# DETERMINISABILITY OF REGISTER AND TIMED AUTOMATA [*]

LORENZO CLEMENTE, SLAWOMIR LASOTA, AND RADOSLAW PIÓRKOWSKI

University of Warsaw, Poland
*e-mail address*: clementelorenzo@gmail.com
*URL*: `https://orcid.org/0000-0003-0578-9103`

University of Warsaw, Poland
*e-mail address*: sl@mimuw.edu.pl
*URL*: `https://orcid.org/0000-0001-8674-4470`

University of Warsaw, Poland
*e-mail address*: r.piorkowski@mimuw.edu.pl
*URL*: `https://orcid.org/0000-0002-9643-182X`

*Key words and phrases:* Timed automata, register automata, determinisation, deterministic membership problem.

[*] This paper is an extended version of the conference paper [20].

Partially supported by the Polish NCN grant 2017/26/D/ST6/00201.

Partially supported by the Polish NCN grant 2019/35/B/ST6/02322 and by the ERC grant LIPA, agreement no. 683080.

Partially supported by the Polish NCN grant 2017/27/B/ST6/02093.

Abstract. The deterministic membership problem for timed automata asks whether the timed language given by a nondeterministic timed automaton can be recognised by a deterministic timed automaton. An analogous problem can be stated in the setting of register automata. We draw the complete decidability/complexity landscape of the deterministic membership problem, in the setting of both register and timed automata. For register automata, we prove that the deterministic membership problem is decidable when the input automaton is a nondeterministic one-register automaton (possibly with epsilon transitions) and the number of registers of the output deterministic register automaton is fixed. This is optimal: We show that in all the other cases the problem is undecidable, i.e., when either 1) the input nondeterministic automaton has two registers or more (even without epsilon transitions), or 2) it uses guessing, or 3) the number of registers of the output deterministic automaton is not fixed. The landscape for timed automata follows a similar pattern. 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 output deterministic timed automaton is fixed. Again, this is optimal: 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.

## 1. Introduction

### 1.1. **Automata over infinite alphabets.**

**Timed automata.** *Nondeterministic timed automata* (NTA) are one of the most widespread models 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 1994 paper [3]. This paved the way for the automatic verification of timed systems, leading to mature tools such as UPPAAL [9], UPPAAL Tiga (timed games) [17], and PRISM (probabilistic timed automata) [40]. The reachability problem is still a very active research area these days [26, 35, 2, 31, 32, 34], as are its expressive generalisations thereof, such as the binary reachability problem [23, 25, 39, 29]. As a testimony to the model's importance, the authors of [3] received the 2016 Church Award [1] for the invention of timed automata.

*Deterministic timed automata* (DTA) form a strict subclass of NTA where the successive 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/equivalence/inclusion problems, and closure under complementation [3]. Moreover, the more restrictive nature of DTA is necessary for several applications of timed automata, such as test generation [47], fault diagnosis [14], and learning [56, 52], winning conditions in timed games [5, 36, 15], and in a notion of recognisability of timed languages [43]. 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 [51]. 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 $\text{NTA}_1$ is incomparable with DTA: For instance, $\text{NTA}_1$ are not closed under complement (unlike DTA), and there are very simple DTA languages that are not recognisable by any $\text{NTA}_1$. Nonetheless, $\text{NTA}_1$, like DTA, have decidable inclusion, equivalence, and universality problems [48, 42], although the complexity is non-primitive recursive [42, Corollary 4.2] (see also [49, Theorem 7.2] for an analogous lower bound for the satisfiability problem of metric temporal logic). Moreover, the nonemptiness problem for $\text{NTA}_1$ is NLogSpace-complete (vs. PSpace-complete for unrestricted NTA and DTA, already with two clocks [26]), and the binary reachability relation of $\text{NTA}_1$ can be computed as a formula of existential linear arithmetic of polynomial size, which is not the case in the general case [19].

**Register automata.** The theory of register automata shares many similarities with that of timed automata. *Nondeterministic register automata* (NRA) have been introduced by Kaminski and Francez around the same time as timed automata. They were defined as an extension of finite automata with finitely many registers that can store input values and be compared with equality and disequality constraints. The authors have shown, amongst other things, that nonemptiness is decidable [37, Theorem 1]. It was later realised that the problem is, in fact, PSpace-complete [24, Theorems 4.3 and Theorem 5.1]. The class of NRA recognisable languages is not closed under complementation [37, Proposition 5]; moreover, universality (and thus equivalence and inclusion) of NRA is undecidable [46, Theorem 5.1] (already for NRA with two registers [24, Theorem 5.4]).

One way to regain decidability is to consider *deterministic register automata* (DRA), which are effectively closed under complement and thus have a decidable inclusion (and thus universality and equivalence) problem[1]. DRA also provide the foundations of learning algorithms for data languages [45]. A recent result completing the theory of register automata has shown that a data language is DRA recognisable if, and only if, both this language and its complement are NRA recognisable [38].

As in the case of timed automata, it has been observed that restricting the number of registers results in algorithmic gains. Already in the seminal work of Kaminski and Francez, it has been proved that the inclusion problem $L(A) \subseteq L(B)$ is decidable when $A$ is an NRA and $B$ is an NRA with one register [37, Appendix A], albeit the complexity is non-primitive recursive in this case [24, Theorem 5.2].

## 1.2. **The deterministic membership problem.**

[1]In fact, even the inclusion problem $L(A) \subseteq L(B)$ with $A$ an NRA and $B$ a DRA is decidable.

**Timed automata.** 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 $\mathrm{DTA}_k$ and $\mathrm{DTA}_{k,m}$ *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 the 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 untimed regular languages are deterministic, the $\mathrm{DTA}_k$ membership problem can be seen as a quantitative generalisation of the regularity problem. For instance, the $\mathrm{DTA}_0$ membership problem is precisely 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 [50, Theorem 6.6.6], labelled Petri nets under reachability semantics [55], Parikh automata [16], etc. One way to obtain decidability is to either restrict the input model to be deterministic (e.g., [54, 55, 8]), or to consider more refined notions of equivalence, such as bisimulation (e.g., [33]).

This negative situation is generally confirmed for timed automata. For every number of clocks $k \in \mathbb{N}$ and maximal constant $m$, the DTA, $\mathrm{DTA}_k$, and $\mathrm{DTA}_{k,m}$ membership problems are known to be undecidable when the input NTA has $\geq 2$ clocks, and for 1-clock NTA with epsilon transitions [27, 53]. To the best of our knowledge, the deterministic membership problem was not studied before when the input automaton is $\mathrm{NTA}_1$ without epsilon transitions.

**Register automata.** The situation for register automata is similar to, and simpler than, timed automata. The $\mathrm{DRA}_k$ *membership problem* asks, given a NRA, whether there exists a DRA with $k$ registers recognising the same language, and the DRA *membership problem* is the same problem with no apriori bound on the number of registers of the deterministic acceptor. Deterministic membership problems for register automata do not seem to have been considered before in the literature.

## 1.3. **Contributions.**

We complete the study of the decidability border for the deterministic membership problem initiated for timed automata in [27, 53], and we extend these results to register automata.

**Upper bounds.** Our main result is the following.

**Theorem 1.1.** *The* $\mathrm{DTA}_k$ *and* $\mathrm{DTA}_{k,m}$ *membership problems are decidable for* $\mathrm{NTA}_1$ *languages.*

Our decidability result contrasts starkly with the abundance of undecidability results for the regularity problem. We establish decidability by showing that if a $\mathrm{NTA}_{1,m}$ recognises a $\mathrm{DTA}_k$ language, then, in fact, it recognises a $\mathrm{DTA}_{k,m}$ language and, moreover, there is a computable bound on the number of control locations of the deterministic acceptor (c.f. Lemma 7.1). This provides a decision procedure since there are finitely many different

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 $\text{NTA}_k$. These automata are required to reset at least one clock at every transition and are thus of expressive power intermediate between $\text{NTA}_{k-1}$ and $\text{NTA}_k$. Always resetting $\text{NTA}_2$ are strictly more expressive than $\text{NTA}_1$: 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 $\text{NTA}_2$ but by no $\text{NTA}_1$. Despite their increased expressive power, always resetting $\text{NTA}_2$ still have a decidable universality problem (the well-quasi order approach of [48] goes through), which is not the case for $\text{NTA}_2$. Thanks to this restricted form, we are able to provide in Lemma 7.1 an elegant characterisation of those $\text{NTA}_1$ languages which are recognised by an always resetting $\text{DTA}_k$.

We prove a result analogous to Theorem 1.1 in the setting of register automata.

**Theorem 1.2.** *The $\text{DRA}_k$ membership problem is decidable for $\text{NRA}_1$ languages.*

Thanks to the effective elimination of $\varepsilon$-transition rules from $\text{NRA}_1$ (c.f. Lemma 3.3), the decidability result above also holds for data languages presented as $\text{NRA}_1$ with $\varepsilon$-transition rules.

**Lower bounds.** We complement the decidability results above by showing that the deterministic membership problem becomes undecidable if we do not restrict the number of clocks/registers of the deterministic acceptor.

**Theorem 1.3.** *The DTA and $\text{DTA}_{\_,m}$ $(m > 0)$ membership problems are undecidable for $\text{NTA}_1$ without epsilon transitions.*

Theorem 1.3 improves on an analogous result from [27, Theorem 1] for $\text{NTA}_2$. We obtain a similar undecidability result in the setting of register automata:

**Theorem 1.4.** *The DRA membership problem is undecidable for $\text{NRA}_1$.*

The following lower bounds further refine the analysis from [27] in the case of a fixed number of clocks of a deterministic acceptor.

**Theorem 1.5.** *For every fixed $k, m \in \mathbb{N}$, the $\text{DTA}_k$ and $\text{DTA}_{k,m}$ membership problems are:*
- *undecidable for $\text{NTA}_2$,*
- *undecidable for $\text{NTA}_1^\varepsilon$ (with epsilon transitions),*
- HYPERACKERMANN-*hard for $\text{NTA}_1$.*

A similar landscape holds for register automata, where the deterministic membership problem for a fixed number of registers of the deterministic acceptor remains undecidable when given in input either a $\text{NRA}_2$ or a $\text{NRA}_1$ with guessing. In the decidable case of a $\text{NRA}_1$ input, the problem is nonetheless not primitive recursive.

**Theorem 1.6.** *Fix a $k \geq 0$. The $\text{DRA}_k$ membership problem is:*
(1) *undecidable for $\text{NRA}_2$,*

(2) *undecidable for* $\text{NRA}_1^{\mathrm{g}}$ *(*$\text{NRA}_1$ *with guessing), and*
(3) *not primitive recursive (*ACKERMANN*-hard) for* $\text{NRA}_1$.

**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 [21]. This contrasts with undecidability of the corresponding membership problem. The deterministic separability problem for register automata has not been studied in the literature. Decidability of the deterministic separability problem when the number of clocks/registers of the separator is not provided remains a challenging open problem.

## 2. Automorphisms, orbits, and invariance

This section contains preliminary definitions needed both for register and timed automata.

**Atoms.** Let $\mathbb{A}$ be a structure (whose elements are called *atoms*) providing a data domain on which register and timed automata operate. In the case of register automata, we will primarily be concerned with *equality atoms* $(\mathbb{A}, =)$, where $\mathbb{A}$ is a countable set, and the signature contains the equality symbol only. However—as we discuss at the end of Section 7—our positive results generalise to other atoms, for instance to densely ordered atoms $(\mathbb{R}, \leq)$, where $\mathbb{R}$ is the set of real numbers with the natural order "$\leq$". In the case of timed automata, we will consider *timed atoms* $(\mathbb{R}, \leq, +1)$, which extend densely ordered atoms with the increment function "$+1$".

**Automorphisms.** Let $S \subseteq \mathbb{A}$ be a (possibly empty) finite set of atoms. An *$S$-automorphism* is a bijection $\pi : \mathbb{A} \to \mathbb{A}$ that is the identity on $S$, i.e., $\pi(a) = a$ for every $a \in S$, and which preserves the atom structure. The latter condition is trivially satisfied for equality atoms. In the case of densely ordered atoms, it also demands monotonicity: $a \leq b$ implies $\pi(a) \leq \pi(b)$. In the case of timed atoms automorphisms additionally preserve integer differences: $\pi(a + 1) = \pi(a) + 1$, for every $a, b \in \mathbb{R}$. For instance, if $\pi(3.4) = 4.5$, then the last condition necessarily implies $\pi(5.4) = 6.5$ and $\pi(-3.6) = -2.5$. When $S = \emptyset$, we just say that $\pi$ is an automorphism. Let $\text{Aut}_S(\mathbb{A})$ denote the set of all $S$-automorphisms of $\mathbb{A}$, and let $\text{Aut}(\mathbb{A}) = \text{Aut}_\emptyset(\mathbb{A})$.

Informally speaking, by *sets with atoms* we mean sets whose elements are either sets or atoms (for a formal definition of the hierarchy thereof, we refer to [11]). If $X$ is a set with atoms, then $\pi(X)$ is the set with atoms which is obtained by replacing every atom $a \in \mathbb{A}$ occurring in $X$ by $\pi(a) \in \mathbb{A}$. We present some commonly occurring concrete examples, relying on standard set-theoretic encodings of tuples, functions, etc. Let $\Sigma$ be a finite alphabet. A *data word* is a finite sequence

$$w = (\sigma_0, a_0) \cdots (\sigma_n, a_n) \in (\Sigma \times \mathbb{A})^* \tag{2.1}$$

of pairs $(\sigma_i, a_i)$ consisting of an input symbol $\sigma_i \in \Sigma$ and an atom $a_i \in \mathbb{A}$. An automorphism $\pi$ acts on a data word $w$ as above point-wise: $\pi(w) = (\sigma_0, \pi(a_0)) \cdots (\sigma_n, \pi(a_n))$.

Let $\mathtt{X}$ be a finite set of register names and let $\mathbb{A}_\bot = \mathbb{A} \cup \{\bot\}$, where $\bot \notin \mathbb{A}$ represents an undefined value. A *register valuation* is a mapping $\mu : \mathbb{A}_\bot^{\mathtt{X}}$ assigning an atom (or $\bot$) $\mu(x)$ to every register $x \in \mathtt{X}$. An automorphism $\pi$ acts on a register valuation $\mu$ as $\pi(\mu)(x) = \pi(\mu(x))$ for every $x \in \mathtt{X}$, i.e., $\pi(\mu) = \pi \circ \mu$, where we assume that $\pi(\bot) = \bot$.

More generally, if $X$ is a set with atoms, then $\pi$ acts on $X$ pointwise as $\pi(X) = \{\pi(x) \mid x \in X\}$.

**Orbits and invariance.** A set with atoms $X$ is $S$*-invariant* if $\pi(X) = X$ for every $S$-automorphism $\pi \in \mathrm{Aut}_S(\mathbb{A})$. Notice that $\pi$ does not need to be the identity on $X$ for $X$ to be $S$-invariant. A set $X$ is *invariant*[2] if it is $S$-invariant with $S = \emptyset$. The $S$*-orbit* of an element $x \in X$ (which can be an arbitrary object on which the action of automorphisms is defined) is the set

$$\textsc{orbit}_S(x) = \{\pi(x) \in X \mid \pi \in \mathrm{Aut}_S(\mathbb{A})\}$$

of all elements $\pi(x)$ which can be obtained by applying some $S$-automorphism $\pi$ to $x$. The *orbit* of $x$ is just its $S$-orbit with $S = \emptyset$, written $\textsc{orbit}(x)$. Clearly $x, y \in X$ have the same $S$-orbit $\textsc{orbit}_S(x) = \textsc{orbit}_S(y)$ if, and only if, $\pi(x) = y$ for some $\pi \in \mathrm{Aut}_S(\mathbb{A})$.

The $S$*-orbit closure* of a set $X$ (or just $S$*-closure*) is the union of the $S$-orbits of its elements:

$$\mathrm{Cl}_S(X) = \bigcup_{x \in X} \textsc{orbit}_S(x).$$

In particular, the $S$-orbit of $x$ is the $S$-closure of the singleton set $\{x\}$: $\textsc{orbit}_S(x) = \mathrm{Cl}_S(\{x\})$. The $\emptyset$-closure $\mathrm{Cl}(X)$ we briefly call the *closure* of $X$. The following fact characterises invariance in term of closures.

**Fact 2.1.** A set $X$ is $S$-invariant if, and only if, $\mathrm{Cl}_S(X) = X$.

*Proof.* The "only if" direction follows the definition of $S$-invariance. For the "if" direction, observe that $\mathrm{Cl}_S(X) = X$ implies $\pi(X) \subseteq X$. The opposite inclusion stems from $S$-automorphisms' closure under inverse: $\pi^{-1}(X) \subseteq X$, hence $X \subseteq \pi(X)$. □

## 3. Register automata

In this section, we define register automata over equality atoms $(\mathbb{A}, =)$. However, all the definitions below can naturally be generalised to any atoms satisfying some mild assumptions, as explained later in Section 4.2.

[2]The term *equivariant* is also often used in the literature instead of *invariant*. Also, in the case of $S$-invariant $X$, the literature often calls $S$ a *support* of $X$, see e.g. [11, 12].

**Constraints.** A *constraint* is a quantifier-free formula $\varphi$ generated by the grammar

$$\varphi, \psi \ ::\equiv \ \textbf{true} \mid \textbf{false} \mid \mathtt{x} = \mathtt{y} \mid \mathtt{x} = \bot \mid \neg\varphi \mid \varphi \wedge \psi \mid \varphi \vee \psi, \tag{3.1}$$

where $\mathtt{x}, \mathtt{y}$ are variables and $\bot$ is a special constant denoting an undefined data value. The *semantics* of a constraint $\varphi(\mathtt{x}_1, \dots, \mathtt{x}_n)$ with $n$ free variables $\mathtt{X} = \{\mathtt{x}_1, \dots, \mathtt{x}_n\}$ is the set of valuations satisfying it: $[\![\varphi]\!] = \{\mu \in \mathbb{A}_\bot^{\mathtt{X}} \mid \mu \models \varphi\}$. Using [11, Lemma 7.5] we easily deduce:

**Claim 3.1.** Subsets of $\mathbb{A}_\bot^{\mathtt{X}}$ definable by constraints are exactly the invariant subsets of $\mathbb{A}_\bot^{\mathtt{X}}$.

**Register automata.** Let $\Sigma$ be a finite alphabet, and let $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$ be $\Sigma$ with the addition of the empty word $\varepsilon$. A (nondeterministic) *register automaton* (NRA) is a tuple $A = (\mathtt{X}, \Sigma, L, L_I, L_F, \Delta)$ where $\mathtt{X} = \{\mathtt{x}_1, \dots, \mathtt{x}_k\}$ is a finite set of register names, $\Sigma$ is a finite alphabet, $L$ is a finite set of *control locations*, of which we distinguish those which are *initial* $L_I \subseteq L$, resp., *final* $L_F \subseteq L$, and $\Delta$ is a set of *transition rules.* We have two kinds of transition rules. A *non-$\varepsilon$-transition rule* is of the form

$$(p, \sigma, \varphi, \mathtt{Y}, q) \in \Delta \tag{3.2}$$

and it means that from control location $p \in L$ we can go to $q \in L$ by reading input symbol $\sigma \in \Sigma$, provided that the transition constraint $\varphi(\mathtt{x}_1, \dots, \mathtt{x}_k, \mathtt{y})$ holds between the current registers $\mathtt{x}_1, \dots, \mathtt{x}_k$ and the input data value $\mathtt{y}$ being currently read; finally, all registers in $\mathtt{Y} \subseteq \mathtt{X}$ are updated to store the input data value $\mathtt{y}$. An *$\varepsilon$-transition rule* is of the form $(p, \varepsilon, \varphi, q) \in \Delta$ and it means that from control location $p \in L$ we can go to $q \in L$, but no input is read, provided that the transition constraint $\varphi(\mathtt{x}_1, \dots, \mathtt{x}_k)$ holds between the current registers $\mathtt{x}_1, \dots, \mathtt{x}_k$.

Formally, the semantics of a register automaton $A$ as above is provided in terms of an infinite transition system $[\![A]\!] = (C, C_I, C_F, \to)$, where $C = L \times \mathbb{A}_\bot^{\mathtt{X}}$ is the set of *configurations*, which are pairs $(p, \mu)$ consisting of a control location $p \in L$ and a register valuation $\mu \in \mathbb{A}_\bot^{\mathtt{X}}$. Amongst them, $C_I = L_I \times \{\lambda\mathtt{x} \cdot \bot\} \subseteq C$ is the set of *initial configurations*, i.e., configurations of the form $(p, \mu)$ with $p \in L_I$ and $\mu(\mathtt{x}) = \bot$ for all registers $\mathtt{x}$, and $C_F = L_F \times \mathbb{A}_\bot^{\mathtt{X}}$ is the set of *final configurations*, i.e., configurations of the form $(p, \mu)$ with $p \in L_F$ (without any further restriction on $\mu$). The set of transitions "$\to$" is determined as follows. For a valuation $\mu \in \mathbb{A}_\bot^{\mathtt{X}}$, a set of registers $\mathtt{Y} \subseteq \mathtt{X}$, and an atom[3] $a \in \mathbb{A}$, let $\mu[\mathtt{Y} \mapsto a]$ be the valuation which is $a$ on $\mathtt{Y}$ and agrees with $\mu$ on $\mathtt{X} \setminus \mathtt{Y}$. Every non-$\varepsilon$-transition rule (3.2) induces a transition between configurations

$$(p, \mu) \xrightarrow{\sigma, a} (q, \mu[\mathtt{Y} \mapsto a])$$

labelled by $(\sigma, a) \in \Sigma \times \mathbb{A}$, provided that the current valuation $\mu$ satisfies the constraint $\varphi$ when variable $\mathtt{y}$ holds the input atom $a$, i.e., $\mu[\mathtt{y} \mapsto a] \models \varphi(\mathtt{x}_1, \dots, \mathtt{x}_k, \mathtt{y})$. Similarly, an $\varepsilon$-transition rule $(p, \varepsilon, \varphi, q) \in \Delta$ induces an $\varepsilon$-labelled transition $(p, \mu) \xrightarrow{\varepsilon} (q, \mu)$ whenever $\mu \models \varphi(\mathtt{x}_1, \dots, \mathtt{x}_k)$. Finally, in order to deal with $\varepsilon$-transitions, we stipulate that whenever we have transitions $(p, \mu) \xrightarrow{\varepsilon} \_ \xrightarrow{b} \_ \xrightarrow{\varepsilon} (q, \nu)$ with $b \in (\Sigma \times \mathbb{A}) \cup \{\varepsilon\}$, then we also have the

[3]It suffices to consider non-$\bot$ values $a \neq \bot$ since we never need to reset a register to the undefined value $\bot$.

transition $(p, \mu) \xrightarrow{b} (q, \nu)$. A *run* of $A$ *over* a data word $w$ as in (2.1) *starting* in configuration $(p, \mu)$ and *ending* in configuration $(q, \nu)$ is a labelled path $\rho$ in $[\![A]\!]$ of the form

$$\rho = (p, \mu) \xrightarrow{\sigma_0, a_0} (p_0, \mu_0) \xrightarrow{\sigma_1, a_1} \cdots \xrightarrow{\sigma_n, a_n} (q, \nu). \tag{3.3}$$

The run $\rho$ is accepting if its ending configuration is accepting. The language *recognised* by configuration $(p, \mu)$ is the set of data words labelling accepting runs:

$$L(p, \mu) = \{w \in (\Sigma \times \mathbb{A})^* \mid [\![A]\!] \text{ has an accepting run over } w \text{ starting in } (p, \mu)\}.$$

The language recognised by the automaton $A$ is the union of the languages recognised by its initial configurations, $L(A) = \bigcup_{c \in C_I} L(c)$. A configuration is *reachable* if it is the ending configuration in a run starting in an initial configuration.

**Remark 3.2.** Register automata, as defined above, are *without guessing*, i.e. an automaton can not store in a register an atom not previously seen in the input.

It turns out that NRA with $\varepsilon$-transition rules are as expressive as NRA without $\varepsilon$-transition rules.

**Lemma 3.3** (Excercise 2 in [11]). *For every $k \in \mathbb{N}$ and* NRA$_k$ *one can effectively build an* NRA$_k$ *without $\varepsilon$-transition rules recognising the same language.*

For this reason, from this point on, we deal exclusively with NRA without $\varepsilon$-transition rules, and we tacitly assume that an NRA does not contain $\varepsilon$-transition rules.

**Deterministic register automata.** A register automaton $A$ is *deterministic* (DRA) if it has precisely one initial location $L_I = \{p_I\}$ and, for every two rules $(p, \sigma, \varphi, \mathtt{Y}, q)$ and $(p, \sigma, \varphi', \mathtt{Y}', q')$ starting in the same location $p$, over the same input symbol $\sigma$ and with jointly satisfiable guards $[\![\varphi \wedge \varphi']\!] \neq \emptyset$, we necessarily have $\mathtt{Y} = \mathtt{Y}'$ and $q = q'$. Hence $A$ has at most one run over every data word $w$. A DRA can be easily transformed into a *total* one, i.e., one where for every location $p \in L$ and input symbol $\sigma \in \Sigma$, the sets defined by the constraints $\{[\![\varphi]\!] \mid \exists \mathtt{Y}, q \cdot p \xrightarrow{\sigma, \varphi, \mathtt{Y}} q\}$ are a partition of all valuations $\mathbb{A}_\perp^{\mathtt{X} \cup \{\mathtt{y}\}}$. Thus, a total DRA has exactly one run over every timed word $w$.

We write NRA$_k$, resp. DRA$_k$, for the class of $k$-register NRA, resp. DRA, and we say that a data language is an NRA language, DRA language, DRA$_k$ language, etc., if it is recognised by a register automaton of the respective type.

**Example 3.4.** Let $\Sigma = \{\sigma\}$ be a unary alphabet. As an example of a language $L$ recognised by an NRA$_1$, but not by any DRA, consider the set of data words where the last atom reappears earlier, i.e., words of the form: $(\sigma, a_1) \cdots (\sigma, a_n)$ where $a_i = a_n$ for some $1 \leq i < n$. The language $L$ is recognised by the NRA$_1$ $A = (\mathtt{X}, \Sigma, L, L_I, L_F, \Delta)$ with one register $\mathtt{X} = \{\mathtt{x}\}$ and three locations $L = \{p, q, r\}$, of which $L_I = \{p\}$ is initial and $L_F = \{r\}$ is final, and transition rules

$$(p, \sigma, \mathbf{true}, \emptyset, p) \qquad (p, \sigma, \mathbf{true}, \{\mathtt{x}\}, q) \qquad (q, \sigma, \mathtt{x} \neq \mathtt{y}, \emptyset, q) \qquad (q, \sigma, \mathtt{x} = \mathtt{y}, \emptyset, r).$$

Intuitively, the automaton waits in $p$ until it guesses that the next input $a_i$ will be appearing at the end of the word as well, at which point it moves to $q$ by storing $a_i$ in the register.

From $q$, the automaton can accept by going to $r$ exactly when the atom stored in the register reappears in the input. The language $L$ is not recognised by any DRA since, intuitively, any deterministic acceptor needs to store unboundedly many different atoms $a_i$.

**One-register automata.** Nondeterministic register automata with just one register enjoy stronger algorithmic properties than the full class of nondeterministic register automata. It was already observed in Kaminski and Francez's seminal paper that the inclusion problem becomes decidable[4].

**Theorem 3.5** (c.f. [37, Appendix A]). *For $A \in$ NRA and $B \in \text{NRA}_1$ the language inclusion problem $L(A) \subseteq L(B)$ is decidable.*

We immediately obtain the following corollary, which we will use in Section 4.

**Corollary 3.6.** *For $A \in$ DRA and $B \in \text{NRA}_1$ the language equality problem $L(A) = L(B)$ is decidable.*

*Proof.* The inclusion $L(A) \subseteq L(B)$ can be checked as a special instance of Theorem 3.5. The reverse inclusion $L(B) \subseteq L(A)$ reduces to checking emptiness of a product construction of $B$ with the complement of $A$. □

**Invariance of register automata.** The following lemma expresses the fundamental invariance properties of register automata. Given a valuation $\mu$ of registers $\mathtt{X}$, by $\mu(\mathtt{X}) \subseteq \mathbb{A}$ we mean the set of atoms stored in registers: $\mu(\mathtt{X}) = \{\mu(\mathtt{x}) \mid \mathtt{x} \in \mathtt{X},\ \mu(\mathtt{x}) \in \mathbb{A}\}$. Automorphisms act on configurations by preserving the control location: $\pi(p, \mu) = (p, \pi(\mu)) = (p, \pi \circ \mu)$.

**Lemma 3.7** (Invariance of NRA). (1) *The transition system $[\![A]\!]$ is invariant: If $c \xrightarrow{\sigma,a} d$ in $[\![A]\!]$ and $\pi$ is an automorphism, then $\pi(c) \xrightarrow{\sigma,\pi(a)} \pi(d)$ in $[\![A]\!]$.*
(2) *The function $L(\_)$ mapping a configuration $c$ to the language $L(c)$ it recognises from $c$ is invariant: For all automorphisms $\pi$, $L(\pi(c)) = \pi(L(c))$.*
(3) *The language $L(p, \mu)$ recognised from a configuration $(p, \mu)$ is $\mu(\mathtt{X})$-invariant: For all $\mu(\mathtt{X})$-automorphims $\pi$, $\pi(L(p, \mu)) = L(p, \mu)$.*

We refrain from proving Lemma 3.7, since proofs of analogous invariance properties, in the more involved setting of timed automata, are provided later (Facts 6.2–6.4 in Section 6). For the proof of (1) in the setting of equality atoms, we refer the reader to [11, Sect.1.1]; the other points are readily derivable from (1).

[4]A *window* in the terminology of [37] corresponds to a register in this paper's terminology. [37, Appendix A] shows that the inclusion problem $L(A) \subseteq L(B)$ is decidable when $B$ is a two-window automaton. Due to the semantics of window reassignment of [37], two-window automata are of intermediate expressive power between one-register automata and two-register automata.

**Deterministic membership problems.** Let $\mathcal{X}$ be a subclass of NRA. We are interested in the following family of decision problems:

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

We study the decidability status of the $\mathcal{X}$ membership problem where $\mathcal{X}$ ranges over DRA and $\text{DRA}_k$ (for every fixed number of registers $k$). Example 3.4 shows that there are NRA languages that cannot be accepted by any DRA. Moreover, there is no computable bound for the number of registers $k$ which suffice to recognise a $\text{NRA}_1$ language by a $\text{DRA}_k$ (when such a number exists), which follows from the following three observations:
(1) the DRA membership problem is undecidable for $\text{NRA}_1$ (Theorem 1.4),
(2) the problem of deciding equivalence of a given $\text{NRA}_1$ to a given DRA is decidable by Theorem 3.6 and
(3) if an $\text{NRA}_1$ is equivalent to some $\text{DRA}_k$ then it is in fact equivalent to some $\text{DRA}_k$ with computably many control locations (by Lemma 4.1 from the next section).

## 4. Decidability of $\text{DRA}_k$ membership for $\text{NRA}_1$

In this section we prove our main decidability result for register automata, which we now recall.

**Theorem 1.2.** *The $\text{DRA}_k$ membership problem is decidable for $\text{NRA}_1$ languages.*

The technical development of this section will also serve as a preparation for the more involved case of timed automata from Sections 5 and 6. The key ingredient used in the proof of Theorem 1.2 is the following characterisation of those $\text{NRA}_1$ languages which are also $\text{DRA}_k$ languages. In particular, this characterisation provides a bound on the number of control locations of a $\text{DRA}_k$ equivalent to a given $\text{NRA}_1$ (if any exists).

**Lemma 4.1.** *Let $A$ be a $\text{NRA}_1$ with $n$ control locations, and let $k \in \mathbb{N}$. The following conditions are equivalent:*
(1) *$L(A) = L(B)$ for some $\text{DRA}_k$ $B$.*
(2) *For every data word $w$, there is $S \subseteq \mathbb{A}$ of size at most $k$ s.t. the left quotient $w^{-1}L(A) = \{v \mid w \cdot v \in L(A)\}$ is $S$-invariant.*
(3) *$L(A) = L(B)$ for some $\text{DRA}_k$ $B$ with at most $f(k, n) = (k+1)! \cdot 2^{n \cdot (k+1)}$ control locations.*

Using the above lemma we derive a proof of Theorem 1.2:

*Proof of Theorem 1.2.* Given a $\text{NRA}_1$ $A$, the decision procedure enumerates all $\text{DRA}_k$ $B$ with at most $f(k, n)$ locations and checks whether $L(A) = L(B)$, using Theorem 3.6. If no such $\text{DRA}_k$ $B$ is found, the procedure answers negatively. □

**Remark 4.2** (Complexity)**.** The decision procedure for $\textsc{nra}_1$ invokes the Ackermann subroutine to check equivalence between a $\textsc{nra}_1$ and a candidate $\textsc{dra}$. This is in a sense unavoidable, since we show in Theorem 1.6 that the $\textsc{dra}_k$ membership problem is Ackermann-hard for $\textsc{nra}_1$.

## 4.1. **Proof of Lemma 4.1.**

Let us fix a $\textsc{nra}_1$ $A = (\mathtt{X}, \Sigma, L, L_I, L_F, \Delta)$ and $k \in \mathbb{N}$. Let $n = |L|$ be the number of control locations of $A$. The implication (3)⇒(1) follows by definition. For the implication (1)⇒(2) assume that $L(A) = L(B)$ for a $\textsc{dra}_k$ $B$. Every left quotient $w^{-1}L(A)$ equals $L_B(c)$ for some configuration $c = (p, \mu)$. The latter is $\mu(\mathtt{X})$-invariant by Lemma 3.7(3), and clearly $|\mu(\mathtt{X})| \leq k$.

It thus remains to prove the implication (2)⇒(3), which is the content of the rest of the section. Assuming (2), we are going to define a $\textsc{dra}_k$ $B'$ with registers $\mathtt{X} = \{\mathtt{x}_1, \ldots, \mathtt{x}_k\}$ and with at most $f(k, n)$ locations such that $L(B') = L(A)$. We start from the transition system $\mathcal{X}$ obtained by the finite powerset construction underlying the determinisation of $A$. Next, after a series of language-preserving transformations, we will obtain a transition system isomorphic to the reachable part of $[\![B']\!]$ for some $\textsc{dra}_k$ $B'$. As the last step, we extract from this deterministic transition system a syntactic definition of $B'$. This is achievable due to the invariance properties witnessed by the transition systems in the course of the transformation.

**Macro-configurations.** For simplicity, we will abuse the notation and write $c = (p, a)$ for a configuration $c = (p, \{\mathtt{x}_1 \mapsto a\})$ of $A$, where $p \in L$ and $a \in \mathbb{A} \cup \{\bot\}$. A *macro-configuration* is a (not necessarily finite) set $X$ of configurations $(p, a)$ of $A$. We use the notation $L_A(X) := \bigcup_{c \in X} L_A(c)$.

Let $\textsc{succ}_{\sigma,a}(X) := \{c' \in [\![A]\!] \mid c \xrightarrow{\sigma,a} c' \text{ for some } c \in X\}$ be the set of successors of configurations in $X$. We define a deterministic 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: $X_0 = \{(p, \bot) \mid p \in L_I\} \in \mathcal{X}$;
- $\mathcal{X}$ is closed under successor: for every $X \in \mathcal{X}$ and $(\sigma, a) \in \Sigma \times \mathbb{A}$, there is a transition $X \xrightarrow{\sigma,a} \textsc{succ}_{\sigma,a}(X)$ in $\mathcal{X}$.

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

**Claim 4.3.** $L_A(X) = L_{\mathcal{X}}(X)$ for every $X \in \mathcal{X}$. In particular $L(A) = L_{\mathcal{X}}(X_0)$.

For a macro-configuration $X$ we write $\textsc{Val}(X) := \{a \in \mathbb{A} \mid \exists p \cdot (p, a) \in X\}$ to denote the set of atoms appearing in $X$.

**Pre-states.** By assumption (2), for every macro-configuration $X \in \mathcal{X}$, $L_A(X)$ 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, $\textsc{Val}(X) \subseteq S$, which is impossible in general when $X$ is arbitrarily large while the size of $S$ is bounded (by $k$). Intuitively, in order to assure $S$-invariance we will replace $X$ by its $S$-closure $\mathrm{Cl}_S(X)$ (recall Fact 2.1).

The *least support* of a set with atoms $X$ is the least $S \subseteq \mathbb{A}$ w.r.t. set inclusion s.t. $X$ is $S$-invariant. In the case of equality atoms every set with atoms has the least support (see [12, Cor. 10.4] or [11, Thm. 6.1]). By assumption, the least support of every macro-configuration $X$ in $\mathcal{X}$ has size at most $k$.

A *pre-state* is a pair $Y = (X, S)$, where $X$ is a macro-state whose least support is $S$. Thus $X$ is $S$-invariant which, together with the fact that $S$ has size at most $k$, implies that there are only finitely many pre-states up to automorphism. We define the deterministic transition system $\mathcal{Y}$ as the smallest set of pre-states and transitions between them such that:

- $\mathcal{Y}$ contains the initial pre-state: $Y_0 = (X_0, \emptyset) \in \mathcal{Y}$;
- $\mathcal{Y}$ is closed under the closure of successor: For every $(X, S) \in \mathcal{Y}$ and $(\sigma, a) \in \Sigma \times \mathbb{A}$, the pre-state $(X', S')$ is in $\mathcal{Y}$ together with transition $(X, S) \xrightarrow{\sigma,a} (X', S')$, where $S'$ is the least support of the language $L' = (\sigma, a)^{-1} L_A(X) = L_A(\textsc{succ}_{\sigma,a}(X))$, and $X' = \mathrm{Cl}_{S'}(\textsc{succ}_{\sigma,a}(X))$.

**Example 4.4.** Suppose that $k = 3$, a successor of some macro-configuration $X$ has the shape $\textsc{succ}_{\sigma,a_1}(X) = \{(p, a_1), (q, a_1), (r, a_2), (s, a_3)\}$ and the least support $S'$ of $L'$ is $\{a_1, a_3\}$, where $a_1, a_2, a_3 \in \mathbb{A}$ are pairwise-different. Then $X' = \{(p, a_1), (q, a_1)\} \cup \{(r, a) \mid a \in \mathbb{A} \setminus \{a_1, a_3\}\} \cup \{(s, a_3)\}$.

By assumption, $L'$ is $T$-invariant for some $T \subseteq \mathbb{A}$ with $|T| \leq k$. Since $X$ is $S$-invariant, $L'$ is also $(S \cup \{a\})$-invariant. By the least support property of equality atoms, finite supports are closed under intersection, and hence $S' \subseteq (S \cup \{a\}) \cap T$, which implies $|S'| \leq k$.

By Lemma 3.7 we deduce:

**Claim 4.5** (Invariance of $\mathcal{Y}$)**.** For every two transitions $(X_1, S_1) \xrightarrow{\sigma,a_1} (X_1', S_1')$ and $(X_2, S_2) \xrightarrow{\sigma,a_2} (X_2', S_2')$ in $\mathcal{Y}$ and an automorphism $\pi$, if $\pi(X_1) = X_2$ and $\pi(S_1) = S_2$ and $\pi(a_1) = a_2$, then we have $\pi(X_1') = X_2'$ and $\pi(S_1') = S_2'$.

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

**Claim 4.6.** $L_{\mathcal{X}}(X_0) = L_{\mathcal{Y}}(Y_0)$.

**States.** We now introduce *states*, which are designed to be in one-to-one correspondence with configurations of the forthcoming $\textsc{dra}_k$ $B'$. Intuitively, a state differs from a pre-state $(X, S)$ only by allocating the values from (some superset of) $S$ into $k$ registers. Thus, while a pre-state contains a set $S$, the corresponding state contains a register assignment $\mu : \mathbb{A}_\perp^{\mathtt{X}}$ with image $S \subseteq \mu(\mathtt{X})$.

Let $\mathtt{X} = \{\mathtt{x}_1, \ldots, \mathtt{x}_k\}$ be a set of $k$ registers. A *state* is a pair $Z = (X, \mu)$, where $X$ is a macro-configuration, $\mu : \mathbb{A}_\perp^{\mathtt{X}}$ is a register assignment, and $X$ is $\mu(\mathtt{X})$-invariant. Thus every state $(X, \mu)$ determines uniquely a corresponding pre-state $\tau(X, \mu) = (X, S)$ where $S \subseteq \mu(\mathtt{X})$ is the least support of $X$.

**Example 4.7.** We continue Example 4.4. States corresponding to the pre-state $(X', S')$ feature the macro-configuration $X'$, but can have different register valuations. One of them is $(X', \mu')$, where $\mu' = \{\mathtt{x}_1 \mapsto a_1, \mathtt{x}_2 \mapsto a_3, \mathtt{x}_3 \mapsto a_1\}$.

We now define a deterministic transition system $\mathcal{Z}$. Its states are all those $(X, \mu)$ satisfying $\tau(X, \mu) \in \mathcal{Y}$, and transitions are determined as follows: $\mathcal{Z}$ contains a transition $(X, \mu) \xrightarrow{\sigma, a} (X', \mu')$ if $\mathcal{Y}$ contains the corresponding transition $\tau(X, \mu) \xrightarrow{\sigma, a} \tau(X', \mu') = (X', S')$, and $\mu' = \mu[\mathtt{Y} \mapsto a]$, where

$$\mathtt{Y} \ = \ \{\mathtt{x}_i \in \mathtt{X} \mid \mu(\mathtt{x}_i) \notin S' \text{ or } \mu(\mathtt{x}_i) = \mu(\mathtt{x}_j) \text{ for some } j > i\}. \tag{4.1}$$

The equation (4.1) defines a deterministic update policy[5] of the register assignment $\mu$ that amounts to updating with the current input atom $a$ all registers $\mathtt{x}_i$ whose value is either no longer needed (because $\mu(\mathtt{x}_i) \notin S'$), or is shared with some other register $x_j$, for $j > i$ and is thus redundant. It is easy to see that the above register update policy guarantees that $S' \subseteq \mu'(\mathtt{X}) \subseteq S' \cup \{a\}$. Using Claim 4.5 we derive:

**Claim 4.8** (Invariance of $\mathcal{Z}$)**.** For every two transitions $(X_1, \mu_1) \xrightarrow{\sigma, a_1} (X_1', \mu_1')$ and $(X_2, \mu_2) \xrightarrow{\sigma, a_2} (X_2', \mu_2')$ in $\mathcal{Z}$ and an automorphism $\pi$, if $\pi(X_1) = X_2$ and $\pi \circ \mu_1 = \mu_2$ and $\pi(a_1) = a_2$, then we have $\pi(X_1') = X_2'$ and $\pi \circ \mu_1' = \mu_2'$.

Let the initial state be $Z_0 = (X_0, \lambda \mathtt{x}.\perp)$, 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 data 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 $Z_0$.

**Orbits of states.** Recall that the action of automorphisms on macro-configurations and reset-point assignments is extended to states as $\pi(X, \mu) = (\pi(X), \pi \circ \mu)$, and that the orbit of a state $Z$ is defined as $\textsc{orbit}(Z) = \{\pi(Z) \mid \pi \in \Pi\}$.

While a state is designed to correspond to a configuration of the forthcoming $\textsc{dra}_k$ $B'$, its orbit is designed to play the role of control location of $B'$. We therefore need to prove that the set of orbits $\{\textsc{orbit}(Z) \mid Z \in \mathcal{Z}\}$ is finite and its size is bounded by $f(k, n)$.

Let $M_k$ denote the number of orbits of register valuations $\mathbb{A}_\perp^{\mathtt{X}}$, which is the same as the number of orbits of $k$-tuples $\mathbb{A}_\perp^k$. In case of equality atoms we have $M_k \leq (k+1)!$ Indeed, at the first position there are two possibilities, $\perp$ or an atom; at the second position there are at most three possibilities: $\perp$, the same atom at the one at the first position, or a fresh atom; and so on, until the last position where there are at most $k+1$ possibilities.

[5]There are in general many correct deterministic update policies, but for our purposes it suffices to define one such deterministic update policy.

Every $l$-element subset $S = \{a_1, a_2, \ldots, a_l\} \subseteq \mathbb{A}$ of atoms induces, in the case of equality atoms, exactly $l+1$ different $S$-orbits of atoms:

$$S\text{-ORBITS} \;=\; \{\text{ORBIT}_S(a) \mid a \in \mathbb{A}\} \;=\; \{\{a_1\}, \{a_2\}, \ldots, \{a_l\}, \mathbb{A} - S\}.$$

Therefore, each $S \subseteq \mathbb{A}$ of size at most $k$ induces at most $N_k := k+1$ different $S$-orbits of atoms.

Consider a state $Z = (X, \mu)$ and let $S = \mu(\mathtt{X})$. We define the characteristic function $\text{char}_Z : S\text{-ORBITS} \to \mathcal{P}(L)$ as follows:

$$\text{char}_Z(o) = \{l \in L \mid (l, a) \in X \text{ for some } a \in o\}.$$

Since $X$ is $S$-invariant, the choice of the atom $a \in o$ is irrelevant, and we conclude:

**Claim 4.10.** Every state $Z = (X, \mu)$ is uniquely determined by its register valuation $\mu$ and by the characteristic function $\text{char}_Z$.

**Claim 4.11.** The size of $\{\text{ORBIT}(Z) \mid Z \in \mathcal{Z}\}$ is at most $M_k \cdot 2^{n \cdot N_k}$.

*Proof.* We show that there are at most $M_k \cdot (2^n)^{N_k}$ different orbits of states. Consider two states $Z = (X, \mu)$ and $Z' = (X', \mu)$ and let $S = \mu(\mathtt{X})$ and $S' = \mu'(\mathtt{X})$. Suppose that the register valuations $\mu$ and $\mu'$ are in the same orbit: $\pi \circ \mu = \mu'$ for some automorphism $\pi$. Thus $\pi(S) = S'$, and moreover $\pi$ induces a bijection $\widetilde{\pi}$ between $S$-ORBITS and $S'$-ORBITS. Note that, once $S$ is fixed, there are at most $(2^n)^{N_k}$ possible characteristic functions of $Z$, and likewise for $S'$ and $Z'$. Supposing further that the characteristic functions agree, i.e., satisfy $\text{char}_Z = \text{char}_{Z'} \circ \widetilde{\pi}$, using Claim 4.10 we derive $\pi(Z) = Z'$, i.e., $Z$ and $Z'$ are in the same orbit. Therefore, since the number of orbits of register valuations $\mu, \mu'$ is at most $M_k$, and for each such orbit the number of different characteristic functions is at most $(2^n)^{N_k}$, the number of different orbits of states is bounded as required. □

For future use we observe that every state is uniquely determined by its register valuation and its orbit:

**Claim 4.12.** Let $Z = (X, \mu)$ and $Z' = (X', \mu)$ be two states in $\mathcal{Z}$ with the same register valuation. If $\pi(X) = X'$ and $\pi \circ \mu = \mu$ for some automorphism $\pi$ then $X = X'$.

*Proof.* Indeed, $X$ is $\mu(X)$-invariant and hence $\pi(X) = X$, which implies $X = X'$. □

**Construction of the** DRA. As the last step we define a $\text{DRA}_k$ $B' = (\mathtt{X}, \Sigma, L', \{o_0\}, L'_F, \Delta')$ such that the reachable part of $[\![B']\!]$ is isomorphic to $\mathcal{Z}$. Let locations $L' = \{\text{ORBIT}(Z) \mid Z \in \mathcal{Z}\}$ be the orbits of states from $\mathcal{Z}$, the initial location be the orbit $o_0$ of $Z_0$, and final locations $L'_F = \{\text{ORBIT}(Z) \mid Z \in F_{\mathcal{Z}}\}$ be orbits of final states. Let each transition $Z = (X, \mu) \xrightarrow{\sigma, a} (X', \mu') = Z'$ in $\mathcal{Z}$ induce a transition rule in $B'$

$$(o, \sigma, \psi, \mathtt{Y}, o') \in \Delta' \tag{4.2}$$

where $o = \text{ORBIT}(Z)$, $o' = \text{ORBIT}(Z')$, $\mathtt{Y} = \{\mathtt{x} \in \mathtt{X} \mid \mu'(\mathtt{x}) = a\}$, and the constraint $\psi(\mathtt{x}_1, \ldots, \mathtt{x}_k, \mathtt{y})$ defines the orbit of $(\mu(\mathtt{x}_1), \ldots, \mu(\mathtt{x}_k), a)$ (here we rely on Claim 3.1). We argue that the automaton $B'$ is deterministic:

**Claim 4.13.** Suppose that $B'$ has two transition rules $o \xrightarrow{\sigma,\psi_1,\mathtt{Y}_1} o'_1$ and $o \xrightarrow{\sigma,\psi_2,\mathtt{Y}_2} o'_2$ with the same source location $o$ and jointly satisfiable constraints ($[\![\psi_1 \wedge \psi_2]\!] \neq \emptyset$). Then the target locations are equal ($o'_1 = o'_2$), and the same registers are updated ($\mathtt{Y}_1 = \mathtt{Y}_2$).

*Proof.* Since the constraints are jointly satisfiable, both transition rules are enabled in some configuration $c = (o, \mu)$ and for some input atom $a \in \mathbb{A}$. By Claim 4.12, $c$ determines a corresponding state $Z = (X, \mu)$ with $o = \textsc{orbit}(Z)$ and, since the system $\mathcal{Z}$ is deterministic, both transition rules are induced by a common transition $(X, \mu) \xrightarrow{\sigma,a} (X', \mu')$ in $\mathcal{Z}$. This in turn implies $o_1 = o_2$ and $\mathtt{Y}_1 = \mathtt{Y}_2$, as required. ☐

**Claim 4.14.** $\mathcal{Z}$ is isomorphic to the reachable part of $[\![B']\!]$.

*Proof.* For a state $Z = (X, \mu)$, let $\iota(Z) = (\textsc{orbit}(Z), \mu)$. By Claim 4.12, the mapping $\iota(\_)$ is a bijection between $\mathcal{Z}$ and its image $\iota(\mathcal{Z}) \subseteq [\![B']\!]$. By (4.2), $\mathcal{Z}$ is isomorphic to a subsystem of the reachable part of $[\![B']\!]$. The converse inclusion follows by invariance of $\mathcal{Z}$ (Claim 4.8) and the observation that $\mathcal{Z}$ is total: for every $(\sigma_1, a_1) \dots (\sigma_n, a_n) \in \Sigma \times \mathbb{A}$, there is a sequence of transitions $(X_0, \mu_0) \xrightarrow{\sigma_1,a_1} \cdots \xrightarrow{\sigma_n,a_n} (X_n, \mu_n)$ in $\mathcal{Z}$. ☐

Claims 4.3, 4.6, 4.9, and 4.14 jointly imply $L(A) = L(B')$, which completes the proof of Lemma 4.1.

## 4.2. **Other atoms.**

The proof of Theorem 1.2 straightforwardly generalises to any relational structure of atoms $\mathbb{A}$ satisfying the following conditions:

- $\mathbb{A}$ is homogeneous [28];
- $\mathbb{A}$ preserves WQO: finite induced substructures of $\mathbb{A}$ labelled by elements of an arbitrary WQO, ordered by label-respecting embedding, are again a WQO (for details we refer the reader to [41, item (A3), Sect.5]);
- $\mathbb{A}$ is effective: it is decidable, if a given finite structure over the vocabulary of $\mathbb{A}$ is an induced substructure thereof;
- $\mathbb{A}$ has the least support property.

As an example, the structure of densely ordered atoms $\mathbb{A} = (\mathbb{R}, \leq)$ satisfies all the conditions and hence Theorem 1.2 holds for register automata over this structure of atoms.

We briefly discuss the adjustments needed. The syntax of constraints (3.1) is extended by adding atomic constraints for all relations in $\mathbb{A}$, and Claim 3.1 holds by homogeneity of $\mathbb{A}$. The decision procedure checking equivalence of a NRA$_1$ and a DRA from Theorem 3.6, invoked in the proof of Theorem 1.2, works assuming that $\mathbb{A}$ preserves WQO and is effective. The least support assumption is required in the definition of pre-states. Finally, again due to homogeneity of $\mathbb{A}$ the bounds $M_k$ and $N_k$ used in Claim 4.11 are finite (but dependent on $\mathbb{A}$).

## 5. Timed automata

The register-based model which is the closest to timed automata is based on timed atoms $(\mathbb{R}, \leq, +1)$ [13]. However, due to the syntactic and semantic restrictions which are traditionally imposed on timed automata (such as monotonicity of time, nonnegative timestamps, the special status of the initial timestamp 0, and the concrete syntax of transition constraints), the latter can be seen only as as a strict subclass of the corresponding register model. For this reason, we present our results on the deterministic membership probem in the syntax and semantics of timed automata.

**Timed words and languages.** Fix a finite alphabet $\Sigma$. Let $\mathbb{R}$ and $\mathbb{R}_{\geq 0}$ denote the reals, resp., the nonnegative reals[6]. Timed words are obtained by instantiating data words to timed atoms, and imposing additional conditions: non-negativeness and monotonicity: A *timed word* over $\Sigma$ is any sequence of the form

$$w \;=\; (\sigma_1, t_1) \ldots (\sigma_n, t_n) \;\in\; (\Sigma \times \mathbb{R}_{\geq 0})^* \tag{5.1}$$

which is *monotonic*, in the sense that the timed atoms (called *timestamps* henceforth) $t_i$'s satisfy $0 \leq t_1 \leq t_2 \leq \cdots \leq t_n$. For $w$ a timed word as in (5.1) and an increment $\delta \in \mathbb{R}_{\geq 0}$, let $w + \delta = (\sigma_1, t_1 + \delta) \ldots (\sigma_n, t_n + \delta)$ be the timed word obtained from $w$ by increasing all timestamps by $\delta$. Let $\mathbb{T}(\Sigma)$ be the set of all timed words over $\Sigma$, 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 any subset of $\mathbb{T}(\Sigma)$.

The concatenation $w \cdot v$ of two timed words $w$ and $v$ is defined only when the first timestamp 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 $\mu \in \mathbb{R}_{\geq 0}^{\mathtt{X}}$ 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 $\mu$ satisfies a constraint $\varphi$, written $\mu \models \varphi$, if interpreting each clock $\mathtt{x}_i$ by $\mu(\mathtt{x}_i)$ makes $\varphi$ a tautology. An $k, m$*-region* is a non-empty set of valuations $[\![\varphi]\!]$ satisfied by a constraint $\varphi$ 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 $(2, 5)$.

[6]Equivalently, the rationals $\mathbb{Q}$ could be considered in place of reals.

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

$$(p, \sigma, \varphi, \mathtt{Y}, q) \tag{5.2}$$

with $p, q \in L$ control locations, $\sigma \in \Sigma$, $\varphi$ 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, $\text{NTA}_k$ 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 resets some clock ($\mathtt{Y} \neq \emptyset$ in (5.2)), and *greedily resetting* if, for every clock $\mathtt{x}$, whenever $\varphi$ implies that $\mathtt{x}$ belongs to $\{0, \ldots, m\} \cup (m, \infty)$, then $\mathtt{x} \in \mathtt{Y}$. Intuitively, a greedily reseting automaton resets every clock whose value is either an integer, or exceeds the maximal constant $m$.

**Reset-point semantics.** We introduce a semantics based on reset points instead of clock valuations. A *reset-point assignment* is a function $\mu \in \mathbb{R}_{\geq 0}^{\mathtt{X}}$ storing, for each clock $\mathtt{x} \in \mathtt{X}$, the timestamp $\mu(\mathtt{x})$ when $\mathtt{x}$ was last reset. Reset-point assignments and clock valuations have the same type $\mathbb{R}_{\geq 0}^{\mathtt{X}}$, however we find it technically more convenient to work with reset points than with clock valuations. The reset-point semantics has already appeared in the literature on timed automata [30] and it is the foundation of the related model of timed-register automata [13].

A *configuration* of an NTA $A$ is a tuple $(p, \mu, t_0)$ consisting of a control location $p \in L$, 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 $\mu$ 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 $\mu$ on $\mathtt{X} \setminus \mathtt{Y}$. A reset-point assignment $\mu$ together with $t_0$ induces the clock valuation $t_0 - \mu$ defined as $(t_0 - \mu)(\mathtt{x}) = t_0 - \mu(\mathtt{x})$ for all clocks $\mathtt{x} \in \mathtt{X}$.

Every transition rule (5.2) induces a *transition* between configurations

$$(p, \mu, t_0) \xrightarrow{\sigma, t} (q, \nu, t)$$

labelled by $(\sigma, t) \in \Sigma \times \mathbb{R}_{\geq 0}$ 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 (5.1) *starting* in configuration $(p, \mu, t_0)$ and *ending* in configuration $(q, \nu, t_n)$ is a path $\rho$ in $\llbracket A \rrbracket$ of the form $\rho = (p, \mu, t_0) \xrightarrow{\sigma_1, t_1} \ldots \xrightarrow{\sigma_n, t_n} (q, \nu, t_n)$. The run $\rho$ is accepting if its last configuration satisfies $(q, \nu, t_n) \in F$. The language *recognised* by configuration $(p, \mu, t_0)$ 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 $L_{[\![A]\!]}(p, \mu, t_0) \subseteq \mathbb{T}_{\geq t_0}(\Sigma)$. We write $L_A(c)$ instead of $L_{[\![A]\!]}(c)$. 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 NTA$_{\_,m}$, every reachable configuration $(p, \mu, t_0)$ satisfies $t_0 \in \mu(\mathtt{X})$, where $\mu(\mathtt{X}) = \{\mu(\mathtt{x}) \mid \mathtt{x} \in \mathtt{X}\}$; in a greedily resetting one, 1) $(p, \mu, t_0)$ 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 $(p, \sigma, \varphi, \mathtt{Y}, q), (p, \sigma', \varphi', \mathtt{Y}', q') \in \Delta$, if $\sigma = \sigma'$ and the two transition constraints are jointly saatisfiabe $[\![\varphi \wedge \varphi']\!] \neq \emptyset$, 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 $p \in L$ and $\sigma \in \Sigma$, the sets defined by clock constraints $\{[\![\varphi]\!] \mid \exists \mathtt{Y}, q \cdot (p, \sigma, \varphi, \mathtt{Y}, q) \in \Delta\}$ are a partition of $\mathbb{R}^{\mathtt{X}}_{\geq 0}$. 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$_k$, DTA$_{\_,m}$, and 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 5.1.** This is a timed analog of Example 3.4. Let $\Sigma = \{\sigma\}$ 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 $(\sigma, t_1) \cdots (\sigma, t_n)$ where $t_n - t_i = 1$ for some $1 \leq i < n$. The language $L$ is recognised by the NTA$_1$ $A = (\mathtt{X}, \Sigma, L, L_I, L_F, \Delta)$ with a single clock $\mathtt{X} = \{\mathtt{x}\}$ and three locations $L = \{p, q, r\}$, of which $L_I = \{p\}$ is initial and $L_F = \{r\}$ is final, and transition rules

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

Intuitively, in $p$ the automaton waits until it guesses that the next input will be $(\sigma, t_i)$, at which point it moves to $q$ by resetting the clock. From $q$, the automaton can accept by going to $r$ only if exactly one time unit elapsed since $(\sigma, t_i)$ 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.** Decision problems for NTA we are interested in are analogous to ones for NRA, but additionally a bound on the maximal constant appearing in a DTA may be specified as a parameter. 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, $\text{DTA}_k$ (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 5.1 shows that there are NTA languages which cannot be accepted by any DTA. Moreover, similarly as in case of NRA, there is no computable bound for the number of clocks $k$ which suffice to recognise a $\text{NTA}_1$ language by a $\text{DTA}_k$ (when such a number exists).

## 6. Invariance of timed automata

A fundamental tool used below is invariance properties of timed languages recognised by NTA with respect to timed automorphisms. In this section we establish these properties, as extension of analogous properties of register automata. We also prove a timed analog of the least support property, and relate regions to orbits of configurations.

Recall that timed automorphisms are monotonic bijections $\mathbb{R} \to \mathbb{R}$ that preserve integer differences. A timed automorphism $\pi$ acts on input letters $\Sigma$ as the identity $\pi(a) = a$, and is extended point-wise to timed words $\pi((\sigma_1, t_1) \dots (\sigma_n, t_n)) = (\sigma_0, \pi(t_1)) \dots (\sigma_n, \pi(t_n))$, configurations $\pi(p, \mu, t_0) = (p, \pi \circ \mu, \pi(t_0))$, transitions $\pi(c \xrightarrow{\sigma,t} c') = \pi(c) \xrightarrow{\sigma,\pi(t)} \pi(c')$, and sets $X$ thereof $\pi(X) = \{\pi(x) \mid x \in X\}$.

**Remark 6.1.** In considerations about timed automata we restrict to nonnegative reals, while a timed automorphism $\pi$ can in general take a nonnegative real $t \geq 0$ to a negative one. In the sequel whenever we write $\pi(x)$, for $x$ being any object like a timestamp, a configuration, a timed word, etc., we always implicitly assume that $\pi$ is well-defined on $x$, i.e., yields a timestamp, a configuration, a timed word, etc. In other words, for invariance properties we restrict to those timed automorphisms that preserve nonnegativeness of all the involved timestamps.

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 $\Pi_S$ 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 6.2.** The timed transition system $[\![A]\!]$ is invariant.

*Proof.* Suppose $c = (p, \mu, t_0) \xrightarrow{\sigma,t} (p', \mu', t) = c'$ due to some transition rule of $A$ whose clock constraint $\varphi$ compares values of clocks $\mathtt{x}$, i.e., the differences $t - \mu(\mathtt{x})$, to integers. Since a timed automorphism $\pi$ 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 $(p, \pi \circ \mu, \pi(t_0)) \xrightarrow{\sigma,\pi(t)} (p, \pi \circ \mu', \pi(t)) = \pi(c')$. □

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

*Proof.* Consider a timed automorphism $\pi$ and an accepting run of $A$ over a timed word $w = (\sigma_1, t_1) \ldots (\sigma_n, t_n) \in \mathbb{T}_{\geq t_0}(\Sigma)$ starting in $c = (p, \mu, t_0)$:

$$(p, \mu, t_0) \xrightarrow{\sigma_1, t_1} \cdots \xrightarrow{\sigma_n, t_n} (q, \nu, t_n),$$

After $\sigma_i$ 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 $\pi$ preserves integer differences, by executing the same sequence of transition rules we obtain the run over $\pi(w)$ starting in $\pi(c) = (p, \pi \circ \mu, \pi(t_0))$:

$$(p, \pi\circ\mu, \pi(t_0)) \xrightarrow{\sigma_1, \pi(t_1)} \cdots \xrightarrow{\sigma_n, \pi(t_n)} (q, \pi\circ\nu, \pi(t_n)),$$

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 $\pi^{-1}$ 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. □

**Fact 6.4** (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 automorphism $\pi$ 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 6.3, 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. □

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 (c.f. the least support property of, e.g., equality atoms).

**Lemma 6.5.** *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 $\pi$. We need to show

$$w \in L \quad \text{iff} \quad w' \in L,$$

which follows immediately by the following claim:

**Claim 6.6.** Every $(F \cap F')$-timed automorphism $\pi$ decomposes into $\pi = \pi_n \circ \cdots \circ \pi_1$, where each $\pi_i$ is either an $F$- or an $F'$-timed automorphism.

Composition of times automorphisms makes $\Pi$ into a group. In short terms, Claim 6.6 states that $\Pi_{F \cap F'} \subseteq \Pi_F + \Pi_{F'}$, where $\Pi_F + \Pi_{F'}$ is the smallest subgroup of $\Pi$ including both $\Pi_F$ and $\Pi_{F'}$. We state below in Claim 6.8 a stronger fact which implies Claim 6.6; its proof is based on the proof of Theorem 10.3 in [12]. An important ingredient of the proof of Claim 6.8 is the following fact where, instead of dealing with decomposition of $\pi$, we analyse the individual orbit of $F \setminus F'$, in the special case when both $F \setminus F'$ and $F' \setminus F$ are singleton sets:

**Claim 6.7.** Let $F, F' \subseteq [0, 1)$ be finite sets s.t. $F \setminus F' = \{t\}$ and $F' \setminus F = \{t'\}$. For every $(F \cap F')$-timed automorphism $\pi$ we have $\pi(t) = (\pi_n \circ \cdots \circ \pi_1)(t)$, for some $\pi_1, \ldots, \pi_n$, each of which is either an $F$- or an $F'$-timed automorphism (i.e., belongs to $\Pi_F + \Pi_{F'}$).

*Proof of Claim 6.7.* We split the proof 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 $(L_F \cap F')$-timed automorphism $\pi$; 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

- $\pi_1$ is some $F'$-timed automorphism that acts as identity on $[t', l+1]$ and s.t. $t < \pi_1(t) < t'$,
- $\pi_2$ 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 $\pi$; 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 $\pi_1$ 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 $\pi_2$ 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. □

**Claim 6.8.** Let $F, F' \subseteq [0, 1)$ be finite sets and let $G \subseteq \Pi$ be a subgroup of $\Pi$. If $\Pi_F \subseteq G$ and $\Pi_{F'} \subseteq G$ then $\Pi_{F \cap F'} \subseteq G$.

*Proof of Claim 6.8.* The proof is by induction on the size of the (finite) set $F \cup F'$. If $F \subseteq F'$ or $F' \subseteq F$, then the conclusion follows trivially. Otherwise, consider any $t \in F \setminus F'$ and $t' \in F' \setminus F$; obviously $t \neq t'$. Define $E = (F \cup F') \setminus \{t, t'\}$. We have $F \subseteq E \cup \{t\}$ and $F' \subseteq E \cup \{t'\}$ hence

$$\Pi_{E\cup\{t\}} \subseteq \Pi_F \subseteq G \qquad \Pi_{E\cup\{t'\}} \subseteq \Pi_{F'} \subseteq G.$$

We shall now prove that $\Pi_E \subseteq G$. To this end, consider any $\pi \in \Pi_E$. By Claim 6.7, there exists a permutation

$$\tau = \pi_n \circ \cdots \circ \pi_1 \in \Pi_{E\cup\{t\}} + \Pi_{E\cup\{t'\}}$$

such that $\pi(t) = \tau(t)$. In other words, each of $\pi_1, \ldots, \pi_n$ is either a $(E \cup \{t\})$- or a $(E \cup \{t'\})$-timed automorphism. Since $\Pi_{E\cup\{t\}} \subseteq G$ and $\Pi_{E\cup\{t'\}} \subseteq G$, all $\pi_i \in G$, hence also $\tau \in G$.

On the other hand, clearly $\Pi_{E\cup\{t\}} \subseteq \Pi_E$ and $\Pi_{E\cup\{t'\}} \subseteq \Pi_E$, so all $\pi_i \in \Pi_E$, therefore $\tau \in \Pi_E$. As a result, $\pi^{-1} \circ \tau \in \Pi_E$. Since $(\pi^{-1} \circ \tau)(t) = t$, we obtain $\pi^{-1} \circ \tau \in \Pi_{E\cup\{t\}}$, therefore $\pi^{-1} \circ \tau \in G$. Together with $\tau \in G$ proved above, this gives $\pi \in G$. Thus we have proved $\Pi_E \subseteq G$.

It is now easy to show that $\Pi_{F\cap F'} \subseteq G$. Indeed, $|F \cup E| = |F \cup F'| - 1$, so by the inductive assumption for $F$ and $E$, we have $\Pi_{F\setminus\{t\}} \subseteq G$ (note that $F \setminus \{t\} = F \cap E$). Further, $|(F \setminus \{t\}) \cup F'| = |F \cup F'| - 1$, so $\Pi_{F\cap F'} \subseteq G$ (note that $(F \setminus \{t\}) \cap F' = F \cap F'$), as required. □

Claim 6.8 immediately implies Claim 6.6 by taking $G = \Pi_F + \Pi_{F'}$. Lemma 6.5 is thus proved. □

Finally, recall $S$-orbits and orbits of elements, as defined abstractly in Section 2. Every bounded region corresponds to an orbit of configurations. Hence, in case of greedily resetting NTA where all reachable regions are bounded, orbits of reachable configurations are in bijective correspondence with reachable regions:

**Fact 6.9.** Assume $A$ is a greedily resetting $\text{NTA}_{k,m}$. Two reachable configurations $(p, \mu, t_0)$ and $(p, \mu', t_0')$ 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.

## 7. Decidability of $\text{DTA}_k$ and $\text{DTA}_{k,m}$ membership for $\text{NTA}_1$

In this section we prove our main decidability result for timed automata, which we now recall.

**Theorem 1.1.** *The* $\text{DTA}_k$ *and* $\text{DTA}_{k,m}$ *membership problems are decidable for* $\text{NTA}_1$ *languages.*

Both results are shown using the following key characterisation of those $\text{NTA}_1$ languages which are also $\text{DTA}_k$ languages.

**Lemma 7.1.** *Let $A$ be a* 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* 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).*

As in case of register automata, this characterisation provides a bound on the number of control locations of a DTA$_k$ equivalent to a given NTA$_1$ (if any exists).

The proof of Theorem 1.1 builds on Lemma 7.1 and on the following fact:

**Lemma 7.2.** *The* DTA$_k$ *and* 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 DTA$_{k,m}$ separability problems are decidable for NTA [21, 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., 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 DTA$_{k,m}$ languages. □

*Proof of Theorem 1.1.* We solve both problems in essentially the same way. Given a NTA$_{1,m}$ $A$, the decision procedure enumerates all always resetting DTA$_{k+1,m}$ $B$ with at most $f(k, m, n)$ locations and checks whether $L(A) = L(B)$ (which is decidable by [48, Theorem 17]). If no such DTA$_{k+1}$ $B$ is found, the $L(A)$ is not an always resetting DTA$_{k+1}$ language, due to Lemma 7.1, 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 a DTA$_k$ membership (resp. DTA$_{k,m}$ membership) test is performed on $B$, which is decidable due to Lemma 7.2. □

**Remark 7.3** (Complexity)**.** The decision procedure for NTA$_1$ invokes the HyperAckermann subroutine of [48] to check equivalence between a NTA$_1$ and a candidate DTA. This is in a sense unavoidable, since we show in Theorem 1.5 that DTA$_k$ and DTA$_{k,m}$ membership problems are HyperAckermann-hard for NTA$_1$.

In the rest of this section we present the proof of Lemma 7.1. The proof is a suitable extension and refinement of the argument used in case of register automata in Section 4.

## 7.1. Proof of Lemma 7.1.

Let us fix a NTA$_{1,m}$ $A = (\Sigma, L, \{\mathtt{x}\}, L_I, L_F, \Delta)$, 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⇒1 follows by definition. For the implication 1⇒2 suppose, by assumption, $L(A) = L(B)$ for a total always resetting $\text{DTA}_k$ $B$. Every left quotient $w^{-1}L(A)$ equals $L_B(c)$ for some configuration $c$, hence Point 2 follows by Fact 6.4. Here we use the fact that $B$ is always resetting in order to apply the second part of Fact 6.4; 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⇒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 achieved thanks to the invariance properties of the transition systems in the course of the transformation.

**Macro-configurations.** Configurations of the $\text{NTA}_1$ $A$ 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 $\text{NOW}(X) = t_0$. We use the notation $L_A(X) := \bigcup_{c \in X} L_A(c)$. Let $\text{SUCC}_{\sigma,t}(X) := \{c' \in [\![A]\!] \mid c \xrightarrow{\sigma,t} c' \text{ for some } c \in X\}$ 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: $X_0 = \{(p, 0, 0) \mid p \in L_I\} \in \mathcal{X}$;
- $\mathcal{X}$ is closed under successor: for every $X \in \mathcal{X}$ and $(\sigma, t) \in \Sigma \times \mathbb{R}_{\geq 0}$, there is a transition $X \xrightarrow{\sigma,t} \text{SUCC}_{a,t}(X)$ in $\mathcal{X}$.

Due to the fact that $[\![A]\!]$ is finitely branching, i.e. $\text{SUCC}_{\sigma,t}(\{c\})$ is finite for every fixed $(\sigma, t)$, 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 7.4.** $L_A(X) = L_{\mathcal{X}}(X)$ for every $X \in \mathcal{X}$. In particular $L(A) = L_{\mathcal{X}}(X_0)$.

For a macro-configuration $X$ we write $\text{VAL}(X) := \{u \mid (p, u, \text{NOW}(X)) \in X\} \cup \{\text{NOW}(X)\}$ to denote the reals appearing in $X$. Since $A$ is greedily resetting, every macro-configuration $X \in \mathcal{X}$ satisfies $\text{VAL}(X) \subseteq (\text{NOW}(X) - m, \text{NOW}(X)]$. 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, $\mathsf{fract}(\text{VAL}(X)) \subseteq \mathsf{fract}(S)$, 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 $\mathrm{Cl}_S(X)$ (recall Fact 2.1).

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 the 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 $(\sigma, t) \in \Sigma \times \mathbb{R}_{\geq 0}$, there is a transition $(X, S) \xrightarrow{\sigma,t} (X', S')$, where $S'$ is the least, with respect to set inclusion, subset of $S \cup \{t\}$ containing $t$ such that the language $L' = (\sigma, t)^{-1} L_A(X) = L_A(\textsc{succ}_{\sigma,t}(X))$ is $S'$-invariant, and $X' = \mathrm{Cl}_{S'}(\textsc{succ}_{\sigma,t}(X))$.

Observe that the least such fraction-independent subset $S'$ exists due to the following facts: Since $X$ is $S$-invariant, due to Fact 6.3 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 6.5, 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 $k$.

**Example 7.5.** Suppose $k = 3$, $m = 2$, $\textsc{succ}_{\sigma,t}(X) = \{(p, 3.7, 5), (q, 3.9, 5), (r, 4.2, 5)\}$ 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* (as defined below) is $(X', \mu')$, where $\mu' = \{\mathtt{x}_1 \mapsto 3.7, \mathtt{x}_2 \mapsto 4.2, \mathtt{x}_3 \mapsto 5\}$.

By Fact 6.3, we deduce:

**Claim 7.6** (Invariance of $\mathcal{Y}$)**.** For every two transitions $(X_1, S_1) \xrightarrow{\sigma,t_1} (X_1', S_1')$ and $(X_2, S_2) \xrightarrow{\sigma,t_2} (X_2', S_2')$ in $\mathcal{Y}$ and a timed permutation $\pi$, 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 $X_i' = \mathrm{Cl}_{S_i'}(\widetilde{X}_i)$. By invariance of $[\![A]\!]$ (Fact 6.2) and invariance of semantics (Fact 6.3) 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 $\pi(X_1') = X_2'$. □

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

**Claim 7.7.** $L_{\mathcal{X}}(X_0) = L_{\mathcal{Y}}(Y_0)$.

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 $\text{DTA}_k$ $B'$. 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 reset-point 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 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 $\rho(Z) = (X, S)$ with $S = \mu(\mathtt{X})$. We define the deterministic timed transition system $\mathcal{Z}$ consisting of those states $Z$ s.t. $\rho(Z) \in \mathcal{Y}$, and of transitions determined as follows: $(X, \mu) \xrightarrow{\sigma, t} (X', \mu')$ if the corresponding pre-state has a transition $(X, S) \xrightarrow{\sigma, t} (X', S')$ in $\mathcal{Y}$, where $S = \mu(\mathtt{X})$, and

$$\mu'(\mathtt{x}_i) := \begin{cases} t & \text{if } \mu(\mathtt{x}_i) \notin S' \text{ or } \mu(\mathtt{x}_i) = \mu(\mathtt{x}_j) \text{ for some } j > i \\ \mu(\mathtt{x}_i) & \text{otherwise.} \end{cases} \tag{7.1}$$

Intuitively, the equation (7.1) defines a deterministic update of the reset-point assignment $\mu$ 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 $\text{DTA}_k$ $B'$ will be always resetting. Using Claim 7.6 we derive:

**Claim 7.8** (Invariance of $\mathcal{Z}$)**.** For every two transitions $(X_1, \mu_1) \xrightarrow{\sigma, t_1} (X'_1, \mu'_1)$ and $(X_2, \mu_2) \xrightarrow{\sigma, t_2} (X'_2, \mu'_2)$ in $\mathcal{Z}$ and a timed permutation $\pi$, 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 7.6 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 (7.1) is invariant:

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

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

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 the 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 7.9.** $L_{\mathcal{Y}}(Y_0) = L_{\mathcal{Z}}(Z_0)$.

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 $\text{DTA}_k$ $B'$ to be always resetting.

**Orbits of states.** While a state is designed to correspond to a configuration of the forthcoming $\text{DTA}_k$ $B'$, 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 $\{\text{ORBIT}(Z) \mid Z \in \mathcal{Z}\}$ is finite and its size is bounded by $f(k, m, n)$. We start by deducing an analogue of Fact 6.9:

**Claim 7.10.** For two states $Z = (X, \mu)$ and $Z' = (X', \mu')$ in $\mathcal{Z}$, their reset-point 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 $\text{NOW}(X) - \mu$ and $\text{NOW}(X') - \mu'$ belong to the same $k, m$-region.

(In passing note that, since in every state $(X, \mu)$ 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 reset-point 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 $\text{ORBIT}(Z) = \{\pi(Z) \mid \pi \in \Pi\}$.

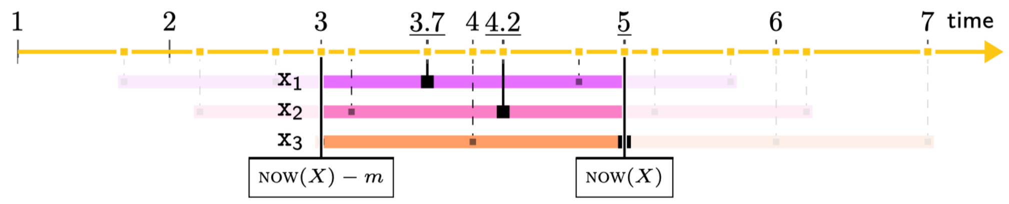
**Claim 7.11.** The number of orbits of states in $\mathcal{Z}$ is bounded by $f(k, m, n)$.

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

**Fact 7.12.** For every $u \in \mathbb{R}_{\geq 0}$ and $S \subseteq \mathbb{R}_{\geq 0}$, the $S$-orbit[7] $\text{ORBIT}_S(u)$ is either the singleton $\{u\}$ (when $u \in S$) or an open interval with end-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 $\text{ORBIT}_{\mu(\mathtt{X})}(u)$ 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 7.13.** Continuing Example 7.5, 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

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

consecutive $S$-orbits $\textsc{orbit}_{\mu(\mathtt{X})}(u)$ 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 $(O,\mu)$, where

$$O = \{(o_1, P_1), \ldots, (o_{2l+1}, P_{2l+1})\} \tag{7.2}$$

assigns to each orbit $o_i$ the set of locations $P_i = \{p \mid (p,u,t_0) \in X \text{ for some } u \in o_i\} \subseteq L$, (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 (7.2) and the reset-point assignment $\mu$.

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

$$P_1,\ P_2,\ \ldots,\ P_{2km+1} \tag{7.3}$$

induced by the assignment $O$ as in (7.2). Therefore, the number of orbits is bounded by the number of orbits of $\mu$ (which is bounded, due to Claim 7.10, by $\mathsf{Reg}(k,m)$) times the number of different sequences of the form (7.3) (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$_k$ $B' = (\Sigma, L', \mathtt{X}, \{o_0\}, L'_F, \Delta')$ such that the reachable part of $[\![B']\!]$ is isomorphic to $\mathcal{Z}$. Let locations $L' = \{\textsc{orbit}(Z) \mid Z \in \mathcal{Z}\}$ be orbits of states from $\mathcal{Z}$, the initial location be the orbit $o_0$ of $Z_0$, and final locations $L'_F = \{\textsc{orbit}(Z) \mid Z \in F_{\mathcal{Z}}\}$ be orbits of final states. A transition $Z=(X,\mu) \xrightarrow{\sigma,t} (X',\mu') = Z'$ in $\mathcal{Z}$ induces a transition rule in $B'$

$$(o, a, \psi, \mathtt{Y}, o') \in \Delta' \tag{7.4}$$

whenever $o = \textsc{orbit}(Z)$, $o' = \textsc{orbit}(Z')$, $\psi$ 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 $o$, $\sigma$ and $\psi$ uniquely determine $\mathtt{Y}$ and $o'$:

**Claim 7.14.** Suppose that two transitions $(X_1,\mu_1) \xrightarrow{\sigma,t_1} (X'_1,\mu'_1)$ and $(X_2,\mu_2) \xrightarrow{\sigma,t_2} (X'_2,\mu'_2)$ in $\mathcal{Z}$ induce transition rules $(o,\sigma,\psi,\mathtt{Y}_1,o'_1), (o,\sigma,\psi,\mathtt{Y}_2,o'_2) \in \Delta'$ with the same source location $o$ and constraint $\psi$, i.e,

$$t_1 - \mu_1 \in [\![\psi]\!] \qquad t_2 - \mu_2 \in [\![\psi]\!]\,. \tag{7.5}$$

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

(Notice that we only consider two transition rules with the same constraint $\psi$, instead of two different jointly satisfiable constraints $\psi, \psi'$ as in the definition of deterministic timed automata, due to the fact that each constraint of $B'$ is a single $k,m$-region.)

*Proof.* We use the invariance of semantics of $A$ and Claim 7.8. Let $o = \textsc{orbit}(X_1,\mu_1) = \textsc{orbit}(X_2,\mu_2)$. Thus there is a timed automorphism $\pi$ such that

$$X_2 = \pi(X_1) \qquad \mu_2 = \pi{\circ}\mu_1. \tag{7.6}$$

It suffices to show that there is a (possibly different) timed permutation $\pi'$ satisfying the following equalities:

$$t_2 = \pi'(t_1) \quad \{i \mid \mu'_1(\mathtt{x}_i) = t_1\} = \{i \mid \mu'_2(\mathtt{x}_i) = t_2\} \quad \mu'_2 = \pi' \circ \mu'_1 \quad X'_2 = \pi'(X'_1). \tag{7.7}$$

We now rely the fact that both $t_{01} = \textsc{now}(X_1) \in \mu_1(\mathtt{X})$ and $t_{02} = \textsc{now}(X_2) \in \mu_2(\mathtt{X})$ are assigned to the same clock due to the second equality in (7.6): $t_{01} = \mu_1(\mathtt{x}_i)$ 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 and easier since all clock are reset due to greedy resetting), which implies $t_2 - t_{02} \leq m$ due to (7.5). In this case we may assume w.l.o.g., due to (7.5) and the equalities (7.6), that $\pi$ is chosen so that $\pi(t_1) = t_2$. We thus take $\pi' = \pi$ for proving the equalities (7.7). Being done with the first equality, we observe that the last two equalities in (7.7) hold due to the invariance of $\mathcal{Z}$ (c.f. Claim 7.8). The remaining second equality in (7.7) is a consequence of the third one. □

**Claim 7.15.** Let $Z = (X, \mu)$ and $Z' = (X', \mu)$ be two states in $\mathcal{Z}$ with the same reset-point assignment. If $\pi(X) = X'$ and $\pi \circ \mu = \mu$ for some timed automorphism $\pi$ then $X = X'$.

**Claim 7.16.** $\mathcal{Z}$ is isomorphic to the reachable part of $[\![B']\!]$.

*Proof.* For a state $Z = (X, \mu)$, let $\iota(Z) = (o, \mu, t)$, where $o = \textsc{orbit}(Z)$ and $t = \textsc{now}(X)$. By Claim 7.15, the mapping $\iota(\_)$ is a bijection between $\mathcal{Z}$ and its image $\iota(\mathcal{Z}) \subseteq [\![B']\!]$. By (7.4), $\mathcal{Z}$ 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 $(\sigma_1, t_1) \ldots (\sigma_n, t_n) \in \mathbb{T}(\Sigma)$, there is a sequence of transitions $(X_0, \mu_0) \xrightarrow{\sigma_1, t_1} \cdots \xrightarrow{\sigma_n, t_n}$ in $\mathcal{Z}$. □

Claims 7.4, 7.7, 7.9, and 7.16 imply $L(A) = L(B')$, which completes the proof of Lemma 7.1.

## 8. Undecidability and hardness

In this section we complete the decidability and complexity landscape for the deterministic membership problem by providing matching undecidability and complexity hardness results, both for register and timed automata.

### 8.1. Lossy counter machines.

Our undecidability and hardness results will be obtained by reducing from the finiteness problem for lossy counter machines, which is known to be undecidable. A *k-counters lossy counter machine* ($k$-$\mathsf{LCM}$) 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 $\Delta$ is a finite set of instructions of the form $(p, \mathsf{op}, q)$, where $\mathsf{op}$ is one of $c$++, $c$--, and $c \overset{?}{=} 0$. A configuration of an $\mathsf{LCM}$ $M$ 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 $\mathsf{LCM}$ $M$ 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 $\delta = (p, \mathsf{op}, q) \in \Delta$, whenever

(1) $\mathsf{op} = c$++ and $v \leq u[c \mapsto u(c) + 1]$, or

(2) $\mathsf{op} = c\,\texttt{--}$ and $v \leq u[c \mapsto u(c) - 1]$, or
(3) $\mathsf{op} = c \overset{?}{=} 0$ and $u(c) = 0$ and $v \leq u$.

We omit $\delta$ in $\xrightarrow{\delta}$ when it is irrelevant, and write $\rightarrow^*$ for the transitive closure of $\rightarrow$. The *finiteness problem* (a.k.a. space boundedness) for an LCM $M$ amounts to decide whether the reachability set $\mathrm{Reach}(M) = \{(p, u) \mid (q_0, u_0) \rightarrow^* (p, u)\}$ is finite, where $u_0$ is the constantly 0 counter valuation.

**Theorem 8.1** ([44, Theorem 13])**.** *The 4-LCM finiteness problem is undecidable.*

8.2. **Register automata.**

8.2.1. *Undecidability of* DRA *membership for* NRA$_1$. We show that it is undecidable whether a NRA$_1$ language can be recognised by some DRA. In the following, it will be useful to be able to refer to projection of a data word on the finite component of the alphabet. To this end, for a data word $w = (\sigma_0, a_0) \cdots (\sigma_n, a_n) \in (\Sigma \times \mathbb{A})^*$, let $\mathsf{undata}(w) = \sigma_0 \cdots \sigma_n \in \Sigma^*$ be the word obtained by removing the atom component. As already announced, we reduce from the finiteness problem for lossy counter machines. Consider a lossy counter machine $M = (C, Q, q_0, \Delta)$ with 4 counters $C = \{c_1, c_2, c_3, c_4\}$. We use the following encoding of LCM runs as data words over the alphabet $\Sigma = Q \cup \Delta \cup C$ comprising the control locations, transitions, and counters of $M$. (A similar encoding has been used in the proof of Theorem 5.2 in [24] to show that the universality problem for NRA$_1$ is not primitive recursive.) We encode a counter valuation $u \in \mathbb{N}^C$ as the word over $C \subseteq \Sigma$

$$\mathsf{enc}(u) \;=\; \underbrace{c_1 c_1 \cdots c_1}_{u(c_1)\text{ letters}}\; \underbrace{c_2 c_2 \cdots c_2}_{u(c_2)\text{ letters}}\; \underbrace{c_3 c_3 \cdots c_3}_{u(c_3)\text{ letters}}\; \underbrace{c_4 c_4 \cdots c_4}_{u(c_4)\text{ letters}} \in \{c_1\}^*\{c_2\}^*\{c_3\}^*\{c_4\}^*. \tag{8.1}$$

Consider a LCM run

$$\pi \;=\; (p_0, u_0) \xrightarrow{\delta_1} (p_1, u_1) \xrightarrow{\delta_2} \cdots \xrightarrow{\delta_n} (p_n, u_n).$$

The set $\mathsf{Enc}(\pi) \subseteq (\Sigma \times \mathbb{A})^*$ of *reversal-encodings* of $\pi$ contains all data words $w \in (\Sigma \times \mathbb{A})^*$ satisfying the following conditions:

(E1) the finite part of $w$ is of the form

$$\mathsf{undata}(w) = p_n \delta_n \mathsf{enc}(u_n) \quad \cdots \quad p_1 \delta_1 \mathsf{enc}(u_1) \quad p_0 \mathsf{enc}(u_0) \in \Sigma^*; \tag{8.2}$$

(E2) all atoms appearing in a single $\mathsf{enc}(u_i)$ block are distinct;
(E3) for every transition $\delta_i = (p_{i-1}, \mathsf{op}_i, p_i)$ and for every counter $c_j \in \{c_1, c_2, c_3, c_4\}$:
 (E3.1) if $\mathsf{op}_i = c_j\,\texttt{++}$ increments counter $c_j$, then (recalling that the lossy semantics amounts to $u_i(c_j) - 1 \leq u_{i-1}(c_j)$) we require that for each occurrence of $c_j$ in $\mathsf{enc}(u_i)$ there is a matching occurrence of $c_j$ in $\mathsf{enc}(u_{i-1})$ with the same atom, with the exception of the *last* occurrence of $c_j$ in $\mathsf{enc}(u_i)$;

(E3.2) if $\mathsf{op}_i = c_j$ -- decrements counter $c_j$, then (recalling that the lossy semantics amounts to $u_i(c_j) + 1 \leq u_{i-1}(c_j)$) we require that for each occurrence of $c_j$ in $\mathsf{enc}(u_i)$ there is a matching occurrence of $c_j$ in $\mathsf{enc}(u_{i-1})$ with the same atom, which is *not the last* occurrence of $c_j$ in $\mathsf{enc}(u_{i-1})$.

(E3.3) if $\mathsf{op}_i = c_j \overset{?}{=} 0$ tests whether counter $c_j$ is zero, then $u_{i-1}(c_j) = u_i(c_j) = 0$, and thus we require that neither $\mathsf{enc}(u_i)$ nor $\mathsf{enc}(u_{i-1})$ contain any occurrence of $c_j$;

(E3.4) otherwise the operation $\mathsf{op}_i$ does not modify counter $c_j$, i.e., $u_i(c_j) \leq u_{i-1}(c_j)$, and we require that for each occurrence of $c_j$ in $\mathsf{enc}(u_i)$ there is a matching occurrence of $c_j$ in $\mathsf{enc}(u_{i-1})$ with the same atom.

The intuition underlying the first two items in condition (E3) is that the effect of an increment or decrement of $c_j$ is encoded by creating or removing *the last* occurrence of $c_j$ in a block.

Let $L = \bigcup_{\pi \text{ a run of } M} \mathsf{Enc}(\pi)$ be the set of all reversal-encodings of runs of $M$. Under this encoding, we can build a NRA$_1$ $A$ recognising the *complement* of $L$. Indeed, $A$ can determine bad encodings $w \notin L$ by guessing one of finitely many reasons for this to occur:

(F1) the projection $\mathsf{undata}(\mathsf{enc}(\pi))$ to the finite alphabet $\Sigma$ is not a word in the regular language $(Q\Delta\{c_1\}^*\{c_2\}^*\{c_3\}^*\{c_4\}^*)\{q_0\}$ (notice that $\mathsf{enc}(u_0)$ is the empty string, since the initial valuation $u_0$ assigns 0 to every counter), or there is a transition $\delta_i = (p, \_, q)$ s.t. either the source is incorrect $p \neq p_{i-1}$ or the destination is incorrect $q \neq p_i$. This is even a regular language of finite words over $\Sigma$ (i.e., without atoms). Thus in the remaining cases below we can assume that the finite part of $w$ is of the form as in (8.2);

(F2) there is a block $\mathsf{enc}(u_i)$ containing the same atom twice;

(F3) there is a transition $\delta_i = (p_{i-1}, \mathsf{op}_i, p_i)$ and a counter $c_j \in \{c_1, c_2, c_3, c_4\}$ s.t. one of the following condition holds:

(F3.1) $\mathsf{op} = c_j$ ++ but there is a non-last occurrence of $c_j$ in $\mathsf{enc}(u_i)$ without matching occurrence of $c_j$ in $\mathsf{enc}(u_{i-1})$ with the same atom;

(F3.2) $\mathsf{op} = c_j$ -- but some occurrence of $c_j$ in $\mathsf{enc}(u_i)$ either has no matching occurrence of $c_j$ in $\mathsf{enc}(u_{i-1})$ with the same atom, or has a matching occurrence which is the last occurrence of $c_j$ in $\mathsf{enc}(u_{i-1})$.

(F3.3) $\mathsf{op} = c_j \overset{?}{=} 0$ but either $\mathsf{enc}(u_i)$ or $\mathsf{enc}(u_{i-1})$ contains an occurrence of $c_j$;

(F3.4) the operation $\mathsf{op}$ does not modify counter $c_j$, but there is an occurrence of $c_j$ in $\mathsf{enc}(u_i)$ without a matching occurrence of $c_j$ in $\mathsf{enc}(u_{i-1})$ with the same atom.

One register is sufficient to recognise each of the possible mistakes above.

**Lemma 8.2.** *The set of reachable configurations Reach$(M)$ is finite if, and only if, $L(A)$ is a* DRA *language.*

*Proof.* The "only if" implication follows from the fact that if Reach$(M)$ is finite, i.e., there is a finite bound $k$ on the sum of values of all counters of $M$ in a run, then the complement of $L(A)$ (which encodes all correct reversal-encodings) is a DRA$_k$ language, and thus $L(A)$ itself is a DRA language since DRA languages are closed under complement. For the "if" implication, assume that Reach($M$) is infinite, and by way of contradiction assume that $L(A)$ is a DRA language. In this case, the complement $L$ of $L(A)$ is recognised by some $\text{DRA}_k$ $B$ with a finite number of registers $k$. Since Reach($M$) is infinite, there are runs where counter (say) $c_1$ is unbounded. In particular, there is a run $\pi$ reaching a counter valuation $u_i$ with $u_i(c_1) \geq k+2$. When $B$ reads the corresponding encoding $w \in \mathsf{Enc}(\pi)$, after reading the first $k+1$ atoms $a_1, \ldots, a_{k+1}$ in the $\mathsf{enc}(u_i)$ block, it must forget at least one such atom, say $a_j$. The next atom $a_{k+2}$ in the $\mathsf{enc}(u_i)$ block can thus be replaced by $a_j$ and $B$ still accepts the corresponding data word $w'$. However, $w'$ is not the reversal encoding of any run of $M$, since it violates condition (E2). This contradicts that $B$ recognises $L$, and thus $L(A)$ is not a DRA language, as required. □

We thus have a reduction from the LCM finiteness problem to the DRA membership problem for $\text{NRA}_1$, which is thus undecidable thanks to Theorem 8.1.

**Corollary 8.3.** *The* DRA *membership problem for* $\text{NRA}_1$ *languages is undecidable.*

8.2.2. *Undecidability and hardness for* $\text{DRA}_k$ *membership.* The NRA *universality problem* asks whether a given NRA $A$ recognises every data word $L(A) = (\Sigma \times \mathbb{A})^*$. All the lower bounds in this section leading to undecidability and hardness results for the $\text{DRA}_k$ membership problem are obtained by a reduction from the universality problem for corresponding classes of data languages.

**Lemma 8.4** (c.f. [27, Theorem 1]). *Let $k \in \mathbb{N}$ and let $\mathcal{Y}$ be a class of invariant data languages that*
(1) *contains all the* $\text{DRA}_0$ *languages,*
(2) *is closed under union and concatenation, and*
(3) *contains some non-*$\text{DRA}_k$ *language.*

*The universality problem for data languages in $\mathcal{Y}$ reduces in polynomial time to the* $\text{DRA}_k$ *membership problem for data languages in $\mathcal{Y}$.*

*Proof.* Let $L \in \mathcal{Y}$ be a data language over a finite alphabet $\Sigma$. We show that the universality problem for $L$ reduces to $\text{DRA}_k$ membership. Thanks to the last assumption, let $M \in \mathcal{Y}$ be a data language over some finite alphabet $\Gamma$ which is not recognised by any $\text{DRA}_k$. Consider the following language over the extended alphabet $\Sigma' = \Sigma \cup \Gamma \cup \{\$\}$:

$$N \ := \ L \cdot (\{\$\} \times \mathbb{A}) \cdot (\Gamma \times \mathbb{A})^* \ \cup \ (\Gamma \times \mathbb{A})^* \cdot (\{\$\} \times \mathbb{A}) \cdot M,$$

where $\$ \notin \Sigma \cup \Gamma$ is a fixed fresh alphabet symbol. Since $\mathcal{Y}$ contains the universal language, by its closure properties the language $N$ belongs to $\mathcal{Y}$. We conclude by proving the following equivalence:

$$L = (\Sigma \times \mathbb{A})^* \quad \text{if, and only if,} \quad N \text{ is recognised by a } \text{DRA}_k.$$

For the "only if" direction, if $L$ is universal, then $N = (\Sigma \times \mathbb{A})^* \cdot (\{\$\} \times \mathbb{A}) \cdot (\Sigma \times \mathbb{A})^*$ is clearly recognised by a $\text{DRA}_k$. For the "if" direction suppose, towards reaching a contradiction, that $N$ is recognised by a $\text{DRA}_k$ $A$ but $L$ is not universal. Choose an arbitrary data word $w \notin L$

over $\Sigma$ and consider an arbitrary extension $u = w \cdot (\$, a)$ of $w$ by one letter. Since \$ does not belong to the finite alphabet $\Sigma \cup \Gamma$, the left quotient $u^{-1}N = \{v \mid uv \in N\}$ equals $M$. Let $(p, \mu)$ be the configuration reached by $A$ after reading $u$, which thus recognises $L(p, \mu) = M$. Since $M$ is invariant as a language in $\mathcal{Y}$, $M$ is a $\textsc{dra}_k$ language, which is a contradiction. □

From Lemma 8.4 we immediately obtain the undecidability and hardness results for the $\textsc{dra}_k$ membership problem, which we now recall.

**Theorem 1.6.** *Fix a $k \geq 0$. The $\textsc{dra}_k$ membership problem is:*
(1) *undecidable for $\textsc{nra}_2$,*
(2) *undecidable for $\textsc{nra}_1^{\mathrm{g}}$ ($\textsc{nra}_1$ with guessing), and*
(3) *not primitive recursive (*Ackermann*-hard) for $\textsc{nra}_1$.*

*Proof.* For the first point, consider the class $\mathcal{Y}$ consisting of all the $\textsc{nra}_2$ languages. Clearly this class contains all $\textsc{dra}_0$ languages and it is closed under union and concatenation. Thanks to Example 3.4 we know that there are $\textsc{nra}_1$ (and thus $\textsc{nra}_2$) languages which are not $\textsc{dra}$ languages. Thus the conditions of Lemma 8.4 are satisfied and the universality problem for $\textsc{nra}_2$ reduces in polynomial time to the $\textsc{dra}_k$ membership problem for $\textsc{nra}_2$. Since the former problem is undecidable [24, Theorem 5.4], undecidability of the latter one follows. For the other two points we can proceed in an analogous way, by using the fact that the universality problem is undecidable for $\textsc{nra}_1^{\mathrm{g}}$ ($\textsc{nra}_1$ with guessing) [11, Exercise 9], and not primitive recursive for $\textsc{nra}_1$ [24, Theorem 5.2]. □

8.3. **Timed automata.** We now focus on timed automata. Following similar lines as in case of register automata in Section 8.2, in Section 8.3.1 we prove undecidability of the $\textsc{dta}$ membership problem for $\textsc{nta}_1$ (c.f. Theorem 1.3) and in Section 8.3.2 we prove HyperAckermann-hardness of the $\textsc{dta}_k$ membership problem for $\textsc{nta}_1$ (c.f. Theorem 1.5).

8.3.1. *Undecidability of* $\textsc{dta}$ *and* $\textsc{dta}_{\_,m}$ *membership for* $\textsc{nta}_1$. It has been shown in [27, Theorem 1] that it is undecidable whether a $\textsc{nta}_2$ timed language can be recognised by some $\textsc{dta}$. This was obtained by a reduction from the $\textsc{nta}_2$ universality problem, which is undecidable. While the universality problem becomes decidable for $k = 1$, we show in this section that, as announced in Theorem 1.3, the $\textsc{dta}$ 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. As in the case of register automata, we reduce from the finiteness problem of a $\mathsf{LCM}$ $M$ with 4 counters, which is undecidable by Theorem 8.1. In the case of timed automata, we can use the reversal encoding from [42, Definition 4.6] showing that we can build a $\textsc{nta}_1$ $A$ recognising the *complement* of the set of reversal-encodings of the runs of $M$. Since this construction has already been presented in full details in [22, App. C] and since it is in complete analogy to the construction in Section 8.2.1 for register automata, we omit it here. One can then prove the following property, which is analogous to Lemma 8.2 in the case of register automata.

**Lemma 8.5.** *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 reduction above uses only constant 1, the reduction works also for the $\text{DTA}_{\_,m}$ membership problem for every fixed $m > 0$, thus proving Theorem 1.3. 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. [48, Lemma 19]), and thus $\text{DTA}_{\_,0}$ membership reduces to $\text{DTA}_{1,0}$ membership, which is decidable for $\text{NTA}_1$ by Theorem 1.1.

8.3.2. *Undecidability and hardness for* $\text{DTA}_k$ *and* $\text{DTA}_{k,m}$ *membership.* All the lower bounds in this section are obtained by a reduction from the universality problem for suitable language classes (does a given timed language $L \subseteq \mathbb{T}(\Sigma)$ satisfy $L = \mathbb{T}(\Sigma)$?), in complete analogy to Section 8.2.2 dealing with register automata. Our starting point is the following result.

**Theorem 8.6** ([27, Theorem 1])**.** *The* DTA *membership problem is undecidable for* NTA *languages.*

We provide a suitable adaptation, generalization, and simplification of the result above which will allow us to extend undecidability to the $\text{DTA}_k$ membership problem for every fixed $k \geq 0$, and also to obtain a complexity lower bound for $\text{NTA}_1$ input languages. Fix two languages $L \subseteq \mathbb{T}(\Sigma)$ and $M \subseteq \mathbb{T}(\Gamma)$, and a fresh alphabet symbol $\$ \notin \Sigma \cup \Gamma$. The *composition* $L \rhd M$ is the timed language over $\Sigma' = \Sigma \cup \{\$\} \cup \Gamma$ defined as follows:

$$L \rhd M \ = \ \{u(\$, t)(v + t) \in \mathbb{T}(\Sigma') \mid u \in L, v \in M, t \in \mathbb{R}_{\geq 0}\},$$

where $t$ is necessarily larger or equal than the last timestamp of $u$ by the definition of $\mathbb{T}(\Sigma')$. The following lemma exposes some abstract conditions on classes of timed languages which are sufficient to encode the universality problem.

**Lemma 8.7.** *Let $k, m \in \mathbb{N}$ and let $\mathcal{Y}$ be a class of timed languages that*
(1) *contains all the* $\text{DTA}_0$ *languages,*
(2) *is closed under union and composition, and*
(3) *contains some non-*$\text{DTA}_k$ *(resp. non-*$\text{DTA}_{k,m}$*) language.*
*The universality problem for languages in $\mathcal{Y}$ reduces in polynomial time to the* $\text{DTA}_k$ *(resp.* $\text{DTA}_{k,m}$*) membership problem for languages in $\mathcal{Y}$.*

Lemma 8.7 is entirely analogous to Lemma 8.4 for data languages, except that invariance of languages in $\mathcal{Y}$ is not required; moreover, notice that the notion of composition of timed languages that we need to state and prove the lemma above is a bit more complicated than the straightforward notion of concatenation that appears in the analogous statement for data languages from Lemma 8.4.

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

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

where $\$ \notin \Sigma \cup \Gamma$ is a fixed fresh alphabet symbol. Since $\mathcal{Y}$ contains all the $\mathrm{DTA}_0$ languages thanks to the assumption 1, and it is closed under union and composition thanks to the assumption 2, the language $N$ belongs to $\mathcal{Y}$. We claim that the universality problem for $L$ is equivalent to the $\mathrm{DTA}_k$ membership problem for $N$:

$$L = \mathbb{T}(\Sigma) \quad \text{if, and only if,} \quad N \text{ is recognised by a } \mathrm{DTA}_k.$$

For the "only if" direction, if $L = \mathbb{T}(\Sigma)$ then clearly $N = \mathbb{T}(\Sigma) \rhd \mathbb{T}(\Gamma)$. Thus $N$ is a $\mathrm{DTA}_0$ languages, and thus also $\mathrm{DTA}_k$ for any $k \geq 0$. For the "if" direction suppose, towards reaching a contradiction, that $N$ is recognised by a $\mathrm{DTA}_k$ $A$ but $L \neq \mathbb{T}(\Sigma)$. Assume, w.l.o.g., that $A$ is greedily resetting. Choose an arbitrary timed word $w = (\sigma_1, t_1) \ldots (\sigma_n, t_n) \notin L$ over $\Sigma$. Therefore, for any extension $v = (\sigma_1, t_1) \ldots (\sigma_n, t_n)(\$, t_n + t)$ of $w$ by one letter, we have

$$v^{-1}N = t + M = \{(\sigma'_1, t + u_1) \ldots (\sigma'_m, t + u_m) \mid (\sigma'_1, u_1) \ldots (\sigma'_m, u_m) \in M\}.$$

Choose $t$ larger than the largest absolute value $m$ of constants appearing in clock constraints in $A$, and let $(p, \mu)$ be the configuration reached by $A$ after reading $v$. Since $t > m$, 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 $\mathrm{DTA}_k$ $A'$ would accept the language $M$. But this contradicts our initial assumption that $M$ is not recognised by a $\mathrm{DTA}_k$, thus finishing the proof. ☐

We can now prove the following refinement of Theorem 8.6 claimed in the introduction.

**Theorem 1.5.** *For every fixed $k, m \in \mathbb{N}$, the $\mathrm{DTA}_k$ and $\mathrm{DTA}_{k,m}$ membership problems are:*
- *undecidable for $\mathrm{NTA}_2$,*
- *undecidable for $\mathrm{NTA}_1^\varepsilon$ (with epsilon transitions),*
- HyperAckermann-*hard for $\mathrm{NTA}_1$.*

*Proof.* Each of the three points follows by an application of Lemma 8.7. For instance, for the first point take as $\mathcal{Y}$ the class of languages recognised by $\mathrm{NTA}_2$. This class contains all $\mathrm{DTA}_0$ languages, is closed under union and composition, and is not included in $\mathrm{DTA}_k$ for any $k$ nor in $\mathrm{DTA}_{k,m}$ for any $k, m$ (c.f. the $\mathrm{NTA}_1$ language from Example 5.1 which is not recognised by any DTA). Since the universality problem is undecidable for $\mathrm{NTA}_2$ [3, Theorem 5.2], by Lemma 8.7 the $\mathrm{DTA}_k$ and $\mathrm{DTA}_{k,m}$ membership problems are undecidable for $\mathrm{NTA}_2$. The second and third points follow in the same way, using the fact that universality is undecidable for $\mathrm{NTA}_1^\varepsilon$ ($\mathrm{NTA}_1$ with epsilon transitions) [42, Theorem 5.3], resp., HyperAckermann-hard for $\mathrm{NTA}_1$ (by combining the same lower bound for the reachability problem in lossy channel systems [18, Theorem 5.5], together with the reduction from this problem to universality of $\mathrm{NTA}_1$ given in [42, Theorem 4.1]). ☐

## 9. Conclusions

We have shown decidability and undecidability results for several variants of the deterministic membership problem for timed and register automata. Regarding undecidability, we have extended the previously known results [27, 53] by proving that the DTA membership problem is undecidable already for $\mathrm{NTA}_1$ (Theorem 1.3). 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 1.5). We have depicted a similar scenario for register automata, in regards of both decidability (Theorem 1.2), undecidability and hardness (Theorems 1.4 and 1.6).

Our deterministic membership algorithm is based on a characterisation of $\mathrm{NTA}_1$ languages which happen to be $\mathrm{DTA}_k$ (Theorem 7.1), which is proved using a semantic approach leveraging on notions from the theory of sets with atoms [13]. It would be interesting to compare this approach to the syntactic determinisation method of [7]. Analogous decidability results for register automata have been obtained with similar techniques.

Finally, our decidability results extend to the slightly more expressive class of always resetting $\mathrm{NTA}_2$, which have intermediate expressive power strictly between $\mathrm{NTA}_1$ and $\mathrm{NTA}_2$.

## Acknowledgment

We thank S. Krishna for fruitful discussions.

## 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. URL: `https://doi.org/10.1109/QEST.2006.59`, `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. URL: `https://doi.org/10.1007/s10703-014-0220-1`, `doi:10.1007/s10703-014-0220-1`.

[11] Mikołaj Bojańczyk. Slightly infinite sets. URL: `https://www.mimuw.edu.pl/~bojan/paper/atom-book`.

[12] Mikolaj Bojanczyk, Bartek Klin, and Slawomir Lasota. Automata theory in nominal sets. *Log. Methods Comput. Sci.*, 10(3), 2014. URL: `https://doi.org/10.2168/LMCS-10(3:4)2014`, `doi:10.2168/LMCS-10(3:4)2014`.

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

[14] 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. URL: `https://doi.org/10.1007/978-3-540-31982-5_14`, `doi:10.1007/978-3-540-31982-5_14`.

[15] 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.

[16] 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.

[17] 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.

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

[19] 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`.

[20] Lorenzo Clemente, Sławomir Lasota, and Radosław Piórkowski. Determinisability of One-Clock Timed Automata. In Igor Konnov and Laura Kovács, editors, *Proc. of CONCUR'20*, volume 171 of *LIPIcs*, pages 42:1–42:17, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. URL: `https://drops.dagstuhl.de/opus/volltexte/2020/12854`, `doi:10.4230/LIPIcs.CONCUR.2020.42`.

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

[22] Lorenzo Clemente, Sławomir Lasota, and Radosław Piórkowski. Timed games and deterministic separability. *arXiv e-prints*, page arXiv:2004.12868, April 2020. `arXiv:2004.12868`.

[23] 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.

[24] Stéphane Demri and Ranko Lazic. LTL with the freeze quantifier and register automata. *ACM Trans. Comput. Log.*, 10(3):16:1–16:30, 2009. URL: `https://doi.org/10.1145/1507244.1507246`, `doi:10.1145/1507244.1507246`.

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

[26] 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`.

[27] 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`.

[28] R. Fraïssé. *Theory of relations*. North-Holland, 1953.

[29] 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`.

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

[31] 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`.

[32] 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.

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

[34] 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`.

[35] 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`.

[36] 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`.

[37] Michael Kaminski and Nissim Francez. Finite-memory automata. *Theor. Comput. Sci.*, 134(2):329–363, 1994.

[38] Bartek Klin, Sławomir Lasota, and Szymon Toruńczyk. Nondeterministic and co-nondeterministic implies deterministic, for data languages. *Proc. of FOSSACS'21*, 12650:365–384, 03 2021. URL: `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7984108/`, `doi:10.1007/978-3-030-71995-1{\_}19`.

[39] 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`.

[40] 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.

[41] Slawomir Lasota. Decidability border for petri nets with data: WQO dichotomy conjecture. In Fabrice Kordon and Daniel Moldt, editors, *Proc. PETRI NETS 2016*, volume 9698 of *Lecture Notes in Computer Science*, pages 20–36. Springer, 2016. URL: `https://doi.org/10.1007/978-3-319-39086-4_3`, `doi:10.1007/978-3-319-39086-4\_3`.

[42] 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`.

[43] 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`.

[44] 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`.

[45] Joshua Moerman, Matteo Sammartino, Alexandra Silva, Bartek Klin, and Michał Szynwelski. Learning nominal automata. In *Proc. of POPL'17*, POPL 2017, pages 613–625, New York, NY, USA, 2017. ACM. URL: `http://doi.acm.org/10.1145/3009837.3009879`, `doi:10.1145/3009837.3009879`.

[46] Frank Neven, Thomas Schwentick, and Victor Vianu. Finite state machines for strings over infinite alphabets. *ACM Trans. Comput. Logic*, 5(3):403–435, July 2004. URL: `https://doi.org/10.1145/1013560.1013562`, `doi:10.1145/1013560.1013562`.

[47] 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. URL: `https://doi.org/10.1007/s10009-002-0094-1`, `doi:10.1007/s10009-002-0094-1`.

[48] 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. URL: `https://doi.org/10.1109/LICS.2004.1319600`, `doi:10.1109/LICS.2004.1319600`.

[49] Joël 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`.

[50] Jeffrey Shallit. *A Second Course in Formal Languages and Automata Theory.* 2008.

[51] 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. URL: `https://doi.org/10.1007/978-3-540-85778-5_7`, `doi:10.1007/978-3-540-85778-5_7`.

[52] 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.

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

[54] 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`.

[55] 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`.

[56] 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.