On Timed Automata with Discrete Time—Structural and Language Theoretical Characterization

Hermann Gruber[1], Markus Holzer[1], Astrid Kiehn[2], and Barbara König[3]*

[1] Institut für Informatik, Technische Universität München, Boltzmannstraße 3, D-85748 Garching bei München, Germany
email: {gruberh,holzer}@in.tum.de

[2] Department of Computer Science and Engineering, Indian Institute of Technology Delhi, Hauz Khas, New Delhi 110016, India
email: astrid@cse.iitd.ernet.in

[3] Institut für Formale Methoden der Informatik, Universität Stuttgart, Universitätsstraße 38, D-70569 Stuttgart, Germany
email: koenigba@fmi.uni-stuttgart.de

Abstract. We develop a structural and language theoretical characterization of timed languages over discrete time in terms of a variant of Büchi automata and languages. The so-called tick automaton is a standard Büchi automaton with a special "clock-tick"-input symbol modeling the discrete flow of time. Based on these characterizations we give an alternative proof for the fact that the class of regular timed languages is closed under complementation and formulate a time-warp lemma which, similar to a pumping lemma, can be used to show that a timed language is not regular. The characterizations hold alike for timed automata with and without periodic clock constraints.

1 Introduction

Timed automata have been introduced in [1] in order to model real-time systems from a quantitative perspective, to make specification and verification of models of real-time systems easier. A timed automaton is a Büchi automaton with a finite set of clocks, which can be independently reset, and where the automaton can keep track of the time elapsed since the last reset. Several generalizations have been investigated as, in particular, timed automata with silent transitions [5] and timed automata with periodic clock constraints [6]. Both these extensions are equally powerful and strictly increase the expressive power of timed automata, regardless of the used time semantics, i.e., the interpretation of the clocks, which assigns some value to each clock. The two major semantics discussed in the literature are the discrete-time and the dense-time model. Although the latter is more natural from a physical point of view and allows an easy modelling of real-time systems, it requires decision methods for real-time

* Research partially supported by DFG project SANDS and EPSRC grant R93346/01.

logics, which in general are undecidable [2]. In fact, the class of timed languages accepted by automata interpreted under the dense-time model is closed under all positive Boolean operations, but is not closed under complementation [1]. The latter is also true for timed languages over discrete time, except for the complementation closure, which was shown to hold in [9] in terms of a variant of monadic second order logic.

However, as recently pointed out by Asarin [3], despite the availability of tools such as Uppaal and Kronos, the foundations of the theory of timed languages are comparatively weak compared to the classical theory of finite automata and regular languages. For instance Asarin states in his open question 3: "Give a simple and natural algebraic characterization of a known class of timed languages—confirming that it is the correct class of languages." With the present paper we provide such a structural characterization for the class of discrete-time languages accepted by timed automata with or without periodic clock constraints (or equivalently with or without silent transitions). To this end we transform timed automata into so-called tick automata. A tick automaton is a Büchi automaton where the input alphabet is equipped with an additional "clock-tick"-symbol ($\surd$) modelling the discrete flow of time.[4] Intuitively, actions are occurring inbetween such clock ticks, hence, our time semantics is such that it allows (finite or infinite) stuttering of actions on a particular time stamp during the computation. It turns out that any tick automaton induced by a timed automaton has a particular $\surd$-transition structure, namely deterministic tick-paths ending in (trivial or non-trivial) tick-loops. On the other hand, we can show that any tick automaton (not necessarily induced by a timed automaton) can be converted into this normal form. This nicely corresponds to the intuitive behaviour of time as a sort of deterministic or non-branching flow.

While timed automata in general induce non-trivial tick-loops, the tick-loops become trivial for aperiodic timed automata. Besides this structural characterization of timed languages in terms of tick automata, we provide a language theoretical characterization for languages accepted by aperiodic timed automata by a simple condition, which will be called $\surd$-stretchiness. This condition is closely related to the aperiodicity condition of regular languages on finite words. Loosely speaking, $\surd$-stretchiness tells us that tick actions in direct sequence can only be counted up to a given threshold, which depends on the language only. We hope that these characterizations give some further insights into the behaviour of timed languages.

The provided characterizations can be used to show an alternative way to reprove the complementation closure of timed aperiodic languages, directly in the framework of Büchi automata, thus reproving the result given in [9]. Moreover, we develop a sort of pumping lemma, the so called time-warp lemma, which shows that certain actions in a run of a word on the timed automaton can be moved along the time axis to the future keeping the acceptance invariant. To illustrate the strength of the time-warp lemma, we give very simple proofs for languages that cannot be accepted by any timed (aperiodic) automaton. In

[4] Note that our model of time is called the fictitious-clock model in [1].

addition we mention that the time-warp lemma is not limited to the discrete time semantics, but holds for dense-time semantics as well. Hence this is a (partial) answer to open question 14—"Develop simple techniques allowing to prove that a given timed language is not regular"—stated in [3]. All presented results are effectively constructible and therefore can be used for the verification of discrete timed systems.

The paper is organized as follows: The next section contains preliminaries on timed automata and timed languages. Section 3 introduces tick automata, and shows how to transform timed automata to tick automata, by using the region automata. The next section is devoted to the structural and language theoretical characterization of timed languages in terms of tick automata. As a byproduct we provide the reader with an alternative proof of the complementation problem of timed languages, by automata theoretical constructions only. Then in Section 5 we develop the time-warp lemma and give some further applications. Finally we summarize our results in Section 6.

2 Definitions

Timed automata can be considered as Büchi-automata which have been equipped with a finite number of clocks. These clock run simultaneously and can individually be reset to 0 by a change of state, which in turn depends on the current values of the clocks. Timed automata are usually interpreted under a dense time domain, that is, a clock value can be any real number. This paper only considers a discrete time semantics which assumes an underlying fictitious clock. For a discussion of the different time models we refer to [1, 5, 6].

Many variations of timed automata can be found in the literature. The most general form allows for silent (ε) transitions and as constraints any Boolean expression over atomic assertions of the form $x = c$, $x < c$, and $x =_m c$ (comparison modulo m), where x is a clock and $c, m \in I\!N$, or even direct comparisons of different clock values. It has been shown in [1] that direct comparisons of clock values provide an even more succinct representation, and in [5] and [6] that silent transitions and modulo constraints are mutually expressible and do increase the power of classical timed automata introduced in [1] (which coincide with aperiodic timed automata in our setting). To keep the technical presentation as simple as possible we use the results of [6, 7] and assume just one clock and a restricted constraint language. Every timed automaton can effectively brought into this form though its size might increase substantially.

Definition 1. *A* timed automaton $A = \langle Z, \Sigma, E, z_S, R, \{x\} \rangle$ *is given by a finite set of states Z, the input alphabet Σ, a start state $z_S \in Z$, an acceptance set $R \subseteq Z$, one clock variable x and the set of transitions $E \subseteq Z \times \Sigma \times 2^{\Phi} \times 2^{\{x\}} \times Z$, where $\Phi = \{x = i, x \neq i, x =_m i, x \neq_m i \mid 0 \leq i < m\}$ and $m \in I\!N$. An* aperiodic *timed automaton differs only in the constraint universe $\Phi = \{x = i, x \neq i, x \geq m \mid 0 \leq i < m\}$.*

We will sometimes use the term *periodic timed automaton* in order to distinguish timed automata from aperiodic timed automata.

We denote $\langle z, a, \varphi, X, z' \rangle \in E$ by $z \xrightarrow{a,\varphi,X} z'$, where φ is called the constraint set (or simply constraint) and X the reset set. A configuration $\langle z, v \rangle$ of A consists of a state and a clock assignment. As we consider just one clock, a clock assignment reduces to an integer $v \in I\!N$. A *timed run* on a timed automaton is an infinite sequence

$$\pi = \langle z_0, v_0 \rangle \xrightarrow{a_1,t_1} \langle z_1, v_1 \rangle \xrightarrow{a_2,t_2} \langle z_2, v_2 \rangle \xrightarrow{a_3,t_3} \cdots ,$$

where $z_0 = z_S$, $a_i \in \Sigma$ and $t_i \in I\!N$ such that $v_0 = 0$ and for each $i \geq 0$ there is an underlying transition $z_i \xrightarrow{a_{i+1},\varphi,X} z_{i+1}$ with (a) $v_i + \Delta_i \models \varphi$ and (b) $v_{i+1} = v_i + \Delta_i$, if $X = \emptyset$, and $v_{i+1} = 0$, if $X = \{x\}$, where $\Delta_i := t_{i+1} - t_i$ and $t_0 = 0$, with $\Delta_i \geq 0$. Condition (a) expresses that the guarding constraint φ is satisfied before the transition is taken, while (b) ensures that the clock is adjusted correctly. A clock assignment satisfies a set of constraints if it satisfies every constraint contained in the set.

A timed run π is called *accepting* if $\mathit{Inf}(\pi) \cap R \neq \emptyset$, where $\mathit{Inf}(\pi)$ denotes the set of states occurring infinitely often in π. From a timed run π we extract the timed word $w(\pi) = \langle a_1, t_1 \rangle \langle a_2, t_2 \rangle \langle a_3, t_3 \rangle \ldots \in (\Sigma \times I\!N)^\omega$, which, as usual, is the constituent of the timed language of A:

$$L^\omega(A) = \{\, w(\pi) \in (\Sigma \times I\!N)^\omega \mid \pi \text{ is an accepting run on } A \,\}.$$

Note, that by definition the time stamp sequence is monotonously increasing. Moreover, we diverge from the literature in that we do not demand non-zenoness, that is, each run indeed diverges in time. By doing so, we simplify the constructions to come. As shown in [5], non-zenoness can be enforced by an additional automata theoretical construction.

3 From Timed Automata to Tick Automata

The fictitious clock semantics suggests another way of behaviour description: rather than equipping an event with the current clock time, the ticking of the clock can be modeled by a particular *tick-action*, $\surd$. For instance, the finite timed word $\langle a, 1 \rangle \langle b, 3 \rangle \langle a, 3 \rangle \langle a, 7 \rangle$ corresponds to $\surd a \surd \surd b a \surd \surd \surd \surd a$. We call the latter representation the *ticked-version* of the former.

Definition 2. *Let $w = \langle a_1, t_1 \rangle \langle a_2, t_2 \rangle \ldots \langle a_i, t_i \rangle \ldots$ be a timed word. Its ticked version is $w_\surd = \surd^{\Delta_0} a_1 \surd^{\Delta_1} a_2 \ldots a_{i-1} \surd^{\Delta_{i-1}} a_i \ldots$ with $\Delta_i = t_{i+1} - t_i$ and $t_0 = 0$. Let $\mathit{untick}(w_\surd) = w$ be the inverse operation. Then the ticked version of a timed language L is defined as $L_\surd := \{\, w_\surd \in (\Sigma \uplus \{\surd\})^\omega \mid \mathit{untick}(w_\surd) \in L \,\}$, where $\uplus$ denotes the disjoint union of sets.*

We show in this section that every (discrete) timed automaton A can be represented as a Büchi-automaton with a distinguished $\surd$-action such that $(L^\omega(A))_\surd$ coincides with the ω-language of the Büchi automaton. We denote this Büchi automaton by $A_\surd$ and call it the *tick automaton of A*. In general, a tick automaton is a Büchi automaton with a new action $\surd$ representing a tick of an underlying fictitious clock. A tick automaton B is given by $\langle Z, \Sigma \uplus \{\surd\}, E, z_S, R\rangle$, where Z, Σ, z_S and R are interpreted as in the case of timed automata but E simplifies to $E \subseteq Z \times \Sigma \uplus \{\surd\} \times Z$. A word w is in the accepted language $L^\omega(B)$ if (1) the word w contains infinitely many non-$\surd$ actions and (2) the underlying run of w contains infinitely many occurrences of acceptance states (the usual Büchi acceptance condition). Note that condition (1) and (2) give an adequate counterpart to languages accepted by timed automata. Since we do not require non-zenoness there, we do not demand infinitely many $\surd$-actions occurring in w as acceptance condition.

To express the timed behaviour of an automaton as a tick automaton we use the concept of regions which have been introduced in [1] in order to describe the untimed behaviour of a timed automaton. A region equates all those clock valuations which are undistinguishable under progress of time or clock resetting with respect to the evaluation of constraints, i.e., a region is an equivalence class.

As we deal with one clock x only, the regions are simply described by the region expressions $x = i$ and $x \geq m \wedge x =_m i$, for $0 \leq i < m$, and in case of aperiodic timed automata by $x = i$ and $x \geq m$, for $0 \leq i < m$, where m is the constant in the definition of an automaton. Note that in both cases, the clock regions provide a partition on the time domain $I\!N$. Two clock valuations v and v' are equivalent, $v \sim v'$, if they satisfy the same region expression. The following lemma ensures soundness of our construction.

Lemma 3. *Let v and v' be clock valuations with $v \sim v'$. Then*

1. *$v \models \varphi$ if and only if $v' \models \varphi$ for any clock constraint φ occurring in A, and*
2. *$(v + \Delta) \sim (v' + \Delta)$ for any $\Delta \in I\!N$.*

For the tick automaton $A_\surd$ we pair the states of A with its regions. The transitions are induced by the transitions of A which are now split into a delay and an action part.

Definition 4 (Automaton $A_\surd$). *Let $A = \langle Z, \Sigma, E, z_S, R, \{x\}\rangle$ be a timed automaton. Then $A_\surd$ is the tick automaton $\langle Z', \Sigma \uplus \{\surd\}, E', z'_S, R'\rangle$, where the set of states is $Z' = \{\, \langle z, \alpha\rangle \mid z \in Z, \alpha$ a region expression$\}$, the initial state equals $z'_S = \langle z_S, x = 0\rangle$, $R' = \{\, \langle z, \alpha\rangle \mid z \in R\,\}$ and E' contains*

1. *$\langle z, \alpha\rangle \xrightarrow{a} \langle z', \alpha\rangle$ if there is $z \xrightarrow{a,\varphi,\emptyset} z'$ in A such that $\alpha \models \varphi$ (non-reset transitions),*
2. *$\langle z, \alpha\rangle \xrightarrow{a} \langle z', x = 0\rangle$ in $A_\surd$ if there is $z \xrightarrow{a,\varphi,\{x\}} z'$ in A such that $\alpha \models \varphi$ (reset transitions),*
3. *$\langle z, \alpha\rangle \xrightarrow{\surd} \langle z, \alpha + 1\rangle$ for all states $\langle z, \alpha\rangle \in Z'$ (tick transitions).*

Observe that the non-reset and reset transitions are executed without any delay. The delay that might be necessary in A to move from state z to z' is performed in $A_\surd$ by the respective number of $\surd$-transitions. These transitions lead to the immediate time successor of the current region but do not leave state z. A time successor $\alpha+k$ of a region expression α is defined by $\alpha+k = [v+k]$ for $\alpha = [v]$.

Theorem 5. *Let A be a timed automaton. Then*

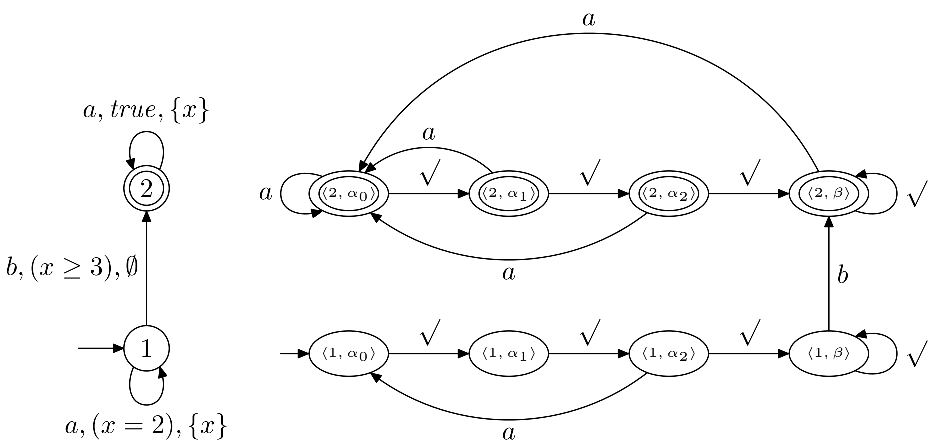
$$(L^\omega(A))_{\surd} = L^\omega(A_{\surd}) \cap \{\, w \in (\Sigma \uplus \{\surd\})^\omega \mid w \text{ contains infinitely many } a \in \Sigma \,\}.$$

The size of $A_{\surd}$ is determined by the number of states and transitions. The number of states of $A_{\surd}$ is $|Z'| = |Z| \times$ (number of regions). We have $|Z'| = |Z| \cdot 2m$ for the case of periodic clock constraints, and $|Z'| = |Z| \cdot (m+1)$ for automata using only aperiodic constraints. The number of transitions is bound by $|E| \cdot (m+1) + |Z| \cdot (m+1) = (|E| + |Z|)(m+1)$ for the aperiodic case, and by $|E| \cdot 2m + |Z| \cdot 2m = (|E| + |Z|) \cdot 2m$ in general.

Example 6. Figure 1 shows an aperiodic timed automaton with accepted language

$$\begin{aligned} L_1 = \{\, \langle a_i, t_i \rangle_{i \geq 1} \mid \exists k \geq 0 : (\forall i < k : a_i = a \wedge t_i = 2i) \\ \wedge\, (a_k = b \wedge t_k > 2k) \wedge (\forall j > k : a_j = a) \,\}. \end{aligned}$$

Its tick automaton is also given in Figure 1, where $\alpha_0 = (x = 0)$, $\alpha_1 = (x = 1)$, $\alpha_2 = (x = 2)$, and $\beta = (x \geq 3)$ are the region expressions.



$a, true, \{x\}$
2
$b, (x \geq 3), \emptyset$
1
$a, (x = 2), \{x\}$
$\langle 2, \alpha_0 \rangle$
$\langle 2, \alpha_1 \rangle$
$\langle 2, \alpha_2 \rangle$
$\langle 2, \beta \rangle$
$\langle 1, \alpha_0 \rangle$
$\langle 1, \alpha_1 \rangle$
$\langle 1, \alpha_2 \rangle$
$\langle 1, \beta \rangle$
a
b
$\surd$

Fig. 1. A timed automaton with aperiodic clock constraints (Example 6) and its corresponding tick automaton.

We now have a closer look at the structure of the tick automaton $A_{\surd}$. It follows immediately from the definition that there is no branching with respect to tick-transitions. For aperiodic automata we can additionally observe that there are no non-trivial cycles in which the transitions are labelled by $\surd$ only, while in arbitrary timed automata the length of tick-cycles is m, where m is the constant of the modulo constraints. Formally, these properties read as follows:

1. Tick determinism: $z \xrightarrow{\surd} z'$ and $z \xrightarrow{\surd} z''$ imply $z' = z''$.
2. Tick-cycle freedom: $z_1 \xrightarrow{\surd} z_2 \xrightarrow{\surd} \cdots \xrightarrow{\surd} z_n$ and $z_1 \neq z_2$ imply $z_1 \neq z_n$.
3. Constant tick-cycle length: $z_1 \xrightarrow{\surd} \cdots \xrightarrow{\surd} z_n \xrightarrow{\surd} z_1$ and $z_1, \ldots, z_n$ are pairwise distinct implies $n = m$ where m is the constant given in the constraint universe of periodic timed automata.

Note that a tick-loop, i.e., a transition $z \xrightarrow{\surd} z$, does not count as a tick-cycle.

Theorem 7. *Let A be a timed automaton and $A_{\surd}$ its corresponding tick automaton. Then the following implications hold:*

1. *If A is aperiodic then $A_{\surd}$ is tick-deterministic and tick-cycle free, and*
2. *if A is periodic then $A_{\surd}$ is tick-deterministic and has a constant tick-cycle length.*

The converse of the theorem given above is shown in Section 4 on structural characterizations. Observe that for timed languages in general, due to the correspondence of ε-transitions and modulo constraints [6], there is an easy construction that transforms an arbitrary tick automaton into an equivalent timed automaton. We introduce a new clock x, replace each tick transition $z \xrightarrow{\surd} z'$ in the tick automaton by an ε-transition $z \xrightarrow{\varepsilon, x=1, \{x\}} z'$ and then use the fact that timed automata with silent actions can be transformed into equivalent ε-free timed automata with modulo constraints. In passing we have hence shown the following corollary:

Corollary 8. *For each tick automaton A there is a periodic timed automaton B such that $L^\omega(A) = (L^\omega(B))_{\surd}$.*

4 Characterizations of Aperiodic Timed Languages

In this section we develop a structural and language theoretical characterization of aperiodic timed languages in terms of tick automata and languages, respectively. An application of both characterizations to the complementation problem is given in Subsection 4.3.

4.1 Structural Characterization of Aperiodic Timed Languages

We give a structural characterization of timed languages accepted by aperiodic timed automata. We state first that every tick-cycle free aperiodic tick automaton can be made tick-deterministic.

Lemma 9. *Each tick-cycle free tick automaton A can be effectively transformed into a tick automaton B such that $L^\omega(A) = L^\omega(B)$ and B is tick-deterministic and tick-cycle free.*

For a tick-deterministic tick automaton A, which is tick-cycle free, it is now straightforward to define a timed automaton B with one clock x such that $(L^\omega(B))_{\surd} = L^\omega(A)$. Here B contains the set of states of A. For each path $z \xrightarrow{a} z_0 \xrightarrow{\surd} \cdots \xrightarrow{\surd} z_n \xrightarrow{b} z_{n+1}$ where a and b are non-$\surd$ actions, we either introduce a transition $z_0 \xrightarrow{b,x=n,\{x\}} z_{n+1}$ if there is no tick loop at z_n or $z_0 \xrightarrow{b,x\geq n,\{x\}} z_{n+1}$ in the presence of such a loop. For the initial state z_S we proceed similarly. Furthermore we introduce appropriate acceptance states and convert the constraints into the form of Definition 1. Hence we can give the following structural characterization of aperiodic timed languages.

Theorem 10. *A tick automaton A is language equivalent to a tick-deterministic and tick-cycle free tick automaton if and only if there is an aperiodic timed automaton B such that $L^\omega(A) = (L^\omega(B))_{\surd}$.*

4.2 Language Theoretical Characterization of Aperiodic Timed Languages

We give a language theoretical characterization of timed language accepted by aperiodic timed automata. To be more precise, we define a condition, which holds exactly for all languages which are tick versions of aperiodic timed languages and *vice versa.* The condition reads as follows:

Definition 11 ($\surd$-stretchy). *A language $L \subseteq (\Sigma \uplus \{\surd\})^\omega$ is called $\surd$-stretchy if and only if there exists an $n \in \mathbb{N}$ such that (1) for each infinite sequence $w_1, w_2, w_3, \ldots$ of finite words over the alphabet $\Sigma \uplus \{\surd\}$ and for each infinite sequence $i_1, i_2, i_3, \ldots$ of nonnegative numbers*

$$w_1 \surd^n w_2 \surd^n w_3 \surd^n \ldots \in L \iff w_1 \surd^{n+i_1} w_2 \surd^{n+i_2} w_3 \surd^{n+i_3} \ldots \in L$$

and (2) for every $w \in (\Sigma \uplus \{\surd\})^*$, $v \in (\Sigma \uplus \{\surd\})^\omega$ and each nonnegative number i,

$$w \surd^n v \in L \iff w \surd^{n+i} v \in L.$$

The following lemma immediately follows by definition.

Lemma 12. *A language $L \subseteq (\Sigma \uplus \{\surd\})^\omega$ is $\surd$-stretchy if and only if the complement of L, i.e., the language $(\Sigma \uplus \{\surd\})^\omega \setminus L$, is $\surd$-stretchy.*

Moreover, it is not hard to see that every language accepted by a tick automaton $A_{\surd}$, where A is aperiodic, is $\surd$-stretchy. This is due to the fact, that whenever $A_{\surd}$ has read a block of $\surd$ which is long enough it must be in a state, which corresponds to the maximal clock region. Since this state must have a tick-loop, one can enlarge the $\surd$-block by an arbitrary number of $\surd$'s without changing the acceptance of the original word. Thus, the $\surd$-stretchy condition is satisfied for $n = m + 1$, where m is the maximal constant occurring in the set of clock constraints of the timed aperiodic automaton A. In terms of $A_{\surd}$, one can choose the number of its states as a suitable n.

Before we prove the converse relation, i.e., that every $\surd$-stretchy ω-regular language over the alphabet $\Sigma \uplus \{\surd\}$ accepted by a tick automaton $A_\surd$ corresponds to an aperiodic timed language, we need the following technical lemma.

Lemma 13. *Let $L \subseteq (\Sigma \cup \{\surd\})^\omega$ be a $\surd$-stretchy language accepted by a tick automaton A. Then a tick automaton B can be effectively constructed from A such that $L^\omega(B) = L$ and B is tick-cycle free.*

Proof. Let n be the constant from the $\surd$-stretchy condition, which is satisfied by L. It suffices to show that every non-trivial strongly connected component of $\surd$-transitions can be eliminated from the tick automaton A, without changing the accepted language.

Fix one non-trivial strongly connected component of $\surd$-transitions and let S be the set of all states contained within this component. Without loss of generality we may assume that there is no a-transition leading from z to z' with $z, z' \in S$. Define $R_{s,s'}$ be the set of all words representing a path from s to s', which lies completely within S. Observe that $R_{s,s'}$ is regular for every $s, s' \in S$. Then for every $s, s' \in S$ such that s has an in-going non-$\surd$ transition and s' an outgoing a-transition to a state t, which is not contained in S, we proceed in three steps: (1) Introduce a new edge from s to t labeled with the regular expressions: (i) All expressions of the form wa with $w \in R_{s,s'}$ and $|w| < n$ and (ii) $\surd^n \surd^* a$. (2) If there is an accepting state $s'' \in S$, then introduce a new accepting state t', which is appropriately connected to all successors of t, and introduce a new edge from s to t' labeled with the regular expressions: (i) All expressions of the form wa with $w \in R_{s,s''}R_{s'',s'}$ and $|w| < n$ and (ii) $\surd^n \surd^* a$. (3) Finally, one removes all $\surd$-transitions, which were once part of the strongly connected cycle, and converts the regular expression on the newly introduced edges to paths, the structure of which contains no non-trivial tick cycles. This completes the description of the construction. It remains to verify the correctness.

Consider step (1) in more detail. The only way to eventually accept a word w, which is not in the language is to use a word from the expression $\surd^n \surd^* a$ going from s to t *via* s'. Then we distinguish two cases: The new edge is traversed infinitely or finitely often. We only prove the former case, since the latter can be shown by similar arguments. Now assume that the edge under consideration is traversed infinitely often and $w = w_1 \surd^{n+c_1} a w_2 \surd^{n+c_2} a w_3 \surd^{n+c_3} \ldots$ where the substrings of the form $\surd^{n+c_i} a$, for $c_i \geq 0$, are due to the new edge. Then we distinguish two cases: If $w_1 \surd^n a w_2 \surd^n a w_3 \surd^n \ldots$ is in L, then so is w. Otherwise, assume that $w_1 \surd^n a w_2 \surd^n a w_3 \surd^n \ldots$ is not a member of L, but w is accepted by the new machine. Then instead of using the newly introduced edge, we alter the computation such that one particular path from s to t *via* s'' within the strongly connected cycle is taken. Thus, we end up with a word $w' = w_1 \surd^{n+c} a w_2 \surd^{n+c} a w_3 \surd^{n+c} \ldots$ for some constant c, which is also accepted by the original Büchi automaton, and hence lies in L—the accepting states that were seen infinitely often in the run of w do not belong to $S \setminus \{s\}$. Then we immediately obtain a contradiction, because by the $\surd$-stretchy condition also the word $w_1 \surd^n a w_2 \surd^n a w_3 \surd^n \ldots$ has to be accepted, which was ruled out by our assumption. Thus, step (1) does not alter the accepted language.

Furthermore, when removing the $\surd$-transitions that were once part of the strongly connected component in step (3), we have to consider computations that visit a possible acceptance state belonging to S. This is done in step (2). By a similar reasoning as above one observes that the newly introduced edge together with the acceptance state t' does not alter the underlying language. Moreover, the same holds for step (3), where the $\surd$-transitions that were part of the strongly connected component are deleted. Thus, $L^\omega(B) = L^\omega(A)$. $\square$

Finally, we state the main result of this section, which is an immediate consequence of our previous considerations.

Theorem 14. *A language $L \subseteq (\Sigma \uplus \{\surd\})^\omega$ is a $\surd$-stretchy language accepted by a tick automaton if and only if the language $untick(L)$ is accepted by an aperiodic timed automaton, where $untick(L) := \{\, untick(w_\surd) \mid w_\surd \in L \,\}$.*

4.3 Application to Complementation

We show that the (discrete) languages obtained from timed automata are closed under complementation, thus reproving Wilke's result [9] by automata theoretical constructions only—observe that the complementation of timed languages is done with respect to timed words, where the time stamp sequence is monotonously increasing.

Theorem 15. *The classes of languages obtained from timed automata and aperiodic timed automata are closed under complementation.*

Proof. In the aperiodic case convert a timed automaton into a tick automaton according to the algorithm given in the previous section. Then we complement the Büchi automaton with any complementation algorithm (for instance the one described in [8]). This results in a tick automaton (with n states), where Lemma 13 can be applied, resulting in a Büchi automaton which is tick-cycle free. Finally, applying the conversion for a tick-deterministic and tick-cycle free Büchi automaton into an aperiodic timed automaton solves the complementation problem *via* our structural characterization. Observe that all steps are effectively constructible since the constant for the $\surd$-stretchy condition can be estimated by the size of the automaton (which is n in our case).

In the periodic case the proof is analogous and even simpler since it does not require Lemma 13. $\square$

5 Time-Warp of Timed Languages

Based on our language theoretical characterization of timed languages we show that certain actions in a timed word can be "time-warped," i.e., all time stamps are shifted by a common distance to the future along the time axis. This leads us to the time-warp lemma, which is similar in flavour to a pumping lemma, and thus allows us to identify certain languages as not acceptable by any timed automaton—compare with the pumping lemmata of [4].

Before we introduce the time-warp lemma we need some more notations in order to simplify the presentation. For a timed word $w = \langle a_1, t_1\rangle\langle a_2, t_2\rangle \dots \langle a_i, t_i\rangle \dots$ define its Δ-timed representation as $w_\Delta = \langle a_1, \Delta_1\rangle\langle a_2, \Delta_2\rangle \dots \langle a_i, \Delta_i\rangle \dots$, where $\Delta_i = t_i - t_{i-1}$ with $t_0 = 0$. Then the Δ-version of the timed language L is defined as $L_\Delta = \{\, w_\Delta \in (\Sigma \times \mathbb{N})^\omega \mid w \in L \,\}$. Now we are ready for the time-warp lemma, the proof of which immediately follows from the given characterizations of languages accepted by timed and aperiodic timed automata in Corollary 8 and Theorem 10. Thus we omit the proof.

Lemma 16 (Time-warp lemma[5]). *Let L be a the language accepted by an aperiodic timed automaton. Then there exists a constant n, such that for every word $w_\Delta = \langle a_i, \Delta_i\rangle_{i\geq 1}$, every index set $J \subseteq \{\, i \in \mathbb{N} \mid \Delta_i \geq n \,\}$, and for every function $f : J \to \mathbb{N}$ we have $w_\Delta \in L_\Delta$ if and only if time-warp$_{J,f}(w_\Delta) \in L_\Delta$, where time-warp$_{J,f}(w_\Delta)$ is defined to be the Δ-timed word $\langle a_i, \Delta'_i\rangle_{i\geq 1}$ with $\Delta'_i = \Delta_i + f(i)$, if $i \in J$, and $\Delta'_i = \Delta_i$ otherwise. The statement remains valid in case L is a language accepted by a periodic timed automaton in general, provided that the all quantified functions $f : J \to \mathbb{N}$ obey the additional property* $\mathrm{range}(f) \subseteq \{\, kn \mid k \geq 0\}$.

With the time-warp lemma we can prove that certain languages are not acceptable by any timed automaton. For instance, consider the language of "convergent response time," which is defined as follows—the language is taken from [1]:

$$L = \{\, \langle a_i, t_i\rangle_{i\geq 1} \mid \forall i \geq 1 : a_{2i-1} = a \wedge a_{2i} = b \\ \wedge\, \exists c \geq 0 : \exists i \geq 1 : \forall j > i : t_{2j} < t_{2j-1} + c \,\}.$$

We show that L is *not* acceptable by any timed automaton. Assume to the contrary that the language L is accepted by some timed automaton A. Then let n be the constant mentioned in Lemma 16. Now consider the Δ-timed word $w_\Delta = \langle a, 1\rangle\langle b, n\rangle\langle a, 1\rangle\langle b, n\rangle\langle a, 1\rangle\langle b, n\rangle \dots$, which obviously is in L_Δ. Let $J = \{\, 2i \mid i \geq 1\,\}$ and define the function $f : \mathbb{N} \to \mathbb{N}$ by $f(i) = i \cdot n$. But then *time-warp*$_{J,f}(w_\Delta)$ is not an element of the Δ-representation of the language under consideration, since the response times clearly diverge for the warped word. Thus, language L is not acceptable by any (aperiodic) timed automaton.

For practical purposes it would be nice if at least the complement of L is acceptable by a timed automaton, as this would be enough to do timed model checking. Unfortunately, this is not the case as the closure under complementation and the following (stronger) theorem show.

Theorem 17. *Let $L \subseteq (\Sigma \times \mathbb{N})^\omega$ be a non-empty timed language such that $L \subseteq D$, where language D is implicitly defined via its Δ-representation, which is $D_\Delta = \{\, \langle a_i, \Delta_i\rangle_{i\geq 1} \mid \forall c \geq 0 : \exists i \geq 1 : \Delta_i > c\,\}$. Then L is not acceptable by any timed automaton.*

[5] It is worth mentioning, that the time-warp lemma generalizes to *dense time* semantics, i.e., real valued clocks. Note that the statements in the remaining part of this section are also valid for the dense time semantics, although in the results and arguments discrete time is used, only.

Finally, we come back to the language L of convergent response time. Obviously, language L can be parameterized according to the response time c. This leads us to languages L_c, for $c \geq 0$, which are appropriately defined. In [1] it was shown that these languages are acceptable by deterministic timed Muller automata—for a formal definition of timed Muller automata we refer to [1]—and moreover it was conjectured that these languages are not acceptable by any deterministic timed (Büchi) automaton. The theorem given below solves this conjecture.

Theorem 18. *Let $c \geq 2$. Then the timed language L_c of constant response time c is not acceptable by any* deterministic *timed automaton.*

6 Conclusions

We have given structural and language theoretical characterizations for regular discrete timed languages, in the periodic as well as in the aperiodic case by means of introducing so-called tick automata and tick languages. The characterizations have several applications and furthermore we have developed the time-warp lemma, a tool very similar to a pumping lemma which can be conveniently used in order to show that certain languages can not be accepted by (aperiodic) timed automata. We hope that these results contribute to a more basic theory for timed languages as envisioned in [3].

References

1. R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.
2. R. Alur and T. Henzinger. Logics and models of real time: a survey. In *Real Time: Theory in Practice*, number 600 in LNCS. Springer, 1992.
3. Eugene Asarin. Challenges in timed languages: From applied theory to basic theory? *EATCS Bulletin*, 83:106–120, June 2004. Appeared in The Concurrency Column.
4. D. Beauquier. Pumping lemmas for timed automata. In *Proc. of FOSSACS '98*, number 1378 in LNCS, pages 81–94. Springer, January 1998.
5. B. Berard, A. Petit, V. Diekert, and P. Gastin. Characterization of the expressive power of silent transitions in timed automata. *Fundamenta Informaticae*, 36(2-3):145–182, 1998.
6. Ch. Choffrut and M. Goldwurm. Timed automata with periodic clock constraints. *Journal of Automata, Languages and Combinatorics*, 5(4):371–404, 2000.
7. Th. A. Henzinger, P. W. Kopke, and H. Wong-Toi. The expressive power of clocks. In *Proc. of ICALP '95*, number 944 in LNCS, pages 417–428, Springer, July 1995.
8. A. Pnueli and M. Vardi. Automata-theoretic approach to automated verification—lecture notes. *http://www.cs.rice.edu/~vardi/av.html*, 1999.
9. Th. Wilke. Specifying time state sequences in powerful logics and timed automata. In *Formal Techniques in Real-Time and Fault-Tolerant Systems*, number 863 in LNCS, pages 694–715. Springer, 1994.