A machine-independent characterization of timed languages

Mikołaj Bojańczyk* and Sławomir Lasota**

Institute of Informatics, University of Warsaw

Abstract. We use a variant of Fraenkel-Mostowski sets (known also as nominal sets) as a framework suitable for stating and proving the following two results on timed automata. The first result is a machine-independent characterization of languages of deterministic timed automata. As a second result we define a class of automata, called by us timed register automata, that extends timed automata and is effectively closed under minimization.

1 Introduction

This paper studies minimization of deterministic timed automata [2]. Existing approaches to this problem explicitly minimize various resources used by an automaton, such a locations or clocks, see [1, 8, 14–16]. We take a different approach, which abstracts away from the syntax of a timed automaton, and focuses on the recognized language, and specifically its Myhill-Nerode equivalence relation. Our notion of minimality is described by the following definition.

Definition 1. *An automaton for a language L is called* minimal *if for every two words w, w' the following conditions are equivalent:*

- *The words are equivalent with respect to Myhill-Nerode equivalence.*
- *The states reached after reading the words are equal.*

In the case of a deterministic timed automaton, the term "state" refers to the location (or control state) and the valuation of clocks. One of the main contributions of this paper is a minimization algorithm for deterministic timed automata. Of course in the case of timed automata, Myhill-Nerode equivalence has infinitely many equivalence classes, e.g. in the language

$$\{t_1 \cdots t_n \in \mathbb{R}^* : t_i = t_{i-1} + 1 \text{ for all } i \in \{2, \ldots, n\}\},$$

the equivalence class of a word is determined by its last letter.

A new automaton model. There is a technical problem with minimizing deterministic timed automata: the minimization process might leave the class of timed automata, as witnessed by the following example.

* Supported by the ERC Starting Grant "Sosna".
** Supported by the FET-Open grant agreement FOX, number FP7-ICT-233599.

Example 1. Consider the following language $L \subseteq \mathbb{R}^*$. A word belongs to L if and only if it has exactly three letters $t_1, t_2, t_3 \in \mathbb{R}$, and the following conditions hold.

- The letter t_2 belongs to the open interval $(t_1; t_1 + 2)$;
- The letter t_3 belongs to the open interval $(t_1 + 2; t_1 + 3)$;
- The letters t_2 and t_3 have the same fractional part, i.e. $t_3 - t_2 \in \mathbb{Z}$.

This language is recognized by a deterministic timed automaton. After reading the first two letters t_1 and t_2, the automaton stores t_1 and t_2 in its clocks. This automaton is not minimal in the sense of Definition 1. The reason is that the words $(0, 0.5)$ and $(0, 1.5)$ are equivalent with respect to Myhill-Nerode equivalence, but the automaton reaches two different states. Any other timed automaton would also reach different states, as timed automata may reset clocks only on time-stamps seen in the input word (unless ε-transitions are allowed).

Because of the example above, we need a new definition of automata. We propose a straightforward modification of timed automata, which we call *timed register automata*. Roughly speaking, a timed register automaton works like a timed automaton, but it can modify its clocks, e.g. increment or decrement them by integers[1]. For instance, in language L from Example 1, the minimal automaton stores not the actual letter t_2, but the unique number in the interval $(t_1; t_1 + 1)$ that has the same fractional part as t_2.

We prove that timed register automata can be effectively minimized.

Typically, minimization corresponds to optimization of resources of an automaton. In case of timed automata, the resources seem to be locations and clocks, but maybe also constants used in the guards, anything else? One substantial novelty of our approach is that the kind of resource we optimize is not chosen ad hoc, but derived directly from Myhill-Nerode equivalence. Myhill-Nerode equivalence is an abstract concept; and therefore we need a tool that is well-suited to abstract concepts. The tool we use is *Fraenkel-Mostowski sets*.

Fraenkel-Mostowski sets. By these we mean a set theory different from the standard one, originating in the work of Fraenkel and Mostowski (see [10] for the references), and thus called by us *Fraenkel-Mostowski sets* (FM sets in short). Much later a special case of this set theory has been rediscovered by Gabbay and Pitts [11, 10] in the semantics community, as a convenient way of describing binding of variable names. Motivated by this important application, Gabbay and Pitts use the name *nominal sets* for the special case of FM sets they consider. Finally, FM sets (under the name "nominal G-sets") have been used in [3] to minimize automata over infinite alphabets, such as Francez-Kaminski finite-memory automata [9]. The paper [3] is the direct predecessor of the present paper.

In the setting of [3] (see also the full version [4]), FM sets are parametrized by a *data symmetry*, consisting of a set of data values together with a group G of permutations of this set. For instance, finite-memory automata are suitably represented in the data symmetry of all permutations of data values. To

[1] A certain restriction to the model is required to avoid capturing Minsky machines.

model timed automata, and even timed register automata, we choose the *timed symmetry*, based on the group of automorphisms of the structure[2]

$$(\mathbb{R}, <, +1).$$

Despite that this data symmetry presents several technical challenges, we show that FM sets can be used to solve nontrivial algorithmic problems, such as the minimization problem. A more accurate description of this paper is that we study automata in FM sets under the timed symmetry; and these automata happen to capture timed automata, and even timed register automata. In particular, we study languages where the timestamps appearing in a word are not necessarily increasing.

The second principal contribution of this paper is an exact characterization of the languages recognized by deterministic timed automata. The characterization is in the style of the Myhill-Nerode theorem, and is machine-independent, in the sense that it does not refer to any notion of recognizing device.

Summary of contributions. Below are the main contributions of our paper.

1. We introduce a new class of automata, called timed register automata, which generalize timed automata.
2. We prove that, unlike for deterministic timed automata, deterministic timed register automata are closed under minimization. We also give a minimization algorithm for timed register automata (Theorem 3 in Section 2).
3. We study automata in Fraenkel-Mostowski sets, under the timed symmetry.
4. We prove a kind of Myhill-Nerode theorem, which characterizes exactly the languages of deterministic timed automata (Theorem 5 in Section 4).

Related research. We only mention here a few related papers we are aware of. Minimization of (nondeterminstic) timed automata has been studied in particular in [1, 14, 16], with respect to bisimulation equivalence. As we mention later, our approach extends easily to bisimulation. On the negative side, minimization of nondeterministic automata with respect to language equivalence is undecidable, cf. [15, 8]. A characterization of deterministic timed languages using finite monoids has been proposed in [6]. Our characterization is of a different nature, being based on *orbit-finiteness* of the set of equivalence classes of Myhill-Nerode equivalence. Another machine-independent characterization of deterministic timed languages has been given in [13].

2 Timed register automata

In this paper, we study timed automata as a special case of automata where the alphabet is of the form $A \times \mathbb{R}$, where A is a finite set and $\mathbb{R}$ is the real numbers. In a letter $(a, t) \in A \times \mathbb{R}$, we call a the label and t the timestamp. Timed automata accept only words where the timestamps increase from left to

[2] Studying this group has been suggested to us by James Worrell.

right, call such words *monotonic*. Unlike timed automata, some of the automata we study in this paper can accept non-monotonic words.

Constraints. A *constraint* over variables $x_1, \ldots, x_n$ is any quantifier free formula that uses the variables, the binary predicate $\leq$, and the unary function $+1$. Examples of constraints include

$$x \leq (y+1)+1 \quad \wedge \quad (y+1)+1 \leq x.$$

When writing constraints, we sometimes use syntactic sugar, for instance writing the above constraint as $x = y + 2$. A constraint over variables $x_1, \ldots, x_n$ defines a subset $X \subseteq \mathbb{R}^n$.

A constraint φ is called *maximal* if every other constraint on the same variables is either implied by φ, or inconsistent with φ. An example of a maximal constraint is

$$x_2 = x_1 + 1 \quad \wedge \quad x_2 < x_3 < x_2 + 1.$$

The constraint $x < y < x + 2$ is not maximal, since it is independent with $y < x + 1$. Not every constraint is equivalent to a finite disjunction of maximal constraints, for instance the constraint $x < y$.

Clearly, maximal constraints describe those *regions* that are bounded.

Timed register automata. We now define an automaton model, which can recognize languages over alphabets of the form $A \times \mathbb{R}$. A *(nondeterministic) timed register automaton* $\mathcal{A}$ is given by the following ingredients.

- A finite set A of *labels*.
- A finite set Loc of *locations*, also called *control states*.
- Subsets of the locations for the *initial* and *final* locations.
- For each location $l \in \text{Loc}$, a set X_l of register names[3].
- For every two locations $l, k \in \text{Loc}$, and every label $a \in A$, a constraint (not necessarily maximal) which defines a subset

$$\delta_{l,a,k} \subseteq \mathbb{R}^{X_l} \times \mathbb{R} \times \mathbb{R}^{X_k}.$$

We assume that every initial location has an empty set of register names.

A *state* of the automaton is defined to be a pair (l, η), where l is a location and η is a function, called the *register valuation*, of the form $\eta : X_l \to \mathbb{R}$. We write $Q_{\mathcal{A}}$ for the set of states of an automaton $\mathcal{A}$. This set is infinite if the automaton uses registers.

The semantics of the automaton is defined in the standard way. One defines the *transition relation*

$$\delta_{\mathcal{A}} \subseteq Q_{\mathcal{A}} \times (A \times \mathbb{R}) \times Q_{\mathcal{A}},$$

[3] A simplified version, where the set of register names does not depend on the location, would not minimize well. The reason is that the number of reals necessary to remember may depend on location. Ignoring some minor differences, the simplified version resembles updatable timed automata of [5].

to be the set of triples $(l, \eta), (a, t), (k, \mu)$ such that $(\eta, t, \mu) \in \delta_{l,a,k}$. A run over an input word from $(A \times \mathbb{R})^*$ is a sequence of states that starts in an initial state and is consistent with the transition relation.

A timed register automaton is called *deterministic* if there is one initial location and the transition relation $\delta_{\mathcal{A}}$ is a function $\delta_{\mathcal{A}} : Q_{\mathcal{A}} \times (A \times \mathbb{R}) \to Q_{\mathcal{A}}$.

Timed register automata, as defined above, are too powerful (a similar undecidability result is shown in [5]):

Theorem 1. *Emptiness is undecidable for deterministic timed register automata.*

Proof. By simulating a Minsky machine. The automaton has three register names: x, y, z. The idea is that z represents zero, $x - z$ is the value of the first counter and $y - z$ is the value of the second counter. Since the automaton can use the $+1$ in its transition relation, it can increment and decrement the counters. The zero tests are simulated by testing $x = z$ or $y = z$. □

The reason why the undecidability proof above works is that we allow a state to store, at the same time, real numbers which are very far from each other. This motivates a restriction on timed register automata to be defined now.

Constrained timed register automata. In a *constrained timed register automaton*, for each location l there is a maximal constraint φ_l over the register names of l, called the *legality constraint*. In a constrained automaton, the notion of state is changed: a state (l, μ) must be such that the register valuation μ satisfies the constraint φ_l. Despite the different semantics, a constrained timed register automaton can be easily seen to be a special case of a timed register automaton, because legality constraints can be enforced by the transition relation.

The idea of adding legality constraints might seem an ugly fix. As we shall see later, constrained timed register automata have an elegant interpretation in terms of FM sets. Also, they are powerful enough to simulate timed automata.

Theorem 2. *Emptiness is decidable for constrained timed register automata.*

As our first main result, we state:

Theorem 3. *The class of constrained timed register automata is closed under minimization. There is an algorithm that computes, for a given constrained timed register automaton, the minimal automaton.*

Speaking abstractly, the minimal automaton is the syntactic automaton, or, in other words, the quotient of a given automaton by language equivalence; this will become apparent when in Section 3 we will observe that timed register automata are a subclass of automata in FM sets under the timed symmetry. Speaking concretely, we minimize the number of locations, and the number of register variables in each location.

Our minimization algorithm adopts the classical idea of iterative partition refinement, and works equally well for bisimulation of nondeterministic automata.

Timed automata. Timed automata [2] are defined similarly as timed register automata above. A timed automaton has a number of clock variables, that may

be used to store the current timestamp and to compare it against timestamps read later on. The transition relation of a timed automaton is described using a subset of constraints, in the sense of the above definition. With these respects, timed automata seem to be a subclass of constrained timed register automata.

Timed automata have however one additional feature, not reflected in our definitions above: the clock variables are initially set to 0. In consequence, only non-negative timestamps are considered. Intuitively, a timed automaton is aware of the time that has elapsed from some *absolute* moment 0, while our automata are only aware of the *relative* time separating timestamps in the input. In particular, languages recognized by timed register automata are always closed under translations, i.e., for any $d \in \mathbb{R}$, the permutation $x \mapsto x + t$ preserves L:

$$L + t = L.$$

A language $L \subseteq (A \times \mathbb{R}^{\geq 0})^*$ can be encoded as the following language closed under translations, which has essentially the same structure as L:

$$\overrightarrow{L} \;=\; \bigcup_{\substack{t \in \mathbb{R} \\ a \in A}} ((a,0)\; L) + t \;\;=\;\; \bigcup_{\substack{t \in \mathbb{R} \\ a \in A}} (a,t)\; (L + t) \;\;\subseteq\;\; (A \times \mathbb{R})^*.$$

Thus, in this paper we only consider languages that are closed under translations. On the level of timed automata, this property may be enforced by assuming that all the clock variables are *uninitialized* (that is, initially undefined), similarly like in finite memory automata of Francez and Kaminski [9].

Theorem 4. *For every (deterministic) timed automaton with uninitialized clocks one can compute an equivalent (deterministic) constrained timed register automaton.*

The idea of the proof is to translate regions of a timed automaton to locations of a timed register automaton. Unbounded regions are eliminated by projecting onto bounded coordinates. One additional register checks monotonicity.

Constrained timed register automata are strictly more expressible than timed automata, as shown in the example below.

Example 2. Let A be a singleton, thus $A \times \mathbb{R}$ is essentially $\mathbb{R}$. The language

$$L = \{t_1 \dots t_n \;:\; n \geq 2,\; t_n - t_1 \in \mathbb{N},\; t_{i+1} - t_i \leq 1 \text{ for } i < n\}$$

is not recognized by a timed automaton, but is recognized by a deterministic constrained timed register automaton with two registers. The automaton stores initially t_1 in its register, and then increments its value, say t, by 1 at every input letter greater than t. It accepts whenever an input letter equals t.

Due to Theorem 4, the minimization algorithm of Theorem 3 works for deterministic timed automata as well. How does the definition of minimality from Definition 1 correspond to resources of a timed automaton? The most appropriate to say is that we minimize the number of regions, and the number of clocks

in each region. Indeed, as regions of timed automata are translated to locations of timed register automata, each region may be optimized independently. We however honestly note that the number of locations of the minimal automaton may be greater than the number of locations of an original timed automaton.

3 Fraenkel-Mostowski sets and their automata

The definition of Fraenkel-Mostowski sets (FM sets) is parametrized by a *data symmetry* $(\mathbb{D}, G)$, which consists of a set $\mathbb{D}$ of *data values* and a subgroup G of the group of all bijections of $\mathbb{D}$. Examples of data symmetries include:

- The *classical symmetry*, where the set of data values is empty, and the group has only the identity. FM sets in the classical symmetry are going to be normal sets.
- The *equality symmetry*, where the set of data values is a countably infinite set, and the group contains all bijections. FM sets in the equality symmetry are essentially the same thing as nominal sets in [11] or FM sets in [10].
- The *timed symmetry*, where the set of data values is the real numbers, and the group contains all permutations of real numbers that preserve the order relation $\leq$ and the successor function $x \mapsto x + 1$ (we call such permutations *timed permutations*). This is the data symmetry that we use in this paper.

Intuitively speaking, normal sets are built out of empty sets and brackets { and }. The intuition behind FM sets is that they can in addition use data values as atomic elements. Our presentation below is motivated by [10].

Fix a data symmetry $(\mathbb{D}, G)$. Consider first the *cumulative hierarchy* of sets with data values, which is a hierarchy of sets indexed by ordinal numbers and defined as follows. The empty set is the unique set of rank 0. A set of rank α is any set whose elements are sets of rank smaller than α, or data values. A permutation $\pi \in G$ can be applied to a set X in the hierarchy, by renaming the data values belonging to X, and the data values belonging to elements of X, and so on. The resulting set, which has the same rank, is denoted by $X \cdot \pi$.

A set C of data values is said to be a *support* of a set X in the cumulative hierarchy if $X \cdot \pi = X \cdot \sigma$ holds for every permutations $\pi, \sigma \in G$ which agree on elements of C. A set is called *finitely supported* if it has some finite support. We use the name *FM set* for a set in the cumulative hierarchy which is hereditarily finitely supported, which means that it is finitely supported, the sets belonging to it are finitely supported, and so on.

The support of an FM set is not unique, e.g. supports are closed under adding data values. A set with empty support is called *equivariant*.

Example 3. An example of an equivariant FM set in the timed symmetry is $\mathbb{R}$ itself. Another example is $\mathbb{R}^*$. A tuple $(x_1, \ldots, x_n) \in \mathbb{R}^*$ is supported by the set $\{x_1, \ldots, x_n\}$. The set $\mathbb{R} - \{0\}$ is not equivariant; it is supported by $\{0\}$.

For some data symmetries, including the classical and equality ones, one can show that every FM set has the least support. However, FM sets in the

timed symmetry do not have least supports. For instance, the set $\mathbb{R} - \{0\}$ is not supported by the empty set, but it is supported by the sets $\{0\}$ or $\{1\}$. This is because if π is a timed permutation, then $\pi(1) = 1$ is equivalent to $\pi(0) = 0$.

In many respects, FM sets behave like normal sets. For instance, if X, Y are FM sets, then $X \times Y$, $X \cup Y$, X^* and the finite powerset of X are all FM sets. Another example is the family of subsets of X that have finite supports. The appropriate notion of a function between FM sets X and Y is that of a *finitely supported function*, which is a function from X to Y whose graph is an FM set.

Orbit-finite FM sets. From our perspective, the key property of FM sets is their more relaxed notion of finiteness. Suppose that X is an FM set. For a set of data values C, define the C-orbit of an element $x \in X$ to be the set

$$\{x \cdot \pi : \pi \in G \text{ and } \pi \text{ is the identity on } C\}.$$

If C supports X, then the C-orbits form a partition of X. The set X is called orbit-finite if the partition into C-orbits has finitely many parts, for some C which supports X. Observe that the number of C-orbits increases as the set C grows. Therefore, a set is orbit-finite if it has a finite number of orbits for some minimal set C that supports it. In particular, an equivariant set is orbit-finite if and only if it has finitely many $\emptyset$-orbits.

In many data symmetries orbit-finite sets are closed under product, but not in the timed symmetry as illustrated in Example 4.

Example 4. The set $\mathbb{R}$ is orbit-finite, namely it has one $\emptyset$-orbit. The set $\mathbb{R}^2$ is not orbit-finite. The $\emptyset$-orbits are of the following form:

$$\{(x, y) : x - y = k\} \qquad \{(x, y) : x - y \in (k, k + 1)\} \qquad \text{for all } k \in \mathbb{Z}.$$

Observe that two orbits of the first kind, say $\{(x, y) : x - y = k\}$ and $\{(x, y) : x - y = l\}$, are equivariantly isomorphic via the mapping $(x, y) \mapsto (x, y + (k - l))$. Likewise, every two orbits of the second kind are mutually isomorphic. Another example of two isomorphic but distinct orbits in $\mathbb{R}^*$ is $\mathbb{R}$ and $\{(x, x, x) : x \in \mathbb{R}\}$. There are infinitely many equivariant isomorphisms between these two orbits, including $x \mapsto (x, x, x)$ and $x \mapsto (x + 1, x + 1, x + 1)$.

Automata. The definition of automata in FM sets is exactly like the definition of automata in normal sets, except that the notion of finiteness is relaxed to orbit-finiteness. Specifically, a *nondeterministic FM automaton* is a tuple

$$(A, Q, I, F, \delta) \qquad I, F \subseteq Q \qquad \delta \subseteq Q \times A \times Q$$

where the alphabet A, states Q, initial states $I \subseteq Q$, final states $F \subseteq Q$ and transitions $\delta \subseteq Q \times A \times Q$ are FM sets, and all of them except for δ are required to be orbit-finite. (We come back to the orbit-finiteness of δ in Example 5.) The definition of acceptance is as usual for automata. An automaton is called *equivariant* if all of its components are equivariant. From now on, we only study equivariant automata.

Example 5. Consider the language $L \subseteq \mathbb{R}^*$ which contains words where some letter appears twice. This language is recognized by a nondeterministic FM automaton whose states are: an initial state q, one state q_x for each real number x, and a single accepting state $\top$. The transition relation contains triples

$$(q, x, q) \qquad (q, x, q_x) \qquad (q_x, y, q_x) \qquad (q_x, x, \top) \qquad (\top, x, \top)$$

for every real numbers x, y. The transition relation is not orbit-finite, because the set of transitions (q_x, y, q_x) is isomorphic to $\mathbb{R}^2$. In general, the transition relation will necessarily have infinitely many orbits in any automaton which stores real numbers in its state, and which reads arbitrary input letters.

A *deterministic FM automaton* is the special case of a nondeterministic one, where the transition relation is a function $\delta : Q \times A \to Q$, and where the set of initial states contains only one state. From now on, we only study equivariant deterministic automata and work exclusively in the timed symmetry.

Comparing the models. So far, we have introduced two kinds of automata. In Section 2, we have introduced timed register automata, and we have identified a subclass of constrained timed register automata. In Section 3, we have introduced automata in FM sets. In this section, we show that in the specific case of FM sets in the timed symmetry, the two kinds of automata are closely related. We only study the deterministic case, but the nondeterministic case is analogous. The results are summed up in Figure 1.

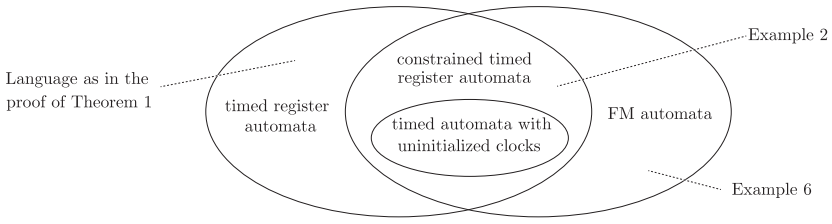

Fig. 1. Timed register automata, and FM automata in the timed symmetry.

We first show that a deterministic timed register automaton is almost a special case of a deterministic FM automaton. The input alphabet, which is a set of the form $A \times \mathbb{R}$, for a finite set A is an equivariant orbit-finite FM set. The number of orbits is the size of A, because permutations of data values (= timestamps) do not change the labels. Recall that a state of a timed register automaton consists of a location and a valuation of registers. Thus the set of all states is an equivariant FM set, since it is basically a set of tuples of real numbers. In the same way, the initial and accepting states are equivariant subsets, because they are identified by their locations, and locations are not changed by

permutations of data values. Finally, transition function of a timed register automaton is equivariant, because it is defined in terms of the order and successor, both preserved by timed permutations.

So why is a deterministic timed register automaton not necessarily an FM automaton? Because the states are not, in general, an orbit-finite FM set. For instance, if an automaton has two registers in some location, then its states will not be orbit-finite for the same reason as $\mathbb{R}^2$. This is where the constraints on the register valuations, as defined in Section 2, come in. The following lemma shows that maximal constraints can be used to enforce an orbit-finite state space.

Lemma 1. *The following conditions are equivalent for a subset $X \subseteq \mathbb{R}^n$:*

- *X is equivariant and has one orbit.*
- *X is defined by a maximal constraint.*

As a conclusion, a constrained timed register automaton is exactly the same thing as a timed register automaton, whose state space is orbit-finite.

So far we have shown that constrained timed register automata are included in FM automata. The inclusion is strict, as the transition function in a timed register automaton is defined by constraints, while in the abstract definition, the transition function is only required to be equivariant. Not all equivariant transition functions are definable by constraints, as shown in the following example.

Example 6. Suppose that $K \subseteq \mathbb{Z}$ is any set of integers, e.g. the prime numbers. Consider the language $\operatorname{diff}(K) = \{t_1\, t_2 \in \mathbb{R}^2 \ : \ t_2 - t_1 \in K\}$. Regardless of K, this language can be recognized by a deterministic FM automaton. The state space of the automaton has four orbits: an initial state ϵ, an accepting state $\top$, a rejecting sink state $\bot$, and one state q_t for every real number t. The automaton starts in the initial state ϵ. The transition function is:

$$\delta(\epsilon, t) = q_t \qquad \delta(\bot, t) = \bot \qquad \delta(q_s, t) = \begin{cases} \top & \text{if } t - s \in K \\ \bot & \text{otherwise} \end{cases} \qquad \delta(\top, t) = \top$$

The transition function is easily seen to be equivariant. For most K, however, it is not defined by a constraint (one argument is that there are uncountably many choices for K, and only countably many choices for a constraint).

Example 6 implies that the abstract definition of a deterministic FM automaton is too powerful. For instance, arbitrary FM automata cannot be represented in a finite way. Restricting equivariant functions to those definable by constraints makes the automata manageable, but it is not necessarily the only solution to the problem. We do not investigate other solutions in this paper.

4 Characterization of deterministic timed automata

In this section we provide a machine-independent characterization of the class of languages recognized by deterministic timed automata.

Every language recognized by a deterministic timed automaton with uninitialized clocks is equivariant and contains only monotonic words. Finally, the set of equivalence classes of Myhill-Nerode equivalence is orbit-finite. As shown in Example 6, these conditions are not sufficient even to characterize nondeterministic orbit-finite timed register automata. One additionally needs to say, roughly, that only recent timestamps can be remembered, and older timestamps must be forgotten. Our formulation of this condition is as follows.

For two nonempty words $u, w \in (A \times \mathbb{R})^+$ (think of monotonic words) and $M \in \mathbb{N}$ we write $u <_M w$ to mean that the first timestamp in w is larger than the last timestamp in u, by at least M.

Definition 2. *Let $M \in \mathbb{N}$. A language $L \subseteq (A \times \mathbb{R})^*$ is called* M-forgetful *if for every words $u, w \in (A \times \mathbb{R})^+$, $v \in (A \times \mathbb{R})^*$ and a timed permutation π such that $v \cdot \pi = v$, $u <_M w$ and $u \cdot \pi <_M w$, it holds:*

$$u\,v\,w \in L \;\Leftrightarrow\; (u \cdot \pi)\,v\,w \in L. \tag{1}$$

Observe that M-forgetfulness implies M'-forgetfulness for all $M' > M$. Note that $v \cdot \pi = v$ implies $(u\,v) \cdot \pi = (u \cdot \pi)\,v$ and that if L is equivariant then the property (1) may be equivalently written as $u\,v\,w \in L \;\Leftrightarrow\; u\,v\,(w \cdot \pi) \in L$.

Example 7. The language L from Example 2 in Section 2 is not M-forgetful for any $M \geq 0$. Indeed, instantiating Definition 2 with

$$u = 0.4 \qquad v = 1.2\ 2.2\ \ldots\ M{+}0.2\ M{+}1.2 \qquad w = M{+}1.4$$

and any timed permutation π satisfying $\pi(0.4) = 0.3$ and $\pi(0.2) = 0.2$, we get a contradiction, as $0.4\ v\ w \in L$ while $0.3\ v\ w \notin L$.

Example 8. The language of all monotonic words is 0-forgetful. The language "for some timestamp t, both t and $t + 3$ appear in the word" is 3-forgetful but not 2-forgetful.

Theorem 5. *Let A be a finite set of labels. For a language $L \subseteq (A \times \mathbb{R})^*$, the following conditions are equivalent:*

- *L is recognized by a deterministic timed automaton with uninitialized clocks.*
- *L satisfies simultaneously the following conditions:*
  1. *L is equivariant;*
  2. *L contains only monotonic words;*
  3. *L is M-forgetful for some threshold $M > 0$; and*
  4. *the set of equivalence classes of the Myhill-Nerode equivalence $\sim_L$ is orbit-finite.*

Note that the set of equivalence classes of $\sim_L$ is an (equivariant) FM set when L is an (equivariant) FM set. Even in presence of condition 3, condition 4 is still necessary, as shown by the following example.

Example 9. Consider the language containing all monotonic timed words of the form $t_1\,t_2\,\ldots\,t_n\,(t_1{+}1)\,(t_2{+}1)\,\ldots\,(t_n{+}1)$, for $n \geq 0$. The language is 1-forgetful, but its syntactic automaton is orbit-infinite.

5 Future work

Our approach based on Fraenkel-Mostowski sets may be further elaborated.

We consider a subclass of orbit-finite automata where the transition function (or relation) is definable by constraints. These restrictions are sufficient to capture timed automata, but there may be other manageable restrictions that are more liberal. As a natural continuation of this work we plan to pursue automata with semi-linear transition functions. We suppose that one would be able to capture in this framework, among the others, periodic time constraints, cf. [7], or some subclasses of hybrid automata, like linear hybrid automata [12].

Another urgent challenge is to relate our approach to the previous work, in particular to minimization of [1, 14, 16] and to characterizations of [6] and [13].

Acknowledgments. We kindly thank anonymous reviewers for insightful comments and valuable suggestions.

References

1. R. Alur, C. Courcoubetis, N. Halbwachs, D. L. Dill, and H. Wong-Toi. Minimization of timed transition systems. In *CONCUR*, pages 340–354, 1992.
2. R. Alur and D. L. Dill. A theory of timed automata. *Theor. Comput. Sci.*, 126(2):183–235, 1994.
3. M. Bojańczyk, B. Klin, and S. Lasota. Automata with group actions. In *Proc. LICS'11*, pages 355–364, 2011.
4. M. Bojańczyk, B. Klin, and S. Lasota. Automata theory in nominal sets. 2012. Submitted. Accessible at `http://www.mimuw.edu.pl/~sl/PAPERS/lics11full.pdf`.
5. P. Bouyer, C. Dufourd, E. Fleury, and A. Petit. Updatable timed automata. *Theor. Comput. Sci.*, 321(2-3):291–345, 2004.
6. P. Bouyer, A. Petit, and D. Thérien. An algebraic approach to data languages and timed languages. *Inf. Comput.*, 182(2):137–162, 2003.
7. C. Choffrut and M. Goldwurm. Timed automata with periodic clock constraints. *Journal of Automata, Languages and Combinatorics*, 5(4):371–404, 2000.
8. O. Finkel. Undecidable problems about timed automata. *CoRR*, abs/0712.1363, 2007.
9. N. Francez and M. Kaminski. Finite-memory automata. *TCS*, 134(2):329–363, 1994.
10. M. Gabbay. Foundations of nominal techniques: logic and semantics of variables in abstract syntax. *Bulletin of Symbolic Logic*, 17(2):161–229, 2011.
11. M. Gabbay and A. M. Pitts. A new approach to abstract syntax with variable binding. *Formal Asp. Comput.*, 13(3-5):341–363, 2002.
12. T. A. Henzinger. The theory of hybrid automata. In *LICS*, pages 278–292, 1996.
13. Oded Maler and Amir Pnueli. On recognizable timed languages. In *FoSSaCS*, pages 348–362, 2004.
14. J. Springintveld and F. W. Vaandrager. Minimizable timed automata. In *FTRTFT*, pages 130–147, 1996.
15. S. Tripakis. Folk theorems on the determinization and minimization of timed automata. *Inf. Process. Lett.*, 99(6):222–226, 2006.
16. M. Yannakakis and D. Lee. An efficient algorithm for minimizing real-time transition systems. *Formal Methods in System Design*, 11(2):113–136, 1997.

The content of the remaining sections is as follows. In Section A we define aperiodic sets and provide a characterization thereof. In Section B we give a characterization of those relation between aperiodic sets that are definable by constraints. These two sections prepare a ground for the proofs of Theorems 3 and 5, that are to be found in Sections C and E, respectively. In Section D we spell out the proof of Theorem 4.

A Aperiodic sets

A nominal set is called *aperiodic* if its every orbit X is either a singleton, or admits an equivariant function $X \to \mathbb{R}$. The aim of this section is to derive a representation of orbit-finite aperiodic sets, stated in Proposition 1 below.

For convenience consider an equivalence relation on $\mathbb{R}$: $x \asymp y$ if $y - x$ is an integer. Thus two real numbers are related if their fractional parts are equal.

Call a subset $X \subseteq \mathbb{R}$ *non-redundant* if no two elements of X are related by $\asymp$. Clearly, if a support of an element is minimal wrt. set inclusion, it is necessarily non-redundant.

An element of a nominal set may have many different minimal supports, for instance $\{4.5, 8.3\} \in \mathcal{P}_{\text{fin}}(\mathbb{R})$ is supported by itself, by also by $\{0.5, 0.3\}$ or $\{3.5, -6.7\}$. In Lemma 2 we claim however that the minimal support is determined uniquely up to $\asymp$.

The relation lifts naturally to subsets of $\mathbb{R}$: two sets are related if there is a bijective matching that relates elements of one set to elements of the other set. Observe that the set of supports of an element of a nominal set is closed under $\asymp$. We claim that the minimal supports of an element are just a single abstraction class of $\asymp$:

Lemma 2. *In the timed symmetry, all minimal supports of an element of a nominal set are related by* $\asymp$.

Proof. Consider the interval $\mathbb{I} = [0, 1)$. If a support of an element is a subset of $\mathbb{I}$ we call it *fraction support*. Clearly, if an element x is supported by C, then x has also a fraction support $D \subseteq \mathbb{I}$ with $D \asymp C$. It is thus sufficient to show the following claim:

Claim. In the timed symmetry, every element of a nominal set has the least fraction support wrt. set inclusion.

We need to argue that if C and D are finite fraction supports of an element of a nominal set, then $C \cap D$ is a support too. An equivalent condition is spelled out in the claim below. For $C \subseteq \mathbb{I}$, by G_C we mean the subgroup of timed group containing those permutations π that fix C, that is

$$\pi(x) = x \quad \text{for all } x \in C.$$

Claim. For all finite $C, D \subseteq \mathbb{I}$, the subgroup $G_{C \cap D}$ is included in $G_C + G_D$, the smallest subgroup containing G_C and G_D.

As it has been proved in [4], instead of comparing the whole subgroups, it is sufficient to compare the individual orbits, in the special case when both $C \setminus D$ and $D \setminus C$ are singleton sets. Even if our setting is slightly different from that considered in [4], as we restrict to fraction supports only, the proof of Theorem 10.3 in [4] may be repeated here to prove that the last claim above is implied by the following one:

Claim. For all finite $E \subseteq \mathbb{I}$ and $c, d \in \mathbb{I} \setminus E$ such that $c \neq d$,

$$c \cdot G_E \subseteq c \cdot \left(G_{E \cup \{c\}} + G_{E \cup \{d\}} \right).$$

We embark on the proof of the last claim from now on. Consider a finite $E \subseteq \mathbb{I}$ and $c, d \in \mathbb{I} \setminus E$ such that $c \neq d$. Let $e \in c \cdot G_E$. We need to define a timed permutation

$$\pi \;=\; \sigma_1 \theta_1 \sigma_2 \theta_2 \ldots \sigma_n \theta_n, \tag{2}$$

with $\sigma_i \in G_{E \cup \{d\}}$, $\theta_i \in G_{E \cup \{c\}}$, and $\pi(c) = e$. The proof is split into two cases, $E \neq \emptyset$ and $E = \emptyset$.

Case $E = \emptyset$. This case is proved in essentially the same way as for total order symmetry, see Corollary 10.5 in [4]. Let l be the greatest element of E smaller than c, and let h be the smallest element of E greater than c, assuming they both exist. (If l does not exist put $l := h' - 1$, where h' is the greatest element of E; symmetrically, if h does not exists put $h := l' + 1$, where l' is the smallest element of E.) Then $c \cdot G_E$ is the open interval (l, h). Take any $e \in (l, h)$; without loss of generality assume that $e > c$. We need to show some π as in (2) such that $\pi(c) = e$.

The only interesting case is $d \in (c, e]$. In this case, take some $d' \in (c, d)$ and put $\pi = \sigma\theta$, where

- σ is some timed permutation that acts as identity on $([d, l + 1)$ (so $\sigma \in G_{E \cup \{d\}}$) and such that $\sigma(c) = d'$,
- θ is some timed permutation that acts as identity on $(h-1, c]$ (so $\theta \in G_{E \cup \{c\}}$) and such that $\theta(d') = e$.

Case $E = \emptyset$. This case needs a special treatment, as G_0 is the whole timed group in this case. The orbit $c \cdot G_0$ is thus the whole set $\mathbb{R}$.

Assume wlog. $c < e$. Assume additionally $c < d$ (the case $d < c$ is shown similarly). The only interesting case now is $c < d < e$. Note that e may be arbitrarily large.

Let σ_1 be an arbitrary permutation that fixes d and maps c to some $d_1 \in (c, d)$. Note that d_1 may be any value in (c, d). Similarly, let θ_1 be an arbitrary permutation that fixes c and maps d_1 to some $c_1 \in (d, c + 1)$. Again, c_1 may be any value in $(d, c + 1)$. The intuition is that d_1 may be chosen arbitrarily close to d and c_1 may be chosen arbitrarily close to $c + 1$. By repeating this process sufficiently many times one finally reaches e as required. □

Proposition 1. *In the timed symmetry, every aperiodic one-orbit set is isomorphic to an orbit of $\mathbb{R}^*$.*

Proof. Let X be an aperiodic one-orbit nominal set in the timed symmetry. We will show that X is isomorphic to an orbit of $\mathcal{P}_{\text{fin}}(\mathbb{R})$, which in turn is clearly isomorphic to an orbit of $\mathbb{R}^*$.

Choose an arbitrary element $x_0 \in X$ and its arbitrary minimal support $y_0 \subseteq \mathbb{R}$. Wlog. assume the minimal support to be nonempty (otherwise X is a singleton, clearly isomorphic to the orbit $\{\emptyset\}$). The pair $\langle y_0, x_0 \rangle$ determines an equivariant relation containing all the pairs

$$\langle y_0 \cdot \pi, x_0 \cdot \pi \rangle$$

where π ranges over the timed group. Note that whenever a pair $\langle y, x \rangle$ is in this relation, the set y is a minimal support of x, as the action of a group necessarily preserves minimal supports. It is easily seen that the relation is a surjective function from one orbit of $\mathcal{P}_{\text{fin}}(\mathbb{R})$ to X. Indeed, if $\langle y, x_1 \rangle$ and $\langle y, x_2 \rangle$ are two related pairs then using an arbitrary permutation π that maps x_1 to x_2, and equivariance, one obtains $y \cdot \pi = y$, and since the set of elements of y supports x_1, we obtain $x_1 = x_1 \cdot \pi = x_2$.

Call the function f and its domain Y. We aim at showing that f is also injective.

Assume two different elements $y_1, y_2 \in Y$ that are both mapped by f to some $x \in X$. As elements of the same orbit, y_1 and y_2 are related by some permutation

$$y_1 \cdot \pi = y_2. \tag{3}$$

By the very definition of f, elements appearing in y_1 are a minimal support of x, and likewise for y_2. By Lemma 2 we obtain $y_1 \asymp y_2$, i.e., the same fractional parts appear in y_1 and y_2. Denote by F this set of fractional parts.

By (3) the permutation π maps elements of y_1 to elements of y_2. Thus π induces the unique permutation of the fractional parts F. Clearly, some power π^k of π induces identity on F, thus π^k is an integer translation of $\mathbb{R}$.

We will use now our assumption about existence of an equivariant function $g : X \to \mathbb{R}$, to deduce $y_1 = y_2$. As $x \cdot \pi^k = x$, by equivariance of g we get $g(x) \cdot \pi^k = g(x)$, thus π^k is necessarily identity. In consequence, π itself is an identity too, which implies $y_1 = y_2$ as required.

We have thus proved that X and Y are isomorphic. □

B Functions definable by constraints

In this section, we study relations on aperiodic sets that are definable by constraints, as defined in Section 2. The first problem is that constraints were defined for subsets of $\mathbb{R}^*$, while the notion of an aperiodic set is more abstract. We prove that there is a canonical extension of constraints to the abstract notion of aperiodic sets.

We define the *presentation* of an aperiodic single-orbit set X to be an equivariant injection $f : X \to \mathbb{R}^*$. By Proposition 1, every aperiodic set has a representation. Suppose that X, Y are aperiodic single orbit states, with presentations

f, g. We say a relation $R \subseteq X \times Y$ is *definable by constraints with respect to the presentations* f, g if there is a subset of $S \subseteq \mathbb{R}^* \times \mathbb{R}^*$ that is definable by constraints such that

$$(x, y) \in R \qquad \text{iff} \qquad (f(x), f(y)) \in S.$$

A set might have several different presentations. However, the following lemma shows that the representations are not important for definability by constraints.

Lemma 3. *If a relation $R \subseteq X \times Y$ is definable by constraints with respect to presentations f, g, then the same holds for any other choice of presentations.*

Thanks to the above lemma, we can simply say that a relation on $X \times Y$ is definable by constraints, without mentioning the presentations. The definition lifts to orbit-finite sets X and Y as follows. We say that a relation $R \subseteq X \times Y$ is definable by constraints if for every orbit U of X and every orbit V of Y, the restriction of R to $U \times V$ is definable by constraints. Finally, we say a function is definable by constraints if it its graph is a relation that is definable by constraints.

Consider an equivariant function from $\mathbb{R}^n$ to $\mathbb{R}^m$ defined by:

$$(x_1 \dots x_n) \mapsto (x_{i_1} + k_1, \dots, x_{i_m} + k_m). \tag{4}$$

Every function of this form we call *simple*. Observe that a simple function is described by a constraint:

$$y_1 = x_{i_1} + k_1 \ \wedge \ \dots \ \wedge \ y_m = x_{i_m} + k_m$$

over variables $x_1 \dots x_n, y_1 \dots y_m$. Clearly, not every equivariant function from $\mathbb{R}^n$ to $\mathbb{R}^m$ is simple. However, every function from a single orbit of $\mathbb{R}^n$ is simple:

Lemma 4. *Every equivariant function from an orbit of $\mathbb{R}^*$ to an orbit of $\mathbb{R}^*$ is simple, and thus definable by a maximal constraint.*

Proof. Let f be an equivariant function from an orbit of $\mathbb{R}^n$ to an orbit of $\mathbb{R}^m$. Suppose

$$f(x_1 \dots x_n) = (y_1 \dots y_m)$$

for some argument $(x_1 \dots x_n) \in \mathbb{R}^n$. By equivariance, f preserves supports, and thus for every y_j there is some x_{i_j} such that $x_{i_j} - y_j$ is an integer. Let k_j be the differences, for $j = 1 \dots m$. Again by equivariance, the function f is precisely the simple mapping (4). □

As an immediate corollary we have:

Lemma 5. *Let X, Y be aperiodic orbit-finite sets. Every equivariant function $f : X \to Y$ is definable by constraints.*

Above we have studied binary relations, but the definitions and results extend naturally to ternary relations and so on. The case of ternary relations is particularly important, since a function

$$f : X \times Y \to Z, \tag{5}$$

such as the transition function of a deterministic orbit-finite automaton, is a special case of a ternary relation. The goal of this section is Theorem 6, which characterizes those functions f which are definable by constraints. Before we can state the theorem, we need to define a notion of limit, presented in Section B.1.

B.1 Left and right limits

Consider a (not necessarily equivariant) function $f : X \to Y$, where X, Y are nominal sets. We say that an element $y \in Y$ is the *right limit* of f in x, if there is some threshold $M \in \mathbb{N}$ such that

$$\pi(0) > M \qquad \text{implies} \qquad f(x \cdot \pi) = y.$$

The right limit, which does not necessarily exist, is denoted by $f(x + \infty)$. In a similar way, we define the *left limit*, denoted by $f(x - \infty)$.

Suppose that $f : X \times Y \to Z$ is an equivariant function. We write $f(x, y+\infty)$ for the right limit of the function $y \mapsto f(x, y)$. Likewise, we define

$$f(x, y - \infty) \qquad f(x + \infty, y) \qquad f(x - \infty, y).$$

Example 10. Consider the function $\max : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$. For every $x, y \in \mathbb{R}$:

$$\begin{aligned} \max(x, y + \infty) &\text{ is undefined} \\ \max(x, y - \infty) &= x \\ \max(x + \infty, y) &\text{ is undefined} \\ \max(x - \infty, y) &= y \end{aligned}$$

Example 11. Consider the function $f : \mathbb{R} \times \mathbb{R} \to \mathbb{R} \cup \{\bot\}$ defined by

$$f(x, y) = \begin{cases} y & \text{if } x - y \text{ is an integer, which is a prime number} \\ \bot & \text{otherwise} \end{cases}$$

This is the transition function of the automaton from Example 6. The values

$$f(x, y + \infty) \quad f(x, y - \infty) \quad f(x + \infty, y) \quad f(x - \infty, y)$$

are undefined for every $x, y \in \mathbb{R}$.

B.2 Binary operations definable by constraints

In this section, we characterize those functions (5) that are definable by constraints.

Theorem 6. *Let X, Y, Z be aperiodic orbit-finite nominal sets. A function*

$$f : X \times Y \to Z$$

is definable by constraints if and only if for every $x \in X$ and $y \in Y$

$$\wedge \begin{cases} \vee \begin{cases} f(x, y+\infty) & \text{is defined} \\ f(x-\infty, y) & \text{is defined} \end{cases} \\ \\ \vee \begin{cases} f(x, y-\infty) & \text{is defined} \\ f(x+\infty, y) & \text{is defined} \end{cases} \end{cases} \tag{6}$$

Recall the discussion at the end of Section 3, which said that not all orbit-finite automata are equivalent to timed register automata. From Theorem 6 we get a more precise version of the discussion, as stated in Corollary 1.

Corollary 1. *An orbit-finite nominal automaton is equivalent to a timed register automaton if and only if:*

- *its state space is aperiodic; and*
- *its transition function satisfies criterion* (6) *from Theorem 6.*

The rest of Section B.2 is devoted to proving Theorem 6. Fix the aperiodic sets X, Y, Z and the function $f : X \times Y \to Z$. The top-down implication is given in the following lemma.

Lemma 6. *If f is definable by constraints, then* (6) *holds.*

Proof. By inspecting the syntax of constraints. It is important that the image of the function f is an orbit-finite set Z, and therefore its representation has bounded span. By the *span* of a vector $(x_1, \ldots, x_n) \in \mathbb{R}^n$ we mean here the maximal difference between two coordinates. For instance, the span of $(1, 14.2, -2)$ is 16.2. □

The rest of the proof is devoted to the more interesting converse implication: if (6) holds, then f is definable by constraints. By the definition of a function defined by constraints, we need to show that for every orbit of X and every orbit of Y, the function f restricted to the product of these orbits is definable by constraints. Therefore, without loss of generality we can assume that X and Y have one orbit.

Lemma 7. *Let $x \in X$. If $f(x, y+\infty)$ is defined for some $y \in Y$, then $f(x, y+\infty)$ is defined for all $y \in Y$, and has the same value.*

Proof. Using the assumption that X has one orbit. □

Thanks to the above lemma, we can use notation $f(x, +\infty)$ without specifying y. Likewise for $f(x, -\infty)$, $f(+\infty, y)$ and $f(-\infty, y)$. Suppose that (6) holds. Choose some $x \in X$ and $y \in Y$. By rewriting condition (6) in disjunctive normal form, there are four possibilities:

$$\begin{gathered} f(x, +\infty),\ f(x, -\infty) \text{ are defined} \\ f(-\infty, y),\ f(x, -\infty) \text{ are defined} \\ f(x, +\infty),\ f(+\infty, y) \text{ are defined} \\ f(-\infty, y),\ f(+\infty, y) \text{ are defined.} \end{gathered}$$

We only study the first possibility, the other three are shown the same way.

If X is a singleton, then f is actually an equivariant function from Y to Z, and it is therefore definable by constraints thanks to Lemma 5. Likewise if Y is a singleton. The interesting case is when neither of the sets X and Y is a singleton. By definition of aperiodicity, this means that there are functions

$$g : X \to \mathbb{R} \qquad h : Y \to \mathbb{R}.$$

Define a function by

$$\Delta : X \times Y \to \mathbb{N} \qquad \Delta(x, y) = \text{floor}(h(y) - g(x)).$$

This function is equivariant, under assuming that the image $\mathbb{N}$ is equipped with the trivial action.

Lemma 8. *Suppose that $f(x, +\infty)$ is defined. There is some $M_x \in \mathbb{N}$ such that*

$$f(x, y) = f(x, +\infty) \qquad \text{for every } y \text{ with } \Delta(y, x) > M_x.$$

Proof. This lemma simply restates the definition of $f(x, +\infty)$ in terms of Δ.

Lemma 9. *The partial function $x \mapsto f(x, +\infty)$ is an equivariant partial function from X to Z. In particular, it is defined on all or no arguments.*

A corollary of the above lemma is that the constant M_x which appears in Lemma 8 does not depend on x.

We are now ready to complete the proof of Theorem 6. Recall the assumption that both $f(x, +\infty)$ and $f(x, -\infty)$ are defined. By Lemma 9, the function $x \mapsto f(x, +\infty)$ is an equivariant function from $X \to Z$ that is defined on all arguments. Like any equivariant function with a one-orbit domain, the function $x \mapsto f(x, +\infty)$ is definable by constraints. By Lemma 8, there is some $M \in \mathbb{N}$ such that the functions $f(x, y)$ and $f(x, +\infty)$ agree on arguments from the set

$$\Sigma_1 = \{(x, y) \in X \times Y : \Delta(y, x) > M\}.$$

Since the set Σ_1 is definable by a constraint, and the function $f(x, +\infty)$ is definable by a constraint, it follows that the restriction of f to Σ_1 is definable by a constraint. Using the same argument for the function $x \mapsto f(x, -\infty)$, we see that there is some $N \in \mathbb{N}$ such that the restriction of f to the subset

$$\Sigma_2 = \{(x, y) \in X \times Y : \Delta(x, y) > N\}$$

is definable by a constraint. Finally, consider the remaining arguments:

$$\Sigma_3 \stackrel{\text{def}}{=} \qquad X \times Y - (\Sigma_1 \cup \Sigma_2).$$

Lemma 10. *The set Σ_3 is orbit-finite.*

By the above lemma, and by Lemma 5, the function f restricted to Σ_3 set is definable by constraints.

Summing up, we have partitioned the domain of f into three subsets, and shown that it is definable by constraints when restricting to these subsets. Therefore, the function f itself is definable by constraints. This finishes the proof of Theorem 6.

C Proof of Theorem 3

Characterization of aperiodic sets. A timed permutation π is called *bounded* if it has at least one fixed point, i.e., if $\pi(t) = t$ for some $t \in \mathbb{R}$. Otherwise, a permutation is called *unbounded*.

Observe that if π is unbounded then for any $x \in \mathbb{R}$ the sequence $(\pi^n(x))_n$ is an unbounded *increasing* sequence; or for any $x \in \mathbb{R}$ the sequence $(\pi^n(x))_n$ is an unbounded *decreasing* sequence. In the first case $\pi(x) > x$ for all $x \in \mathbb{R}$. In the second case, $\pi(x) < x$ for all $x \in \mathbb{R}$.

For $x \in X$, let

$$G_x = \{\pi : x \cdot \pi = x\}$$

denote the stabilizer of x, a subgroup of the timed group. A subgroup, in particular a stabilizer, we call *bounded* if it only contains bounded permutations. Otherwise, a subgroup is called unbounded.

Lemma 11. *For a non-singleton one-orbit set X, the following conditions are equivalent:*

- *X is aperiodic;*
- *all elements of X have bounded stabilizers.*

Proof. We show the top-down implication first. Assume some $x \in X$ with unbounded stabilizer. This means that there is some unbounded timed permutation π with $x \cdot \pi = x$. Any equivariant function $f : X \to \mathbb{R}$ would satisfy

$$f(x) = f(x \cdot \pi) = \pi(f(x)),$$

which is in contradiction with unboundedness of π. In consequence, X is not aperiodic.

For the bottom-up implication, assume that all elements of X have bounded stabilizers. We will show that X is aperiodic by constructing an equivariant function $f : X \to \mathbb{R}$.

First observe that no element of X has empty support. Fix an arbitrary $x_0 \in X$ and an arbitrary element y_0 of some minimal support C_0 of x_0. Extend the pair $x_0 \mapsto y_0$ to the equivariant function:

$$f : \quad x_0 \cdot \pi \quad \mapsto \quad \pi(y_0). \tag{7}$$

It only remains to show that this is a well-defined function. For this we need some preparation. For $C \subseteq \mathbb{R}$, let G_C be the subgroup containing those timed permutations that fix the set C:

$$G_C = \{\pi : \pi(t) = t \text{ for all } t \in C\}.$$

We know that $G_{C_0} \subseteq G_{x_0}$. We will argue that by minimality of C_0, and by our assumption it follows:

Fact 7 $G_{x_0} \subseteq G_{C_0}$.

Proof. For the sake of contradiction, suppose that for some $\pi \notin G_{C_0}$ it holds $x_0 \cdot \pi = x_0$. Let $t \in C_0$ be such that $\pi(t) \neq t$. By assumption on bounded stabilizers, permutation π is bounded. We distinguish two cases. Wlog. assume that $\pi(x) > x$ for all $x \in \mathbb{R}$, otherwise consider π^{-1}.

Case $C_0 = \{t\}$**.** We know that the generated subgroup $G_{C_0} + \{\pi\}$, the smallest subgroup of the timed group that includes $G_{C_0} \cup \{\pi\}$, is a subgroup of G_{x_0}:

$$G_{C_0} + \{\pi\} \subseteq G_{x_0}.$$

We will obtain a contradiction, once we prove:

Claim. $G_{C_0} + \{\pi\}$ is unbounded.

Indeed, from any point $x \in \mathbb{R}$, one can go arbitrarily far, by applying, in an alternating manner, either a permutation from G_{y_0} that necessarily fixes $t, t+1$, etc., or π that may cross $t, t+1$, etc.

Case $C_0 \neq \{t\}$**.** We will focus on the set

$$D = (C_0 \setminus \{t\}) + \mathbb{Z}.$$

First observe that we may assume wlog. that $\pi(t) \notin D$. Indeed, if $\pi(t) \in D$, then replace t by $\pi(t)$. If again $\pi(t) \in D$, repeat the replacement. After a finite number of steps we finally get $\pi(t) \notin D$, as otherwise π would be unbounded.

As a second step, we observe that one may assume wlog. that π fixes $C_0 \setminus \{t\}$ and that between t and $\pi(t)$ there is no element of D:

$$\pi \in G_{C_0 \setminus \{t\}} \qquad \wedge \qquad (t, \pi(t)) \cap D = \emptyset. \tag{8}$$

Indeed, take a permutation $\tau \in G_{C_0}$ that slightly modifies $\pi(t)$, and replace π by:

$$\pi\, \tau\, \pi^{-1}.$$

Now we will argue that by composing π and π^{-1} with permutations from G_{C_0} one obtains all permutations from $G_{C_0 \setminus \{t\}}$. This would mean that $C_0 \setminus \{t\}$ supports x_0, a contradiction with minimality of C_0. Thus it is sufficient to show:

Claim. $G_{C_0 \setminus \{t\}} \subseteq G_{C_0} + \{\pi\}$.

For the proof of this claim, consider an arbitrary permutation from $G_{C_0 \setminus \{t\}}$, and focus on its restriction ρ to some open interval (l, u), determined by some two consecutive element of the set D. Clearly ρ is a permutation of (l, u). Assume wlog. that $t \in (l, u)$. We have at our disposal arbitrary permutations from G_{C_0}, which restricted to the interval (l, u) means arbitrary permutations of (l, u) that fix t, and the permutations π and π^{-1} (note that $\pi(t) \in (l, u)$ and $\pi^{-1}(t) \in (l, u)$ by (8)). We observe that ρ may be always presented as a composition of permutations at our disposal.

This completes the proof of Fact 7. □

Now we easily demonstrate that (7) is a well-defined function. Indeed, by general properties of group actions and by the above claim we get

$$G_{x_0 \cdot \pi} \subseteq G_{\pi(y_0)}.$$

Suppose $x_0 \cdot \pi = x_0 \cdot \tau = x$. Thus $(\pi^{-1}; \tau) \in G_{x_0 \cdot \pi}$. By the inclusion above we obtain $(\pi^{-1}; \tau) \in G_{y_0 \cdot \pi}$, and hence $y_0 \cdot \pi = y_0 \cdot \tau$.

This completes the proof of Lemma 11. □

Approximants are aperiodic. We consider the class of deterministic orbit-finite timed register automata in this section.

Lemma 12. *Timed register automata are a subclass of nominal automata, both in the deterministic and nondeterministic cases.*

Proof. There is a natural action of the timed group on states of a timed register automaton: given a permutation π from the timed group, and a state (l, η), the state $(l, \eta) \cdot \pi$ is defined leaving the location the same, and applying π to each register value. By Lemma 1, the transition relation of a timed register automaton is equivariant. □

The above proof works for the unconstrained model of timed register automata. Lemma **??** implies that constrained timed register automata have an orbit-finite state space. In particular:

Corollary 2. *Constrained register timed automata are a subclass of orbit-finite nominal automata, both in the deterministic and nondeterministic cases.*

It is not difficult to see that a converse implication holds: constrained timed register automata have the same expressive power as timed register automata with an orbit-finite state space, both in the deterministic and nondeterministic case. b R*selying on Corollary 2, we treat orbit-finite timed register automata as a subclass of orbit-finite nominal automata. The latter class, being syntax-independent, allows us to use standard notions, like quotient by an equivariant equivalence, or automaton homomorphism.

Fix a deterministic orbit-finite timed register automaton $\mathcal{A}$ in the sequel, with orbit-finite state space Q described by maximal constraints, and transition function $\delta : Q \times (A \times \mathbb{R}) \to Q$.

Let $\sim \subseteq Q \times Q$ denote the language equivalence of states of $\mathcal{A}$ from now on.

We will use the standard approximating equivalences $\sim_n \subseteq Q \times Q$ of the language equivalence. The initial approximant $\sim_0$ relates every two accepting states, and every two non-accepting ones. Inductively, $\sim_{n+1}$ is defined using $\sim_n$:

$$q \sim_{n+1} p \quad \iff \quad \forall (a,t) \in A \times \mathbb{R}.\ \delta(q,(a,t)) \sim_n \delta(p,(a,t)). \tag{9}$$

In other words, two states are related by $\sim_n$ if and only if no word of length at most n can distinguish the two states. As usual

$$\sim \ = \ \bigcap_n \sim_n . \tag{10}$$

Lemma 13. *For every n, the quotient set $Q/\sim_n$ is aperiodic.*

Proof. The proof is by induction on n. For $n = 0$ the quotient is a finite set, hence aperiodic.

Assume $Q/\sim_n$ is aperiodic. We aim at showing that $Q/\sim_{n+1}$ is aperiodic too, using the characterization of Lemma 11.

By Lemma 4, the quotient function $f_n : Q \to Q/\sim_n$ is definable by constraints. As a consequence, the composition of functions

$$(\delta; f_n) : Q \times (A \times \mathbb{R}) \to Q/\sim_n \tag{11}$$

is also definable by constraints. Towards a contradiction, assume that $Q/\sim_{n+1}$ is not aperiodic. Let $O \subseteq Q$ be a non-aperiodic orbit of Q. Using Lemma 11 we obtain some unbounded π and $x \in O/\sim_{n+1}$ with $x \cdot \pi = x$. Using equivariance of the quotient function $f_{n+1} : Q \to Q/\sim_{n+1}$ restricted to the orbit O, this is equivalent to saying that there is some unbounded π and a state $q \in O$ such that

$$q \sim_{n+1} q \cdot \pi. \tag{12}$$

We will now apply reasoning similar to Theorem 6 to the function (11) restricted to the orbit O, to deduce that $O/\sim_{n+1}$ is a singleton, thus aperiodic, which is a contradiction with our assumption above. In other words, we aim at showing that every two states in O are related by $\sim_{n+1}$.

Clearly $\pi(0) \neq 0$. Assume wlog. that $\pi(0) > 0$ (otherwise consider π^{-1} instead of π). As $\sim_{n+1}$ is an equivariant equivalence, we could replace π with π^n, for any n, while preserving condition (12). Knowing that $(\pi^n(0))_n$ is an unbounded increasing sequence, we may assume wlog. that $\pi(0)$ is arbitrarily large. Denote by f the function (11) restricted to the orbit O. Let x range over O and let y range over $A \times \mathbb{R}$. Expanding (9) for $q \sim_{n+1} q \cdot \pi$ we know that

$$f(q, y) = f(q \cdot \pi, y) \qquad \text{for all } y. \tag{13}$$

By Theorem 6 we know that one of the limits exists,

$$f(x, -\infty) \qquad \text{or} \qquad f(+\infty, y), \tag{14}$$

for every x, y. As $\pi(0)$ may be chosen arbitrarily large in (13), it follows that it is the second limit $f(+\infty, y)$ in (14) that exists for all y. Similarly, one shows that the limit

$$f(-\infty, y)$$

exists for all y. Now, using (13) one obtains

$$f(x, y) = f(x', y)$$

for all $x, x' \in O$ and all y, thus $O/\sim_{n+1}$ is a singleton as required. □

Syntactic automaton is aperiodic. We will argue now that the quotient $Q/\sim$ is an aperiodic set. By Lemma 13 and equation (10) it is sufficient to prove that for some n the sequence of quotients stabilizes, i.e., there is an isomorphism:

$$Q/\sim_n \ \simeq \ Q/\sim_{n+1}.$$

The stabilization will easily follow from the following simple observation. By a *characteristic* of a one-orbit aperiodic set X we mean here the cardinality of a minimal support of elements of X. In other words, the characteristic is the smallest n such that X is isomorphic to an orbit of $\mathbb{R}^n$.

Lemma 14. *Let X, Y be orbit-finite aperiodic sets. Suppose $f : X \to Y$ is equivariant, non-bijective and surjective. Then at least one of the following conditions holds:*

- *Y has strictly less orbits than X,*
- *for some orbit O of X, the image orbit has strictly smaller characteristic than O.*

Proof. Follows immediately using Lemma 4. □

Every consecutive equivalence $\sim_{n+1}$ refines the previous one $\sim_n$, which is witnesses by equivariant surjective quotient function

$$g_n : Q/\sim_{n+1} \ \to Q/\sim_n.$$

On the other hand, for every n, the quotient function $f_n : Q \to Q/\sim_n$ is also equivariant and surjective. By Lemma (14), only finitely many of functions f_n are non-bijective. As functions f_n and g_n commute, we conclude that there are only finitely many functions g_n that are non-bijective. Thus the state space of the minimal automaton is aperiodic.

Syntactic automaton is a timed register automaton. By now we have shown that every orbit of the state space $Q/\sim$ of the minimal automaton is isomorphic to an orbit of $\mathbb{R}^*$. It only remains to show that the transition function

$$\gamma : (Q/\sim) \times (A \times \mathbb{R}) \to Q/\sim$$

of the minimal automaton is definable by constraints.

Let $h : Q \to Q/\sim$ be the quotient function. The equation

$$\gamma(h(q), (a,t)) \ = \ h(\delta(q, (a,t)))$$

holds for all $q \in Q$ and $(a,t) \in A \times \mathbb{R}$. As h is simple due to Lemma 4, the function γ is definable by constraints, as required.

Minimization algorithm. We finally describe a minimization algorithm. In the sequel we merely concentrate on decidability, not on complexity. However, an efficient algorithm may be actually derived from our considerations below. We leave this issue for future research.

In consequence of Lemma 4 and 13, every approximating equivalence $\sim_n$ is definable by constraints, as stated in Lemma 15 below. We will be explicit about what we mean by describing an equivalence over Q by constraints. As the number of register names varies from one orbit (location) to another, there will be a separate constraint $\phi_{l,k}$ for every pair of locations l, k. We distinguish two cases. If $k \neq l$, the constraint $\phi_{l,k}$ will be a finite conjunction of equalities

$$x = y + k$$

for $x \in X_l$ and $y \in X_k$. If $k = l$, the constraint $\phi_{l,l}$ will be a finite conjunction of equalities of a special form:

$$x = x'$$

where $x \in X_l$. The diagonal constraints are over variables $X_l \cup X'_l$, where X'_l contains a copy x' of every $x \in X_l$. A set of constraints $\{\phi_{l,k}\}_{l,k}$, of the form described above, where l, k range over locations of $\mathcal{A}$, we will call *equivalence constraint*.

Lemma 15. *Every approximating equivalence $\sim_n$ is definable by an equivalence constraint.*

The minimization algorithm will follow form the two lemmas below. First, note that from the proof of Lemma 13 it actually follows that if an equivariant equivalence $\equiv \subseteq Q \times Q$ is defined by an equivalence constraint, then the *expansion* of $\equiv$, defined according to (9) by

$$q \equiv' p \quad \iff \quad \forall (a,t) \in A \times \mathbb{R}.\ \delta(q,(a,t)) \equiv \delta(p,(a,t)), \tag{15}$$

is also definable by an equational constraint. We claim that the expansion is computable:

Lemma 16. *There is an algorithm that for a given equivalence constraint computes an equivalence constraint that defines the expansion.*

Proof. Knowing that an equivalence constraint defining the expansion exists, the algorithm simply enumerates all equivalence constraints, and for each of them checks if it defines the expansion $\equiv'$ as defined by formula (15). The latter question is decidable, using a decision procedure for theory of $(\mathbb{R}, <, +1)$. □

A second lemma is an immediate consequence of the above one:

Lemma 17. *There is an algorithm that for a given equivalence constraint decides if the defined equivalence coincides with its expansion.*

Proof. Compute the expansion and check if the defined equivalence coincides with the relation defined by the given equivalence constraint. □

Basing on Lemmas (16) and (17), we obtain a partition refinement algorithm that always terminates and computes an equivalence constraint that defines language equivalence. From the equivalence constraint, it is easy to construct a presentation of the minimal automaton.

D Proof of Theorem 4

Except for the fact that a run starts with all clocks undefined, timed automata with uninitialized clocks are exactly like timed automata of [2].

We start by defining timed automata with uninitialized clocks explicitly, but in a slightly unusual notation similar to that used for timed register automata.

Clock constraints. We will use constraints over the structure $(\mathbb{R}, <, +1)$ extended with an additional element $\perp$ standing for undefined value. Thus we additionally allow for atomic constraints of the form $x = \perp$.

Let $\mathcal{X}$ be a finite set of clock names, and let $\mathcal{X}'$ contain a copy x' of every clock name $x \in \mathcal{X}$. By a *clock constraint* over $\mathcal{X}$ we mean any constraint over variables $\mathcal{X} \cup \{\mathtt{t}\} \cup \mathcal{X}'$ that implies, for every clock name $x \in \mathcal{X}$, the following constraint:

$$(x' = x \quad \vee \quad x' = \mathtt{t} \quad \vee \quad x' = \perp) \quad \wedge \quad (x' = \perp \implies x = \perp).$$

Intuitively, variable $\mathtt{t}$ represents current timestamp and variables $\mathcal{X}'$ represent new clock values.

Timed automata with uninitialized clocks. A *(nondeterministic) timed automaton with uninitialized clocks* $\mathcal{A}$ is given by the following ingredients.

- A finite set A of *labels*.
- A finite set Loc of *locations*.
- Subsets of the locations for the *initial* and *final* locations.
- A finite set $\mathcal{X}$ of clock names.
- For every two locations $q, p \in \mathrm{Loc}$, and every label $a \in A$, a clock constraint, which defines a subset

$$\delta_{q,a,p} \subseteq (\mathbb{R} \cup \perp)^{\mathcal{X}} \times \mathbb{R} \times (\mathbb{R} \cup \perp)^{\mathcal{X}}.$$

A *clock valuation* is an element of $(\mathbb{R} \cup \perp)^{\mathcal{X}}$. A *state* of $\mathcal{A}$ is a pair $(l, \mathbf{v})$ consisting of a location and a clock valuation. Let $Q_{\mathcal{A}}$ denote the set of states of $\mathcal{A}$. The semantics of the automaton is defined in a standard way. One defines

$$\delta_{\mathcal{A}} \subseteq Q_{\mathcal{A}} \times (A \times \mathbb{R}) \times Q_{\mathcal{A}}$$

to be the set of triples $(q, \mathbf{v})$, (a, t), $(p, \mathbf{w})$ such that

$$(\mathbf{v}, t, \mathbf{w}) \in \delta_{q,a,p}.$$

A run is a sequence of states consistent with the transition relation, that starts in a state consisting of an initial location and the completely undefined clock valuation.

In the sequel, automata defined above are called briefly timed automata.

From timed automata to constrained timed register automata. A timed automaton $\mathcal{A}$ as defined above may be easily transformed to an equivalent constrained timed register automaton as follows.

First of all, we need to guarantee that the timed register automaton will recognize only monotonic words. For this purpose we introduce one additional clock to store the most recent timestamp, and transitions to an additional sink state when the monotonicity is violated. Now the clocks become registers. Note that value $\perp$ may be eliminated from states, by reducing the set of clocks/registers in certain locations.

Second, every location is split into a finite number of locations, one per every clock region. Some clock regions are represented by maximals constraints, and some are not. In the latter case, eliminate those clocks that are unbounded. As projection of a clock region is a clock region, this will result in a maximal constraint. Note that information about region is stored explicitly in the location.

E Proof of Theorem 5

The top-down implication is straightforward: it is not difficult to see that the four conditions 1-4 hold for any language recognized by a deterministic timed automaton. The rest of the section is devoted to proving the converse.

Fix a finite set A_{fin} of labels, and a language L over the alphabet $A_{\text{fin}} \times \mathbb{R}$, as in the theorem. Assume that L satisfies conditions 1-4. We will show that L is recognized by a timed automaton.

Let us write A for $A_{\text{fin}} \times \mathbb{R}$. Let Q be the state space of the syntactic automaton, and let δ be its transition relation. Fix this notation for the rest of this section.

The proof of Theorem 5 is done in two steps. In Section E.1, we show that the syntactic automaton is actually (isomorphic to) a timed register automaton. In Section E.2, we show that after adding some new locations, the syntactic automaton becomes a timed automaton.

E.1 Getting a timed register automaton

Proposition 2. *The syntactic automaton is a timed register automaton.*

We use the standard notion of residue for a word $w \in A^*$

$$w^{-1}L \stackrel{\text{def}}{=} \{v \in A^* : wv \in L\}.$$

Two words are L-equivalent if and only if they have the same residues. Define a partial function

$$\mathit{minfuture} : A^* \dot{\to} \mathbb{R}$$

which maps a word w to

$$\inf\{t \in \mathbb{R} : t \text{ is a timestamp that appears in } w^{-1}L\}.$$

It is not difficult to see that the function *minfuture* is equivariant. Because the value $\mathit{minfuture}(w)$ depends only on the set $w^{-1}L$, it follows that two L-equivalent words have the same values under *minfuture*. Therefore, by abuse of notation, we can think of *minfuture* as being a function

$$\mathit{minfuture} : Q \dot{\to} \mathbb{R}.$$

A state $q \in Q$ we call equivariant if $\{q\}$ is equivariant as subset of Q. Equivalently, one can say that q has empty support.

Lemma 18. *A state $q \in Q$ is equivariant if and only if $\mathit{minfuture}(q)$ is undefined.*

Proof. The left-to-right implication is immediate. Suppose that q is equivariant, and that $\mathit{minfuture}(q)$ is defined and equal to a real number x. Then

$$x \cdot \pi = \mathit{minfuture}(q) \cdot \pi = \mathit{minfuture}(q \cdot \pi) = \mathit{minfuture}(q) = x$$

holds for every permutation π in the timed group. There is no real number x with this property.

Consider now the right-to-left implication. Consider a state $q \in Q$, which is the L-equivalence class of some word $w \in A^*$. We will prove that either $\mathit{minfuture}(q)$ is defined, or q is equivariant. The state is equivariant q if and only if the residue $w^{-1}L$ is equivariant, i.e. it is a set of words that is stable under the action of the timed group. Let us first look at three cases concerning the residue $w^{-1}L$.

- If $w^{-1}L$ is empty, then it is equivariant.
- If w is the empty word, then $w^{-1}L$ is L, which is equivariant by assumption.
- If w^{-1} contains only the empty word, then it is equivariant.

Suppose then that none of the cases above holds, which means that w is a nonempty word and $w^{-1}L$ is a nonempty language. Since w is a nonempty word, it has some last timestamp, call it t. By condition 2 on monotonicity, all timestamps that appear in $w^{-1}L$ are bigger than t. By nonemptiness of $w^{-1}L$, there is at least one timestamp that appears in $w^{-1}L$. It follows that the lower bound in the definition of *minfuture* is well defined, as an infimum over a nonempty set of reals that is bounded from below.

Proof (of Proposition 2). First we show that the state space is aperiodic. Choose an orbit of the state space Q. If the orbit is a singleton, then it is aperiodic. If the orbit is not a singleton, then *minfuture* is defined on the orbit by Lemma 18. Therefore, the orbit is aperiodic.

Because the language L is monotone, we know that for every state $q \in Q$ and every letter a, the value $\delta(q, a - \infty)$ is defined and equal to the rejecting sink state. From condition 3, we see that for every $q \in Q$ and every letter $a \in A$, the value $\delta(q - \infty, a)$ is defined. Therefore, by Corollary 1, the syntactic automaton is actually an orbit-finite timed register automaton.

□

E.2 Getting a timed automaton

So far, we have shown that the syntactic automaton is an orbit-finite register automaton. However, not every orbit-finite register automaton is equivalent to a deterministic timed automaton. The following lemma shows what is needed to get a timed automaton.

Lemma 19. *An deterministic orbit-finite timed register automaton is equivalent to a deterministic timed automaton if*

- *It only accept monotonic words.*
- *There is some $M \in \mathbb{N}$ such that for every word $w \in A^*$, after reading w the register contain only timestamps that appear in w, and the timestamps are at distance at most M from the last timestamp.*

Proof. Suppose that a deterministic orbit-finite timed register automaton satisfied the conditions above. It may be easily transformed to a timed automaton. One needs to add dummy register names to make the number of registers equal in each location. Additionally, one needs to keep track of a simple book-keeping, for instance instead of copying value of a register x to register y, keep this value in the same register while remembering the assignment $y \mapsto x$ in an additional finite control state space. □

In this section, we prove that the syntactic automaton can be modified so that it satisfies the second condition in Lemma 19 while recognizing the same language.

Let $Q_1, \ldots, Q_n \subseteq Q$ be the orbits of Q. The number of orbits is finite by condition 4. Let Q_i be one of the orbits. By Proposition 2, this orbit is aperiodic. By Proposition 1, the orbit Q_i is isomorphic to some orbit $\mathbb{R}^*$, let us denote the latter orbit by $X_i \subseteq \mathbb{R}^*$. Of course all vectors in X_i have the same dimension, i.e. there is some number $n_i \in \mathbb{N}$ such that $X_i \subseteq \mathbb{R}^{n_i}$. Summing up, we can assume that the state space of the automaton is

$$Q = \bigcup_{i \in \{1,\ldots,n\}} \{i\} \times X_i.$$

Essentially, the above is a presentation of the state space, as defined in Section B.

As previously, we will write $x \asymp y$ when x and y have the same fractional parts.

Lemma 20. *Let the state of the automaton after reading a nonempty word w be*

$$(i, x_1, \ldots, x_{n_i}).$$

For every $j \in \{1, \ldots, n_i\}$ there is a timestamp y_j in w such that $x_j \asymp y_j$. Furthermore, this distance between y_j and the last timestamp in w is at most M, where M is the constant from condition 3.

Proof. Decompose w as $w = uv$ where v is be the maximal suffix of w which contains all the timestamps at distance at most M from the last timestamp in w. Let $t_1, \ldots, t_k$ be all the timestamps that appear in the suffix v. We need to prove that for every $j \in \{1, \ldots, n_i\}$, there is some $l \in \{1, \ldots, k\}$ with $x_j \asymp t_l$.

Suppose that $j \in \{1, \ldots, n_i\}$ is such that $x_j \not\asymp t_l$ for all $l \in \{1, \ldots, k\}$. We will derive a contradiction. Choose a permutation π in the timed group which preserves all the timestamps $t_1, \ldots, t_k$, but does not preserve the timestamp x_j. Such a permutation π exists, it moves x_j by a very small distance. The transition function of the syntactic automaton is equivariant, and therefore the state of the syntactic automaton after reading $w \cdot \pi$ is

$$(i, \pi(x_1), \ldots, \pi(x_{n_i})),$$

which is a different state than the one after reading w (because the two states are different on coordinate j). Stated differently, the words w and $w \cdot \pi$ are not L-equivalent. Knowing that $v \cdot \pi = v$, we may write:

$$w = uv \qquad \not\sim_L \qquad w \cdot \pi = (u \cdot \pi)(v \cdot \pi) = (u \cdot \pi)v.$$

If u is empty, then we get a contradiction immediately. If u is nonempty, then we get a contradiction with condition 4 in Theorem 5. □

Let w and $(i, x_1, \ldots, x_{n_i})$ be as in Lemma 20. Define a vector

$$\lambda(w) = (y_1, \ldots, y_{n_i})$$

by choosing for y_j the last timestamp in w that has the same fractional part as x_j. The length of the vector $\lambda(w)$ depends on the orbit i. Lemma 20 says that the vector $\lambda(w)$ is always defined, and also all of the coordinates of $\lambda(w)$ are at distance at most M from the last timestamp in w. Finally, define a vector

$$\Delta(w) = (x_1 - y_1, \ldots, x_{n_i} - y_{n_i}). \tag{16}$$

By definition of $\lambda(w)$, the vector $\Delta(w)$ contains only integers.

Lemma 21. *The set* $\Delta \stackrel{def}{=} \{\Delta(w) : w \in A^*\}$ *is finite.*

Proof. Using Lemma 4 we immediately obtain:

Claim. Let $n \in \mathbb{N}$, let X be a single orbit in $\mathbb{R}^n$, and let $f : X \to \mathbb{R}$ be an equivariant function. The following set has bounded span.

$$\{(x_1, \ldots, x_n, f(x_1, \ldots, x_n)) : (x_1, \ldots, x_n) \in X\} \subseteq \mathbb{R}^{n+1}.$$

Consider the function which maps a word w to three pieces of information:

1. The state $(i, x_1, \ldots, x_{n_i})$ of the syntactic automaton after reading w.
2. The number $\mathit{minfuture}(w)$.
3. The last timestamp in w.

Using property 3, one shows that the distance between the *minfuture*(w) and the last timestamp in w is bounded by M. As we have observed previously, the second piece of information is a function of the first piece of information, and therefore by the claim above, the possible values for the first and second pieces of information have bounded span, when ranging over all words. It follows that the values for all three pieces of information have bounded span, which gives the lemma. □

Consider now the function γ, which maps a word to

$$\gamma(w) = (i, \Delta(w), \lambda(w)),$$

where $i \in \{1, \ldots, n\}$ identifies the orbit of the state $\delta^*(w)$.

Lemma 22. *The function γ is equivariant, and its image $\gamma(A^*)$ is orbit-finite.*

Proof. The function λ has orbit-finite image, while the other components i and $\Delta(w)$ come from a finite set. □

Lemma 23. *The function δ^* factors through γ, i.e. there is an equivariant function $f : \gamma(A^*) \to Q$ such that $\delta^* = f \circ \gamma$.*

Proof. The orbit is already stored in $\gamma(w)$, while the register values in $\delta^*(w)$ can be computed by adding the vectors $\lambda(w)$ and $\Delta(w)$:

$$f(i, \boldsymbol{x}, \boldsymbol{y}) = (i, \boldsymbol{x} + \boldsymbol{y}),$$

according to the equality (16). □

Lemma 24. *The function γ is compositional in the following sense. There is an equivariant function*

$$\hat{\gamma} : \gamma(A^*) \times A \to \gamma(A^*) \tag{17}$$

such that for every word $w \in A^*$ and every letter $a \in A$,

$$\gamma(w \cdot a) = \hat{\gamma}(\gamma(w), a)$$

Proof. Recall that the transition function δ of $\mathcal{A}$ is defined by constraints. Recall also that the function $f : \gamma(A^*) \to Q$ acts by adding vectors. Then the constraints that define δ may be lifted along f to define a function (17) such that $\hat{\gamma}$, δ and f commute:

$$f \circ \hat{\gamma} = \delta \circ f.$$

Being defined by constaints, $\hat{\gamma}$ is automatically equivariant.

□

We define an automaton $\mathcal{B}$ as follows. Its states are $\gamma(A^*)$; the state space is orbit-finite by Lemma 22. The transition function is $\hat{\gamma}$ from Lemma 24. The accepting and initial states are obtained by taking the inverse images under the function f from Lemma 23. The automaton recognizes the same language as the syntactic automaton thanks to Lemma 23. By definition of $\lambda(w)$, the registers of $\mathcal{B}$ only store recently seen timestamps, and therefore the automaton $\mathcal{B}$ is equivalent to a timed automaton thanks to Lemma 19.