Thin and Thick Timed Regular Languages

Nicolas Basset[1] and Eugene Asarin[2]

[1] LIGM, Université Paris-Est and CNRS
nbasset@dptinfo.ens-cachan.fr
[2] LIAFA, Université Paris Diderot and CNRS
asarin@liafa.jussieu.fr

Abstract. In previous literature on timed automata, it was noticed that they are in several aspects too precise, which leads sometimes to strange artifacts, mathematical pathologies or unrealistic models. In particular, some timed automata are non-implementable, non-robust, behave badly under discretization, have many Zeno runs etc. In this paper, we propose a unifying approach to most of these issues for deterministic timed automata. We classify these automata either as **thin** or as **thick**. In thin automata, all the infinite trajectories are, in some weak sense, Zeno; the discretization of long trajectories is difficult, since it requires very small discretization step. In thick automata, most of trajectories are non-Zeno and behave well under discretization; such automata satisfy a sort of pumping lemma. Formally, the thin-thick alternative is based on the notion of entropy of timed regular languages introduced by E. Asarin and A. Degorre in [3,4]. Thin languages have the entropy $= -\infty$ while thick have a larger one. An important application of thin-thick alternative is again the entropy theory of timed languages. We show that the entropy can be computed with a desired precision using discretization and thus it is computable, which closes a question left open in [3,4].

1 Introduction

Timed automata [2] using exact continuous clocks, exact guards and resets are a beautiful mathematical object and a useful model of real-time systems. However, from the very beginning of the timed automata research, it was clear that they are in several aspects too precise, which leads sometimes to strange artifacts, mathematical pathologies or unrealistic models. Several lines of research have partially elucidated these issues.

Thus, the state space of a timed automaton being infinite, some long (or infinite) runs never revisit the same state. For this reason, as stated in [7], usual **pumping lemmata** do not hold, and should be replaced by rather involved analogues. In a run, infinitely many events can happen during a finite amount of time, or two events can happen again and again with the time interval between them tending to 0. Such a run reminds of Zeno's aporias and is often called a **Zeno run**, see [12] and reference therein. Pathological runs do not support well discretization of clocks, see [14,6].

In order to rule out bad behaviors, restricted classes of timed automata and alternative semantics were considered by several authors. Thus, in [13,15], a **tube**

U. Fahrenberg and S. Tripakis (Eds.): FORMATS 2011, LNCS 6919, pp. 113–128, 2011.

language semantics is introduced. In [19] a **robust semantics**, based on small imprecisions is considered. It reappears in a different flavor as **implementability**, see [22,21], and in another version in [1].

With the same objective to rule out bad behaviors, restrictions are often put on all the cycles in the automaton, by requiring that each cycle takes at least one time unit (strongly non-Zeno condition), or resets all the clocks (progress cycle condition), or even resets all the clocks at one and the same transition (regeneration or synchronization condition).

In this paper, we propose a unifying approach to most of these issues for deterministic timed automata. We classify each automaton either as *thin* or as *thick* (the classification is decidable).

In thin automata. all the infinite trajectories, are, in some weak sense Zeno; the digitization of long trajectories is difficult, since it requires very small discretization step.

In thick automata. most of trajectories are non-Zeno, behave well under digitization and satisfy a sort of pumping lemma.

The main technical tool used to characterize thin and thick timed languages is their (volumetric) entropy introduced in [3,4]. Let us briefly recall this notion: a timed language L, for a given number of events n, can be seen as several polytopes in $\mathbb{R}^n$, their total volume is denoted by V_n. In most cases, for $n \to \infty$ this volume behaves exponentially: $V_n \approx 2^{n\mathcal{H}}$. The growth rate (i.e. $\mathcal{H}$) is referred to as *entropy*, and characterizes the size and the information contents of the timed language. With the notion of entropy, the definition of thin and thick languages is simple: a language is thin if its entropy equals $-\infty$ (that is the volume V_n decays faster than any exponent), and thick otherwise.

We identify a novel notion of a forgetful cycle, that is a cyclic path allowing forgetting the clock values. We state that any path in a timed automaton which is thick and long (in some precise sense), necessarily contains a forgetful cycle (Thm. 1), which can be seen as a weak version of pumping lemma. Based on this pumping lemma, we obtain our first main result (Thm. 2): thickness of a language is equivalent to many other nice properties briefly described above (good discretization, existence of forgetful cycle etc.).

The proof of Thms. 1-2 is rather technical, and uses together with "timed" techniques inspired by [19,1], the monoid version of Ramsey theory, namely Simon factorization forests [20].

The thin-thick alternative leads to a more precise analysis of a timed automaton when applied to its strongly connected components. In general, a timed automaton can be decomposed into several strongly connected components (some of them are thin, others thick) and acyclic pathways between them. We show that most of the long enough runs spend most of the time in thick components, and only few pathological runs wander in thin components (Thm. 3).

Finally, we apply the thin-thick alternative to the analysis of entropy of timed languages. In [4] it was shown that the entropy of a timed language can be lower and upper bounded using entropies of two discrete languages L_ε^- and L_ε^+, corresponding to a shrunk and a bloated discretizations of L:

$$h(L_\varepsilon^-) + \log(\varepsilon) \le \mathcal{H} \le h(L_\varepsilon^+) + \log(\varepsilon).$$

Here (Thm. 4) we strengthen this result and establish that on a thick component the entropy of a timed language can be approximated with a good precision using the entropy $h(L_\varepsilon)$ of the discretized language:

$$\mathcal{H} = h(L_\varepsilon) + \log(\varepsilon) + o(1).$$

As a corollary we obtain a converging algorithm allowing computation of $\mathcal{H}$ for any timed regular language with any precision required. This answers the open question from [4]: $\mathcal{H}$ is always a computable real number.

Paper organization. In Sect. 2, we recall some basic definitions, define thin and thick languages and give some motivating examples. In Sect. 3 we describe three more involved constructions: polytopes associated to paths in timed automata as in [18], region split automaton as in [3] and monoid of orbit graphs inspired by [19]. In the central Sect. 4 we state the thin-thick alternative for timed automata and a sort of pumping lemma for thick automata (Thm. 1). In Sect. 5 we apply these results to entropy of timed languages. We conclude with some perspectives in Sect. 6.

2 Preliminaries

2.1 Timed Languages and Their Measures

A *timed word* $\alpha = (t_1, a_1) \dots (t_n, a_n)$ is a word on the alphabet $\mathbb{R}^+ \times \Sigma$ where Σ is a finite alphabet of *events*. Times t_i represent *delays* between events a_{i-1} and a_i. Throughout this paper, delays will be bounded[1] by an integer constant M. We will sometimes write the same timed word α as $(\boldsymbol{t}, w)$ with $\boldsymbol{t} \in [0, M]^n$ and $w \in \Sigma^n$. A timed language L is a set of timed words. We will denote by L_n the language L restricted to words of length n.

Volume and volumetric entropy. Let L be a timed language. For each word of events $w \in \Sigma^n$, let $L(w)$ be the set $\{\boldsymbol{t} \in [0, M]^n \mid (\boldsymbol{t}, w) \in L\}$. This subset of $\mathbb{R}^n$ has a volume[2] (Lebesgue's measure) denoted by $\mathtt{Vol}(L(w))$. The *volume* of L_n is $\mathtt{Vol}(L_n) = \sum_{w \in \Sigma^n} \mathtt{Vol}(L(w))$. Bounding the delays by M leads to a volume bounded by $(|\Sigma|\, M)^n$.

In this paper, we will work with factor closed languages[3], i.e. such that if $(t_1, a_1) \dots (t_n, a_n) \in L$ then for all $1 \le i \le j \le n$ $(t_i, a_i) \dots (t_j, a_j) \in L$. For each couple of words $w_1, w_2 \in \Sigma^*$ the language inclusion $L(w_1 \cdot w_2) \subseteq L(w_1)L(w_2)$

[1] Our approach to timed languages is based on volumes, and does not apply, in its present form, to unbounded delays which lead to infinite volumes.

[2] Under the condition that the set is measurable; timed languages considered in this paper are all measurable as unions of polytopes.

[3] This roughly corresponds to automata where all states are both initial and final.

holds, and then $\mathtt{Vol}(L(w_1 \cdot w_2)) \leq \mathtt{Vol}(L(w_1))\mathtt{Vol}(L(w_2))$. The volumetric entropy of a timed language is a number in $[-\infty, +\infty)$ defined as

$$\mathcal{H}(L) = \lim_{n \to +\infty} \frac{1}{n} \log_2(\mathtt{Vol}(L_n)).$$

The limit (finite or $-\infty$) exists due to subadditivity of $\log_2 \mathtt{Vol}(L_n)$ wrt n and is upper bounded by $\log_2(|\Sigma|\, M)$. A timed language L is called *thin* if $\mathcal{H}(L) = -\infty$ ant *thick* otherwise.

To deal with finite objects, one can discretize all the previous continuous languages. Given an $\varepsilon = \frac{1}{N}$, ($N \in \mathbb{N}$) called *discretization step*, let us define $L_{\varepsilon,n} = L_n \cap (\varepsilon\mathbb{N} \times \Sigma)^n$, i.e. the set of words in L_n with all delays multiple of ε, $L_\varepsilon = \bigcup_{n \in \mathbb{N}} L_{\varepsilon,n}$, $L_\varepsilon(w) = L(w) \cap (\varepsilon\mathbb{N})^n$. The *$\varepsilon$-entropy* h_ε is defined as

$$h_\varepsilon(L) = \lim_{n \to \infty} \frac{1}{n} \log_2(|L_{\varepsilon,n}|).$$

We will relate the discrete entropy h_ε to the continuous one $\mathcal{H}$ in Sect. 5.

Open, closed and punctual languages. A language is said to be *open* if for all n, the set L_n is an open subset (for the product topology) of $\mathbb{R}^n \times \Sigma^n$. In other words, L_n should be a finite union of $O \times \{w\}$ where w is a word of events and O an open subset of $\mathbb{R}^n$. Interior and closure of L, denoted $\mathtt{int}(L)$ and $\overline{L}$ are defined in a natural way.

Taking closure or interior of a finite union of polytopes of $\mathbb{R}^n$ has no effect on its volume. Nevertheless the number of discrete points can drastically change if punctuality is allowed (see [5]). We call a language to be *punctual* if $\overline{\mathtt{int}(L)} \neq \overline{L}$.

2.2 Timed Automata and Their Languages

Clocks, zones and regions. Let X be a finite set of variables called *clocks*. Clocks have non negative values bounded by a constant M_c. A rectangular constraint is a formula of the form $x \sim c$ where $x \in X$, $c \in \mathbb{N}$, $\sim \in \{\leq, <, =, >, \geq\}$. A diagonal constraint is a formula of the form $x - y \sim c$ where $x, y \in X$. A guard is a finite conjunction of rectangular constraints. We denote by G the set of all guards. A *zone* is a set of clock vectors $\boldsymbol{x} \in [0, M_c]^X$ satisfying a finite conjunction of rectangular and diagonal constraints. A *region* is an inclusion-minimal zone. A region (which is always a simplex) is uniquely defined by a point with integer coordinates $\lfloor \boldsymbol{x} \rfloor \in \{0, \dots, M_c\}^X$ giving integer part of clocks and an order on the fractional part of clocks $0 \sim_0 \{x_{i_1}\} \sim_1 \{x_{i_2}\} \sim_2 \cdots \sim_{|X|} 1$ where $\sim_0, \dots, \sim_{|X|} \in \{<, =\}$. The closure of a region (abusively called closed region) can be obtained by replacing $<$ by $\leq$ and $>$ by $\geq$ in its definition.

Timed automata. A *timed automaton* is a tuple $(Q, X, \Sigma, E, \mathcal{L}, I, F)$ with Q a finite set of *locations*; X a set of bounded clocks; $E \subseteq Q \times G \times 2^X \times Q$ a finite set of edges; $\mathcal{L} : E \to \Sigma$ a labeling function on edges; $I \subseteq Q \times [0, M_c]^X$ the set of initial states; $F \subseteq Q \times [0, M_c]^X$ the set of final states.

By default all the *states* (elements of $Q \times [0, M_c]^X$) are initial and final, otherwise they are given by union of zones.

The clocks grow with the same (unit) speed and some of them are reset to 0 when passing through an edge. More formally, there is a *transition* $(q, \boldsymbol{x}) \xrightarrow{(t,a)} (q', \boldsymbol{x}')$ if there is an edge $e = (q, \mathfrak{g}, R, q') \in E$ with $\mathcal{L}(e) = a$ such that $\boldsymbol{x} + (t, \ldots, t)$ satisfies the guard $\mathfrak{g}$ and for each clock $x \in X$, its new value $x' = 0$ iff $x \in R$, or $x' = x+t$ otherwise. A *run* on the timed word $\alpha = (t_1, a_1) \ldots (t_n, a_n)$ is a sequence of consecutive transitions $(q_0, \boldsymbol{x}_0) \xrightarrow{(t_1,a_1)} (q_1, \boldsymbol{x}_1) \cdots \xrightarrow{(t_n,a_n)} (q_n, \boldsymbol{x}_n)$, where $\boldsymbol{x}_0, \ldots, \boldsymbol{x}_n \in [0, M]^X$, $q_0, \ldots, q_n \in Q$. A timed word is recognized by the automaton if there exists a run on it from an initial state to a final state. The timed language $L(\mathcal{A})$ consists of all the recognized words. We will be interested in its entropy $\mathcal{H}(L(\mathcal{A}))$, that will be abusively denoted $\mathcal{H}(\mathcal{A})$.

We call a TA *right resolving* if any two edges leaving the same location and having the same label have disjoint guards. Adding the condition that there is only one initial state gives the usual definition of determinism. In the rest of the paper, we work with right resolving TA.

Paths and reachability relation. We call a *path* in an automaton any sequence of edges. The "useful" paths are sequences of consecutive edges (such that the starting location of the $(i+1)^{th}$ edge is the ending one of the i^{th}), but we allow arbitrary words of E^* and all objects associated to a non-consecutive sequence will be empty.

Given two clock vectors $\boldsymbol{x}, \boldsymbol{x}'$, a path $\pi \in E^n$ and a sequence of delays of the same length $\boldsymbol{t} = (t_1, \ldots, t_n)$, we write that $\boldsymbol{x} \xrightarrow{\boldsymbol{t},\pi} \boldsymbol{x}'$ whenever exists a run in the automaton of the form $(q_0, \boldsymbol{x}) \xrightarrow{(t_1,a_1)} (q_1, \boldsymbol{x}_1) \cdots \xrightarrow{(t_n,a_n)} (q_n, \boldsymbol{x}')$ following the sequence of edges π.

Several objects are naturally associated with a path. Given a path and two clock vectors, a language (a polytope of all the timings of the path) can be defined: $L(\pi, \boldsymbol{x}, \boldsymbol{x}') = \{\boldsymbol{t} \mid \boldsymbol{x} \xrightarrow{\boldsymbol{t},\pi} \boldsymbol{x}'\}$. If we are not interested in clock values, we get a polytope depending only on the path: $L(\pi) = \{\boldsymbol{t} \mid \exists \boldsymbol{x}, \boldsymbol{x}', \boldsymbol{x} \xrightarrow{\boldsymbol{t},\pi} \boldsymbol{x}'\}$. The other way around, if we do not care about timing, we get the reachability predicate: $\texttt{Reach}(\pi) = \{(\boldsymbol{x}, \boldsymbol{x}') \mid \exists \boldsymbol{t}, \boldsymbol{x} \xrightarrow{\boldsymbol{t},\pi} \boldsymbol{x}'\}$.

A path $\pi \in E^n$ is said to be *punctual* if $L(\pi)$ is not empty but has dimension less than n. The closure (resp. interior) of a path π denoted by $\overline{\pi}$ (resp. $\texttt{int}(\pi)$) is the path obtained from π by taking closure (resp. interior) of all guards of edges in π (i.e. changing strict inequalities in non-strict ones (resp. vice versa)). There is a *limit cycle* (resp. *strong limit cycle*) along π if there exists a clock vector $\boldsymbol{x}$ and time sequence $\boldsymbol{t}$ such that $\boldsymbol{x} \xrightarrow{\boldsymbol{t},\overline{\pi}} \boldsymbol{x}$ (resp. $\boldsymbol{x} \xrightarrow{\boldsymbol{t},\texttt{int}(\pi)} \boldsymbol{x}$). Given $\varepsilon > 0$, in ε-*discrete limit cycles* all the components of $\boldsymbol{x}$ and $\boldsymbol{t}$ should be multiple of ε.

2.3 Thinness, Simplices and Examples

Our analysis of thin languages will start with a simple observation that the volume of k-dimensional simplices tends to 0 faster than any exponent:

Lemma 1. *The volume of a simplex of "type 1" described by inequalities* $0 \leq t_1 + \cdots + t_k \leq 1$, $t_i \geq 0$ *or of a simplex of "type 2" described by inequalities* $0 \leq t_1 \leq \cdots \leq t_k \leq 1$ *is* $\frac{1}{k!}$.

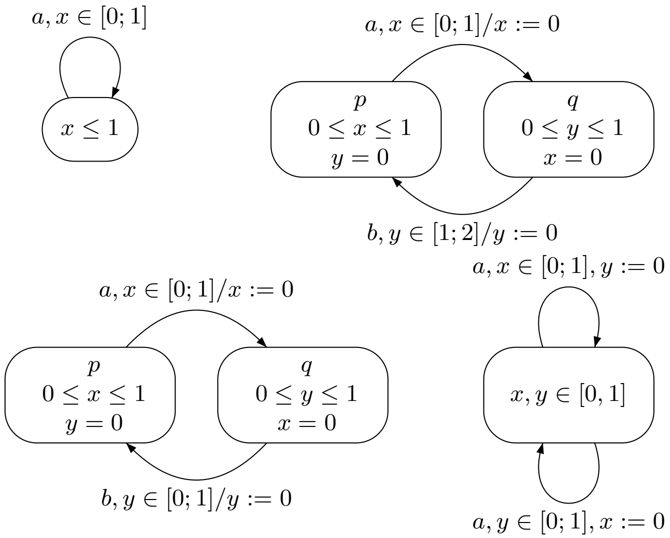

Fig. 1. First row: thin automata $\mathcal{A}_1$, $\mathcal{A}_2$. Second row: thick ones $\mathcal{A}_3$, $\mathcal{A}_4$. Initial states are given by conditions in nodes.

By change of coordinates the lemma can be extended to more general polytopes:

Corollary 1. *Let P be a subset of $\{t_1, \dots, t_n \mid 0 \le t_i \le M\}$. If there exists a subsequence of indices $s(1), \dots, s(k)$ of $1, \dots, n$ and new coordinates $u_{s(i)}$ functions of $t_{s(1)}, \dots, t_{s(i)}$ with $0 \le u_{s(1)} \le u_{s(2)} \le \dots \le u_{s(k)} \le 1$ and $\left|\frac{\partial u_{s(i)}}{\partial t_{s(i)}}\right| \ge 1$ then $\mathtt{Vol}(P) \le \frac{M^{n-k}}{k!}$.*

The automata on Fig. 1 illustrate the concepts of thin and thick. $L_n(\mathcal{A}_1) = \{t_1, \dots, t_n \mid \sum_{i \le n} t_i \le 1\}$ is a simplex of type 1, and thus $L(\mathcal{A}_1)$ is thin. $L_n(\mathcal{A}_2, q) = \{t_1, \dots, t_n \mid \forall i, t_{2i} + t_{2i+1} \le 1 \wedge t_{2i+1} + t_{2i+2} \ge 1\}$, by interchanging even and odd indices we obtain $L_n(\mathcal{A}_2, p)$. Posing $u_{2i+1} = 1 - t_{2i+1}$ and $u_{2i} = t_{2i}$ yields a simplex $0 \le u_1 \le \dots u_n \le 1$. This change of coordinates preserves volume and so $\mathtt{Vol}(L_n(\mathcal{A}_2, q)) = \frac{1}{n!}$. This is an example of automaton satisfying the *progress cycle condition* (i.e. resetting all clocks along each cycle) and nevertheless thin.

Third and fourth examples are thick, their entropies can be computed symbolically because they are $1\frac{1}{2}$ clock (see [3]), they are respectively $\log_2 \frac{2}{\pi}$ and $\log_2 \log_2(e)$. Note that $\mathcal{A}_4$ does not satisfy the progress cycle condition.

3 More on Paths and Cycles

3.1 Region Graph and State Split Automata

Timed variants of the region graph [2] are extensively used in the literature. Here we use so-called region-split automaton given in [3], add several new conditions

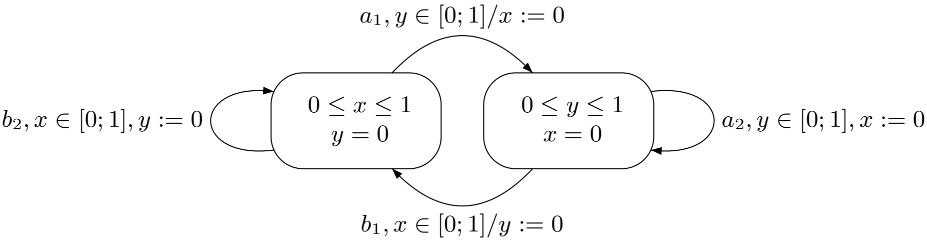
$a_1, y \in [0;1]/x := 0$
$b_2, x \in [0;1], y := 0$
$0 \le x \le 1$
$y = 0$
$0 \le y \le 1$
$x = 0$
$a_2, y \in [0;1], x := 0$
$b_1, x \in [0;1]/y := 0$

Fig. 2. The closed region-split version of $\mathcal{A}_4$

and modify those concerning initial states. A timed automaton is in region-split form if

B1. For every location $q \in Q$ a unique region $\mathbf{r}_q$ (called its *entry region*) exists, such that the set of clock values with which q is entered is exactly $\mathbf{r}_q$.
B2. The guard $\mathfrak{g}$ of every transition $\delta = (q, \mathfrak{g}, R, q') \in E$ is just one region.
B3. All the states of entry regions (and only these states) are both initial and final.
B4. For any location there exists a path leading to some cycle and a path coming from some cycle.
B5. For every transition δ its guard $\mathfrak{g}$ has no constraints of the form $x = c$ in its definition.
B6. The labeling function on edges is identity (and so every two distinct edges have different labels).

Proposition 1. *Given a right resolving TA $\mathcal{A}$ with bounded clocks and all states initial and final, one can construct an automaton $\mathtt{RS}(\mathcal{A})$ called the region-split automaton of $\mathcal{A}$ which satisfies $B1 - B6$ and such that $\mathcal{H}(\mathtt{RS}(\mathcal{A})) = \mathcal{H}(\mathcal{A})$.*

In the following, we replace w.l.o.g. for the computing of $\mathcal{H}$, $\mathcal{A}$ by $\overline{\mathtt{RS}(\mathcal{A})}$ obtained from $\mathtt{RS}(\mathcal{A})$ by taking non-strict inequalities instead of strict ones.

Proposition 2. *For region split automata:*

- *words, paths, and region paths are in natural bijection;*
- *volume of any path is less or equal to* 1*;*
- *every path of consecutive edges has a non empty and non punctual language.*

As for our running examples, $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3$ are already region split automata. $\overline{\mathtt{RS}(\mathcal{A}_4)}$ is depicted in Fig. 2.

3.2 Paths and Polytopes

Let us describe languages associated with paths as polytopes in $\mathbb{R}^n$, following [18].

Contiguous polytopes. Let $T_n = \{t_1, \ldots, t_n\}$ be an ordered set of real variables bounded by M. A sum $S_{j..k} = \sum_{i=j}^{k} t_i$ is called a *contiguous sum* (of *length* $k - j + 1$). A *temporal inequality* is a constraint of the form $S_{j..k} \in [A, B]$ where $A, B \in \mathbb{N}$, $A \ge 0$ and $B \le M$. A *contiguous polytope* is a bounded subset of

$\mathbb{R}^n$ which is composed by all the points satisfying a conjunction of temporal inequalities. We say that the polytope is *d-contiguous* if the length of all sums in the inequalities is bounded by d.

Proposition 3. *For each path $\pi \in E^*$, $L(\overline{\pi})$ is a contiguous polytope.*

The inequality $\sum_{i=j}^{k} t_i \in [A, B]$ comes from testing the guard $x \in [A, B]$ during the k^{th} transition, provided that the last reset of x took place in the j^{th} transition. The subclass of d-contiguous polytopes corresponds to automata with progress cycle condition, where the number of transitions between two resets is bounded by d.

Given a polytope $\mathcal{P}$, we denote by $N\mathcal{P}$ its N-fold dilated copy, i.e. $\{N\boldsymbol{t} \mid \boldsymbol{t} \in \mathcal{P}\}$ and by $E(\mathcal{P}) = \mathcal{P} \cap \mathbb{Z}^n$ the set of points with integer coordinates in $\mathcal{P}$. A contiguous polytope is said to be *N-fat* if there exists an integer point in the interior of $N\mathcal{P}$ (called an *internal point*): $E(\mathtt{int}(N\mathcal{P})) \neq \emptyset$.

3.3 Point to Point Reachability: Algebraic Characterization

In this section, we characterize the relation $\mathtt{Reach}(\overline{\pi})$ in terms of an algebraic object: monoid of orbit graphs. Our analysis is less detailed than those in [10,11,16] and follows the lines of [19].

For a closed region $\overline{\mathbf{r}}$, let us denote by $V(\mathbf{r}) = \{S_1, \ldots, S_p\}$ its vertices. Any point $\boldsymbol{x}$ in the region is uniquely described by its barycentric coordinates $\lambda_1, \ldots, \lambda_p$, i.e. nonnegative numbers such that $\sum_{i=1}^{p} \lambda_i = 1; \quad \boldsymbol{x} = \sum_{i=1}^{p} \lambda_i S_i$.

Given two regions $\overline{\mathbf{r}}$ and $\overline{\mathbf{r}'}$, we call *orbit graph* any graph G with vertices $V(\mathbf{r}) \biguplus V(\mathbf{r}')$ if $\mathbf{r}$ and $\mathbf{r}'$ are different and $V(\mathbf{r})$ otherwise, and with edges going from $V(\mathbf{r})$ to $V(\mathbf{r}')$. Informally, an edge from S to S' means that the clock vector at the vertex S can reach the clock vector at S' along some transition or path.

Orbit graphs compose in the natural way: for G_1 on regions $\overline{\mathbf{r_1}}$ and $\overline{\mathbf{r}'_1}$, and G_2 on regions $\overline{\mathbf{r_2}}$ and $\overline{\mathbf{r}'_2}$, their product $G = G_1 \cdot G_2$ is defined if $\overline{\mathbf{r}'_1} = \overline{\mathbf{r_2}}$. In this case, G is an orbit graph on $\overline{\mathbf{r_1}}$ and $\overline{\mathbf{r}'_2}$. There is an edge from S to S'' in G if and only if there exists S' such that (S, S') and (S', S'') are edges of G_1 and G_2. Whenever $\overline{\mathbf{r}'_1} \neq \overline{\mathbf{r_2}}$, we put $G_1 \cdot G_2$ equal to some special (absorbing) element $\mathbf{0}$. The set $\mathcal{G}$ of orbit graphs, augmented with $\mathbf{0}$ and a neutral element $\mathbf{1}$ has a structure of finite monoid.

An orbit graph G can be represented by its adjacency matrix M of size $|V(\mathbf{r})| \times |V(\mathbf{r}')|$. Products in the monoid of orbit graphs are easy to compute using matrices: $M(G_1 G_2) = M(G_1) \otimes M(G_2)$ where the “product” $\otimes$ is defined by

$$(A \otimes B)_{ij} = \max_k \min(A_{ik}, B_{kj}).$$

There exists a natural morphism $\gamma : E^* \to \mathcal{G}$ from paths to orbit graphs defined as follows. For a transition e between $\overline{\mathbf{r}}$ and $\overline{\mathbf{r}'}$, we define the orbit graph $\gamma(e)$ on $\mathbf{r}$ and $\mathbf{r}'$ with edges $\{(S, S') \in V(\mathbf{r}) \times V(\mathbf{r}') \mid \exists t, S \xrightarrow{(e,t)} S'\}$. For a path $\pi = e_1 \ldots e_n$, we define $\gamma(\pi) = \gamma(e_1) \ldots \gamma(e_n)$ (it will be called the orbit graph of the path π). For the empty path we have $\gamma(\varepsilon) = \mathbf{1}$, and for any non-consecutive path $\gamma(\pi) = \mathbf{0}$.

For example, the orbit graphs of cycles ab and ba of $\mathcal{A}_3$ and $\mathcal{A}_4$ are complete, the orbit graphs of the other running examples are given in Fig. 3.

The orbit graph is crucial for reachability analysis.

Proposition 4. *The orbit graph of a path $\gamma(\pi)$ determines its reachability relation $\mathtt{Reach}(\bar{\pi})$. In particular, $\gamma(\pi)$ is complete iff $\mathtt{Reach}(\bar{\pi}) = \overline{\mathbf{r}} \times \overline{\mathbf{r}'}$, or equivalently iff $\mathtt{Reach}(\mathtt{int}(\pi)) = \mathbf{r} \times \mathbf{r}'$.*

The proof of the first criterion is based on the following remarkable characterization of $\mathtt{Reach}(\bar{\pi})$ in terms of the orbit graph due to Puri [19].

Lemma 2. [4] *Let $\boldsymbol{x}$ and $\boldsymbol{x}'$ be two clock vectors with barycentric coordinates $\boldsymbol{\lambda}$ and $\boldsymbol{\lambda}'$. Then $(\boldsymbol{x}, \boldsymbol{x}') \in \mathtt{Reach}(\bar{\pi})$ iff there exists a stochastic matrix $P \preceq M(\gamma(\pi))$, such that $\boldsymbol{\lambda} P = \boldsymbol{\lambda}'$.*

Here matrix "inequality" $A \preceq B$ means that $B_{ij} = 0 \Rightarrow A_{ij} = 0$ for all i, j.

Adding clock resets. For future use, we must enrich the monoid of orbit graphs by adding information on clock resets. Elements of the monoid $\mathcal{M}$ are couples (orbit graph, subset of clocks) (and also, as before, two special elements $\mathbf{0}, \mathbf{1}$), the product rule is:

$$(G_1, X) \cdot (G_2, Y) = \begin{cases} (G_1 \cdot G_2, X \cap Y), & \text{if } G_1 \cdot G_2 \neq \mathbf{0} \\ \mathbf{0}, & \text{otherwise.} \end{cases}$$

For each $\pi \in E^*$ we denote by $\nu(\pi)$ the set of clocks not reset along the path π. We define a morphism $\mu : E^* \to \mathcal{M}$ as follows: $\mu(\pi) = (\gamma(\pi), \nu(\pi))$.

Idempotents. An idempotent of a monoid is an element m such that $m \cdot m = m$. Every finite monoid contains an idempotent. In our case, an idempotent orbit graph is always associated to a cyclic path, it is a graph G equal to its transitive closure $G^+ = \cup_{n \in \mathbb{N}^+} G^n$.

4 The Thin-Thick Alternative and Its Consequences

In this central section, we characterize thin and thick paths and languages, based on a new notion of a forgetful cycle.

4.1 Forgetful Cycles, and the Others

After reading a timed path $\pi \times \boldsymbol{t}$ from a state s_0, the reached state s depends only on s_0 and on the delays $\boldsymbol{t}$. We will say that π is *forgetful* if s and s_0 are independent, i.e. all the following equivalent conditions hold: $\mathtt{Reach}(\mathtt{int}(\pi)) = \mathbf{r} \times \mathbf{r}'$, $\mathtt{Reach}(\bar{\pi}) = \bar{\mathbf{r}} \times \bar{\mathbf{r}}'$, $\gamma(\pi)$ is complete.

If a cycle is non-forgetful, and moreover its orbit graph is not strongly connected, then it is possible to find a linear Lyapunov function:

[4] An intuition behind this lemma could be as follows. A clock vector with barycentric coordinates $\boldsymbol{\lambda}$ in a region can be seen as a probabilistic distribution over vertices of this region (with probabilities $\boldsymbol{\lambda}$). The lemma says that this distribution, at each cycle, evolves exactly as in some Markov chain.

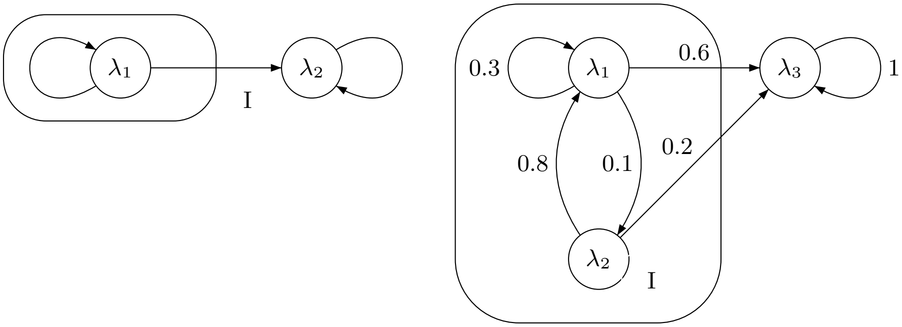

Fig. 3. Two non strongly connected orbit graphs, the first one is the orbit graph of the cycle of $\mathcal{A}_1$, of the cycle ab of $\mathcal{A}_2$ and of the cycles a and b of $\mathcal{A}_4$. States move from initial SCC I to final one. By choosing the convex combination of paths given by the Markov chain on the second orbit graph we pass from state $(\lambda_1 = 0.2, \lambda_2 = 0.5, \lambda_3 = 0.3)$ to state $(\lambda'_1 = 0.46, \lambda'_2 = 0.02, \lambda'_3 = 0.52)$. The sum $\lambda_1 + \lambda_2$ can only decrease.

Lemma 3. *For a cycle π, if $\gamma(\pi)$ is not strongly connected then there exists a non empty $I \subsetneq \{1, \dots, p\}$ such that $\sum_{i \in I} \lambda'_i \leq \sum_{i \in I} \lambda_i$ whenever $(\boldsymbol{x}, \boldsymbol{x}') \in$ $\texttt{Reach}(\bar{\pi})$, where $\boldsymbol{\lambda}$ and $\boldsymbol{\lambda}'$ stand for barycentric coordinates of $\boldsymbol{x}$ and $\boldsymbol{x}'$.*

In this lemma, as before, $\{1, \dots, p\}$ are indices of the vertices of the region where π starts (and ends).

In fact I corresponds to an initial strongly connected component (SCC) of the orbit graph, i.e. an SCC without incoming edges from other SCC. According to the lemma, the state moves from the facet spanned by I towards other vertices of the region and cannot come back.

Comparing to other types of cycles. Two other kinds of cycles are often considered in the literature: in a *progress cycle* each clock should be reset at some edge; in a *synchronizing cycle* all the clocks are reset along one and the same edge of the cycle.

Proposition 5. *progress cycles $\supsetneq$ forgetful cycles $\supsetneq$ synchronizing cycles.*

A remark is in order, in most works using progress or synchronizing cycles, **all the cycles** are **required** to be like that. In our work, **existence** of a forgetful cycle **appears** naturally in "non degenerate" (i.e. thick) automaton.

The condition of progress cycle can be seen as a weaker kind of forgetting: the state after such a cycle is exactly determined by the delays (see following lemma). Nevertheless the orbit graph of a progress cycle can be not strongly connected (e.g. cycle ab of $\mathcal{A}_2$ depicted in Fig. 3); in that case starting states and ending states are still dependent.

Lemma 4. *If all clocks have been reset during reading of $\pi \times (t_1, \dots, t_m)$ then for all non empty $I \subsetneq \{1, \dots, p\}$, there exists $\alpha_1 \dots \alpha_m \in \{-p, \dots, p\}$ and an integer constant C such that $\sum_{i \in I} \lambda_i = C + \sum_{j=1}^{m} \alpha_j t_j$. Moreover one of the α_j is not zero.*

4.2 Pumping Lemma for Long Thick Paths

For a given real $\eta > 0$, we say that a path π is *η-thick* if $\mathtt{Vol}(L(\pi)) \geq \eta^{|\pi|}$. The following "pumping lemma" will play the key role in characterization of thick languages below and can be interesting by itself.

Theorem 1. *For every timed automaton $\mathcal{A}$ and every $\eta > 0$, there exists N_η such that any η-thick path longer than N_η contains a forgetful cycle.*

The rest of this section is devoted to the proof of this result.

Elements of the monoid $\mathcal{M}$ associated to forgetful cycles will be referred to as *forgetful*, they are idempotent. We will first see how repeating a non forgetful idempotent induces a subexponential volume (like the simplex example), then we will use Simon's theorem on factorization forests to factorize paths and find some repeated idempotent. Absence of forgetful cycles in a path will then imply thinness.

Proposition 6. *Let $\pi_1, \dots, \pi_k$ be k cycles of E^* such that $\mu(\pi_1), \dots, \mu(\pi_k)$ are all equal to a same non forgetful idempotent of $\mathcal{M}$, then* $\mathtt{Vol}(L(\pi_1 \dots \pi_k)) \leq \frac{M^{n-k}}{k!}$ *where $n = |\pi_1| + \dots + |\pi_k|$.*

Proof. If G is an idempotent orbit graph (thus equal to its transitive closure), G is complete if and only if G is strongly connected. Thus we will distinguish two disjoint kinds of non forgetful idempotents, those associated to non progress cycles and those associated to progress cycles with non strongly connected orbit graphs. In the former case a clock is not reset all along the path $\pi_1 \dots \pi_k$, thus $L(\pi_1 \dots \pi_k)$ is in a simplex of type 1 and the volume satisfies the inequality to prove. In the latter case, we use Lem. 3,4, and Cor. 1. □

A *factorization forest* of a word π is an unranked labeled tree with leaves labeled by the letters of π, with root labeled by π and with two types of internal nodes:

- binary node labeled by a word $\pi_1 \cdot \pi_2$ with two children labeled by the words π_1 and π_2;
- idempotent node labeled by a word $\pi_1 \dots \pi_k$ with all $\mu(\pi_i)$ equal to a same idempotent and with children labeled by the words $\pi_1, \dots, \pi_k$.

Theorem (Simon [20]). *If μ is a morphism from E^* to a finite monoid $\mathcal{M}$, then every word admits a factorization forest of height at most $h(\mathcal{M}) = 9|\mathcal{M}|$.*

We suppose that there are no forgetful cycles on a long path π and consider its factorization forest of height at most $h(\mathcal{M})$. When its length n grows up, the number of leaves also grows and since the height is bounded, branching of nodes must get larger and larger. These hugely branched nodes are idempotent and satisfy hypotheses of Lem. 6, thus their volume is very small, which implies that $\mathtt{Vol}(L(\pi))$ is also small. The Prop. 7 below quantifies this "smallness" of $\mathtt{Vol}(L(\pi))$ as function of the length of π and height of its factorization forest, and Thm. 1 follows immediately from this proposition.

Let $\texttt{LVol}$ be the function defined on paths by $\texttt{LVol}(\pi) = \log_2 \texttt{Vol}(L(\pi))$. This function is subadditive and non-positive, i.e. $\texttt{LVol}(\pi_1 \cdot \pi_2) \leq \texttt{LVol}(\pi_1) + \texttt{LVol}(\pi_2) \leq 0$. Let $L(n, h)$ be the maximum of $\texttt{LVol}(\pi)$ over paths π of length n that do not contain forgetful idempotents and admit a factorization forest of height at most h.

Proposition 7. *For any height h, for any $C < 0$, there exists $N_{h,C} \in \mathbb{N}$ such that for all $n > N_{h,C}$ the inequality $L(n, h) \leq Cn$ holds.*

Proof. We will define $N_{h,C}$ by induction on the height h. Let a be a factorization forest of height h with n leaves. We consider all the children of the root and their subtrees (all these subtrees have heights $\leq h - 1$), and distinguish two disjoint cases:

1. There are more than $k = \frac{n}{2N_{h-1,2C}}$ subtrees having less than $N_{h-1,2C}$ leaves.
2. There are less than $k = \frac{n}{2N_{h-1,2C}}$ subtrees with less than $N_{h-1,2C}$ leaves. Here the juicy part (sons with enough leaves to satisfy induction hypothesis) has more than $\frac{n}{2}$ leaves.

In the first case: root is an idempotent node and we can apply Lem. 6:

$$\texttt{LVol}(\pi) \leq (n - k) \log_2(M) - \log_2(k!) \leq nC \text{ for } n \text{ large enough.}$$

In the second case: $\texttt{LVol}(\pi) \leq \sum_{i=1}^{k} L(n_i, h_i) \leq \sum_{n_i \geq N_{h-1,2C}} L(n_i, h_i)$. We apply the induction hypothesis:

$$\texttt{LVol}(\pi) \leq 2C \sum_{n_i \geq N_{h-1,2C}} n_i \leq 2C\frac{n}{2} \leq nC \text{ (recall that } C \text{ is negative).} \qquad \square$$

To conclude the proof of Thm. 1, given $\eta > 0$, let $C = \log_2 \eta$ and $h = h(\mathcal{M})$ the bound on height of factorization forest. Using Prop. 7, we obtain that a path longer than $N_{h,C}$ without forgetful idempotents cannot be η-thick. □

4.3 Characterizing Thick Languages

We are ready to describe thick languages now.

Theorem 2. *For a right resolving timed automaton in region split form the following conditions are equivalent and define thick languages:*

1. *$\mathcal{H} > -\infty$;*
2. *there exists a forgetful cycle;*
3. *there exists a strong limit cycle;*
4. *there exists an ε-discrete strong limit cycle with $\varepsilon > 0$.*

Equivalence between 3 and 4 can be found in [16]. 2 ⇒ 3 is straightforward.

Proof of 4 ⇒ 1. There exist $q_0, \ldots, q_{d-1}, \boldsymbol{x}_0, \ldots, \boldsymbol{x}_{d-1}$, $\pi \in E^d, u_1, \ldots, u_d \in \{\varepsilon, 2\varepsilon, \ldots, M-\varepsilon\}$ such that $(q_0, \boldsymbol{x}_0) \xrightarrow{(u_1,w_1)} (q_1, \boldsymbol{x}_1) \ldots \xrightarrow{(u_d,w_d)} (q_0, \boldsymbol{x}_0)$ along π and all the $\boldsymbol{x}_i$ are not on the frontier of regions and have discrete coordinates. First we can see that all clocks have been reset at least once because any non-reset clock would augment during the run, which contradicts its cyclicity. Then for each $n \in \mathbb{N}^*$ the language $L(\pi^n)$ is a d-contiguous polytope with equation of the form $A \le \sum_{i=j}^k t_i \le B$. Extending u periodically permits to have a word in $L(\pi^n)$ such that $A + \varepsilon \le \sum_{i=j}^k u_i \le B - \varepsilon$. Taking $t_i \in [u_i - \frac{\varepsilon}{d}, u_i + \frac{\varepsilon}{d}]$ defines a hypercube included in $L(\pi^n)$ whose volume is therefore greater than $(\frac{2\varepsilon}{d})^{nd}$. Then $\mathcal{H}(\mathcal{A}) \ge \log_2 \frac{2\varepsilon}{d} > -\infty$. □

Proof of 1 ⇒ 2. We notice first that a thick language contains long thick paths.

Lemma 5. *If $\mathcal{H} > -\infty$, there exists $\eta > 0$ such that for all n big enough, there exists an η-thick path of length n.*

Proof. Let $\beta = 2^{\mathcal{H}-1}$. For n large enough $\texttt{Vol}(L_n) \ge \beta^n$. Let $\pi_{n,\max}$ be the path of E^n of maximal volume, then $\texttt{Vol}(L_n) \le \texttt{Vol}(L(\pi_{n,\max}))|E|^n$ and so if we pose $\eta = \frac{\beta}{|E|}$ we have $\texttt{Vol}(L(\pi_{n,\max})) \ge \eta^n$. □

Combining Lem. 5 with Thm. 1 we find a required forgetful cycle. □

4.4 Thin and Thick SCC

The theory developed above can be refined using a decomposition of $\mathcal{A}$ into strongly connected components (SCC) $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_k$.

Proposition 8. *Volumetric entropy of $\mathcal{A}$ equals the maximal volumetric entropy of its SCC. In particular, $\mathcal{A}$ is thin iff so are all the subautomata $\mathcal{A}_i$.*

It is easy to see that long and thick paths spend most of the time in thick SCC.

Theorem 3. *For every timed automaton $\mathcal{A}$ and every $\eta, \alpha > 0$, there exists $N_{\eta,\alpha}$ such that for any η-thick path of length $n > N_{\eta,\alpha}$ at most $n\alpha$ states belong to thin SCC.*

5 Entropies of Thick Languages

In this section, we apply the results of the previous section to show that in thick automata, volumes and entropies can be computed with a good precision using discretization.

Theorem 4. *For a thick strongly connected automaton $\mathcal{A}$ in region split form, the discrete and the volumetric entropies are related as follows*[5]*:*

$$h_\varepsilon = \log_2 \frac{1}{\varepsilon} + \mathcal{H} + o(1).$$

[5] It can be proved that $o(1)$ is in fact $O\left(\varepsilon^{1/3}\left(\log_2 \frac{1}{\varepsilon}\right)^{2/3}\right)$.

Proof ($\geq$ direction). To bound the volume of L_n by the number of discrete points, we will use a beautiful theorem on counting points in polytopes:

Theorem (Ehrhart, see [8]). *For integer N and an integer polytope $\mathcal{P} \subset \mathbb{R}^n$ (i.e. whose vertices have integer coordinates), the number of integer points $|E(N\mathcal{P})|$ is a polynomial in N with non negative coefficients of degree n and whose coefficient of the highest degree is the volume.*

We deduce directly from this theorem that for each path π of length n and $\varepsilon = \frac{1}{N}$ the following holds: $\texttt{Vol}(L(\pi))N^n \leq |E(NL(\pi))| = |L_\varepsilon(\pi)|$. Summing over all words of length n and taking $\lim_{n\to\infty} \frac{1}{n}\log_2$, we obtain that $\mathcal{H}+\log_2\frac{1}{\varepsilon} \leq h_\varepsilon$. □

Proof ($\leq$ direction of Thm. 4).

Upper bounding h_ε by $\mathcal{H} + \log_2\frac{1}{\varepsilon} + o(1)$ is more involved, and we give only a sketch of proof. We fix several integer parameters: a, b, c, d, e (they have to be adjusted in order to obtain the required estimate). Let π be a path of a length $n > a$. At every b transitions, we insert in π a forgetful cycle of length c (it exists by virtue of Thm. 2, and can be made of the same fixed length everywhere for an appropriate choice of c). Thus we obtain a slightly longer path π' (its length is $n' \approx n(1 + c/b)$), satisfying two additional conditions:

- every clock is reset at least every $2d$ transitions (and thus $L(\pi')$ is $2d$-contiguous polytope);
- the polytope $L(\pi')$ is e-fat.

We have three inequalities:

1. The first one:
$$|L_\varepsilon(\pi)| \leq |L_\varepsilon(\pi')|$$
can be proved by constructing an injection from the left-hand side discrete language to the right-hand side one.
2. We choose ε' slightly smaller than ε (another parameter to adjust) and consider the polytope L^- obtained from $L(\pi')$ by pushing all its facets inside by the amount $\delta = \varepsilon' d$ [6]. Using fatness of $L(\pi')$, it is possible to build an injection from its ε-discrete points to ε'-discrete points of L^- (the latter is a bit smaller but its discrete points are slightly denser).
$$|L_\varepsilon(\pi')| \leq \left|L^-_{\varepsilon'}\right|.$$
3. Taking an ε'-cube at every ε'-discrete point of L^-, we get a set included in $L(\pi')$ (this requires $2d$-contiguity of $L(\pi')$). Passing to volumes we conclude that
$$\varepsilon'^{n'}\left|L^-_{\varepsilon'}\right| \leq \texttt{Vol}(L(\pi')).$$

[6] i.e. by replacing each constraint $S_{j..k} \in [A, B]$ in the definition of $L(\pi')$ as a contiguous polytope by $S_{j..k} \in [A + \delta, B - \delta]$ (see [4]).

Combining the three inequalities we get:

$$|L_\varepsilon(\pi)| \le \left(\varepsilon'^{-n'}\mathtt{Vol}(L(\pi'))\right),$$

and with an appropriate choice of parameters, ε' and n' can be made very close to ε and n. Summing up over π and taking $\lim_{n\to\infty}\frac{1}{n}\log_2$ in the previous inequality, we obtain the required result. □

Corollary 2. *For right resolving TA with bounded clocks, $\mathcal{H}(\mathcal{A})$ is computable as function of $\mathcal{A}$. Consequently, $\mathcal{H}(\mathcal{A})$ is a computable real (i.e. one can compute its approximation with any wanted precision).*

Proof. First compute $\mathtt{RS}(\mathcal{A})$. Then compute by fixpoint method the submonoid of orbit graphs $\gamma(E^*) \subset \mathcal{G}$ and see whether there is a complete graph. If there is none, the automaton is thin and $\mathcal{H} = -\infty$. Otherwise, the automaton is thick and it just remains to compute the discrete entropy of $L_\varepsilon(\mathtt{RS}(\mathcal{A}))$ for the wanted precision (similarly to [4]). □

6 Conclusion and Future Work

We have identified the class of thick timed automata (those with non-vanishing language volume). Most runs in such automata are thick and exhibit a nice behavior, they spend most of the time in thick strongly connected components (Thm. 3) and visit from time to time forgetful cycles (Thm. 1). Thick runs are captured (both qualitatively and quantitatively) by ε-discretized automata.

We believe that the notions of thick languages and forgetful cycles will be useful in the operator approach to volume and entropy of [3] and will imply some good properties of operators associated to these forgetful cycle. Similarly, we believe that thickness hypothesis is exactly what is needed for the analysis of probabilistic timed systems in the spirit of [9] but for an unbounded time horizon. Another direction of future work is to extend the thin-thick dichotomy to the case of punctual paths and to find when the two size measures of [5] are defined. We hope also to relate thinness with the notion of mean topological dimension [17]. In the verification context, we believe that when analyzing a thick timed automaton, it suffices to check that the thick paths satisfy the specification, while thin ones can violate it.

Acknowledgements. The authors are thankful to Thomas Colcombet for a key advise: to use Simon factorization forests, and to Dominique Perrin and Aldric Degorre for inspiring discussions.

References

1. Abdulla, P.A., Krcál, P., Yi, W.: Sampled semantics of timed automata. Logical Methods in Computer Science 6(3) (2010)
2. Alur, R., Dill, D.L.: A theory of timed automata. Theor. Comput. Sci. 126, 183–235 (1994)

3. Asarin, E., Degorre, A.: Volume and entropy of regular timed languages: Analytic approach. In: Ouaknine, J., Vaandrager, F.W. (eds.) FORMATS 2009. LNCS, vol. 5813, pp. 13–27. Springer, Heidelberg (2009)
4. Asarin, E., Degorre, A.: Volume and entropy of regular timed languages: Discretization approach. In: Bravetti, M., Zavattaro, G. (eds.) CONCUR 2009. LNCS, vol. 5710, pp. 69–83. Springer, Heidelberg (2009)
5. Asarin, E., Degorre, A.: Two size measures for timed languages. In: FSTTCS, Schloss Dagstuhl - Leibniz-Zentrum für Informatik. LIPIcs, vol. 8, pp. 376–387 (2010)
6. Asarin, E., Maler, O., Pnueli, A.: On discretization of delays in timed automata and digital circuits. In: Sangiorgi, D., de Simone, R. (eds.) CONCUR 1998. LNCS, vol. 1466, pp. 470–484. Springer, Heidelberg (1998)
7. Beauquier, D.: Pumping lemmas for timed automata. In: Nivat, M. (ed.) FOSSACS 1998. LNCS, vol. 1378, pp. 81–94. Springer, Heidelberg (1998)
8. Beck, M., Robins, S.: Computing the continuous discretely: Integer-point enumeration in polyhedra. Springer, Heidelberg (2007)
9. Carnevali, L., Grassi, L., Vicario, E.: State-density functions over DBM domains in the analysis of non-Markovian models. IEEE Trans. Software Eng. 35(2), 178–194 (2009)
10. Comon, H., Jurski, Y.: Timed automata and the theory of real numbers. In: Baeten, J.C.M., Mauw, S. (eds.) CONCUR 1999. LNCS, vol. 1664, pp. 242–257. Springer, Heidelberg (1999)
11. Dima, C.: Computing reachability relations in timed automata. In: LICS, p. 177. IEEE Computer Society, Los Alamitos (2002)
12. Gómez, R., Bowman, H.: Efficient detection of Zeno runs in timed automata. In: Raskin, J.-F., Thiagarajan, P.S. (eds.) FORMATS 2007. LNCS, vol. 4763, pp. 195–210. Springer, Heidelberg (2007)
13. Gupta, V., Henzinger, T.A., Jagadeesan, R.: Robust timed automata. In: Maler, O. (ed.) HART 1997. LNCS, vol. 1201, pp. 331–345. Springer, Heidelberg (1997)
14. Henzinger, T.A., Manna, Z., Pnueli, A.: What good are digital clocks? In: Kuich, W. (ed.) ICALP 1992. LNCS, vol. 623, pp. 545–558. Springer, Heidelberg (1992)
15. Henzinger, T.A., Raskin, J.F.: Robust undecidability of timed and hybrid systems. In: Lynch, N.A., Krogh, B.H. (eds.) HSCC 2000. LNCS, vol. 1790, pp. 145–159. Springer, Heidelberg (2000)
16. Krcál, P.: Infinite Structures in Timed Systems. Ph.D. thesis, University of Uppsala, Dept. of Information Technology (May 2009)
17. Lindenstrauss, E., Weiss, B.: Mean topological dimension. Israel J. Math 115, 1–24 (2000)
18. Maler, O., Pnueli, A.: On recognizable timed languages. In: Walukiewicz, I. (ed.) FOSSACS 2004. LNCS, vol. 2987, pp. 348–362. Springer, Heidelberg (2004)
19. Puri, A.: Dynamical properties of timed automata. Discrete Event Dynamic Systems 10(1-2), 87–113 (2000)
20. Simon, I.: Factorization forests of finite height. Theor. Comput. Sci. 72(1), 65–94 (1990)
21. Wulf, M.D., Doyen, L., Markey, N., Raskin, J.F.: Robust safety of timed automata. Formal Methods in System Design 33(1-3), 45–84 (2008)
22. Wulf, M.D., Doyen, L., Raskin, J.F.: Almost ASAP semantics: from timed models to timed implementations. Formal Asp. Comput. 17(3), 319–341 (2005)