297(1-3):337–354, March 2003. URL: `http://dx.doi.org/10.1016/S0304-3975(02)00646-1`, `doi:10.1016/S0304-3975(02)00646-1`.

37 Brian Nielsen and Arne Skou. Automated test generation from timed automata. *International Journal on Software Tools for Technology Transfer*, 5(1):59–77, Nov 2003. `doi:10.1007/s10009-002-0094-1`.

38 Joël Ouaknine and James Worrell. On the language inclusion problem for timed automata: Closing a decidability gap. In *Proc. of LICS'04*, pages 54–63, 2004. `doi:10.1109/LICS.2004.1319600`.

39 Joel Ouaknine and James Worrell. On the decidability and complexity of Metric Temporal Logic over finite words. *Logical Methods in Computer Science*, Volume 3, Issue 1, February 2007. URL: `https://lmcs.episciences.org/2230`, `doi:10.2168/LMCS-3(1:8)2007`.

40 Jeffrey Shallit. *A Second Course in Formal Languages and Automata Theory*. 2008.

41 P. Vijay Suman, Paritosh K. Pandya, Shankara Narayanan Krishna, and Lakshmi Manasa. Timed automata with integer resets: Language inclusion and expressiveness. In *Proc. of FORMATS'08*, pages 78—92, Berlin, Heidelberg, 2008. Springer-Verlag. `doi:10.1007/978-3-540-85778-5_7`.

42 Martin Tappler, Bernhard K. Aichernig, Kim Guldstrand Larsen, and Florian Lorber. Time to learn - learning timed automata from tests. In Étienne André and Mariëlle Stoelinga, editors, *Proc. of FORMATS'19*, pages 216–235, Cham, 2019. Springer International Publishing.

43 Stavros Tripakis. Folk theorems on the determinization and minimization of timed automata. *Inf. Process. Lett.*, 99(6):222–226, September 2006.

44 Leslie G. Valiant. Regularity and related problems for deterministic pushdown automata. *J. ACM*, 22(1):1–10, January 1975. URL: `http://doi.acm.org/10.1145/321864.321865`, `doi:10.1145/321864.321865`.

45 Rüdiger Valk and Guy Vidal-Naquet. Petri nets and regular languages. *Journal of Computer and System Sciences*, 23(3):299–325, 1981. URL: `http://www.sciencedirect.com/science/article/pii/0022000081900672`, `doi:http://dx.doi.org/10.1016/0022-0000(81)90067-2`.

46 Sicco Verwer, Mathijs de Weerdt, and Cees Witteveen. An algorithm for learning real-time automata. In *Proc of. the Annual Belgian-Dutch Machine Learning Conference (Benelearn'078)*, 2007.

A Proofs for Section 3

▶ **Fact 3.2.** *The timed transition system $[\![A]\!]$ is invariant.*

Proof. Suppose $c = (p, \mu, t_0) \xrightarrow{a,t} (p', \mu', t) = c'$ due to some transition rule of A whose clock constraint φ compares values of clocks $\mathtt{x}$, i.e., the differences $t - \mu(\mathtt{x})$, to integers. Since a timed automorphism π preserves integer distances, the same clock constraint is satisfied in $\pi(c) = (p, \pi\circ\mu, \pi(t_0))$, and therefore the same transition rule is applicable yielding the transition $(p, \pi\circ\mu, \pi(t_0)) \xrightarrow{a,\pi(t)} (p, \pi\circ\mu', \pi(t)) = \pi(c')$. ◀

▶ **Fact 3.4** (Invariance of the language of a configuration). *The language $L_A(p, \mu, t_0)$ is $(\mu(\mathtt{X}) \cup \{t_0\})$-invariant. Moreover, if A is always resetting, then $L_A(p, \mu, t_0)$ is $\mu(\mathtt{X})$-invariant.*

Proof. This is a direct consequence of the invariance of semantics. Indeed, for every $(\mu(\mathtt{X}) \cup \{t_0\})$-timed permutation π the configurations $c = (p, \mu, t_0)$ and $\pi(c) = (p, \pi\circ\mu, \pi(t_0))$ are equal, hence their languages $L_A(c)$ and $L_A(\pi(c))$, the latter equal to $\pi(L_A(c))$ by Fact 3.3, are equal too. Thus, $L = \pi(L)$. Finally, if A is always resetting, then $t_0 \in \mu(\mathtt{X})$, from which the second claim follows. ◀

▶ **Fact 3.3** (Invariance of the language semantics). *The function $c \mapsto L_A(c)$ from $[\![A]\!]$ to languages is invariant, i.e., for all timed permutations π, $L_A(\pi(c)) = \pi(L_A(c))$.*

Proof. Consider a timed permutation π and an accepting run of A over a timed word $w = (a_1, t_1) \ldots (a_n, t_n) \in \mathbb{T}_{\geq t_0}(\Sigma)$ starting in $c = (p, \mu, t_0)$:

$$(p, \mu, t_0) \xrightarrow{a_1, t_1} \cdots \xrightarrow{a_n, t_n} (q, \nu, t_n),$$

After a_i is read, the value of each clock is either the difference $t_i - \mu(\mathtt{x})$ for some $1 \leq i \leq n$ and clock $\mathtt{x} \in \mathtt{X}$, or the difference $t_i - t_j$ for some $1 \leq j \leq i$. Likewise is the difference of values of any two clocks. Thus clock constraints of transition rules used in the run compare these differences to integers. As timed automorphism π preserves integer differences, by executing the same sequence of transition rules we obtain the run over $\pi(w)$ starting in $\pi(c) = (p, \pi\circ\mu, \pi(t_0))$:

$$(p, \pi\circ\mu, \pi(t_0)) \xrightarrow{a_1, \pi(t_1)} \cdots \xrightarrow{a_n, \pi(t_n)} (q, \pi\circ\nu, \pi(t_n)),$$

also accepting as it ends in the same location q. As $w \in \mathbb{T}(\Sigma)$ can be chosen arbitrarily, we have thus proved one of inclusions, namely

$$\pi(L_A(p, \mu, t_0)) \;\subseteq\; L_A(p, \pi\circ\mu, \pi(t_0)).$$

The other inclusion follows from the latter one applied to π^{-1} and $L_A(p, \pi\circ\mu, \pi(t_0))$:

$$\pi^{-1}(L_A(p, \pi\circ\mu, \pi(t_0))) \;\subseteq\; L_A(p, \pi^{-1}\circ\pi\circ\mu, \pi^{-1}(\pi(t_0))) \;=\; L_A(p, \mu, t_0).$$

The two implications prove the equality. ◀

▶ **Lemma 3.5.** *For finite subsets $S, S' \subseteq \mathbb{R}_{\geq 0}$, if a timed language L is both S-invariant and S'-invariant, then it is also S''-invariant where $S'' = \mathsf{fract}(S) \cap \mathsf{fract}(S')$.*

Proof. Let L be an S- and S'-invariant timed language, and let $F = \mathsf{fract}(S)$ and $F' = \mathsf{fract}(S')$. Towards proving that L is an $(F \cap F')$-invariant subset of $\mathbb{T}(\Sigma)$, consider two timed words $w, w' \in \mathbb{T}(\Sigma)$ such that $w' = \pi(w)$ for some $(F \cap F')$-timed automorphism π. We need to show that $w \in L$ iff $w' \in L$, which follows immediately by the following claim:

▷ Claim A.1. Every $(F \cap F')$-timed automorphism π decomposes into $\pi = \pi_n \circ \cdots \circ \pi_1$, where each π_i is either F- or F'-timed automorphism.

Indeed, due to F- and F'-invariance of L, we have $w \in L$ iff $w' \in L$ as required.

As it has been proved in [11], instead of dealing with decomposition of π, it is sufficient to analyse the individual orbit of $F - F'$, in the special case when both $F - F'$ and $F' - F$ are singleton sets. The proof of Theorem 10.3 in [11] may be repeated here to prove that the last claim above is implied by the following one:

▷ Claim A.2. Let $F, F' \subseteq [0, 1)$ be finite sets s.t. $F - F' = \{t\}$ and $F' - F = \{t'\}$. For every $(F \cap F')$-timed automorphism π we have $\pi(t) = (\pi_n \circ \cdots \circ \pi_1)(t)$, for some $\pi_1, \ldots, \pi_n$, each of which is either F- or F'-timed automorphism.

The proof of the claim is split into two cases.

Case $F \cap F' \neq \emptyset$. Let l be the greatest element of $F \cap F'$ smaller than t, and let h be the smallest element of $F \cap F'$ greater than t, assuming they both exist. (If l does not exist put $l := h' - 1$, where h' is the greatest element of $F \cap F'$; symmetrically, if h does not exists put $h := l' + 1$, where l' is the smallest element of $F \cap F'$.) Then the $(F \cap F')$-orbit $\{\pi(t) \mid \pi \text{ is a } (F \cap F')\text{-timed automorphism}\}$ is the open interval (l, h). Take any $(\mathtt{F} \cap F')$-timed automorphism π; without loss of generality assume that $u = \pi(t) > t$. The only interesting case is $t < t' \leq u$. In this case, we show $\pi_2(\pi_1(t))$,where

- π_1 is some F'-timed automorphism that acts as identity on $[t', l+1]$ and s.t. $t < \pi_1(t) < t'$,
- π_2 is some F-timed automorphism that acts as identity on $[h-1, t]$ and s.t. $\pi_2(\pi_1(t)) = u$.

Case $F \cap F' = \emptyset$. Thus $F = \{t\}$ and $F' = \{t'\}$. Take any timed automorphism π; without loss of generality assume that $\pi(t) > t$. Let $z \in \mathbb{Z}$ be the unique integer s.t. $t'+z-1 < t < t'+z$. Let π_1 be an arbitrary $\{t'\}$-timed automorphism that maps t to some $t_1 \in (t, t'+z)$. Note that t_1 may be any value in $(t, t'+z)$. Similarly, let π_2 be an arbitrary $\{t\}$-timed automorphism that maps t_1 to some $t_2 \in (t', t+1)$. Again, t_2 may be any value in $(t', t+1)$. By repeating this process sufficiently many times one finally reaches $\pi(t)$ as required. ◀

B Proofs for Section 4

▷ Claim 4.6 (Invariance of $\mathcal{Y}$). For every two transitions $(X_1, S_1) \xrightarrow{a,t_1} (X'_1, S'_1)$ and $(X_2, S_2) \xrightarrow{a,t_2} (X'_2, S'_2)$ in $\mathcal{Y}$ and a timed permutation π, if $\pi(X_1) = X_2$ and $\pi(S_1) = S_2$ and $\pi(t_1) = t_2$, then we have $\pi(X'_1) = X'_2$ and $\pi(S'_1) = S'_2$.

Proof. Let i range over $\{1,2\}$ and let $\widetilde{X}_i := \textsc{succ}_{a,t_i}(X_i)$. Thus S'_i is the least subset of $S_i \cup \{t_i\}$ containing t_i such that $L_A(\widetilde{X}_i)$ is S'_i-invariant, and $X'_i = \Pi_{S'_i}(\widetilde{X}_i)$. By invariance of $[\![A]\!]$ (Fact 3.2) and invariance of semantics (Fact 3.3) we get

$$\pi(\widetilde{X}_1) = \widetilde{X}_2, \qquad \text{and} \qquad \pi(L_A(\widetilde{X}_1)) = L_A(\widetilde{X}_2),$$

and therefore $\pi(S'_1) = S'_2$, which implies $\pi(X'_1) = X'_2$. ◀

▷ Claim 4.8 (Invariance of $\mathcal{Z}$). For every two transitions $(X_1, \mu_1) \xrightarrow{a,t_1} (X'_1, \mu'_1)$ and $(X_2, \mu_2) \xrightarrow{a,t_2} (X'_2, \mu'_2)$ in $\mathcal{Z}$ and a timed permutation π, if $\pi(X_1) = X_2$ and $\pi \circ \mu_1 = \mu_2$ and $\pi(t_1) = t_2$, then we have $\pi(X'_1) = X'_2$ and $\pi \circ \mu'_1 = \mu'_2$.

Proof. Let i range over $\{1,2\}$. Let $S_i = \mu_i(\mathtt{X})$ and $(X_i, S_i) \xrightarrow{a,t_i} (X'_i, S'_i)$ in $\mathcal{Y}$. By Claim 4.6 we have

$$\pi(X'_1) = X'_2 \qquad \text{and } \pi(S'_1) = S'_2.$$

Since $\pi \circ \mu_1 = \mu_2$ and the definition (3) is invariant:

$$\pi \circ (\mu') = (\pi \circ \mu)',$$

we derive $\pi \circ \mu'_1 = \mu'_2$. ◀

C Proofs for Section 5

▶ **Lemma 5.2.** *The set of reachable configurations Reach(M) is finite if, and only if, $L(A)$ is a deterministic timed language.*

Proof. For the "only if" direction, if $\text{Reach}(M)$ is finite then there is some k s.t. every reachable configuration u has size $u(c_1) + u(c_2) + u(c_3) + u(c_4) + 1 \leq k$, and thus the set of reversals of accepting runs can be recognised by a $\text{DTA}_{(k+1)}$, and thus also its complement can be recognised by a $(k+1)$-DTA.

For the "if" direction, if $\text{Reach}(M)$ is infinite, then there exist reachable configurations with arbitrarily large counter values. Suppose, towards reaching contradiction, that $L(A)$ is recognised by a DTA_k. Thus also its complement, that is the set of reversal-encodings of runs of M, is recognised by some DTA_k B. There exists a run π of M where some counter value exceeds k, and thus when B reads the reversal-encoding of π it must forget some timestamp (say) (c_1, t) in some configuration $p_{i+1}\delta_{i+1}u_{i+1}$. Since t is forgotten, we can perturb its corresponding $(c_1, t+1)$ in $p_i\delta_i u_i$ to any value (c_1, t') s.t. $t' - t \neq 1$ and obtain a new word still accepted by A, but which is no longer the reversal-encoding of a run of M, thus reaching the sought contradiction. ◀

▶ **Lemma 5.5.** *Let $k, m \in \mathbb{N}$ and let $\mathcal{Y}$ be a class of timed languages that*
1. *contains all the timeless timed languages,*
2. *is closed under union and composition, and*
3. *contains some non-DTA_k (resp. non-$\text{DTA}_{k,m}$) language.*

The universality problem for languages in $\mathcal{Y}$ reduces in polynomial time to the DTA_k (resp. $\text{DTA}_{k,m}$) membership problem for languages in $\mathcal{Y}$.

Proof. We consider DTA_k membership (the $\text{DTA}_{k,m}$ membership is treated similarly). Consider some fixed timed language $M \in \mathcal{Y}$ which is not recognised by any DTA_k (relying on the assumption 3), over an alphabet Γ. For a given timed language $L \in \mathcal{Y}$, over an alphabet Σ, we construct the following language over the extended alphabet $\Sigma \cup \Gamma \cup \{\$\}$:

$$N \;:=\; L \triangleright \{\$\} \triangleright \mathbb{T}(\Gamma) \;\cup\; \mathbb{T}(\Sigma) \triangleright \{\$\} \triangleright M \;\subseteq\; \mathbb{T}(\Sigma \cup \Gamma \cup \{\$\}),$$

where $\$ \notin \Sigma \cup \Gamma$ is a fixed fresh alphabet symbol. Since $\mathcal{Y}$ contains all timeless timed languages due to the assumption 1, and is closed under union and composition due to the assumption 2, the language N belongs to $\mathcal{Y}$.

▷ Claim. $L = \mathbb{T}(\Sigma)$ if, and only if, N is recognised by a DTA_k.

For the "only if" direction, if $L = \mathbb{T}(\Sigma)$ then clearly $N = \mathbb{T}(\Sigma) \cdot \{\$\} \cdot \mathbb{T}(\Gamma)$. Thus N is timeless and in consequence N is recognised by a DTA_k, as DTA_k recognise all timeless timed languages for any $k \geq 0$.

For the "if" direction suppose, towards reaching a contradiction, that N is recognised by a DTA_k A but $L \neq \mathbb{T}(\Sigma)$. Assume, w.l.o.g., that A is greedily resetting. Choose an arbitrary timed word $w = (a_1, t_1) \ldots (a_n, t_n) \notin L$ over Σ. Therefore, for any extension $v = (a_1, t_1) \ldots (a_n, t_n)(\$, t_n + t)$ of w by one letter, we have

$$v^{-1}N = t + M = \{(b_1, t + u_1) \ldots (b_m, t + u_m) \mid (b_1, u_1) \ldots (b_m, u_m) \in M\}\,.$$

Choose t larger than the largest absolute value m of constants appearing in clock constraints in A, and let (p, μ) be the configuration reached by A after reading v. As $t > m$, all the clocks are reset by the last transition and hence $\mu(\mathtt{x}) = 0$ for all clocks $\mathtt{x}$. Consequently, if the initial control location of A were moved to the location p, the so modified DTA_k A' would accept the language M. But this contradicts our initial assumption that M is not recognised by a DTA_k, thus finishing the proof. ◀