On Recognizable Timed Languages*

Oded Maler[1] and Amir Pnueli[2,3]

[1] CNRS-VERIMAG, 2 Av. de Vignate, 38610 Gières, France
Oded.Maler@imag.fr
[2] Weizmann Institute of Science, Rehovot 76100, Israel
[3] New York University, 251 Mercer St. New York, NY 10012, USA
Amir.Pnueli@cs.nyu.edu

Abstract. In this work we generalize the fundamental notion of recognizability from untimed to timed languages. The essence of our definition is the existence of a right-morphism from the monoid of timed words into a *bounded* subset of itself. We show that the recognizable languages are exactly those accepted by deterministic timed automata and argue that this is, perhaps, the right class of timed languages, and that the closure of untimed regular languages under projection is a positive accident that cannot be expected to hold beyond the finite-state case.

1 Introduction

Let Σ^* be the free monoid generated by a finite set Σ. A set (language) $L \subseteq \Sigma^*$ is recognizable if there exists a finite deterministic automaton $\mathcal{A} = (Q, \delta, q_0, F)$ that accepts it. The automaton sends words into states via the mapping $\hat{\delta}_{\mathcal{A}} : \Sigma^* \to Q$ defined as $\hat{\delta}_{\mathcal{A}}(\varepsilon) = q_0$ and $\hat{\delta}_{\mathcal{A}}(w \cdot a) = \delta(\hat{\delta}_{\mathcal{A}}(w), a)$. A language L is recognizable if $L = \bigcup_{q \in F} \hat{\delta}_{\mathcal{A}}^{-1}(q)$ for some automaton $\mathcal{A}$.

There are two common ways to express these notions more algebraically. One is to speak of a monoid morphism φ from Σ^* to a finite monoid M satisfying $\varphi(w \cdot w') = \varphi(w) \cdot \varphi(w')$. The disadvantage of this approach is that the object under study is not anymore the "action" of a word w on the initial state, but rather the whole transformation it induces on Q. This object is a much less intuitive (and typically exponentially larger) than the automaton. An alternative, mentioned briefly in [E74], is to speak of *right modules* and of a module morphism from the free module (Σ^*, Σ) to the finite module (Q, Σ).

For the purpose of this paper we define an equivalent variation on this notion that will allow us to extend it easily to timed languages. Our definition is inspired by automaton learning theory [G72,A87] where every state of the automaton is identified with (one of) the first words[1] that reach it from q_0. The standard prefix partial-order on Σ^* is defined as $u \prec u \cdot v$ for every $u, v \in \Sigma^*$. A language is *prefix-closed* if it includes the prefixes

* This work was partially supported by a grant from Intel, by the European Community Projects IST-2001-35304 AMETIST (Advanced Methods for Timed Systems), http://ametist.cs.utwente.nl and by the CNRS project AS 93, Automates, modèles distribués et temporisés.

[1] That is, a word that reaches the state via a cycle-free run.

I. Walukiewicz (Ed.): FOSSACS 2004, LNCS 2987, pp. 348–362, 2004.

of all its elements. The *immediate exterior* of a prefix-closed language P is defined as $ext(P) = P \cdot \Sigma - P$, i.e. the first words that go outside P.

Definition 1 (Recognizable Languages). *A language L is recognizable if there exists a finite prefix-closed subset $P \subseteq \Sigma^*$, a "right"-morphism $\varphi : \Sigma^* \to P$ satisfying*

$$\varphi(w) = w \text{ if } w \in P \qquad \varphi(w \cdot w') = \varphi(\varphi(w) \cdot w')$$

and a subset $F \subseteq P$ such that $L = \bigcup_{w \in F} \varphi^{-1}(w)$.

As an example let us look at the deterministic automaton of Figure 1 and one of its spanning trees. The prefix-closed set $P = \{\varepsilon, b, ba, bb, baa, bbb, bbbb\}$ contains one representative for each of the states $\{q_0, \ldots, q_7\}$. The choice of P is not unique and may depend on the spanning tree chosen. For example, we could replace ba and baa by bba and $bbaa$ as representatives of q_2 and q_4, respectively. The morphism from Σ^* to P is defined, for elements outside P, via rewriting rules ("relations" in the algebraic jargon) that mimic the "non-spanning" transitions in the transition graph. Such a rewriting rule is defined for every element in $ext(P)$. In our example the rules are $a = \varepsilon$, $bba = ba$, $bab = \varepsilon$, $bbba = ba$, $baaa = baab = baa$, $bbbbb = baa$ and $bbbba = bbbb$. These rewriting rules can be applied only at the left of a word, that is, the rule $bba = a$ corresponds to the family $bbaw = aw$ for every $w \in \Sigma^*$.

The recognition of a word by this structure proceeds like reading the word by an automaton: a word w is scanned until a prefix $u \in ext(P)$ is detected, such that $w = uv$. Than the rewriting rule $u = u'$ is applied, reducing w to $w' = u'v$ with $u' = \varphi(u) \in P$ and the process is continued with w' until w is reduced to a word in P which is tested for membership in F (in our example $F = \{bb\}$).

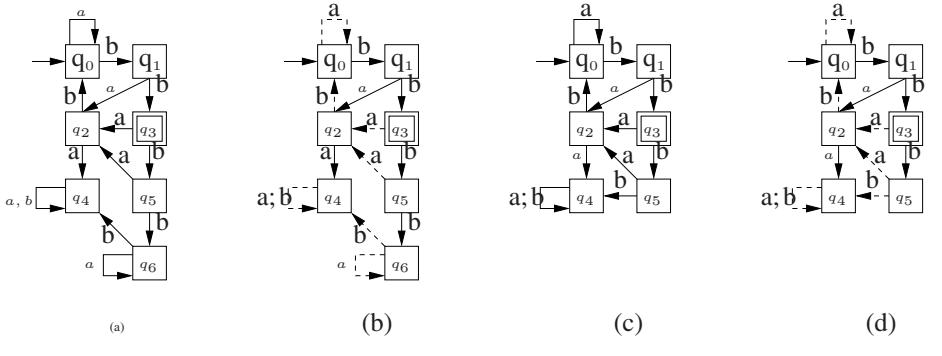

Fig. 1. (a) A deterministic automaton; (b) A spanning tree of the automaton (the solid lines); (c) A minimal automaton for the language accepted by the automaton in (a); (d) A spanning tree for the minimal automaton.

For untimed languages this exercise seems nothing more than a fancy formulation of acceptance by a finite automaton, yet it emphasizes the fundamental property of finite-state systems and languages: the ability to distinguish between a finite number of classes

of input histories. Before adapting this notion for timed languages let us recall some known facts about minimal automata and the notion of a state in a dynamical system.

Every $L \subseteq \Sigma^*$ admits a unique canonical automaton $\mathcal{A}_L$ (not necessarily finite-state) that accepts it. Any other automaton accepting L can be reduced to $\mathcal{A}_L$ by an automaton homomorphism (merging of states). This automaton is defined using the syntactic right-congruence[2] relation induced by L on Σ^*

$$u \sim v \text{ iff } \forall w\ uw \in L \iff vw \in L$$

The states of the minimal automaton for L are the equivalence classes of $\sim$. This is the Nerode part of the Myhill-Nerode characterization of regular languages as those for which $\sim$ has a finite index. A language like $a^n b^n$ can be proved non-recognizable by showing that $a^n \not\sim a^m$ for every $n \neq m$ and hence $\sim$ has an infinite index and no finite set of representatives of its congruence classes exists.

By choosing proper representatives for each class we can have a set P of minimal size. Figure 1-(c) shows a minimal automaton for our example. The corresponding algebraic object is obtained from the non-minimal one by removing $bbbb$ from P, removing the rules $bbbbb = baa$ and $bbbba = bbbb$ and adding the rule $bbbb = baa$.

2 Timed Languages

We consider timed languages as subsets of the *time-event monoid* $\mathcal{T} = \Sigma^* \uplus \mathbb{R}_+$, the free product (shuffle) of the free monoid $(\Sigma^*, \cdot, \varepsilon)$ and the commutative monoid $(\mathbb{R}_+, +, 0)$. This monoid has been introduced in [ACM02] as an alternative semantic domain for timed behaviors, where elements of Σ indicate events and elements of $\mathbb{R}_+$ denote passage of time. Elements of $\mathcal{T}$ can be written as timed words of the form

$$t_0 \cdot a_1 \cdot t_1 \cdot a_2 \cdot t_2 \cdots a_n \cdot t_n \tag{1}$$

with $t_i \geq 0$ and $a_i \in \Sigma \cup \{\varepsilon\}$ for every i. Such a word indicates passage of t_0 time, followed by the occurrence of a_1, followed by passage of t_1 time, etc. The reader may find in [ACM02] more precise details, examples and a definition of a canonical form to which two equivalent timed words can be reduced. For example, $a \cdot 0 \cdot a'$ can be reduced to $a \cdot a'$ and $t \cdot \varepsilon \cdot t'$ reduces to $t + t'$. The prefix partial-order relation on $\mathcal{T}$ is defined as $u \preceq u \cdot v$ for any $u, v \in \mathcal{T}$. Note that, in particular, $w \cdot t \preceq w \cdot t'$ whenever $t \leq t'$.

A timed word w of the form (1) can be projected onto Σ^* and $\mathbb{R}_+$, respectively, via the following two morphisms: The *untime* function, $\mu(w) = a_1 \cdot a_2 \cdots a_n$ and the *duration* function $\lambda(w) = t_0 + t_1 + \cdots + t_n$. For an untimed word u, $|u|$ indicates its logical length (number of letters). These functions are lifted naturally from individual words to sets of words.

It is clear that the notion of *finite* recognizability is useless for timed languages. It suffices to look at the singleton language $\{5 \cdot a\}$, consisting of the word where a occurs at time 5, and see that it has an uncountable number of Nerode classes as $t \not\sim t'$ for every

[2] A right-congruence relation of Σ^* is an equivalence relation such that $u \sim v$ implies $uw \sim vw$ for every w.

$t \neq t'$ where $t, t' < 5$. We believe that the suitable notion for timed languages is that of *boundedness* (which implies finiteness for discrete systems). Intuitively this means that one can distinguish between a finite number of classes of (qualitative) histories and in each of these classes it is possible to distinguish between durations taken from a bounded set.

Definition 2 (Bounded Timed Languages). *A timed language $L \subseteq \mathcal{T}$ is bounded if $\mu(L)$ is finite and $\lambda(L)$ is bounded in the usual sense of $\mathbb{R}_+$.*

We want to generalize Definition 1 to timed languages using a bounded prefix-closed subset P of $\mathcal{T}$ and a morphism to it. Before giving a formal definition let us illustrate the idea using the language

$$\{t \cdot a \cdot w : t \in [1,5], w \in \mathcal{T}\}$$

consisting of all timed words that have no letters until 1 and an occurrence of a somewhere in $[1,5]$. The set P should contain all the time prefixes t with $t \in [0,5]$. All the words of the form $t \cdot a$ with $t < 1$ are Nerode equivalent (they accept nothing) and can be represented by a and the same holds for all t with $t > 5$. Likewise, the words of the form $t \cdot a$ with $t \in [1,5]$ are equivalent (they accept everything) and hence can be represented by $1 \cdot a$. So for this language we have

$$P = \{t : t \in [0,5]\} \cup \{a\} \cup \{1 \cdot a\}, \quad F = \{1 \cdot a\}$$

The immediate exterior of P contains all the a-continuations of P which are outside P, namely the words $t \cdot a$ with $t \in (0,1) \cup (1,5]$ as well as $a \cdot a$ and $1 \cdot a \cdot a$. The immediate exterior via time passage is harder to define due to the density of $(\mathbb{R}, \leq)$. In general, given a timed word w, one cannot[3] characterize its "first" time continuation. One solution would be to take an arbitrarily small positive ϵ and let the exterior of w be $\{w \cdot t : t \in (0, \epsilon)\}$. We will use the notation $w \cdot t$ for that, and denote the corresponding elements of $ext(P)$ by $a \cdot t$, $1 \cdot a \cdot t$, and $5 \cdot t$. The morphism is defined using the following rewriting rules:

$$\{t \cdot a = a : t \in [0,1)\} \qquad \{t \cdot a = 1 \cdot a : t \in [1,5]\}$$

$$a \cdot a = a \cdot t = a \qquad 1 \cdot a \cdot a = 1 \cdot a \cdot t = 1 \cdot a \qquad 5 \cdot t = a$$

A discrete-time interpretation of this object appears in Figure 2. As one can see, we need a formalism to express *parameterized families of words* belonging to P and F as well as *parameterized families of rewriting rules*. The choice of this formalism depends on the type of dense-time automata whose expressive power we want to match. In this work we concentrate on timed automata and before doing so let us give an example of a non-recognizable timed language,

$$L_{bad} = \mathcal{T} \cdot \{a \cdot t_1 \cdot a \cdot t_2 \cdots t_n \cdot a : n \in \mathbb{N} \wedge \sum_{i=1}^{n} t_i = 1\}. \tag{2}$$

[3] Perhaps a definition can be given using non-standard analysis with infinitesimals, or by taking limits on a sequence of discretizations with decreasing time steps.

This language, which can be "accepted" by a non-deterministic timed automaton, was introduced by Alur [A90] to demonstrate the non-closure of timed automata under complementation. It is not hard to see that for every n

$$a \cdot t_1 \cdot a \cdots t_n \cdot a \not\sim a \cdot t_1 \cdot a \cdots t_n \cdot a \cdot t_{n+1} \cdot a$$

whenever $0 < \sum_{i=1}^{n+1} t_i < 1$ and hence for any P, $\mu(P)$ should contain the infinite language $\{a^n : n \in \mathbb{N}\}$ and P cannot be bounded.

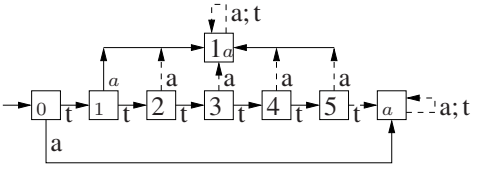

Fig. 2. An acceptor for a discrete time interpretation of $[1, 5] \cdot a \cdot \mathcal{T}$. Transitions labeled by t indicate passage of one time unit. Dashed arrows indicate non-spanning transitions that correspond to the rewriting rules.

3 Timed Automata

We consider Σ-labeled timed automata as acceptors of subsets of $\mathcal{T}$. Timed automata are automata operating in the dense time domain. Their state-space is a product of a finite set of discrete states (locations) and the clock-space $\mathbb{R}_+^m$, the set of possible valuations of a set of clock variables. The behavior of the automaton consists of an alternation of time-passage periods where the automaton stays in the same location and the clock values grow uniformly, and of instantaneous transitions that can be taken when clock values satisfy certain conditions and which may reset some clocks to zero.

The interaction between clock values and discrete transitions is specified by conditions on the clock-space which determine what future evolution, either passage of time or one or more transitions, is possible at a given part of the state-space. The clocks allow the automaton to remember, to a certain extent, some of the quantitative timing information associated with the input word. This ability is bounded due to the finite number of clocks and due to the syntactic restrictions on the form of the clock conditions, namely comparisons of clock values with a finite number of rational constants. This, combined with the monotonicity of clock growth, means that a clock becomes "inactive" after its value crosses the value of the maximal constant κ and it cannot distinguish in that state between time duration of length κ and of length $\kappa + t$ for any positive t.

Let $X = \{x_1, \ldots, x_m\}$ be a set of clock variables. A *clock valuation* is a function $\mathbf{x} : X \to \mathbb{R}_+$. We use $\mathbf{1}$ to denote the unit vector $(1, \ldots, 1)$ and $\mathbf{0}$ for the zero vector $(0, \ldots, 0)$.

Definition 3 (Clock and Zone Constraints). *A clock constraint is either a single clock constraint $x \ll d$ or a clock difference constraint $x_i - x_j \ll d$, where $\ll \in \{<, \leq, =, \geq, >\}$ and d is an integer. A zone constraint is a conjunction of clock constraints.*

Definition 4 (Timed Automaton).
A timed automaton is $\mathcal{A} = (\Sigma, Q, X, q_0, I, \Delta, F)$ *where* Q *is a finite set of states (locations),* X *is a finite set of clocks,* I *is the staying condition (invariant), assigning to every* $q \in Q$ *a zone* I_q*, and* Δ *is a transition relation consisting of elements of the form* (q, a, ϕ, ρ, q') *where* q *and* q' *are states,* $a \in \Sigma \cup \{\varepsilon\}$*,* $\rho \subseteq X$ *and* ϕ *(the transition guard) is a rectangular zone constraint. The initial state is* q_0 *and the acceptance condition* F *is a finite set of pairs of the from* (q, ϕ) *where* ϕ *is a zone constraint.*

A *configuration* of the automaton is a pair $(q, \mathbf{x})$ consisting of a location and a clock valuation. Every subset $\rho \subseteq X$ induces a reset function Reset_ρ on valuations which resets to zero all the clocks in ρ and leaves the other clocks unchanged. A *step* of the automaton is one of the following:

- A discrete step: $(q, \mathbf{x}) \xrightarrow{\delta} (q', \mathbf{x}')$, for some transition $\delta = (q, a, \phi, \rho, q') \in \Delta$, such that $\mathbf{x}$ satisfies ϕ and $\mathbf{x}' = \mathrm{Reset}_\rho(\mathbf{x})$. The label of such a step is a.
- A time step: $(q, \mathbf{x}) \xrightarrow{t} (q, \mathbf{x} + t\mathbf{1})$, $t \in \mathbb{R}_+$ such that $\mathbf{x} + t'\mathbf{1}$ satisfies I_q for every $t' < t$. The label of a time step is t.

A *run* of the automaton starting from the initial configuration $(q_0, \mathbf{0})$ is a finite sequence of steps

$$\xi : \quad (q_0, \mathbf{0}) \xrightarrow{s_1} (q_1, \mathbf{x}_1) \xrightarrow{s_2} \cdots \xrightarrow{s_n} (q_n, \mathbf{x}_n).$$

A run is accepting if it ends in a configuration satisfying F. The timed word carried by the run is obtained by concatenating the step labels. The timed language accepted by a timed automaton $\mathcal{A}$ consists of all words carried by accepting runs and is denoted by $L_{\mathcal{A}}$.

A timed automaton is deterministic if from every reachable configuration every event and "non-event" leads to exactly one configuration. This means that the automaton cannot make both a "silent" transition and a time passage in the same configuration.

Definition 5 (Deterministic Timed Automaton). *A deterministic timed automaton is an automaton whose guards and staying conditions satisfy:*

1. *For every two distinct transitions* $(q, a, \phi_1, \rho_1, q_1)$ *and* $(q, a, \phi_2, \rho_2, q_2)$*,* ϕ_1 *and* ϕ_2 *have an empty intersection (event determinism).*
2. *For every transition* $(q, \varepsilon, \phi, \rho, q') \in \Delta$*, the intersection of* ϕ *with* I_q *is, at most, a singleton (time determinism).*

In deterministic automata any word is carried by exactly one run. We denote the class of timed languages accepted by such automata by DTA.

Before defining the recognizable timed languages let us present a particular atomic type of zones called *regions*, introduced in [AD94], which play a special role in the theory of timed automata. Intuitively a region consists of all clock valuations that are not (and will not be) distinguishable by any clock constraint. A region constraint is a zone constraint where for every x it contains a constraint of one of the following forms: $x = d$, $d < x < d + 1$ or $\kappa < x$ and for every pair of clocks — either $x_i - x_j = d$ or

$d < x_i - x_j < d + 1$. The set of all regions over m clocks with a largest constant[4] κ is denoted by G^m_κ.

Regions are the elementary zones from which all other zones can be built. Two clock valuations that belong to the same region satisfy the same guards and staying conditions. Moreover, by letting time pass from any two such points, the next visited region is the same. Finally, any reset of clocks sends all the elements of one region into the same region. This motivates the definition [AD94] of the "region automaton", a finite-state automaton whose state space is $Q \times G^m_\kappa$ and its transition relation is constructed as follows. First we introduce a special symbol τ which indicates the passage of an under-specified amount of time, and connect two regions $\mathcal{R}$ and $\mathcal{R}'$ by a τ-transition, denoted by $(q, \mathcal{R}) \xrightarrow{\tau} (q, \mathcal{R}')$ if time can progress in $(q, \mathcal{R})$ and $\mathcal{R}'$ is the next region encountered while doing so. Secondly, for every transition (q, a, ϕ, ρ, q') and every $\mathcal{R}$ which satisfies ϕ we define a transition $(q, \mathcal{R}) \xrightarrow{a} (q', \mathcal{R}')$ if $\mathcal{R}'$ is the result of applying Reset_ρ to $\mathcal{R}$. As an example consider the deterministic automaton and its corresponding region automaton appearing on Figure 3. The automaton accepts any word with 3 a's such that the second occurs 1 time after the beginning and the third — 1 time after the first.[5]

4 Recognizable Timed Languages

Let $T_n = \{t_0, \ldots, t_n\}$ be an ordered set of non-negative real variables. A *contiguous sum* over T_n is $S_{j..k} = \sum_{i=j}^{k} t_i$ and the set of all such sums over T_n is denoted by $\mathcal{S}^n$. A *timed inequality* on T_n is a condition of the form $S_{i..j} \in J$ where J is an interval with natural endpoints. A *timed condition* is a conjunction of timed inequalities.

A timed language L is *elementary* if $\mu(L) = \{u\}$ with $u = a_1 \cdots a_n$ and the set $\{(t_0, \ldots t_n) : t_0 \cdot a_1 \cdots a_n \cdot t_n \in L\}$ is definable by a timed condition Λ. We will sometime denote elementary languages by a pair (u, Λ). The immediate exterior $ext(L)$ of an elementary language $L = (u, \Lambda)$ consists of the following sets: for every $a \in \Sigma$, $ext^a(L)$ is the set $(u \cdot a, \Lambda^a)$ where $\Lambda^a = \Lambda \cup \{t_{n+1} = 0\}$. The immediate exterior via time passage is $ext^t(L) = (u, \Lambda^t)$ where Λ^t is obtained from Λ as follows. If Λ contains one or more equality constraints of the form $S_{j..n} = d$, these constraints are replaced by constraints of the form $d < S_{j..n}$. Otherwise, let j be the smallest number such that a constraint of the form $S_{j..n} < d$ appears in Λ. This constraint is replaced by $S_{j..n} = d$.

Definition 6 (Chronometric Subset). *A subset P of $\mathcal{T}$ is chronometric if it can be written as a finite union of disjoint elementary languages.*

[4] There are some simplifications in the description in order to avoid a full exposition of the theory of timed automata. In particular, if some clock $x > \kappa$ in some region, we do not care anymore about its comparisons with other clocks. This way the region automaton has just one terminal state in which all the clocks are larger than κ. Readers interested in all the subtle details may consult [B03].

[5] Note that the existence of two transitions leaving q_2, one labeled with $x = 1$ and one with $x = 1, a$, is not considered a violation of determinism. A word $1 \cdot t$ for an arbitrarily small t will take the former and the word $1 \cdot a$ will take the latter.

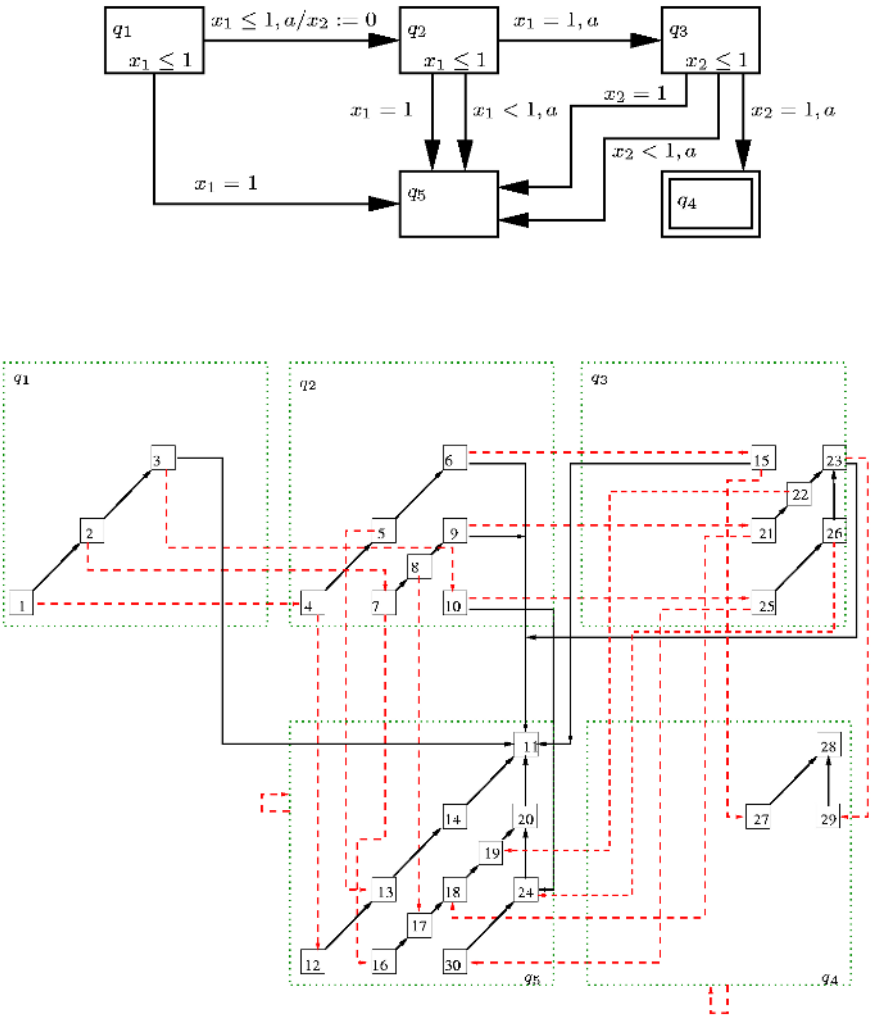

Fig. 3. A timed automaton with 2 clocks and its region automaton. Solid arrows indicate time passage and ε transitions while dashed arrows are a transitions. The a-labeled self-loops from all regions associated with q_4 and q_5 are depicted in a StateChart style. The regions are detailed in Table 1.

Definition 7 (Chronometric Relational Morphism). *Let P be bounded and prefix-closed subset of $\mathcal{T}$. A chronometric (relational) morphism Φ from $\mathcal{T}$ to P is a relation definable by a finite set of tuples $(u, \Lambda, u', \Lambda', E)$ such that each (u, Λ) is an elementary language included in $ext(P)$, each (u', Λ') is an elementary language contained in P, and E is a set of equalities of the form $\sum_{i=j}^{n} t_i = \sum_{i=k}^{n'} t'_i$, where $n = |u|$ and $n' = |u'|$. It is required that all (u, Λ) are disjoint and their union is equal to $ext(P)$. For every $w = t_0 \cdot a_1 \cdots a_n \cdot t_n$ and $w' = t'_0 \cdot b_1 \cdots b_{n'} \cdot t'_{n'}$, $(w, w') \in \Phi$ iff there exists a tuple*

$(u, \Lambda, u', \Lambda', E)$ in the presentation of Φ such that $w \in (u, \Lambda)$, $w' \in (u', \Lambda')$ and the respective time values for w and w' satisfy all the equalities in E. The definition of Φ for words outside $ext(P)$ is done via the identity $\Phi(u \cdot v) = \Phi(\Phi(u) \cdot v)$.

As an example of a component $(u, \Lambda, u', \Lambda', E)$ of a chronometric morphism let

$$(u, \Lambda) = (t_0 \cdot a \cdot t_1 \cdot a \, , \{0 < t_0 < 1, 0 < t_1 < 1, 0 < t_0 + t_1 < 1\}),$$

$$(u', \Lambda') = (r_0 \cdot a \cdot r_1 \cdot a \cdot r_2 \, , \{0 < r_0 < 1, r_1 = 0, 0 < r_2 < 1, 0 < r_0 + r_1 + r_2 < 1\})$$

and

$$E \;\; = \;\; t_0 + t_1 = r_0 + r_1 + r_2.$$

This component corresponds to the non-spanning transition $\mathcal{R}_8 \cdot a = \mathcal{R}_{17}$ in the region automaton of Figure 3.

The relation Φ is said to be *well formed* if the following holds for each tuple $(u, \Lambda, u', \Lambda', E)$ in Φ:

- For every $w \in (u, \Lambda)$, there exists $w' \in (u', \Lambda')$ such that $(w, w') \in \Phi$.
- For every $w' \in (u', \Lambda')$, there exists $w \in (u, \Lambda)$ such that $(w, w') \in \Phi$.

A relation Φ is said to be *compatible* with a chronometric subset F if for every $(u, \Lambda, u', \Lambda', E)$ in Φ, either $(u', \Lambda') \subseteq F$ or $(u', \Lambda') \cap F = \emptyset$.

Remark: From a well formed relational chronometric morphism Φ one can derive a (functional) chronometric morphism $\varphi : \mathcal{T} \to P$ by letting $\varphi(w)$ be some w' such that $(w, w') \in \Phi$. From the relation described above we can derive functional morphisms such as $\varphi(t_0 \cdot a \cdot t_1 \cdot a) = t_0 \cdot a \cdot a \cdot t_1$, or $\varphi(t_0 \cdot a \cdot t_1 \cdot a) = a \cdot a \cdot (t_0 + t_1)$. While functional morphisms follow more closely the spirit of classical theory, relational morphisms are more suitable for the proofs in this paper.

Definition 8 (Recognizable Timed Languages). *A timed language L is recognizable if there is a chronometric prefix-closed set P, a chronometric subset F of P and a chronometric relational morphism $\Phi : \mathcal{T} \to P$ compatible with F such that $L = \bigcup_{w \in F} \Phi^{-1}(w)$.*

4.1 From Deterministic Automata to Recognizable Languages

We are now ready to prove the first result, stating that every language accepted by a DTA is recognizable, by assigning timed words to reachable configurations. The correspondence between values of clock variables in the automaton and values of time variables in an input word of length n is done via a *clock binding* over (X, T_n), a function $\beta : X \to \mathcal{S}^n$ associating with every clock x a contiguous sum of the form $S_{j..n}$. Recall that a region is a conjunction of single clocks constraints and clock difference constraints. By substituting $\beta(x)$ for x, the former become timed inequalities and the latter become inequalities on $S_{j..n} - S_{k..n} = S_{j..k}$ and, hence, timed inequalities as well.

Claim 1 (DTA $\Rightarrow$ REC) *From every deterministic timed automaton $\mathcal{A}$ one can construct a chronometric prefix-closed subset P of $\mathcal{T}$ and a morphism $\Phi : \mathcal{T} \to P$ such that if $(w, w') \in \Phi$ then w and w' lead to the same configuration from the initial state.*

Sketch of Proof: Build the region automaton for $\mathcal{A}$ and pick a spanning tree in which each region is reached via a simple path. Starting from the root we associate with every region an elementary timed language in a prefix-closed manner. More precisely with every region $\mathcal{R}$ of the automaton we associate the triple (u, Λ, β) where (u, Λ) is an elementary timed language with $|u| = n$ and β is a clock binding on (X, T_n). We decompose Λ into two sets of timed inequalities Λ_- and Λ_+ where Λ_- consists of the "anachronistic" inequalities not involving t_n and Λ_+ — of "live" constraints involving t_n. Note that transitions may change the binding and move some inequalities from Λ_+ to Λ_-.

For the initial region $\mathcal{R}_0 = (q_0, \mathbf{0})$, $u = \varepsilon$, $\Lambda = \Lambda_+$ is $t_0 = 0$ and all clocks are bound to t_0. Consider now the inductive step. Given a region $\mathcal{R}$ with (u, Λ, β) we compute (u', Λ', β') for its successor (via a spanning transition) $\mathcal{R}'$. There are two cases:

1. $\mathcal{R}'$ is a simple time successor of $\mathcal{R}$: in this case $u' = u$ and $\beta' = \beta$. We let $\Lambda'_- = \Lambda_-$ and obtain Λ'_+ from the region formula ψ' by replacing every clock x by $\beta(x)$.[6]
2. $\mathcal{R}'$ is a transition successor of $\mathcal{R}$ via an a-labeled[7] transition: in this case $u' = u \cdot a \cdot t_{n+1}$, we have a new time variable t_{n+1} and the (T^{n+1}, X) binding β' is derived from β and from the corresponding transition as follows. If a clock x is not reset by the transition then $\beta'(x) = S_{i..n+1}$ whenever $\beta(x) = S_{i..n}$. If x is reset then $\beta'(x) = t_{n+1}$ (note that $x = 0$ in $\mathcal{R}'$). To compute Λ'_- we add to Λ_- the substitution of $\beta(x)$ for x in ψ and let Λ'_+ be the substitution of $\beta'(x)$ in ψ'.

From this construction it is easy to see that the union of the obtained languages is prefix-closed (we proceed by concatenation and by respecting past timing constraints) and chronometric and that all reachable configurations are covered by words.

Next, we construct the relation Φ based on transitions which correspond to back- or cross-edges in the spanning tree. Consider a non-spanning transition leading from region $\mathcal{R}$ with characteristic (u, Λ, β) into region $\mathcal{R}'$ with characteristic (u', Λ', β'). Let $(u'', \Lambda'', \beta'')$ be the language and binding associated with the successor of $\mathcal{R}$ according to the previously described procedure. This transition contributes to Φ the tuple $(u'', \Lambda'', u', \Lambda', E)$. For each clock x which is not reset by the transition, E contains the equality $\beta'(x) = \beta''(x)$. If x is reset by the transition, then E contains the equality $\beta'(x) = 0$. ∎

Table 1 shows the correspondence between the regions of Figure 3 and elementary languages. The numbering of the regions is consistent with the chosen spanning tree.

4.2 From Recognizable Languages to Deterministic Automata

We will now prove the other direction by building a deterministic timed automaton for a given recognizable language. To facilitate the construction we will use an extended form of timed automata, proposed in [SV96], where transitions can be labelled by assignments

[6] Note that Λ_+ and Λ'_+ are very similar consisting of almost identical sets of inequalities differing from each other only by replacing one or more inequalities of the form $S_{i..n} = d$ by $d < S_{i..n} < d + 1$, etc.

[7] The special case where the transition is not labeled is resolved by introducing a new time variable t_{n+1} such that the word can be written as $t_0 \cdots a_n \cdot t_n \cdot \varepsilon \cdot t_{n+1}$.

of the form $x := 0$ and $x := y$ (clock renaming). As shown in [SV96] such automata can be transformed into standard timed automata (see also [BDFP00]).

Claim 2 ($REC \Rightarrow DTA$) *From every chronometric subset P of $\mathcal{T}$ and a chronometric morphism $\Phi : \mathcal{T} \to P$ one can build a DTA $\mathcal{A}$ such that if two timed words lead to the same configurations in $\mathcal{A}$ then $(w, w') \in \Phi$.*

Sketch of Proof: The construction of the automaton starts with an untimed automaton (with a tree structure) whose set of states is $\mu(P)$ with ε as the initial state and a transition function such that $\delta(u, a) = u \cdot a$ whenever $u \cdot a$ is in $\mu(P)$. We then decorate the automaton with staying conditions, transition guards, and resets as follows. With every transition we reset a new clock so that for every word $t_0 \cdot a_1 \cdots a_n \cdot t_n$, the value of clock x_i at any state $a_1 \cdots a_j$, $i \leq j$ is bound to $S_{i..j}$.

For every state $u = a_1 \cdots a_n \in \mu(P)$ let

$$\Lambda(u) = \{(t_0, \ldots, t_n) : t_0 \cdot a_1 \cdots a_n \cdot t_n \in P\}.$$

By decomposing $\Lambda(u)$ into anachronistic (Λ_-) and live (Λ_+) constraints and substituting x_i instead of every $S_{i..n}$ in Λ_+, we obtain the staying condition for state u.

For every u and a such that $u \cdot a$ is in $\mu(P)$ let

$$H_{u,a} = \{(t_0, \ldots, t_n) : t_0 \cdot a_1 \cdots a_n \cdot t_n \cdot a \in P\}.$$

Without loss of generality we assume that $H_{u,a}$ is definbable by a timed condition.[8] Hence every expression $S_{i..j}$ in it can be replaced by $x_j - x_i$ and the whole condition can be transformed into a zone constraint that will serve as the guard of the transition between u and $u \cdot a$. This way we have an automaton in which every element of P reaches a distinct configuration.

Consider an element $(u \cdot a, \Lambda, u', \Lambda', E) \in \Phi$, such that $(u \cdot a, \Lambda) \in ext^a(P)$, with $|u| = n$ and $|u'| = n'$. Such an element introduces into the constructed automaton an a-labeled transition from u to u'. For every constraint of the form $S_{j..k} \in J$ included in Λ, we include in the transition guard the constraint $x_j - x_k \in J$. For every equality $S'_{j..n'} = S_{k..n}$ included in E, we add to the reset function the assignment $x_j := x_k$.

Likewise every $(u, \Lambda, u', \Lambda', E) \in \Phi$ such the $(u, \Lambda) \subseteq ext^t(P)$ induces a timed transition from u to u' with a guard and a reset function similar to the previous case. ∎

Corollary 1 (REC=DTA). *The recognizable timed languages are those accepted by a deterministic timed automaton.*

5 Discussion

Ever since the introduction of timed automata and the observation that their languages are not closed under complementation, researchers were trying to find a well-behaving subclass of languages.[9] Among the proposals given, we mention the *event-clock automata* of

[8] In general it could be definable by a finite union of timed conditions and we should make several transitions from u to $u \cdot a$.

[9] The question whether a non-deterministic timed automaton can be determinized is undecidable, see [T03].

Table 1. Correspondence between regions in the automaton of Figure 3 and timed words.

$\mathcal{R}$	q	ψ	u	β	Λ
1	q_1	$0 = x_2 = x_1$	t_0	$x_1 = t_0$ $x_2 = t_0$	$t_0 = 0$
2	q_1	$0 < x_2 = x_1 < 1$	t_0	$x_1 = t_0$ $x_2 = t_0$	$0 < t_0 < 1$
3	q_1	$x_2 = x_1 = 1$	t_0	$x_1 = t_0$ $x_2 = t_0$	$t_0 = 1$
4	q_2	$0 = x_2 = x_1$	$t_0 a t_1$	$x_1 = t_0 + t1$ $x_2 = t_1$	$t_0 = t_1 = 0$
5	q_2	$0 < x_2 = x_1 < 1$	$t_0 a t_1$	$x_1 = t_0 + t1$ $x_2 = t_1$	$t_0 = 0\,0 < t_1 < 1$
6	q_2	$x_2 = x_1 = 1$	$t_0 a t_1$	$x_1 = t_0 + t1$ $x_2 = t_1$	$t_0 = 0\,t_1 = 1$
7	q_2	$0 = x_2 < x_1 < 1$	$t_0 a t_1$	$x_1 = t_0 + t1$ $x_2 = t_1$	$0 < t_0 < 1\,t_1 = 0$
8	q_2	$0 < x_2 < x_1 < 1$	$t_0 a t_1$	$x_1 = t_0 + t1$ $x_2 = t_1$	$0 < t_0 < 1\,0 < t_1 < 1\,0 < t_0 + t_1 < 1$
9	q_2	$0 < x_2 < x_1 = 1$	$t_0 a t_1$	$x_1 = t_0 + t1$ $x_2 = t_1$	$0 < t_0 < 1\,0 < t_1 < 1\,t_0 + t_1 = 1$
10	q_2	$x_2 = 0\,x_1 = 1$	$t_0 a t_1$	$x_1 = t_0 + t1$ $x_2 = t_1$	$t_0 = 1\,t_1 = 0$
11	q_5	$x_1 > 1\,x_2 > 1$	$t_0 \varepsilon t_1$	$x_1 = t_0 + t_1$ $x_2 = t_0 + t_1$	$1 < t_0 + t_1$
12	q_5	$x_1 = x_2 = 0$	$t_0 a t_1 a t_2$	$x_1 = t_0 + t_1 + t_2$ $x_2 = t_1 + t_2$	$t_0 = t_1 = t_2 = 0$
13	q_5	$0 < x_1 = x_2 < 1$	$t_0 a t_1 a t_2$	$x_1 = t_0 + t_1 + t_2$ $x_2 = t_1 + t_2$	$t_0 = t_1 = 0\,0 < t_2 < 1$
14	q_5	$x_1 = x_2 = 1$	$t_0 a t_1 a t_2$	$x_1 = t_0 + t_1 + t_2$ $x_2 = t_1 + t_2$	$t_0 = t_1 = 0\,t_2 = 1$
15	q_3	$x_2 = x_1 = 1$	$t_0 a t_1 a t_2$	$x_1 = t_0 + t1 + t_2$ $x_2 = t_1 + t_2$	$t_0 = 0\,t_1 = 1\,t_2 = 0$
16	q_5	$0 = x_2 < x_1 < 1$	$t_0 a t_1 a t_2$	$x_1 = t_0 + t1 + t_2$ $x_2 = t_1 + t_2$	$0 < t_0 < 1\,t_1 = 0\,t_2 = 0$
17	q_5	$0 < x_2 < x_1 < 1$	$t_0 a t_1 a t_2$	$x_1 = t_0 + t1 + t_2$ $x_2 = t_1 + t_2$	$0 < t_0 < 1\,t_1 = 0\,0 < t_2 < 1$ $0 < t_0 + t_2 < 1$
18	q_5	$0 < x_2 < x_1 = 1$	$t_0 a t_1 a t_2$	$x_1 = t_0 + t1 + t_2$ $x_2 = t_1 + t_2$	$0 < t_0 < 1\,t_1 = 0\,0 < t_2 < 1$ $0 < t_0 + t_2 = 1$
19	q_5	$0 < x_2 < 1 < x_1$	$t_0 a t_1 a t_2$	$x_1 = t_0 + t1 + t_2$ $x_2 = t_1 + t_2$	$0 < t_0 < 1\,t_1 = 0\,0 < t_2 < 1$ $0 < t_0 + t_2 > 1$
20	q_5	$0 < x_2 = 1 < x_1$	$t_0 a t_1 a t_2$	$x_1 = t_0 + t1 + t_2$ $x_2 = t_1 + t_2$	$t_0 = 1\,t_1 = t_2 = 0\,t_3 = 1$
21	q_3	$0 < x_2 < x_1 = 1$	$t_0 a t_1 a t_2$	$x_1 = t_0 + t1 + t_2$ $x_2 = t_1 + t_2$	$0 < t_0 < 1\,0 < t_1 < 1$ $t_0 + t_1 = 1\,t_2 = 0$
22	q_3	$0 < x_2 < 1 < x_1$	$t_0 a t_1 a t_2$	$x_1 = t_0 + t1 + t_2$ $x_2 = t_1 + t_2$	$0 < t_0 < 1\,0 < t_1 < 1$ $t_0 + t_1 + t_2 > 1\,0 < t_2 < 1\,t_1 + t_2 < 1$
23	q_3	$0 < x_2 = 1 < x_1$	$t_0 a t_1 a t_2$	$x_1 = t_0 + t1 + t_2$ $x_2 = t_1 + t_2$	$0 < t_0 < 1\,0 < t_1 < 1$ $t_0 + t_1 + t_2 > 1\,0 < t_2 < 1\,t_1 + t_2 = 1$
24	q_5	$0 < x_2 < 1 < x_1 = 1$	$t_0 a t_1 \varepsilon t_2$	$x_1 = t_0 + t1 + t_2$ $x_2 = t_1 + t_2$	$t_0 = 1\,t_1 = t_2 = 0\,0 < t_3 < 1$
25	q_3	$x_1 = 1\,x_2 = 0$	$t_0 a t_1 a t_2$	$x_1 = t_0 + t1 + t_2$ $x_2 = t_1 + t_2$	$t_0 = 1\,t_1 = t_2 = 0$
26	q_3	$0 < x_2 < 1 < x_1$	$t_0 a t_1 a t_2$	$x_1 = t_0 + t1 + t_2$ $x_2 = t_1 + t_2$	$t_0 = 1\,t_1 = 0\,0 < t_2 < 1$
27	q_4	$x_2 = x_1 = 1$	$t_0 a t_1 a t_2 a t_3$	$x_1 = t_0 + t1 + t_2 + t_3$ $x_2 = t_1 + t_2 + t_3$	$t_0 = 0\,t_1 = 1\,t_2 = 0\,t_3 = 0$
28	q_4	$1 < x_1 1 < x_2$	$t_0 a t_1 a t_2 a t_3$	$x_1 = t_0 + t1 + t_2 + t_3$ $x_2 = t_1 + t_2 + t_3$	$t_0 = 1\,t_1 = 0\,t_2 = 1\,t_3 > 0$
29	q_4	$0 < x_2 = 1 < x_1$	$t_0 a t_1 a t_2 a t_3$	$x_1 = t_0 + t1 + t_2 + t_3$ $x_2 = t_1 + t_2 + t_3$	$t_0 = 1\,t_1 = 0\,t_2 = 1\,t_3 = 0$
30	q_5	$x_1 = 1\,x_2 = 0$	$t_0 a t_1 a t_2 a t_3$	$x_1 = t_0 + t_1 + t_2 + t_3$ $x_2 = t_1 + t_2 + t_3$	$t_0 = 1\,t_1 = t_2 = t_3 = 0$

[AFH99] where for each letter in the alphabet, the automaton can measure only the time since its *last* occurrence. It was shown that these languages admit a deterministic timed acceptor. Recognizable timed languages take this idea further by allowing the automaton to remember the occurrence times of a *finite* number of events, not necessarily of distinct types.

The ideas of [AFH99] were developed further in [RS97] and [HRS98], resulting in a rich class of timed languages characterized by a decidable logic. While being satisfactory from a logical point of view, the automaton characterization of this class is currently very complicated, involving cascades of event-recording and event-predicting timed

automata. We feel that our more restricted class of recognizable languages captures the natural extension of recognizability toward timed languages, namely which classes of input histories can be distinguished by a finite number of states and a finite number of bounded clocks.[10]

Deterministic timed languages have not been studied much in the literature due to several reasons. The first is a slight confusion about what *deterministic* means in this context and between acceptors and generators in general. A transition guarded by a "fat" condition of the form $x \in [l, u]$ is non-deterministic only if it is *not* labeled by an input letter. If it is labeled by an input a the transition is deterministic, reacting differently to $t \cdot a$ and $t' \cdot a$ for $t \neq t'$.

Another reason for ignoring deterministic automata is the centrality of the equivalence between DFA and NDFA in the untimed theory which serves to show that regular languages are closed under projection. Recognizable timed languages are indeed *not* closed under projection. The non-recognizable language L_{bad} (2) can be obtained from a recognizable language over $\{a, b\}$ by projecting away b. Not seeing b, the automaton has to "guess" at certain points, whether b has occurred. When this guessing has to be done a finite number of times, the Rabin-Scott subset construction can simulate it by a DFA that goes simultaneously to all possible successors. However when these hidden events can occur unboundedly within a finite interval and their occurrence times should be memorized, finite subset construction is impossible. In this context it is worth mentioning the result of [W94] about the determinizability of timed automata under a uniform bounded variability assumption and also to point out that for the same reasons determinization is always possible under any time discretization.

The closest work to ours, in the sense of trying to establish a semantic input-output definition of a state in a timed system, is [SV96], motivated by testing of timed automata. In that paper the authors give an algorithm for semantic minimization of timed automata and also make useful observations about clock permutations and assignments and about the relevance of clocks in various states. Similar observations were made in [DY96] where clock activity analysis was used to reduce the dimensionality of the clock space in order to save memory during verification.

Another related work is that of [BPT03] which is concerned with *data languages*, languages over an alphabet $\Sigma \times D$ where D is some infinite domain. Based on ideas developed in [KF94], they propose to recognize such languages using automata augmented with auxiliary registers that can store a finite number of data elements but not perform computations on these values. The results in [BPT03] show that acceptance by such automata coincides with their notion of recognizability by a finite monoid. These very general results can be specialized to timed languages by interpreting D as absolute time and every pair $(a, d) \in \Sigma \times D$ as a letter a and a time stamp d. Although the special nature of time can be imposed via monotonicity restrictions on the d's, we feel more comfortable with our more "causal" treatment of time as an entity whose elapse is consumed by the automaton in the same way input events are. Other investigations of the algebraic aspects of timed languages are reported in [D01].

[10] Note that in the untimed theory recognizability implies decidability but not vice versa, for example the emptiness problem for push-down automata is decidable.

To summarize, we have defined what we believe to be the appropriate notion of recognizability for timed systems and have shown that it coincides with acceptance by a deterministic timed automaton. We believe that this is the "right" class of timed languages and we have yet to see a useful and realistic timed language which is outside this class. Our result also makes timed theory closer to the untimed one and opens the way for further algebraic investigations of timed languages.

Let us conclude with some open problems triggered by this work:

1. What happens if contiguous sums are replaced by arbitrary sums or by linear expressions with positive coefficients? Clearly, the former case corresponds to "stopwatch automata" and the latter to some class of hybrid automata and it is interesting to see whether such a study can shed more light on problems related to these automata.
2. Is there a natural restriction of the timed regular expressions of [ACM02] which guarantees recognizability? Unfortunately, dropping the renaming operation will not suffice because the language L_{bad} (2) can be expressed without it.
3. Can our results be used to develop an algorithm for learning timed languages from examples and for solving other related problems such as minimization and test generation?
4. Can recognizability be related to the growth of the index of the Nerode congruence for a discretization of the language as time granularity decreases?

Acknowledgment. This work benefited from discussions and monologues with Eugene Asarin, Stavros Tripakis, Pascal Weil, Yassine Lakhnech, Paul Caspi and Sergio Yovine, as well as from thoughtful comments from anonymous referees that improved the correctness and presentation of the results.

References

[A90] R. Alur, *Techniques for Automatic Verification of Real-Time Systems*, PhD Thesis, Stanford, 1990.

[AD94] R. Alur and D.L. Dill, A Theory of Timed Automata, *Theoretical Computer Science* 126, 183–235, 1994.

[AFH99] R. Alur, L. Fix, and T.A. Henzinger, Event-Clock Automata: A Determinizable Class of Timed Automata, *Theoretical Computer Science* 211, 253-273, 1999.

[A87] D. Angluin, Learning Regular Sets from Queries and Counter-Examples, *Information and Computation* 75, 87-106, 1987.

[ACM02] E. Asarin, P. Caspi and O. Maler, Timed Regular Expressions *The Journal of the ACM* 49, 172-206, 2002.

[B03] P. Bouyer, Untameable Timed Automata!, *Proc. STACS'03*, 620-631, LNCS 2607, Springer, 2003.

[BDFP00] P. Bouyer, C. Dufourd, E. Fleury and A. Petit, Expressiveness of Updatable Timed Automata, *Proc. MFCS'2000*, 232-242, LNCS 1893, Springer, 2000.

[BPT03] P. Bouyer, A. Petit, and D. Thérien, An algebraic Approach to Data Languages and Timed Languages, *Information and Computation* 182, 137-162, 2003.

[D01] C. Dima, Real-Time Automata, *Journal of Automata, Languages and Combinatorics* 6, 3-24, 2001.

[DY96] C. Daws and S. Yovine, Reducing the Number of Clock Variables of Timed Automata, *Proc. RTSS'96*, 73-81, IEEE, 1996.

[E74] S. Eilenberg, *Automata, Languages and Machines, Vol. A*, Academic Press, New-York, 1974.

[G72] E.M. Gold, System Identification via State Characterization, *Automatica* 8, 621-636, 1972.

[HRS98] T.A. Henzinger, J.-F. Raskin, and P.-Y. Schobbens, The Regular Real-time Languages, *Proc. ICALP 98*, 580-591, LNCS 1343, Springer, 1998.

[KF94] M. Kaminski and N. Francez, Finite-memory Automata, *Theoretical Computer Science* 134, 329-363, 1994.

[RS97] J.-F. Raskin and P.-Y. Schobbens, State Clock Logic: A Decidable Real-Time Logic, in *Hybrid and Real-Time Systems (HART)*, 33-47, LNCS 1201, Springer, 1997.

[T03] S. Tripakis, Folk Theorems on the Determinization and Minimization of Timed Automata, *Proc. FORMATS'03*, 2003.

[SV96] J.G. Springintveld and F.W. Vaandrager, Minimizable Timed Automata, *Proc. FTRTFT'96*, 130-147, LNCS 1135, Springer, 1996.

[W94] Th. Wilke, Specifying State Sequences in Powerful Decidable Logics and Timed Automata, *Proc. FTRTFT'94*, LNCS 863, 694-715, Springer, 1994.