A Myhill-Nerode Style Characterization for Timed Automata with Integer Resets

Kyveli Doveri
University of Warsaw, Poland

Pierre Ganty
IMDEA Software Institute, Pozuelo de Alarcón, Madrid, Spain

B. Srivathsan
Chennai Mathematical Institute, India
CNRS IRL 2000, ReLaX, Chennai, India

Abstract

The well-known Nerode equivalence for finite words plays a fundamental role in our understanding of the class of regular languages. The equivalence leads to the Myhill-Nerode theorem and a canonical automaton, which in turn, is the basis of several automata learning algorithms. A Nerode-like equivalence has been studied for various classes of timed languages.

In this work, we focus on timed automata with integer resets. This class is known to have good automata-theoretic properties and is also useful for practical modeling. Our main contribution is a Nerode-style equivalence for this class that depends on a constant K. We show that the equivalence leads to a Myhill-Nerode theorem and a canonical one-clock integer-reset timed automaton with maximum constant K. Based on the canonical form, we develop an Angluin-style active learning algorithm whose query complexity is polynomial in the size of the canonical form.

2012 ACM Subject Classification Theory of computation → Formal languages and automata theory

Keywords and phrases Timed languages, Timed automata, Canonical representation, Myhill-Nerode equivalence, Integer reset

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2024.21

Related Version *Full Version*: `https://arxiv.org/abs/2410.02464` [9]

Funding *Kyveli Doveri*: Supported by the ERC grant INFSYS, agreement no. 950398. Results partly obtained when previously affiliated with the IMDEA Software Institute.
Pierre Ganty: This publication is part of the grant PID2022-138072OB-I00, funded by MCIN/AEI/10.13039/501100011033/FEDER, UE.

1 Introduction

A cornerstone in our understanding of regular languages is the Myhill-Nerode theorem. This theorem characterizes regular languages in terms of the Nerode equivalence $\sim_L$: for a word w we write $w^{-1}L = \{z \mid wz \in L\}$ for the *residual language* of w w.r.t. L; and for two words u, v we say $u \sim_L v$ if $u^{-1}L = v^{-1}L$.

▶ **Theorem 1.1** (Myhill-Nerode theorem). *Let L be a language of finite words.*
- *L is regular iff the Nerode equivalence has a finite index.*
- *The Nerode equivalence is coarser than any other monotonic L-preserving equivalence.*

An equivalence is said to be *monotonic* if $u \approx v$ implies $ua \approx va$ for all letters a and is L-preserving if each equivalence class is either contained in L or disjoint from L. [1] An equivalence over words being monotonic makes it possible to construct an automaton with

[1] The exact term would be "right monotonic" because it only considers concatenation to the right of the word. Throughout the paper we simply write monotonic to keep it short but we mean right monotonic.

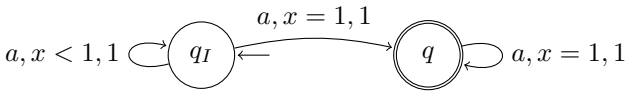

Figure 1 Automaton accepting $L = \{(t_1 \cdot a) \dots (t_n \cdot a) \mid t_1 + \cdots + t_n = 1\}$ with alphabet $\Sigma = \{a\}$.

states being the equivalence classes. The Nerode equivalence being the coarsest makes the associated automaton the minimal (and a canonical) deterministic automaton for the regular language. Our goal in this work is to obtain a similar characterization for certain subclasses of timed languages.

Timed languages and timed automata were introduced by Alur and Dill [1] as a model for systems with real-time constraints between actions. Ever since its inception, the model has been extensively studied for its theoretical aspects and practical applications. In this setting, words are decorated with a delay between consecutive letters. A *timed word* is a finite sequence $(t_1 \cdot a_1)(t_2 \cdot a_2) \cdots (t_n \cdot a_n)$ where each $t_i \in \mathbb{R}_{\geq 0}$ and each a_i is a letter taken from a finite set Σ called an *alphabet*. A timed word associates a time delay between letters: a_1 was seen after a delay of t_1 from the start, the next letter a_2 appears t_2 time units after a_1, and so on. Naturally, a timed language is a set of timed words. A timed automaton is an automaton model that recognizes timed languages. Figures 1 and 2 present some examples (formal definitions appear later). In essence, a timed automaton makes use of *clocks* to constrain time between the occurrence of transitions. In Figure 1, the variable x denotes a clock. The transition labels are given by triples comprising a letter (e.g. a), a clock constraint (e.g. $x = 1$), and a multiplicative factor (0 or 1) for the clock update. Intuitively, the semantics of the transition from q_I to q is as follows: the automaton reads the letter a when the value of the clock held in x is exactly 1 and updates the clock value to $1 \times x$. If the third element of the transition label is 0, then the transition updates the value of x to $0 \times x = 0$. We refer to the second element of the transition label as the transition *guard* and the third element as the *reset*. It is worth mentioning that the transition guards feature constants given by integer values, meaning that a guard like $x = 0.33$ is not allowed. Next, we argue how challenging it is to define a Nerode-style equivalence for timed languages.

Challenge 1. *The Nerode equivalence lifted as it is has infinitely many classes.* For example, the timed automaton of Figure 1 accepts a timed word $(t_1 \cdot a) \dots (t_n \cdot a)$ as long as $t_1 + \cdots + t_n = 1$. The timed language L of that automaton has infinitely many quotients. Indeed let $0 < t_1 < 1$, we have that $(t_1 \cdot a)^{-1} L = \{(t_2 \cdot a) \dots (t_n \cdot a) \mid t_2 + \cdots + t_n = 1 - t_1\}$. Observe that different values for t_1 yield different quotients, hence L has uncountably many quotients.

Challenge 2. *Two words with the same residual languages may never go to the same control state in any timed automaton.* Figure 2 gives an example of a timed language that exhibits this challenge. Consider the words $u = (0.5 \cdot a)$ and $v = (1.5 \cdot a)$. The residual of both these words is the singleton language $\{(0.5 \cdot b)\}$. Suppose both u and v go to the same control state q in the timed automaton. After reading u (resp. v), clocks which are possibly reset will be 0, whereas the others will be 0.5 (resp. 1.5). Suppose v is accepted via a transition sequence $q_I \to q \to q_F$. Since guards contain only integer constants, the guard on $q \to q_F$ should necessarily be of the form $x = 2$ for some clock x which reaches q with value 1.5. The same transition can then be taken from u to give $u(2 \cdot b)$ or $u(1.5 \cdot b)$ depending on the value of x after reading u. A contradiction. This example shows there is no hope to identify states of a timed automaton through quotients of a Nerode-type equivalence. The equivalence that we are aiming for needs to be stronger, and further divide words based on some past history.

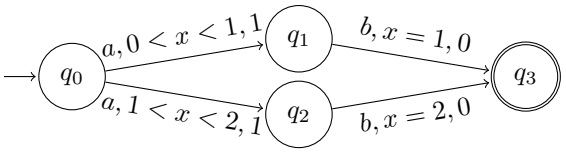

Figure 2 Automaton accepting $L_2 = \{(t_1 \cdot a)(t_2 \cdot b) \mid \text{either } 0 < t_1 < 1 \text{ and } t_1 + t_2 = 1, \text{ or } 1 < t_1 < 2 \text{ and } t_1 + t_2 = 2\}$ with alphabet $\Sigma = \{a, b\}$.

Challenge 3. *The Nerode-style equivalence should be amenable to a timed automaton construction.* In the case of untimed word languages, monotonicity of the Nerode equivalence immediately led to an automaton construction. We need to find the right notion of monotonicity for the class of automata that we want to build from the equivalence.

A machine independent characterization for deterministic timed languages has been studied by Bojańczyk and Lasota [6]. They circumvented the above challenges by considering a new automaton model *timed register automata* that generalizes timed automata. This automaton model makes use of registers to store useful information, for instance for the language in Figure 2, a register stores the value 0.5 after reading $(0.5 \cdot a)$ and $(1.5 \cdot a)$. This feature helps in resolving Challenge 2. For the question of finiteness mentioned in Challenge 1, timed register automata are further viewed as a restriction of a more general model of automata that uses the abstract concept of *Frankel-Mostowski* sets in its definition. Finiteness is relaxed to a notion of *orbit-finiteness*.

The work of An et al. [3] takes another approach to these challenges by considering a subclass of timed languages which are called *real-time languages*. These are languages that can be recognized using timed automata with a single clock that is reset in every transition. Therefore, after reading a letter, the value of the clock is always 0. This helps in solving the challenges, resulting in a canonical form for real-time languages.

Our work. As we have seen, to get a characterization which also lends to an automaton construction, either the automaton model has been modified or the characterization is applied to a class of languages where the role of the clock is restricted to consecutive letters. Our goal is to continue working with the same model as timed automata and apply a characterization to a different subclass.

In this work, we look at languages recognized by timed automata with integer resets (IRTA). These are automata where clock resets are restricted to transitions that contain a guard of the form $x = c$ for some clock x and some integer c [17]. The class of languages recognized by IRTA is incomparable with real-time languages. Moreover, it is known that IRTA can be reduced to 1-clock-deterministic IRTA [15], or 1-IRDTA for short. The proof of this result effectively computes, given an IRTA, a timed language equivalent 1-IRDTA. Here is our main result which gives a Myhill-Nerode style characterization for IRTA languages.

▶ **Theorem 1.2.** *Let L be a timed language.*

- *L is accepted by a timed automaton with integer resets iff there exists a constant K such that $\approx^{L,K}$ is K-monotonic and has a finite index.*
- *The $\approx^{L,K}$ equivalence is coarser than any K-monotonic L-preserving equivalence.*

Intuitively, one should think of K as the largest integer that needs to appear in the guards of an accepting automaton. The goal of the paper is to identify the notion of K-monotonicity and the equivalence $\approx^{L,K}$ that exhibit the above theorem. The characterization also leads to a canonical form for IRTA. In practice, the integer reset assumption allows for modeling multiple situations [17].

To the best of our knowledge, there is no learning algorithm that can compute an IRTA for systems that are known to satisfy the integer reset assumption. We fill this gap and show how Angluin's style learning [4] can be adapted to learn 1-IRDTA.

Related work. Getting a canonical form for timed languages has been studied in several works: [6] and [14] focus on a machine independent characterization for deterministic timed languages, whereas the works [10, 3, 21] extend the study of the canonical forms to an active learning algorithm. Languages accepted by event-recording automata are a class of languages where the value of the clocks is determined by the input word. This helps in coming up with a canonical form [10]. In [21], the author presents a Myhill-Nerode style characterization for deterministic timed languages by making use symbolic words rather than timed words directly. The author shows that the equivalence has a finite index iff the language is recognizable (under the notion of recognizability using right-morphisms proposed by Maler and Pnueli [14]). Further, Maler and Pnueli have given an algorithm to convert recognizable timed languages to deterministic timed automata, which resets a fresh clock in every transition and makes use of clock-copy updates $x := y$ in the transitions. It is known that automata with such updates can be translated to classical timed automata [16, 7].

Learning timed automata is a topic of active research. The foundations of timed automata learning were laid in the pioneering work of Grinchtein et al. [10] by providing a canonical form for event-recording automata (ERAs). These are automata having a clock for each letter in the alphabet, and a clock x_a records the time since the last occurrence of a. The canonical form essentially considers a separate state for each region. Since there are as many clocks as the number of letters, there are at least $|\Sigma|!$ number of regions. This makes the learning algorithm prohibitively expensive to implement. In contrast, as we will see, we are able to convert IRTAs into a subclass of single-clock IRTAs. If K is the maximum constant, there are only $2K + 2$ many regions. Later works on learning ERAs have considered identifying other forms of automata that merge the states of the canonical form [12, 11, 13]. Other models for learning timed systems consider one-clock timed automata [22, 2] and Mealy machines with timers [8, 19]. Approaches other than active learning for timed automata include passive learning of discrete timed automata [20] and learning timed automata using genetic programming [18].

We have considered a subclass of deterministic timed languages. Therefore, our class does fall under the purview of the [21, 14] work – however, the fundamental difference is that we continue to work with timed words and not symbolic timed words. This gives an alternate perspective and a direct and simpler 1-clock IRTA construction. The simplicity and directness also apply when it comes to learning 1-clock IRTA.

Outline of the paper. In Section 3, we define the class of one-clock languages with integer resets, and their acceptors thereof: the 1-clock Timed Automata (or 1-TA) with clock constraints given by region equivalence classes and transitions that always reset on integer clock values. Section 4 puts forward a notion of K-monotonicity and characterizates K-monotonic equivalences as a certain type of integer reset automata. Subsequently, Section 5 presents the Nerode-style equivalence, the Myhill-Nerode theorem and some examples applying the theorem. Finally, in Section 6, we give an algorithm to compute and learn the canonical form.

2 Background

Words and languages. An *alphabet* is a finite set of *letters* which we typically denote by Σ. An *untimed word* is a finite sequence $a_1 \cdots a_n$ of letters $a_i \in \Sigma$. We denote by Σ^* the set of untimed words over Σ. An *untimed language* is a subset of Σ^*. A *timestamp* is a finite sequence of non-negative real numbers. We denote the latter set by $\mathbb{R}_{\geq 0}$ and the set of all timestamps by $\mathbb{T}$. A *timed word* is a finite sequence $(t_1 \cdot a_1) \cdots (t_n \cdot a_n)$ where $a_1 \cdots a_n \in \Sigma^*$ and $t_1 \cdots t_n \in \mathbb{T}$. We denote the set of timed words by $\mathbb{T}\Sigma^*$. Given a timed word $u = (t_1 \cdot a_1) \ldots (t_n \cdot a_n) \in \mathbb{T}\Sigma^*$ we denote by $\sigma(u)$ the sum $t_1 + \cdots + t_n$ of its timestamps. A *timed language* is a set of timed words. As usual, we denote the empty (un)timed word by ϵ. The *residual language* of an (un)timed language L with regard to a (un)timed word u is defined as $u^{-1}L = \{w \mid uw \in L\}$. Therefore it is easy to see that $\epsilon^{-1}L = L$ for every (un)timed language L.

Timed automata [1] are recognizers of timed languages. Since we focus on subclasses of timed automata with a single clock, we do not present the definition of general timed automata. Instead, we give a modified presentation of one-clock timed automata that will be convenient for our work.

One-clock timed automata. A *One-clock Timed Automaton* (1-TA) over Σ is a tuple $\mathcal{A} = (Q, q_I, T, F)$ where Q is a finite set of states, $q_I \in Q$ is the initial state, $F \subseteq Q$ is the set of final states and $T \subseteq Q \times Q \times \Sigma \times \Phi \times \{0, 1\}$ is a finite set of transitions where Φ is the set of clock constraints given by

$$\phi ::= x < m \quad | \quad m < x \quad | \quad x = m \quad | \quad \phi \wedge \phi \ , \text{ where } m \in \mathbb{N}.$$

For a clock constraint ϕ, we write $[\![\phi]\!]$ for the set of non-negative real values for x that satisfies the constraint. Notice that we have disallowed guards of the form $x \leq m$ which appear in standard timed automata literature, since its effect can be captured using two transitions, one with $x = m$ and another with $x < m$. A transition is a tuple (q, q', a, ϕ, r) where ϕ is a clock constraint called the *guard* of the transition and $r \in \{0, 1\}$ denotes whether the single clock x is *reset* in the transition.

We say that a 1-TA with transitions T is *deterministic* whenever for every pair $\theta = (q, q', a, \phi, r)$ and $\theta_1 = (q_1, q_1', a_1, \phi_1, r_1)$ of transitions in T such that $\theta \neq \theta_1$ we have that either $q \neq q_1$, $a \neq a_1$ or $[\![\phi]\!] \cap [\![\phi_1]\!] = \emptyset$.

A *run* of $\mathcal{A}$ on a timed word $(t_1 \cdot a_1) \ldots (t_k \cdot a_k) \in \mathbb{T}\Sigma^*$ is a finite sequence

$$e = (q_0, \nu_0) \xrightarrow{t_1, \theta_1} (q_1, \nu_1) \xrightarrow{t_2, \theta_2} \cdots \xrightarrow{t_k, \theta_k} (q_k, \nu_k) \ ,$$

where $\{q_0, \ldots, q_k\} \subseteq Q$, $\{\nu_0, \ldots, \nu_k\} \subseteq \mathbb{R}_{\geq 0}$ and for each $i \in \{1, \ldots, k\}$ the following hold: $\theta_i \in T$ and θ_i is of the form $(q_{i-1}, q_i, a_i, \phi_i, r_i)$, $\nu_{i-1} + t_i \in [\![\phi_i]\!]$, and $\nu_i = r_i(\nu_{i-1} + t_i)$. Therefore if $r_i = 0$, we have $\nu_i = 0$ and if $r_i = 1$ we have $\nu_i = \nu_{i-1} + t_i$. A pair $(q, \nu) \in Q \times \mathbb{R}_{\geq 0}$ like the ones occurring in the run e is called a *configuration* of $\mathcal{A}$ and the configuration $(q_I, 0)$ is called *initial*. The run e is deemed *accepting* if $q_k \in F$.

For $w \in \mathbb{T}\Sigma^*$ we write $(q, \nu) \rightsquigarrow^w (q', \nu')$ if there is a run of $\mathcal{A}$ on w from (q, ν) to (q', ν'). Observe that if $\mathcal{A}$ is deterministic then for every timed word w there is at most one run on w starting from the initial configuration. We say that $\mathcal{A}$ is complete if every word admits a run. In the rest, we will always assume, without loss of generality, that our timed automata are complete. Finally, given a configuration (q, x) define $\mathcal{L}(q, x) = \{w \in \mathbb{T}\Sigma^* \mid (q, x) \rightsquigarrow^w (p, \nu), p \in F, \nu \in \mathbb{R}_{\geq 0}\}$, hence define $L(\mathcal{A})$ as $\mathcal{L}(q_I, 0)$.

Equivalence relation. A relation $\sim \subseteq S \times S$ on a set S is an *equivalence* if it is reflexive (i.e. $x \sim x$), transitive (i.e. $x \sim y \wedge y \sim z \implies x \sim z$) and symmetric (i.e. $x \sim y \implies y \sim x$). The equivalence class of $s \in S$ w.r.t. $\sim$ is the subset $[s]_\sim = \{s' \in S \mid s \sim s'\}$. A *representative* of the class $[s]_\sim$ is any element $s' \in [s]_\sim$. Given a subset D of S we define $[D]_\sim = \{[d]_\sim \mid d \in D\}$. We say that $\sim$ has *finite index* when $[S]_\sim$ is a finite set. An important notion in the analysis of timed automata is the *region equivalence* which we recall next for one-clock timed automata.

Region equivalence. Given a constant $K \in \mathbb{N}$ define the equivalence $\equiv^K \subseteq \mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0}$ by

$$x \equiv^K y \iff \big(\lfloor x \rfloor = \lfloor y \rfloor \wedge (\{x\} = 0 \Leftrightarrow \{y\} = 0)\big) \vee (x > K \wedge y > K) \ ,$$

where given $x \in \mathbb{R}_{\geq 0}$ we denote by $\lfloor x \rfloor$ its integral part and by $\{x\}$ its fractional part.

3 Languages with Integer Resets

We are interested in timed languages recognized by IRTAs. It is known that IRTAs can be converted to 1-clock deterministic IRTAs [15]. The key idea is that in any reachable valuation of an IRTA, all clocks have the same fractional value. Therefore, the integral values of all clocks can be encoded inside the control state, and the fractional values can be read from a single clock. In the sequel, we simply define IRTAs with a single clock, due to the equi-expressivity.

We define the class of one-clock integer-reset timed automata (1-IRTA) where transitions reset the clock provided its value is an integer. Formally, we say that a 1-TA $\mathcal{A} = (Q, q_I, T, F)$ is a 1-IRTA when for every resetting transition $(q, q', a, \phi, 0) \in T$ the clock constraint ϕ is of the form $x = m$, or, equivalently, $[\![\phi]\!] \in \mathbb{N}$. A deterministic 1-IRTA is called a 1-IRDTA.

▶ **Example 3.1.** The one-clock timed automata in Figures 1, 2 and 3 are all 1-IRTAs.

The definition of a run of a 1-IRTA on a timed word simply follows that of 1-TA. However, for 1-IRTA, we can identify positions in input timed words where resets can potentially happen. The next definition makes this idea precise.

▶ **Definition 3.2.** *Given $d = t_1 \cdots t_n \in \mathbb{T}$ and a $K \in \mathbb{N}$, we define the longest sequence of indices $s_d = \{0 = i_0 < i_1 < \cdots < i_p \leq n\}$ such that for every $j \in \{0, \ldots, p-1\}$ the value $\sum_{i=(i_j)+1}^{i_{(j+1)}} t_i$ is an integer between 0 and K. We refer to the set of positions of the sequence s_d as the* integral positions *of d. Note that s_d is never empty since it always contains 0. Next, define*

$$c^K(d) = \sum_{i=(i_p)+1}^{n} t_i \ .$$

The definitions of integral positions and the function c^K apply equally to timed words by taking the timestamp of the timed word.

Notice that the sequence s_d depends on the constant K (we do not explicitly add K to the notation for simplicity, as in our later usage, K will be clear from the context). Note also that when $i_p = n$ then $c^K(d) = 0$, otherwise $c^K(d)$ can be any real value except an integer value between 0 and K, i.e. $c^K(d) \in \mathbb{R}_{\geq 0} \setminus \{0, \ldots, K\}$.

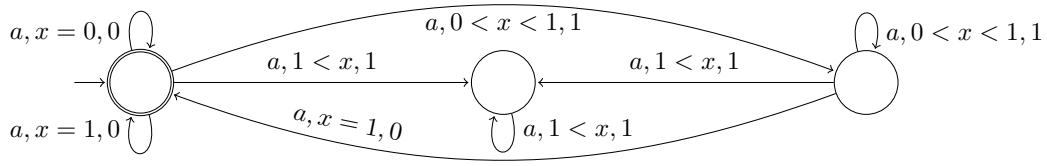

Figure 3 A strict 1-IRTA with alphabet $\Sigma = \{a\}$ accepting $M = \{u \in \mathbb{T}\Sigma^* \mid c^1(u) = 0\}$.

▶ **Example 3.3.** For $K = 1$ and $u = (0.2 \cdot a)(0.8 \cdot a)(0.2 \cdot a)$ we have $s_u = \{0 < 2\}$ and $c^1(u) = 0.2$. For $K = 1$ and $u' = (1.2 \cdot a)(0.8 \cdot a)(0.2 \cdot a)$ we have $s_{u'} = \{0\}$ and $c^1(u') = 2.2$. For $K = 2$, we have $s_{u'} = \{0 < 2\}$ and $c^2(u') = 0.2$.

Consider a run of a 1-IRTA on a word $u = (t_1 \cdot a_1) \cdots (t_n \cdot a_n) \in \mathbb{T}\Sigma^*$ and factor it according to $s_u = \{0 = i_0 < i_1 < \cdots < i_p \leq n\}$:

$$(q_{i_0}, \nu_{i_0}) \xrightarrow{t_{(i_0)+1}, \theta_{(i_0)+1} \cdots t_{i_1}, \theta_{i_1}} (q_{i_1}, \nu_{i_1}) \xrightarrow{t_{(i_1)+1}, \theta_{(i_1)+1} \cdots t_{i_2}, \theta_{i_2}} (q_{i_2} \nu_{i_2}) \to \cdots$$
$$\to (q_{i_p}, \nu_{i_p}) \xrightarrow{t_{(i_p)+1}, \theta_{(i_p)+1} \cdots t_n, \theta_n} (q_n, \nu_n)$$

At each position i_j with $j \in \{0, \ldots, p\}$, $\nu_{i_j} \in \mathbb{N}$ and, moreover, $\nu_{i_j} = 0$ when $r_{i_j} = 0$.

In Definition 3.2 we identified the integral positions at which a 1-IRTA *could potentially* reset the clock. In the following, we recall a subclass of 1-IRTAs called strict 1-IRTAs [5] where every transition with an equality guard ϕ (ϕ is of the form $x = m$ or, equivalently, $[\![\phi]\!] \in \mathbb{N}$) *must* reset the clock. This feature, along with a special requirement on guards forces a reset on every position given by s_u for a word u.

Strict 1-IRTA

A 1-IRTA is said to be *strict* if there exists $K \in \mathbb{N}$ such that for each of its transitions (q, q', a, ϕ, r) the following holds:

1. the clock constraint of the guard ϕ is either $x = m$, $m < x \wedge x < m + 1$, or $K < x$,
2. the clock constraint of the guard ϕ is an equality iff $r = 0$.

▶ **Example 3.4.** The 1-IRTA in Figures 2 and 3 are strict 1-IRTAs whereas the one in Figure 1 is not strict since the transition $q_I \xrightarrow{a, x=1, 1} q$ does not reset the clock. To make it strict while accepting the same language, we replace the transitions of guard $x = 1$ by resetting transitions of guard $x = 0$ and, split the transition $q_I \xrightarrow{a, x<1, 1} q_I$ into $q_I \xrightarrow{a, 0<x<1, 1} q_I$ and $q_I \xrightarrow{a, x=0, 0} q_I$.

A run of a strict 1-IRTA on a word u can be factored similarly as explained for a general 1-IRTA, however now, every r_{i_j} will be a reset transition: notice that we require each transition to be guarded using constraints of a special form, either $x = m$ or $m < x < m + 1$ or $K < x$; therefore, the transition reading (t_{i_1}, a_{i_1}) will necessarily have an equality guard $x = m$ forcing a reset, similarly at i_2 and so on. Therefore, the sequence s_u identifies the exact reset points in the word, no matter which strict 1-IRTA reads it. The quantity $c^K(u)$ gives the value of the clock on reading u by any strict 1-IRTA. This *input-determinism* is a fundamental property of strict 1-IRTAs that helps in the Myhill-Nerode characterization that we present in the later sections.

The question now is how expressive are strict 1-IRTAs. As shown by the proposition below, every language definable by a 1-IRTA is also definable by a strict 1-IRTA. Therefore, we could simply consider strict 1-IRTAs instead of 1-IRTAs. Even though a proof of this equi-expressivity theorem is known [5] we provide one in the full version [9].

▶ **Proposition 3.5** (see also Theorem 1 [5]). *A language accepted by a (deterministic) 1-IRTA is also accepted by a (deterministic) strict 1-IRTA with no greater constant in guards.*

4 From Equivalences to Automata and Back

We start the study of equivalences for languages accepted by integer reset automata. Proposition 3.5 says that every IRTA language can be recognized by a strict 1-IRDTAs. There are two advantages of strict 1-IRDTAs: there is a single clock; and the value of the clock on reading the word is simply determined by the word and not by the automaton that is reading it. This motivates us to restrict our attention to equivalences that make use of the quantity $c^K(u)$, and from which one can construct a strict 1-IRDTA with states as the equivalence classes. In order to be able to do so, we need a good notion of monotonicity (Challenge 3 of the Introduction).

Intuitively, the equivalence should satisfy two conditions whenever u is equivalent to v: (1) when u can elapse time and satisfy a guard, v should be able to elapse some time and satisfy the same guard, and (2) all one step extensions of u and v, say $u' = u(t \cdot a)$ and $v' = v(t' \cdot a)$ such that the clock values $c^K(u')$ and $c^K(v')$ satisfy the same set of guards w.r.t constant K, should be made equivalent. Each guard in a strict 1-IRDTA represents a K-region. All these remarks lead to the following definition of K-monotonicity.

▶ **Definition 4.1** (L-preserving, K-monotonic). *An equivalence $\approx \subseteq \mathbb{T}\Sigma^* \times \mathbb{T}\Sigma^*$ is L-*preserving *when $u \approx v \implies (u \in L \iff v \in L)$. Given a constant $K \in \mathbb{N}$, $\approx \subseteq \mathbb{T}\Sigma^* \times \mathbb{T}\Sigma^*$ is K-*monotonic *when $u \approx v$ implies:*
(a) $c^K(u) \equiv^K c^K(v)$, *and*
(b) $\forall a \in \Sigma, \forall t, t' \in \mathbb{R}_{\geq 0} : \; c^K(u) + t \equiv^K c^K(v) + t' \implies u(t \cdot a) \approx v(t' \cdot a)$.

From a K-monotonic equivalence, we can construct a strict 1-IRDTA whose states are the equivalence classes. Here is additional notation. For a number $t \in \mathbb{R}_{\geq 0}$, we define a clock constraint $\phi(\ [t]_{\equiv^K}\ )$ as:

$$\phi(\ [t]_{\equiv^K}\ ) = \begin{cases} x = t & \text{if } t \leq K \wedge t \in \mathbb{N}\ , \\ \lfloor t \rfloor < x \wedge x < \lfloor t \rfloor + 1 & \text{if } t \leq K \wedge t \notin \mathbb{N}\ , \\ K < x & \text{if } K < t\ . \end{cases}$$

▶ **Definition 4.2** (From equivalence $\approx$ to strict 1-IRDTA $\mathcal{A}_\approx$). *Let $L \subseteq T\Sigma^*$, $K \geq 0$, and $\approx$ a K-monotonic, L-preserving equivalence with finite index. The strict 1-IRDTA $\mathcal{A}_\approx$ has states $\{[u]_\approx \mid u \in T\Sigma^*\}$. The initial state is $[\epsilon]_\approx$. Final states are $\{[u]_\approx \mid u \in L\}$. Between two states $[u]_\approx$ and $[v]_\approx$ there is a transition $([u]_\approx, [v]_\approx, a, g, s)$ if there exists $t \in \mathbb{R}_{\geq 0}$ such that:*
- *$u(t \cdot a) \approx v$, and*
- *$g = \phi(\ [c^K(u) + t]_{\equiv^K}\ )$, and*
- *$s = 0$ if $c^K(u) + t \in \{0, 1, \ldots, K\}$ and $s = 1$ otherwise.*

We now explain why the above definition does not depend on the representative picked from an equivalence class. Suppose $([u]_\approx, [v]_\approx, a, g, s)$ is a transition. Let $t \in \mathbb{R}_{\geq 0}$ be a value which witnesses the transition, that is, it satisfies the conditions of the above definition.

Pick another word u' equivalent to u, that is, $u \approx u'$. By Definition 4.1 (a), we have $c^K(u) \equiv^K c^K(u')$. Therefore, there exists t' such that $c^K(u) + t \equiv^K c^K(u') + t'$. Moreover by (b), $u'(t' \cdot a) \approx u(t \cdot a)$, and hence $u'(t' \cdot a) \approx v$. Therefore, we observe that even if we had chosen u' instead of u, we get a witness t' for the same transition $([u]_\approx, [v]_\approx, a, g, s)$.

▶ **Lemma 4.3.** *Let $\approx$ be an L-preserving, K-monotonic equivalence with finite index. Then $\mathcal{L}(\mathcal{A}_\approx) = L$.*

Proof. By induction on the length of the timed words we show that for every $u \in \mathbb{T}\Sigma^*$, $([\epsilon]_\approx, 0) \rightsquigarrow^u ([u]_\approx, c^K(u))$. Let $u(t \cdot a) \in \mathbb{T}\Sigma^*$. Assume $([\epsilon]_\approx, 0) \rightsquigarrow^u ([u]_\approx, c^K(u))$. By definition of $\mathcal{A}_\approx$, there is a transition $([u]_\approx, [u(t \cdot a)]_\approx, a, g, s)$ such that $g = \phi([c^K(u) + t]_{\equiv^K})$ and, $s = 0$ if $c^K(u) + t \in \{0, 1, \ldots, K\}$ and $s = 1$ otherwise. Since $c^K(u(t \cdot a)) = (c^K(u) + t)s$ we deduce that $([\epsilon]_\approx, 0) \rightsquigarrow^u ([u]_\approx, c^K(u)) \rightsquigarrow^{(t \cdot a)} ([u(t \cdot a)]_\approx, c^K(u(t \cdot a)))$. Finally, since $\approx$ is L-preserving, a word is in L iff $\mathcal{A}_\approx$ accepts it. ◀

We now look at the reverse question of obtaining a monotonic equivalence from an automaton. Given a complete strict 1-IRDTA $\mathcal{B}$, we define an equivalence $\approx_\mathcal{B}$ as $u \approx_\mathcal{B} v$ if $\mathcal{B}$ reaches the same (control) state on reading u and v from its initial configuration. If K is the maximum constant appearing in $\mathcal{B}$, it is tempting to think that $\approx_\mathcal{B}$ is a K-monotonic equivalence. However $\approx_\mathcal{B}$ need not satisfy condition **(a)** of Definition 4.1. For instance, consider a strict 1-IRDTA $\mathcal{B}$ which has two self-looping transitions in its initial state: $q \xrightarrow{a, x=0, 0} q$ and $q \xrightarrow{a, 0<x<1, 1} q$. Observe that $(0 \cdot a) \approx_\mathcal{B} (0.5 \cdot a)$, but $c^1((0 \cdot a)) \neq c^1((0.5 \cdot a))$. Therefore, the state-based equivalence $\approx_\mathcal{B}$ needs to be further refined in order to satisfy monotonicity. This leads us to define an equivalence $\approx_\mathcal{B}^K$ as: $u \approx_\mathcal{B}^K v$ if $u \approx_\mathcal{B} v$ and $c^K(u) = c^K(v)$.

▶ **Lemma 4.4.** *Let $\mathcal{B}$ be a complete strict 1-IRDTA with maximum constant K. The equivalence $\approx_\mathcal{B}^K$ is $\mathcal{L}(\mathcal{B})$-preserving, K-monotonic and has finite index.*

Proof. The equivalence $\approx_\mathcal{B}^K$ is $\mathcal{L}(\mathcal{B})$-preserving because equivalent words reach the same state in $\mathcal{B}$, thus either both are accepted or both are rejected. It has finite index because the number of its equivalence classes is bounded by the number of states of $\mathcal{B}$ multiplied by the number of K regions. Condition **(a)** and **(b)** of Def. 4.1 respectively hold by definition of $\approx_\mathcal{B}^K$ and since $\mathcal{B}$ is deterministic. ◀

The goal of the section was to go from equivalences to automata and back. Lemma 4.3 talks about equivalence-to-automata. For the automata-to-equivalence, we needed to strengthen the state-based equivalence with the region equivalence. A close look at $\mathcal{A}_\approx$ of Definition 4.2 reveals whenever $u \approx v$, we also have $c^K(u) \equiv^K c^K(v)$. So, in the equivalence-to-automata, we get an automaton satisfying a stronger property. This motivates us to explicitly highlight a class of strict 1-IRDTAs where each state can be associated with a unique region. For this class, we will be able to go from equivalence-to-automata-and-back directly.

▶ **Definition 4.5** (K-acceptor). *A K-acceptor $\mathcal{B}$ is a complete strict 1-IRDTA with maximum constant smaller than or equal to K such that for every $u, v \in T\Sigma^*$, $u \approx_\mathcal{B} v$ implies $c^K(u) \equiv^K c^K(v)$. Hence every state q of $\mathcal{B}$ can be associated to a unique K-region, denoted, region(q), i.e., whenever $(q_I, 0) \rightsquigarrow^w (q, c^K(w))$ then $c^K(w) \in$ region(q).*

Every strict 1-IRDTA $\mathcal{B}$ with maximum constant K can be converted into a K-acceptor by starting with the equivalence $\approx_\mathcal{B}^K$ and building $\mathcal{A}_{\approx_\mathcal{B}^K}$. Furthermore K-monotonic equivalences characterize K-acceptors (Lemmas 4.3 and 4.4). Therefore, for the rest of the document, we

will restrict our focus to K-acceptors. As a next task, we look for the coarsest possible K-monotonic equivalence for a language. This will give a minimal K-acceptor for the language, which we deem to be the canonical integer reset timed automaton, with maximum constant K, for the language.

5 A Nerode-style Equivalence

In the previous section, we have established generic conditions required from an equivalence to construct a K-acceptor from it. In this section, we will present a concrete such equivalence: given a language L definable by a 1-IRTA with a maximum constant $K \in \mathbb{N}$ we define a *syntactic equivalence* $\approx^{L,K} \subseteq \mathbb{T}\Sigma^* \times \mathbb{T}\Sigma^*$. "Syntactic" means this equivalence is independent of a specific representation of L. We then show that $\approx^{L,K}$ is the coarsest L-preserving and K-monotonic equivalence.

The idea for defining $\approx^{L,K}$ is to identify two words u and u' whenever $c^K(u) \equiv^K c^K(u')$ and the residuals $u^{-1}L$ and $u'^{-1}L$ coincide modulo some rescaling w.r.t. $c^K(u)$ and $c^K(u')$. We start with examples to give some intuition behind the rescaling function.

Examples. Consider an automaton segment $q_0 \xrightarrow{a,0<x<1,1} q_1 \xrightarrow{b,x=1,0} q_2 \xrightarrow{c,0<x<1,1} q_3$, with q_3 being an accepting state. The words $u = (0.2 \cdot a)$ and $v = (0.6 \cdot a)$ both go to state q_1. Let us assume that all other transitions go to a sink state, and also that the maximum constant $K = 1$. The language L accepted is $\{(t_1 \cdot a)(t_2 \cdot b)(t_3 \cdot c)\}$ where $0 < t_1 < 1$, $t_1 + t_2 = 1$ and $0 < t_3 < 1$. Moreover, $u^{-1}L = \{(0.8 \cdot b)(t_3 \cdot c) \mid 0 < t_3 < 1\}$ and $v^{-1}L = \{(0.4 \cdot b)(t_3 \cdot c) \mid 0 < t_3 < 1\}$. Here we want to somehow "equate" the residual languages $u^{-1}L$ and $v^{-1}L$. The idea is to define a bijection between these two sets $u^{-1}L$ and $v^{-1}L$. In this case, the bijection maps $(0.8 \cdot b)(t_3 \cdot c)$ to $(0.4 \cdot b)(t_3 \cdot c)$ for every t_3. Observe that given u, v the bijection depends on the values $c^K(u)$ and $c^K(v)$, which in this example are 0.2 and 0.6 respectively.

Here is another example. Let u_1, v_1 be words with $c^K(u_1) = 0.2$ and $c^K(v_1) = 0.6$. Let $u_1^{-1}L = \{(t_1 \cdot a)(t_2 \cdot b)(t_3 \cdot c)(t_4 \cdot d) \mid c^K(u_1) + t_1 + t_2 + t_3 = 1\}$ and $v_1^{-1}L = \{(t'_1 \cdot a)(t'_2 \cdot b)(t'_3 \cdot c)(t'_4 \cdot d) \mid c^K(v_1) + t'_1 + t'_2 + t'_3 = 1\}$. The bijection in this case is more complicated than the previous example. The idea is to first start with $c^K(u_1) = 0.2$, $c^K(v_1) = 0.6$ and consider a bijection f of the open unit interval $(0, 1)$ that maps the intervals $(0, 0.8]$ and $(0.8, 1)$ to $(0, 0.4]$ and $(0.4, 1)$ respectively. This bijection is essentially a rescaling of the intervals $(0, 1 - c^K(u_1)]$ and $(1 - c^K(u_1), 1)$ into the intervals $(0, 1 - c^K(v_1)])$ and $(1 - c^K(v_1), 1)$. We now pick the first letters in the residual languages $u_1^{-1}L$ and $v^{-1}L$ and create a mapping: $(t_1 \cdot a) \mapsto (f(t_1) \cdot a)$. Now we consider $c^K(u_1(t_1 \cdot a))$ and $c^K(v_1(f(t_1) \cdot a))$ in the place of $c^K(u_1), c^K(v_1)$, and continue the mapping process one letter at a time.

Rescaling function. We will now formalize this idea. We will start with bijections of the open unit interval.

Let $\lambda, \lambda' \in (0, 1)$ be arbitrary real values. Define a bijection $f_{\lambda \to \lambda'} : (0, 1) \to (0, 1)$ that scales the $(0, \lambda]$ interval to $(0, \lambda']$ and the $(\lambda, 1)$ interval to $(\lambda', 1)$:

$$f_{\lambda \to \lambda'}(t) = \begin{cases} \left(\frac{\lambda'}{\lambda}\right) t & \text{for } 0 < t \leq \lambda \\ \lambda' + \frac{(1-\lambda')}{(1-\lambda)}(t - \lambda) & \text{for } \lambda < t < 1 \end{cases}$$

Now, consider $x, x' \in \mathbb{R}_{\geq 0}$ such that $x \equiv^K x'$. We define a *length-preserving* bijection $\tau_{x\to x'} : \mathbb{T} \to \mathbb{T}$ inductively as follows: for the empty sequence ϵ, we define $\tau_{x\to x'}(\epsilon) = \epsilon$; for a timestamp $d \in \mathbb{T}$ and a $t \in \mathbb{R}_{\geq 0}$, $\tau_{x\to x'}(dt) = d't'$ where $d' = \tau_{x\to x'}(d)$ and t' is obtained as follows: let $y = c^K(xd)$ and $y' = c^K(x'd')$. If $y, y' \in \mathbb{N}$ or $y, y' > K$, then define $t' = t$. Else, define $\lfloor t' \rfloor = \lfloor t \rfloor$ and $\{t'\} = f_{(1-\{y\})\to(1-\{y'\})}(\{t\})$.

Here is an additional notation, before we describe some properties of the rescaling function. For $x_1, x_2, x_3 \in \mathbb{R}_{\geq 0}$ with $x_1 \equiv^K x_2 \equiv^K x_3$, we denote the composed function $(\tau_{x_2\to x_3}) \circ (\tau_{x_1\to x_2})$ as $\tau_{x_1\to x_2\to x_3}$. So, $\tau_{x_1\to x_2\to x_3}(t) = \tau_{x_2\to x_3}(\tau_{x_1\to x_2}(t))$.

▶ **Lemma 5.1.** *The bijection $\tau_{x\to x'}$ satisfies the following properties:*

1. *for an arbitrary timestamp $t_1 t_2 \dots t_n \in \mathbb{T}$, if $\tau_{x\to x'}(t_1 t_2 \dots t_n) = t'_1 t'_2 \dots t'_n$ then, we have $c^K(x + t_1 + \dots + t_{n-1}) + t_n \equiv^K c^K(x' + t'_1 + \dots + t'_{n-1}) + t'_n$,*
2. *$\tau^{-1}_{x\to x'}$ is identical to $\tau_{x'\to x}$,*
3. *$\tau_{x_1\to x_2\to x_3}$ is identical to $\tau_{x_1\to x_3}$.*

The rescaling function $\tau_{x\to x'}$ can be naturally extended to timed words: $\tau_{x\to x'}((t_1 \cdot a_1)(t_2 \cdot a_2) \dots (t_n \cdot a_n)) = (t'_1 \cdot a_1)(t'_2 \cdot a_2) \dots (t'_n \cdot a_n)$ where $t'_1 t'_2 \dots t'_n = \tau_{x\to x'}(t_1 t_2 \dots t_n)$. The next observation follows from Property 1. of Lemma 5.1.

▶ **Lemma 5.2.** *Let $\mathcal{B}$ be a K-acceptor, and q a control state of $\mathcal{B}$. Let $x, x' \in \mathbb{R}_{\geq 0}$ such that $x \equiv^K x'$. Then, for every timed word w: $w \in \mathcal{L}(q, x)$ iff $\tau_{x\to x'}(w) \in \mathcal{L}(q, x')$.*

Syntactic equivalence. For timed words $u, v \in T\Sigma^*$ such that $c^K(u) \equiv^K c^K(v)$, we write $\tau_{u\to v}$ for the bijection $\tau_{c^K(u)\to c^K(v)}$. We now present the main equivalence.

▶ **Definition 5.3** (Equivalence $\approx^{L,K}$)**.** *Let L be a timed language and K a natural number. We say $u \approx^{L,K} v$ if $c^K(u) \equiv^K c^K(v)$ and $\tau_{u\to v}(u^{-1}L) = v^{-1}L$.*

Note that the equivalence $\approx^{L,K}$ is L-preserving. Assume $u \approx^{L,K} v$. We have $u \in L \iff \epsilon \in u^{-1}L$ and $\epsilon \in u^{-1}L \iff \epsilon \in v^{-1}L$ since $\tau_{u\to v}(\epsilon) = \epsilon$ by definition. Thus, $u \in L \iff v \in L$.

▶ **Proposition 5.4.** *When $L \subseteq \mathbb{T}\Sigma^*$ is definable by a K-acceptor, then $\approx^{L,K}$ has finite index and is K-monotonic. Moreover, $\approx^{L,K}$ is the coarsest K-monotonic and L-preserving equivalence.*

Proof. We start by showing that $\approx^{L,K}$ is K-monotonic. Condition **(a)** of Definition 4.1 holds by definition. We move to **(b)**. Let $t_u, t_v \in \mathbb{R}_{\geq 0}$ s.t. $c^K(u) + t_u \equiv^K c^K(v) + t_v$. Let:

$$\begin{aligned} &u_1 = u(t_u \cdot a) \quad v_1 = v(t_v \cdot a) \\ &\text{To show:} \quad \tau_{u_1\to v_1}(u_1^{-1}L) = v_1^{-1}L \end{aligned} \tag{1}$$

Let $t'_v = \tau_{u\to v}(t_u)$ and $v_2 = v(t'_v \cdot a)$. We will prove 1 using these intermediate claims:

▷ Claim 5.5. $\tau_{u_1\to v_2}(u_1^{-1}L) = v_2^{-1}L$

▷ Claim 5.6. $\tau_{v_2\to v_1}(v_2^{-1}L) = v_1^{-1}L$

Hence: $\tau_{u_1\to v_2\to v_1}(u_1^{-1}L) = v_1^{-1}L$. By Lemma 5.1, we conclude $\tau_{u_1\to v_1}(u_1^{-1}L) = v_1^{-1}L$.

Proof of Claim 5.5. Let $w \in T\Sigma^*$. By definition of u_1, we have:

$$w \in u_1^{-1}L \quad \text{iff} \quad (t_u \cdot a)w \in u^{-1}L \tag{2}$$

Since $u \approx^{L,K} v$, we know that $\tau_{u\to v}(u^{-1}L) = v^{-1}L$. Therefore:

$$(t_u \cdot a)w \in u^{-1}L \quad \text{iff} \quad \tau_{u\to v}((t_u \cdot a)w) \in v^{-1}L \tag{3}$$

By the way we have constructed the rescaling function, we have

$$\tau_{u\to v}((t_u \cdot a)w) = (t'_v \cdot a)\tau_{u_1\to v_2}(w) \tag{4}$$

Finally, by definition of v_2:

$$(t'_v \cdot a)\tau_{u_1\to v_2}(w) \in v^{-1}L \quad \text{iff} \quad \tau_{u_1\to v_2}(w) \in v_2^{-1}L \tag{5}$$

From (2), (3), (4) and (5), we conclude $w \in u_1^{-1}L$ iff $\tau_{u_1\to v_2}(w) \in v_1^{-1}L$ for an arbitrary timed word w. This proves the claim. ◁

Proof of Claim 5.6. Let $\mathcal{B}$ be a K-acceptor recognizing L, and let q be the control state reached by $\mathcal{B}$ on reading word v. We first claim that:

$$c^K(v) + t_v \equiv^K c^K(v) + t'_v \tag{6}$$

This is because, by Lemma 5.1, we have $c^K(u) + t_u \equiv^K c^K(v) + t'_v$, and by assumption, we have $c^K(u) + t_u \equiv^K c^K(v) + t_v$. Since $\equiv^K$ is transitive, (6) follows.

The observation made in (6) implies on elapsing t_v or t'_v from v, the same outgoing transition is enabled, as $\mathcal{B}$ is deterministic. Therefore $v_1 = v(t_v \cdot a)$ and $v_2 = v(t'_v \cdot a)$ reach the same control state q'. Hence, (6) can be read as $c^K(v_1) \equiv^K c^K(v_2)$. By Lemma 5.2, for any timed word w, we have $w \in v_2^{-1}L$ iff $\tau_{v_2\to v_1}(w) \in v_1^{-1}L$. The claim follows. ◁

We will now show that $\approx^{L,K}$ has finite index and is also the coarsest K-monotonic, L-preserving equivalence. Let $\mathcal{B}$ be a K-acceptor for L. Recall that $u \approx^{\mathcal{B}} v$ if $\mathcal{B}$ reaches the same control state on reading u and v. We will show:

$$u \approx^{\mathcal{B}} v \quad \text{implies} \quad u \approx^{L,K} v \tag{7}$$

This will immediately prove that $\approx^{L,K}$ has finite index. Secondly, for any K-monotonic, L-preserving equivalence $\approx$, we can build a K-acceptor $\mathcal{A}_{\approx}$ (Lemma 4.3) whose equivalence is identical to $\approx$. Thus, (7) also shows that $\approx^{L,K}$ is the coarsest such equivalence.

Proof of (7) is as follows. Let q be the control state reached by u and v in $\mathcal{B}$, and let $x = c^K(u), x' = c^K(v)$. Since $\mathcal{B}$ is a K-acceptor, we also have $x \equiv^K x'$. From Lemma 5.2, $\tau_{x\to x'}(\mathcal{L}(q,x)) = \mathcal{L}(q,x')$. Hence, we deduce: $\tau_{u\to v}(u^{-1}L) = v^{-1}L$. This proves $u \approx^{L,K} v$. ◀

The above proposition leads to the following Myhill-Nerode style characterization for timed languages recognized by IRTA.

▶ **Theorem 5.7.**

(a) *$L \subseteq \mathbb{T}\Sigma^*$ is a language definable by an IRTA if and only if there is a constant $K \in \mathbb{N}$ such that $\approx^{L,K}$ is K-monotonic and has finite index.*

(b) *$\approx^{L,K}$ is coarser than any K-monotonic and L-preserving equivalence.*

The conversion from a general IRTA to a strict 1-IRTA does not increase the constant. Similarly, a strict 1-IRTA with maximum constant K can be converted to a K-acceptor. Therefore, the overall conversion from an IRTA to an acceptor preserves the constant. From the second part of Theorem 5.7, we deduce that for a language L that is recognized by an

IRTA with constant K, the K-acceptor $\mathcal{A}_{\approx^{L,K}}$ built from the equivalence $\approx^{L,K}$ has the least number of states among all K-acceptors recognizing L. It is therefore a minimal automaton among all K-acceptors for L. We now present some examples that apply this Myhill-Nerode characterization.

▶ **Example 5.8.** Consider the language $L = \{(x \cdot a) \mid x \in \mathbb{N}\}$. The right-and side of Theorem 5.7 **(a)** does not hold. Indeed, there is no $K \in \mathbb{N}$ such that $\approx^{L,K}$ verifies the K-monotonicity **(b)**: For every $K \in \mathbb{N}$ we have $c^K(\epsilon) + K + 1 \equiv^K c^K(\epsilon) + K + 1.1$. Since $(K + 1 \cdot a) \in L$ and $(K + 1.1 \cdot a) \notin L$, by L-preservation, $(K + 1 \cdot a) \not\approx^{L,K} (K + 1.1 \cdot a)$. By Theorem 5.7, L is not 1-IRTA definable.

▶ **Example 5.9.** Consider the language $L = \{(x \cdot a)(1 \cdot b) \mid 0 < x < 1\}$. This language is accepted by a 1-TA that reads a on a guard $0 < x < 1$, resets the clock x and reads b at $x = 1$. This is clearly not an IRTA. We will once again see that K-monotonicity holds for no K. For $u = (0.2 \cdot a)$, $c^K(u) + 1 \equiv^K c^K(u) + 1.1$ holds for every constant $K \in \mathbb{N}$. Since $u(1 \cdot b) \in L$ and $u(1.1 \cdot b) \notin L$, we have $u(1 \cdot b) \not\approx^{L,K} u(1.1 \cdot b)$. Thus, there is no K such that $\approx^{L,K}$ verifies K-monotonicity **(b)**, hence L is not 1-IRTA definable by Theorem 5.7.

▶ **Example 5.10.** Consider the language $M = \{u \in \mathbb{T}\Sigma^* \mid c^1(u) = 0\}$ with alphabet $\Sigma = \{a\}$ which has the following residual languages. For $u \in \mathbb{T}\Sigma^*$,

$$u^{-1}M = M \quad \text{when } c^1(u) = 0, \qquad u^{-1}M = \{v \in \mathbb{T}\Sigma^* \mid \sigma(uv) = 1\} \quad \text{when } 0 < c^1(u) < 1,$$
$$u^{-1}M = \emptyset \quad \text{when } 1 < c^1(u).$$

The equivalence classes for $\approx^{M,1}$ are the following:

$$[\epsilon]_{\approx^{M,1}} = M, \quad [(\tfrac{3}{2} \cdot a)]_{\approx^{M,1}} = \{u \in \mathbb{T}\Sigma^* \mid 1 < c^1(u)\}, \quad [(\tfrac{1}{2} \cdot a)]_{\approx^{M,1}} = \{u \in \mathbb{T}\Sigma^* \mid 0 < c^1(u) < 1\}.$$

The acceptor $\mathcal{A}_{\approx^{M,1}}$ is depicted in Figure 3 where $[\epsilon]_{\approx^{M,1}}$ is the initial (and final) state, $[(\frac{3}{2} \cdot a)]_{\approx^{M,1}}$ the sink state, and $[(\frac{1}{2} \cdot a)]_{\approx^{M,1}}$ the rightmost state.

▶ **Example 5.11.** Consider the language $L = \{(0 \cdot a)^n (0 \cdot b)^n \mid n \in \mathbb{N}\}$. There is no K such that $\approx^{L,K}$ has finite index, although $\approx^{L,0}$ is 0-monotonic: for $u \approx^{L,0} v$ with $0 < c^0(u) \equiv^0 c^0(v)$, no extension of u or v belongs to L and hence monotonicity **(b)** holds; pick u and v with $c^0(u) = c^0(v) = 0$, and let t, t' such that $c^0(u) + t \equiv^0 c^0(v) + t'$. If $t = 0$ then $t' = 0$ and monotonicity holds. Suppose $0 < t, t'$. Then $u(t \cdot \alpha) \notin L$ and $u(t' \cdot \alpha) \notin L$ for any letter α. Once again, **(b)** holds.

We now argue the infinite index, similar to the case of untimed languages. For distinct integers n and m we have $((0 \cdot a)^n)^{-1}L \neq ((0 \cdot a)^m)^{-1}L$. Since $\tau_{0 \to 0}$ is the identity we have $\tau_{0 \to 0}(((0 \cdot a)^n)^{-1}L) \neq ((0 \cdot a)^m)^{-1}L$. Thus, $(0 \cdot a)^n \not\approx^{L,K} (0 \cdot a)^m$.

▶ **Example 5.12.** Consider the language $L = \{u \in \mathbb{T}\Sigma^* \mid t_1 + \cdots + t_n = 1\}$ given in Figure 1. Given a timed word $u = (t_1 \cdot a_1) \dots (t_n \cdot a_n) \in \mathbb{T}\Sigma^*$ we denote by $\sigma(u)$ the sum $t_1 + \cdots + t_n$ of its timestamps. Also we have that $\sigma(\epsilon) = 0$. The residual languages of L are the following. For $u \in \mathbb{T}\Sigma^*$,

$$u^{-1}L = \emptyset \quad \text{when } 1 < \sigma(u), \qquad u^{-1}L = \{v \in \mathbb{T}\Sigma^* \mid \sigma(v) = 0\} \quad \text{when } \sigma(u) = 1,$$
$$u^{-1}L = L \quad \text{when } \sigma(u) = 0, \qquad u^{-1}L = \{v \in \mathbb{T}\Sigma^* \mid \sigma(u) + \sigma(v) = 1\} \quad \text{when } 0 < \sigma(u) < 1.$$

The equivalence classes for $\approx^{L,1}$ are:

$$[(1 \cdot a)]_{\approx^{L,1}} = L, \qquad [(1 \cdot a)(\tfrac{1}{2} \cdot a)]_{\approx^{L,1}} = \{u \in \mathbb{T}\Sigma^* \mid 0 < c^1(u) < 1 \wedge 1 < \sigma(u)\},$$
$$[\epsilon]_{\approx^{L,1}} = \{u \in \mathbb{T}\Sigma^* \mid \sigma(u) = 0\}, \qquad [(\tfrac{1}{2} \cdot a)]_{\approx^{L,1}} = \{u \in \mathbb{T}\Sigma^* \mid 0 < c^1(u), \sigma(u) < 1\},$$
$$[(2 \cdot a)]_{\approx^{L,1}} = \{u \in \mathbb{T}\Sigma^* \mid 1 < c^1(u)\}, \quad [(1 \cdot a)(1 \cdot a)]_{\approx^{L,1}} = \{u \in \mathbb{T}\Sigma^* \mid c^1(u) = 0 \wedge 1 < \sigma(u)\}.$$

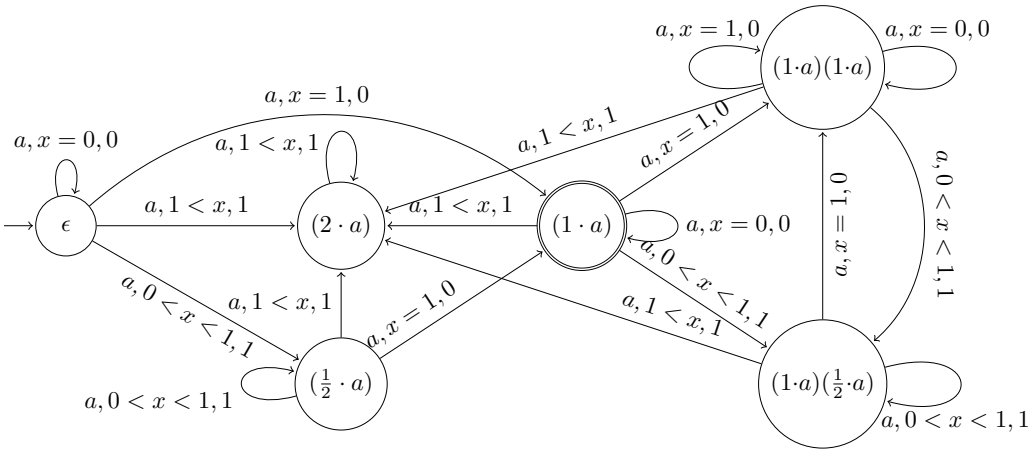

Figure 4 A strict 1-IRDTA $\mathcal{A}_{\approx^{L,1}}$ accepting $L = \{u \in \mathbb{T}\Sigma^* \mid \sigma(u) = 1\}$.

The acceptor $\mathcal{A}_{\approx^{L,1}}$ is depicted in Figure 4 and has six states, whereas the (strict) 1-IRDTA given in Example 3.4 for L only has two.

6 The Canonical Form

Sections 4 and 5 developed the idea of a canonical K-acceptor for a language recognized by an IRTA with constant K. In this section, we will further study this canonical form. We present a crucial property that will help effectively compute the canonical form, and also apply an Angluin-style L^* algorithm.

▶ **Definition 6.1** (Half-integral words). *We call a timed word $(t_1 \cdot a_1)(t_2 \cdot a_2) \dots (t_n \cdot a_n)$ to be a* half-integral *if for each $1 \leq i \leq n$, the fractional part $\{t_i\}$ is either 0 or $\frac{1}{2}$: in other words, each delay is either an integer or a rational with fractional value $\frac{1}{2}$. Also, the empty word ϵ is a half-integral word.*

▶ **Definition 6.2** (Small half-integral words). *Let $K \in \mathbb{N}$. A half-integral word $(t_1 \cdot a_1)(t_2 \cdot a_2) \dots (t_n \cdot a_n)$ is said to be* small *w.r.t K if $t_i < K + 1$ for all $1 \leq i \leq n$.*

For a finite alphabet Σ and $K \in \mathbb{N}$, let $\Sigma_K := \{0, \frac{1}{2}, 1, \dots, K - \frac{1}{2}, K, K + \frac{1}{2}\} \times \Sigma$. Every small integral word is therefore in Σ_K^*. The next lemma is a generalization of the following statement: for every timed word u, there is a small half-integral timed word w such that every K-acceptor reaches the same state on reading both u and w.

▶ **Lemma 6.3.** *Let u_0 be a half-integral word which is small w.r.t. K. For every timed word u, there is a half-integral word w such that $u_0 w$ is small w.r.t K and every K-acceptor reaches the same state on reading $u_0 u$ or $u_0 w$.*

This allows us to identify the canonical equivalence $\approx^{L,K}$ using small integral words.

▶ **Proposition 6.4.** *Let $\mathcal{B}$ be a K-acceptor for a language L. Then, the equivalence $\approx^{\mathcal{B}}$ coincides with $\approx^{L,K}$ iff for all half-integral words $u, v \in (\Sigma_K)^*$: $u \approx^{\mathcal{B}} v$ iff $u \approx^{L,K} v$.*

We make use of Proposition 6.4 to compute the canonical form.

6.1 Computing the Canonical Form

Given a K-acceptor $\mathcal{B}$, we can minimize it using an algorithm which is similar to the standard DFA minimization which proceeds by computing a sequence of equivalence relations on the states.

- Equivalence $\sim^0$: for a pair of states p, q of $\mathcal{B}$ define $p \sim^0 q$ if $region(p) = region(q)$ and either both are accepting states, or both are non-accepting states.
- Suppose we have computed the equivalence $\sim^i$ for some $i \in \mathbb{N}$. For a pair of states p, q, define $p \sim^{i+1} q$ if $p \sim^i q$ and for every letter $(t, a) \in \Sigma_K$, the outgoing transitions $(p, p', a, \phi([t]_{\equiv^K}), s)$ and $(q, q', a, \phi([t]_{\equiv^K}), s)$ in $\mathcal{B}$ satisfy $p' \sim^i q'$.
- Stop when $\sim^{i+1}$ equals $\sim^i$.

The next lemma is a simple consequence of the definition of $\sim^i$ and by an induction on the number of iterations i.

▶ **Lemma 6.5.** *Let p, q be such that $region(p) = region(q)$. Suppose $p \not\sim^i q$. Then there exists a word z of length at most i such that $\delta^*_{\mathcal{B}}(p, z)$ is accepting whereas $\delta^*_{\mathcal{B}}(q, z)$ is not.*

We define the quotient of $\mathcal{B}$ by $\sim^i$ as the K-acceptor whose states are the equivalence classes for $\sim^i$. There is a transition $([q]_{\sim^i}, [p]_{\sim^i}, a, \phi([t]_{\equiv^K}), s)$ if there is $q' \in [q]_{\sim^i}$ and $p' \in [p]_{\sim^i}$ such that $(q', p', a, \phi([t]_{\equiv^K}, s)$ is a transition of $\mathcal{B}$. The initial state is the class of the initial state of $\mathcal{B}$ and the final states are the classes that have a non empty intersection with the set of final states of $\mathcal{B}$. For $i \geq 1$ the quotient of $\mathcal{B}$ by $\sim^i$ is an acceptor for $L(\mathcal{B})$.

Suppose we reach a fixpoint at $m \in \mathbb{N}$. The quotient of $\mathcal{B}$ by $\sim^m$ gives the canonical automaton $\mathcal{A}_{\approx^{L,K}}$. Suppose the quotient does not induce the canonical equivalence. By Proposition 6.4 there are two words $u, v \in (\Sigma_K)^*$ such that u and v go to a different state, but $u \approx^{L,K} v$. Consider the iteration i when the two states were made non-equivalent. There is a small half-integral word z of length i which distinguishes u and v by Lemma 6.5 – a contradiction to $u \approx^{L,K} v$. Let us say $z \in u^{-1}L$, since u, v are half-integral, $\tau_{u \to v}$ is simply the identity function. Since $z \notin v^{-1}L$, we deduce that $\tau_{u \to v}(u^{-1}L) \neq v^{-1}L$, hence $u \not\approx^{L,K} v$.

6.2 Learning the Canonical Form

Our learning algorithm closely follows Angluin's L^* approach, so we assume familiarity with it and provide a brief example of its adaptation to the IRTA setting. Detailed definitions, proof of correctness, and a complete example are provided in the full version [9].

We assume that the Learner is aware of the maximum constant K for the unknown language L. The Learner's goal is to identify the equivalence classes of $\approx^{L,K}$ using small half-integral words, from Σ^*_K. Correspondingly, the rows and columns in an observation table are words in Σ^*_K. In the L^* algorithm, each row of the observation table corresponds to an identified state. Two identical rows correspond to the same state. In order to make a similar conclusion, we add a column to the observation table, that maintains $c^K(u)$ for every string u of a row. There is one detail: once $c^K(u)$ goes beyond K, we want to store it as a single entity $\top$. We define $c^K_\top(u)$ to be equal to $c^K(u)$ when this value is bellow K and equal to $\top$ otherwise.

▶ **Lemma 6.6.** *Let L be a timed language recognized by a K-acceptor, and let $u, v \in (\Sigma_K)^*$. Then $u \approx^{L,K} v$ iff $c^K_\top(u) = c^K_\top(v)$ and for all words $z \in (\Sigma_K)^*$, we have $uz \in L$ iff $vz \in L$.*

An observation table labels its rows and columns with words in $(\Sigma_K)^*$. The row words form a prefix-closed set, and the column words form a suffix-closed set, as in Angluin. Table 1 shows three observation tables. The lower part of these tables includes one-letter extensions of the row words, and their $c^K_\top$ values are shown in the extra column in red.

Table 1 A run of L^* for IRTA.

$\mathcal{T}_0$	ϵ	$c^1_\top(u)$
ϵ	0	0
$(0\cdot a)$	0	0
$(\frac{1}{2}\cdot a)$	0	$\frac{1}{2}$
$(1\cdot a)$	1	0
$(1+\frac{1}{2}\cdot a)$	0	$\top$

$\mathcal{T}_1$	ϵ	$c^1_\top(u)$
ϵ	0	0
$(\frac{1}{2}\cdot a)$	0	$\frac{1}{2}$
$(1\cdot a)$	1	0
$(1+\frac{1}{2}\cdot a)$	0	$\top$
$(0\cdot a)$	0	0
$(\frac{1}{2}\cdot a)(0\cdot a)$	0	$\frac{1}{2}$
$(\frac{1}{2}\cdot a)(\frac{1}{2}\cdot a)$	1	0
$(\frac{1}{2}\cdot a)(1\cdot a)$	0	$\top$
$(\frac{1}{2}\cdot a)(1+\frac{1}{2}\cdot a)$	0	$\top$
$(1\cdot a)(0\cdot a)$	1	0
$(1\cdot a)(\frac{1}{2}\cdot a)$	0	$\frac{1}{2}$
$(1\cdot a)(1\cdot a)$	0	0
$(1\cdot a)(1+\frac{1}{2}\cdot a)$	0	$\top$
$(1+\frac{1}{2}\cdot a)(\Sigma_K\cdot a)$	0	$\top$

$\mathcal{T}_2$	ϵ	$c^1_\top(u)$
ϵ	0	0
$(\frac{1}{2}\cdot a)$	0	$\frac{1}{2}$
$(1\cdot a)$	1	0
$(1+\frac{1}{2}\cdot a)$	0	$\top$
$(1\cdot a)(1\cdot a)$	0	0
$(1\cdot a)(1\cdot a)(1\cdot a)$	0	0
$(0\cdot a)$	0	0
$(\frac{1}{2}\cdot a)(0\cdot a)$	0	$\frac{1}{2}$
$(\frac{1}{2}\cdot a)(\frac{1}{2}\cdot a)$	1	0
$(\frac{1}{2}\cdot a)(1\cdot a)$	0	$\top$
$(\frac{1}{2}\cdot a)(1+\frac{1}{2}\cdot a)$	0	$\top$
$(1\cdot a)(0\cdot a)$	1	0
$(1\cdot a)(\frac{1}{2}\cdot a)$	0	$\frac{1}{2}$
$(1\cdot a)(1+\frac{1}{2}\cdot a)$	0	$\top$
$(1+\frac{1}{2}\cdot a)(\Sigma_K\cdot a)$	0	$\top$
$(1\cdot a)(1\cdot a)(0\cdot a)$	0	0
$(1\cdot a)(1\cdot a)(\frac{1}{2}\cdot a)$	0	$\frac{1}{2}$
$(1\cdot a)(1\cdot a)(1+\frac{1}{2}\cdot a)$	0	$\top$
$(1\cdot a)(1\cdot a)(1\cdot a)(\Sigma_K\cdot a)$	0	..

Suppose the unknown IRTA language is $L = \{u \in \mathbb{T}\Sigma^* \mid \sigma(u) = 1\}$ for $\Sigma = \{a\}$ with a known constant $K = 1$. The learner starts with $\mathcal{T}_0$ containing only the ϵ row which is 0 since $\epsilon \notin L$, and the ϵ column. $\mathcal{T}_0$ is not closed as witnessed by $(1\cdot a)$, whose row is 1, while the row of ϵ is 0. Additionally, $(\frac{1}{2}\cdot a)$ and $(1+\frac{1}{2}\cdot a)$ also witness non-closure because their $c^K_\top$ values are non zero. This highlights the difference from the untimed case: row words are distinguished based on their clock values as well.

To obtain a closed table, the learner successively adds the words $(\frac{1}{2}\cdot a)$, $(1\cdot a)$, $(1+\frac{1}{2}\cdot a)$, forming $\mathcal{T}_1$. Every two words in $\mathcal{T}_1$ are distinguished either by their row or by their $c^K_\top$ value. Thus, $\mathcal{T}_1$ is consistent: if two words have identical rows and same $c^K_\top$ value then their extensions also satisfy this. Since $\mathcal{T}_1$ is closed and consistent the learner conjectures $\mathcal{A}_{\mathcal{T}_1}$ (see [9] for details), the K-acceptor induced by $\mathcal{T}_1$ (formal definition in [9]). The teacher provides a counterexample, assumed to be $(1\cdot a)(1\cdot a)(1\cdot a)$, which is accepted by $\mathcal{A}_{\mathcal{T}_1}$ but not in L. The learner processes this counterexample and computes $\mathcal{T}_2$, which is closed but not consistent as shown by ϵ and $(1\cdot a)(1\cdot a)$ and their extensions by $(1\cdot a)$. Hence, the learner adds a column for $(1\cdot a)$ and computes $\mathcal{T}_3$. The process continues similarly. The rest of the run is detailed in the full version [9].

7 Conclusion

We have presented a Myhill-Nerode style characterization for timed languages accepted by timed automata with integer resets. There are three main technical ingredients: (1) the notion of K-monotonicity (Definition 4.1) that helps characterize equivalences on timed words with automata, that we call K-acceptors. This was possible since each word u determines the value $c^K(u)$ of the clock on reading u, in any K-acceptor; (2) the definition of the rescaling function (Section 5) that gives a Nerode-like equivalence, leading to a Myhill-Nerode theorem for IRTA

languages, and the canonical equivalence $\approx^{L,K}$; (3) understanding canonical equivalence $\approx^{L,K}$ through half-integral words (Section 6), which are, in some sense, discretized words. This helps us to build and learn the canonical form. We believe these technical ingredients provide insights into understanding the languages recognized by IRTA. Typically, active learning algorithms begin by setting up a canonical form. We have laid the foundation for IRTAs. Future work lies in adapting these foundations for better learning algorithms.

References

1. Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994. doi:10.1016/0304-3975(94)90010-8.
2. Jie An, Mingshuai Chen, Bohua Zhan, Naijun Zhan, and Miaomiao Zhang. Learning one-clock timed automata. In *TACAS'20: Proc. 26th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, volume 12078 of *LNCS*, pages 444–462. Springer, 2020. doi:10.1007/978-3-030-45190-5_25.
3. Jie An, Bohua Zhan, Naijun Zhan, and Miaomiao Zhang. Learning Nondeterministic Real-Time Automata. *ACM Transactions on Embedded Computing Systems*, 20(5s):1–26, 2021. doi:10.1145/3477030.
4. Dana Angluin. Learning regular sets from queries and counterexamples. *Information and Computation*, 75(2):87–106, 1987. doi:10.1016/0890-5401(87)90052-6.
5. Devendra Bhave and Shibashis Guha. Adding Dense-Timed Stack to Integer Reset Timed Automata. In *RP'17: Proc. 11th International Conference on Reachability Problems*, volume 10506 of *LNCS*, pages 9–25. Springer, 2017. doi:10.1007/978-3-319-67089-8_2.
6. Mikołaj Bojańczyk and Sławomir Lasota. A Machine-Independent Characterization of Timed Languages. In *ICALP'12: Proc. of the 39th Int. Colloquium of Automata, Languages and Programming*, volume 7392, pages 92–103. Springer, 2012. doi:10.1007/978-3-642-31585-5_12.
7. Patricia Bouyer, Catherine Dufourd, Emmanuel Fleury, and Antoine Petit. Updatable timed automata. *Theoretical Computer Science*, 321(2-3):291–345, 2004. doi:10.1016/j.tcs.2004.04.003.
8. Véronique Bruyère, Bharat Garhewal, Guillermo A. Pérez, Gaëtan Staquet, and Frits W. Vaandrager. Active learning of mealy machines with timers. *CoRR*, abs/2403.02019, 2024. doi:10.48550/arxiv.2403.02019.
9. Kyveli Doveri, Pierre Ganty, and B. Srivathsan. A myhill-nerode style characterization for timed automata with integer resets. *CoRR*, abs/2410.02464, 2024. doi:10.48550/arxiv.2410.02464.
10. Olga Grinchtein, Bengt Jonsson, and Martin Leucker. Learning of event-recording automata. *Theoretical Computer Science*, 411(47):4029–4054, 2010. doi:10.1016/j.tcs.2010.07.008.
11. Olga Grinchtein, Bengt Jonsson, and Paul Pettersson. Inference of event-recording automata using timed decision trees. In *CONCUR'06: Proc. 17th International Conference on Concurrency Theory*, volume 4137 of *LNCS*, pages 435–449. Springer, 2006. doi:10.1007/11817949_29.
12. Shang-Wei Lin, Étienne André, Jin Song Dong, Jun Sun, and Yang Liu. An efficient algorithm for learning event-recording automata. In *ATVA'11: Proc. 9th International Symposium on Automated Technology for Verification and Analysis*, volume 6996 of *LNCS*, pages 463–472. Springer, 2011. doi:10.1007/978-3-642-24372-1_35.
13. Anirban Majumdar, Sayan Mukherjee, and Jean-François Raskin. Greybox learning of languages recognizable by event-recording automata. In *ATVA'24: Proc. 22nd International Symposium on Automated Technology for Verification and Analysis*, LNCS. Springer, 2024.
14. Oded Maler and Amir Pnueli. On Recognizable Timed Languages. In *FoSSaCS'04: Proc. of the Int. Conf. on Foundations of Software Science and Computation Structures*, volume 2987 of *LNCS*, pages 348–362. Springer, 2004. doi:10.1007/978-3-540-24727-2_25.
15. Lakshmi Manasa and Krishna S. Integer Reset Timed Automata: Clock Reduction and Determinizability. *CoRR*, abs/1001.1215, 2010. doi:10.48550/arxiv.1001.1215.

16 Jan Springintveld and Frits W. Vaandrager. Minimizable timed automata. In *FTRTFT'96: Proc. of 4th Int. Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems*, volume 1135 of *LNCS*, pages 130–147. Springer, 1996. doi:10.1007/3-540-61648-9_38.

17 P. Vijay Suman, Paritosh K. Pandya, Shankara Narayanan Krishna, and Lakshmi Manasa. Timed Automata with Integer Resets: Language Inclusion and Expressiveness. In *FORMATS'08: Proc. of the Int. Conf. on Formal Modeling and Analysis of Timed Systems*, volume 5215, pages 78–92. Springer, 2008. doi:10.1007/978-3-540-85778-5_7.

18 Martin Tappler, Bernhard K. Aichernig, Kim Guldstrand Larsen, and Florian Lorber. Time to learn - learning timed automata from tests. In *FORMATS'19: Proc. 17th International Conference on Formal Modeling and Analysis of Timed Systems*, volume 11750 of *LNCS*, pages 216–235. Springer, 2019. doi:10.1007/978-3-030-29662-9_13.

19 Frits W. Vaandrager, Masoud Ebrahimi, and Roderick Bloem. Learning mealy machines with one timer. *Inf. Comput.*, 295(Part B):105013, 2023. doi:10.1016/J.IC.2023.105013.

20 Sicco Verwer. *Efficient Identification of Timed Automata: Theory and practice*. PhD thesis, Delft University of Technology, Netherlands, 2010. URL: http://resolver.tudelft.nl/uuid:61d9f199-7b01-45be-a6ed-04498113a212.

21 Masaki Waga. Active Learning of Deterministic Timed Automata with Myhill-Nerode Style Characterization. In *CAV'23: Proc. of the 35th Int. Conf. on Computer Aided Verification*, volume 13964 of *LNCS*, pages 3–26. Springer, 2023. doi:10.1007/978-3-031-37706-8_1.

22 Runqing Xu, Jie An, and Bohua Zhan. Active learning of one-clock timed automata using constraint solving. In *ATVA'22: Proc. 20th International Symposium on Automated Technology for Verification and Analysis*, volume 13505 of *LNCS*, pages 249–265. Springer, 2022. doi:10.1007/978-3-031-19992-9_16.