Generating Functions of Timed Languages*

Eugene Asarin[1,**], Nicolas Basset[2], Aldric Degorre[1], and Dominique Perrin[2]

[1] LIAFA, University Paris Diderot and CNRS, France
[2] LIGM, University Paris-Est Marne-la-Vallée and CNRS, France

Abstract. In order to study precisely the growth of timed languages, we associate to such a language a generating function. These functions (tightly related to volume and entropy of timed languages) satisfy compositionality properties and, for deterministic timed regular languages, can be characterized by integral equations. We provide procedures for closed-form computation of generating functions for some classes of timed automata and regular expressions.

1 Introduction

Since the introduction of timed automata in [1], these automata and their languages are extensively studied both in theoretical perspective and in applications to verification of real-time systems. However, the natural question of measuring the size of timed languages was addressed only recently in [4,3] and a couple of subsequent works. In these articles we explored the asymptotic behavior of the volume of a timed language when the number of events tends to ∞. We showed that for most deterministic timed automata this volume grows (or decreases) exponentially, defined entropy as its growth rate, characterized this entropy as a logarithm of the spectral radius of an integral operator Ψ and showed how to compute the entropy symbolically or numerically.

We believe that size analysis can be useful in several aspects: entropy is a measure of information content in timed words [4] and a key to a timed code theory (work in progress). Whenever the entropy is not too small, timed automata have nice robustness properties [5]. As a practical perspective, we are exploring applications of size analysis to random generation and compression of timed words. We also find the study of the size of timed languages a natural and mathematically appealing generalization of classical results on regular languages and formal series.

In this article, we make a much more precise size analysis of timed languages accepted by deterministic timed automata. We associate to such a language L the sequence of its volumes $\mathrm{Vol}(L_n)$, and the generating function $f(z) = \sum_n \mathrm{Vol}(L_n) z^n$.

* The support of Agence Nationale de la Recherche under the project EQINOCS (ANR-11-BS02-004) is gratefully acknowledged.

** A part of the work of this author was done during his stay at the Institute of Mathematical Sciences (National University of Singapore) within its Programme "Automata Theory and Applications".

B. Rovan, V. Sassone, and P. Widmayer (Eds.): MFCS 2012, LNCS 7464, pp. 124–135, 2012.

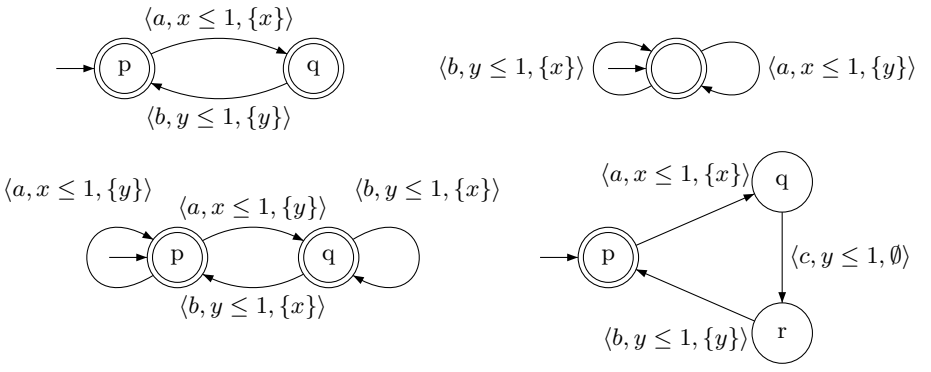

Fig. 1. Timed automata. First line: $\mathcal{A}_1$, $\mathcal{A}_2$; second line: $\mathcal{A}_3$, $\mathcal{A}_4$.

Thus the function $f(z)$ contains a complete information on the "size profile" of $\mathrm{Vol}(L_n)$ as a function of n. To relate it to the previous work, we show that $f(z)$ can be expressed in terms of the resolvent of the operator Ψ, and that the entropy of a timed language depends only on the convergence radius of $f(z)$.

Throughout the paper we use examples in Fig. 1, to illustrate the notions of volume, generating function and techniques for computing the latter. Thus, the timed language recognized by automaton $\mathcal{A}_1$ is $L = \{t_1, a, t_2, b, t_3, a, \ldots | \forall i\ (t_i + t_{i+1} \leq 1)\}$. For any number of events n we have a polytope in $\mathbb{R}^n$: $L_n = \{t_1, t_2, t_3, \ldots, t_n | \forall i\ (t_i + t_{i+1} \leq 1)\}$, the sequence of volumes V_n of these polytopes is $1; 1; \frac{1}{2}; \frac{1}{3}; \frac{5}{24}; \frac{2}{15}; \frac{61}{720}; \frac{17}{315}; \frac{277}{8064} \ldots$, and it was shown in [3] that this sequence behaves asymptotically like $(2/\pi)^n$. The methods developed in this paper yield a closed-form expression for the generating function of volumes: $\tan z + \sec z$. The convergence radius of the series, $\pi/2$, is the inverse of the growth rate of the sequence V_n. This series describes precisely the sequence of volumes, and a closed-form formula for V_n can be deduced: $V_{2n-1} = B_{2n}(-4)^n(1-4^n)/(2n)!$; $V_{2n} = (-1)^n E_{2n}/(2n)!$, where B_s stand for Bernoulli numbers and E_s for Euler numbers.

Generating functions behave in a natural way with respect to simple operations on timed languages (disjoint union, unambiguous concatenation, and unambiguous star). However in order to obtain an exact characterization and eventually closed-form expressions for generating function of timed regular languages a more involved analysis is needed. Such an analysis constitutes the main contribution of the article.

Related Work. Our generating functions generalize those of regular languages, thoroughly studied and applied, [6,8,9]. We are not aware of any work on generating functions of timed languages. Techniques and ideas used in this article build on our previous works on volumes and entropy of timed languages; especially on [3] (however the current article is self-contained). As for automata and languages under study, we investigate timed regular languages of [1], clock languages and expressions as in [7], and subclasses of timed automata: regenerating automata from [10] and $1\frac{3}{4}$-clocks automata, that extend both regenerating automata and $1\frac{1}{2}$-clocks automata from [3].

Whenever the alphabet of a timed automaton contains only one letter, its language can be seen as a sequence of polytopes $P_n \subseteq \mathbb{R}^n$. Some of such sequences, known as Fibonacci polytopes, have been studied (independently of timed automata) in combinatorics (see [11] and references therein). In particular, [11] provides a full analysis of the sequence of polytopes produced by automaton $\mathcal{A}_1$ on Fig. 1. Thus our work points at a connection between timed automata and enumerative combinatorics.

Article Structure. In Sect. 2 we introduce a formalism (inspired by [7]) for timed and clock languages, introduce volume functions of such languages, and investigate the properties of these functions. In Sect. 3 we introduce generating functions of timed languages and investigate their general properties. In Sect. 4 we explain how to compute generating functions for several subclasses of timed automata. We summarize the contributions and discuss the directions of future work in Sect. 5.

A longer version [2] of this article with additional proof details and examples is available on-line.

2 Preliminaries

2.1 Clock Languages and Timed Languages

In this paper, we study timed languages (mostly regular) using an approach based on clock languages introduced in [7]. We present this approach in a slightly different form along with a multi-stage semantics. The general idea is as follows: we are interested in timed languages. Timed languages are obtained as projections of clock languages. Clock languages are homomorphic images of discrete "triplet languages". Triplet languages, in turn, can be generated by automata or regular expressions. Below we define formally all these notions and illustrate them on a running example.

An alphabet of *timed events* is the product $\mathbb{R}^+ \times \Sigma$ where Σ is a finite alphabet. The meaning of a timed event (t, a) is that t is the time delay before the event a. A *timed word* is a sequence of timed events and a *timed language* is just a set of timed words.

Inspired by [7], we enrich timed words and languages with d-dimensional clock vectors. A *clock* is a variable which takes values in $\mathbb{R}^+$. In our setting, values of clocks will be bounded by a positive integer M. A *clock word* is a timed word together with an initial and a final clock vector, i.e. an element of $\mathbb{R}^d \times (\mathbb{R}^+ \times \Sigma)^* \times \mathbb{R}^d$. Two clock words $[\mathbf{x}\|\mathbf{w}\|\mathbf{y}]$ and $[\mathbf{x}'\|\mathbf{w}'\|\mathbf{y}']$ are said to be compatible if $\mathbf{y} = \mathbf{x}'$, in this case we define their product by $[\mathbf{x}\|\mathbf{w}\|\mathbf{y}]\cdot[\mathbf{y}\|\mathbf{w}'\|\mathbf{y}'] = [\mathbf{x}\|\mathbf{w}\mathbf{w}'\|\mathbf{y}']$. A *clock language* is a set of clock words. The product of two clock languages $\mathcal{L}$ and $\mathcal{L}'$ is

$$\mathcal{L} \cdot \mathcal{L}' = \{c \cdot c' \mid c \in \mathcal{L},\ c' \in \mathcal{L}',\ c \text{ and } c' \text{ compatible}\}. \tag{1}$$

The neutral element $\mathcal{E}$ is $\{[\mathbf{x}\|\epsilon\|\mathbf{x}] \mid \mathbf{x} \in \mathbb{R}^d\}$ and the Kleene star of a language $\mathcal{L}$ is as usual $\mathcal{L}^* = \bigcup_k \mathcal{L}^k$ with $\mathcal{L}^0 = \mathcal{E}$.

A clock language $\mathcal{L}$ is said to be *deterministic* whenever for each clock word the final clock vector is uniquely determined by the initial clock vector and the timed word, in other words there exists a function $\sigma_{\mathcal{L}} : \mathbb{R}^d \times (\mathbb{R}^+ \times \Sigma)^* \to \mathbb{R}^d$ such that for any clock word $[\mathbf{x}\|\mathbf{w}\|\mathbf{y}]$ of $\mathcal{L}$, we have that $\mathbf{y} = \sigma_{\mathcal{L}}(\mathbf{x}, \mathbf{w})$. In the following, we work with deterministic clock languages[1].

To a clock language we associate its timed projections. Given $\mathcal{L}$, we define $\mathcal{L}(\mathbf{x}, \mathbf{x}')$ as the timed language leading from $\mathbf{x}$ to an element lower than $\mathbf{x}'$: $\mathcal{L}(\mathbf{x}, \mathbf{x}') = \{w \mid \exists \mathbf{y}\, [\mathbf{x}\|\mathbf{w}\|\mathbf{y}] \in \mathcal{L} \wedge \mathbf{y} \leq \mathbf{x}'\}$. We also define the timed language $\mathcal{L}(\mathbf{x}) = \{w \mid \exists \mathbf{y}\, [\mathbf{x}\|\mathbf{w}\|\mathbf{y}] \in \mathcal{L}\}$ as the language starting from $\mathbf{x}$. Note that $\mathcal{L}(\mathbf{x}) = \mathcal{L}(\mathbf{x}, \boldsymbol{M})$ where $\boldsymbol{M} = (M, \ldots, M)$ is the greatest clock vector possible.

2.2 From Timed Automata to Triplet, Clock and Timed Languages

In this section, following [7], we give a convenient representation of timed automata (such as those on Fig. 1) and their languages.

Triplets and Timed Automata. We define *timed automata* as finite automata over the finite alphabet $\mathcal{T}$ of *triplets*. These triplets are tuples $\langle a, \mathfrak{g}, \mathfrak{r}\rangle$ with: a a letter in Σ; $\mathfrak{g}$ a conjunction of constraints, $x_i \bowtie c$ ($i \in \{1..d\}$, $c \in \{0..M\}$, $\bowtie \in \{<, >, \leq, \geq\}$), called *guard*, and $\mathfrak{r} \subseteq \{1..d\}$ a set of indices of clocks to be reset. We suppose moreover that guards are such that all the clocks remain bounded by M.

Clock Semantics of Triplets. Informally, a triplet $\langle a, \mathfrak{g}, \mathfrak{r}\rangle$ corresponds to the following behaviors: starting from some initial clock vector $\mathbf{x}$ let some time t elapse (all the clocks advance by t), check that $\mathfrak{g}$ is satisfied, emit a and update the clocks according to $\mathfrak{r}$. Formally, the clock language of this triplet is $\mathcal{L}(\langle a, \mathfrak{g}, \mathfrak{r}\rangle) = \{[\mathbf{x}\|(t, a)\|\mathfrak{r}(\mathbf{x} + t)] \mid \mathbf{x} + t \models \mathfrak{g}\}$. Here, for a clock vector $\mathbf{x} = (x_1, \ldots, x_d)$, we denote by $\mathbf{x} + t$ the vector $(x_1 + t, \ldots, x_d + t)$. Clock vectors are updated as follows: $\mathfrak{r}(y_1, \ldots, y_d) = (y'_1, \ldots, y'_d)$ with $y'_i = 0$ if $i \in \mathfrak{r}$ and $y'_i = y_i$ otherwise.

This definition can be extended to all triplet words by: $\mathcal{L}(\epsilon) = \mathcal{E}$ and $\mathcal{L}(\pi_1 \ldots \pi_n) = \mathcal{L}(\pi_1) \ldots \mathcal{L}(\pi_n)$ (using the product of clock languages as defined in (1)). Finally for a language $L \subseteq \mathcal{T}^*$, we define $\mathcal{L}(L) = \{\mathcal{L}(\pi) \mid \pi \in L\}$. In fact, $\mathcal{L}$ is a morphism between the two Kleene algebras of triplet and clock languages.

Timed Automata and Their Languages. Given a *timed automaton* $\mathcal{A}$, its *discrete semantics* L is the language of triplet words accepted by $\mathcal{A}$ seen as a finite automaton over $\mathcal{T}$; its *clock semantics* is $\mathcal{L}_{\mathcal{A}} = \mathcal{L}(L)$ and its *timed semantics* is $\mathcal{L}_{\mathcal{A}}(\mathbf{0})$. The timed automata considered in this article are assumed to be deterministic in the sense of [1], i.e. outgoing transitions from the same state with the same letter must have pairwise incompatible guards. Clock languages of deterministic automata are also deterministic.

A timed regular expression is defined as expression over the finite alphabet $\mathcal{T}$. Its discrete, clock and timed semantics are defined similarly to the automata.

[1] Such are clock languages associated to deterministic timed automata. However, a product of two deterministic clock languages can be non-deterministic, and we will explicitly rule out this situation.

This multi-stage timed semantics is equivalent to the usual semantics of timed automata and timed regular expressions.

Example 1 (Our running example). Automata $\mathcal{A}_2$ and $\mathcal{A}_3$ on Fig. 1 have the same discrete semantics[2] which is captured by a regular expression: $(\langle a, x \leq 1, \{y\}\rangle + \langle b, y \leq 1, \{x\}\rangle)^*$. An example of a clock word recognized by the automaton is $[(0.5, 0.8)\|(0.3, a)(0.1, a)(0.9, b)\|(0, 0.9)]$. The timed language recognized is: $\{(t_1, a)\cdots(t_{n_1}, a)(t_{n_1+1}, b)\cdots(t_{n_2}, b)(t_{n_2+1}, a)\cdots(t_{n_3}, a)\cdots \mid \forall j \geq 0,\ \sum_{i=n_j+1}^{n_{j+1}} t_i \leq 1\}$, with $n_0 = 0$ and possibly $n_1 = 0$.

Matrix Notation. A convenient way to see automata is the matrix form. A timed automaton $\mathcal{A}$ over a set of control states Q and an alphabet of transitions $\mathcal{T}$ is uniquely described by three ingredients:

- a $Q \times Q$-matrix $\mathbb{T}$ whose element $\mathbb{T}_{qq'}$ is the set of triplets labelling transitions from q to q';
- a row vector $\mathbf{I}$ describing initial states: for each control state p, its element $I_p = \{\epsilon\}$ iff p is initial, and $\emptyset$ otherwise;
- a column vector $\mathbf{F}$ describing final states: for each control state q, its element $F_q = \{\epsilon\}$ iff q is final, and $\emptyset$ otherwise.

The coefficient $(\mathbb{T}^n)_{p,q}$ of $\mathbb{T}^n$ contains the language of all the triplet words of length n from p to q. The p^{th} coordinates of the column vector $\mathbb{T}^n\mathbf{F}$ contains the language recognized from state p and $\mathbf{I}\mathbb{T}^n\mathbf{F}$ contains the language of triplet words of length n recognized by $\mathcal{A}$. For instance the matrices for $\mathcal{A}_3$ are:

$$\mathbf{I} = \begin{pmatrix} \{\epsilon\} & \emptyset \end{pmatrix};\ \mathbb{T} = \begin{pmatrix} \{\langle a, x \leq 1, \{y\}\rangle\} & \{\langle b, y \leq 1, \{x\}\rangle\} \\ \{\langle a, x \leq 1, \{y\}\rangle\} & \{\langle b, y \leq 1, \{x\}\rangle\} \end{pmatrix};\ \mathbf{F} = \begin{pmatrix} \{\epsilon\} \\ \{\epsilon\} \end{pmatrix}.$$

2.3 Volume(s) of Timed and Clock Languages

Measurable Timed Languages and Clock Languages. A timed language L is *measurable* if, for any word $w \in \Sigma^*$, the projection $L_w = \{\mathbf{t} \in \mathbb{R}^{|w|} \mid (\mathbf{t}, w) \in L\}$[3] is a Lebesgue-measurable subset of $\mathbb{R}^{|w|}$. A clock language $\mathcal{L}$ is measurable if it is deterministic and for every $w \in \Sigma^*$, $\sigma_{\mathcal{L}}(\cdot, (\cdot, w))$ is a Lebesgue-measurable function of $\mathbb{R}^d \times \mathbb{R}^{|w|} \to \mathbb{R}^d$. We remark that timed languages and deterministic clock languages obtained from triplet languages are measurable because their timed projections are polytopes.

Volumes of a Timed Language [3]. The sequence of volumes $(V_n(L))_{n\in\mathbb{N}}$ associated to a measurable timed language is $V_n(L) = \sum_{w\in\Sigma^n} \mathrm{Vol}(L_w)$, where Vol is the hyper-volume (i.e. Lebesgue measure) in $\mathbb{R}^n$. For dimension 0 we define $V_0(L) = 1$ if $\epsilon \in L$, and $V_0(L) = 0$ otherwise.

Now, for a clock language $\mathcal{L}$ and a word $w \in \Sigma^*$ of length $n \geq 0$, we define the clock language $\mathcal{L}(w) = \{[\mathbf{x}\|(\mathbf{t}, v)\|\mathbf{x}'] \mid v = w\}$.

[2] The difference will appear in section 4.1 since $\mathcal{A}_3$ is $1\frac{3}{4}$-clocks and $\mathcal{A}_2$ is not.
[3] by a slight abuse of notation, since $(\mathbb{R} \times \Sigma)^n$ is isomorphic to $\mathbb{R}^n \times \Sigma^n$.

Volumes Constrained by Initial and Final Clock Vectors. Timed regular languages considered below come from clock languages (which themselves come from triplet languages). The information about clock vectors is crucial to compute the volume of timed languages in a compositional manner.

Thus we define parametric volumes depending on initial and final clock vectors as follows $V_n^2(\mathbf{x},\mathbf{x}') = V_n(\mathcal{L}(\mathbf{x},\mathbf{x}'))$. We call this function the *cumulative volume function* (CVF)[4] of $\mathcal{L}$. We also allow the following notations: for a clock language $\mathcal{L}$ and a discrete events word w, $V^2_{\mathcal{L}(w)}(\mathbf{x},\mathbf{x}') = V^2_{|w|}(\mathcal{L}(w)(\mathbf{x},\mathbf{x}'))$; and for a triplets word π, $V^2_\pi(\mathbf{x},\mathbf{x}') = V_{|\pi|}(\mathcal{L}(\pi)(\mathbf{x},\mathbf{x}'))$. The notion of parametric volumes can be also applied to the clock language constrained only by initial clock vector $\mathcal{L}(\mathbf{x})$: $V_n^1(\mathbf{x}) = V_n(\mathcal{L}(\mathbf{x}))$. Clearly $V_n^1(\mathbf{x}) = V_n^2(\mathbf{x},\infty) = V_n^2(\mathbf{x},M)$.

CVFs for a Triplet Word. According to the following result, a CVF is easy to compute for a triplet word, and hence for a finite triplet language.

Proposition 1. *For a triplet word π the CVF V^2_π is piecewise polynomial with rational coefficients of degree $\leq |\pi|$. The pieces are polytopes, and an expression of this function is computable.*

Composing CVFs. In order to define a composition for CVF corresponding to the concatenation of triplet words and languages, we proceed as follows. We define composition of two functions of $\mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ as: $V_1^2 \star V_2^2(\mathbf{x},\mathbf{x}') = \int_{\mathbf{y}} V_2^2(\mathbf{y},\mathbf{x}')V_1^2(\mathbf{x},d\mathbf{y})$, where the integral is the Lebesgue-Stieltjes integral[5]. We also define $V_1^2 \star v(\mathbf{x}) = \int_{\mathbf{y}} v(\mathbf{y})V_1^2(\mathbf{x},d\mathbf{y})$, when v is defined on $\mathbb{R}^d$. Then we can state the key lemma (to transpose concatenation of words to the CVFs world):

Proposition 2. *For any measurable clock languages $\mathcal{L}_1$ and $\mathcal{L}_2$ and discrete words w_1 and w_2, $V^2_{\mathcal{L}_1(w_1)} \star V^2_{\mathcal{L}_2(w_2)}$ is well defined and satisfies: $V^2_{\mathcal{L}_1(w_1)} \star V^2_{\mathcal{L}_2(w_2)} = V^2_{\mathcal{L}_1(w_1)\cdot\mathcal{L}_2(w_2)}$.*

Volume Functions in Timed Automata. As we did for languages, we introduce a Q-vector $\mathbf{V}_n(\mathbf{x})$ of volumes of clock languages and a $Q \times Q$-matrix $\mathbb{V}(\mathbf{x},\mathbf{x}')$ of cumulative volume functions of elements of the transition matrix $\mathbb{T}$: formally $\mathbf{V}_{n,q}(\mathbf{x}) = V_n(\mathcal{L}_q(\mathbf{x}))$ and $\mathbb{V}_{qq'}(\mathbf{x},\mathbf{x}') = V_1(\mathcal{L}(\mathbb{T}_{qq'})(\mathbf{x},\mathbf{x}'))$, with $\mathcal{L}_q = \mathcal{L}((\mathbb{T}^n\mathbf{F})_q)$. It follows from the proposition above that the matrix element $(\mathbb{V}^{\star n})_{pq}$ (of the matrix power wrt $\star$) contains the CVF of $\mathcal{L}((\mathbb{T}^n)_{pq})$, that is of the language of all the clock words of length n leading from p to q. Finally, we get the formula for volumes:

$$\mathbf{V}_n = \mathbb{V}^{\star n} \star \mathbf{V}_F, \tag{2}$$

with $\mathbf{V}_F$ a column vector with $\mathbf{V}_{F,p} = 1$ if p is final, and $\mathbf{V}_{F,p} = 0$ otherwise.

The following not so obvious property of volume functions will be used in the sequel.

Proposition 3. *In a timed automaton $\mathcal{A}$, for $n \geq 1$, the volume functions $\mathbf{V}_n(\mathbf{x})$ and $\mathbb{V}^{\star n}(\mathbf{x},\mathbf{x}')$ are continuous wrt the initial clock vector $\mathbf{x}$.*

[4] similarly to cumulative distribution functions in probability theory.

[5] By definition, the Lebesgue-Stieltjes integral $\int f(\mathbf{x})g(d\mathbf{x})$ is the Lebesgue integral of f wrt the measure μ having cumulative distribution function g.

3 Generating Functions

3.1 Definitions

To study volume sequences associated to timed and clock languages we define their generating functions. As usual for generating functions, they allow recovering the sequence, its growth rate, momenta etc; and they have nice compositional properties. Given a timed language L its *generating function* is defined as follows: $f_L(z) = \sum_k z^k V_k(L)$. Given a clock language $\mathcal{L}$, we define a *(parametric) generating function* with a given initial clock vector $f^1_{\mathcal{L}}(z, \mathbf{x}) = \sum_k z^k V_k(\mathcal{L}(\mathbf{x})) = f_L(z)$, with $L = \mathcal{L}(\mathbf{x})$. For a clock language $\mathcal{L}$ we also define another *cumulative generating function* with a given initial clock vector and a bound on the final clock vector: $f^2(z, \mathbf{x}, \mathbf{x}') = \sum_k z^k V^2_k(\mathbf{x}, \mathbf{x}') = f_L(z)$, with $L = \mathcal{L}(\mathbf{x}, \mathbf{x}')$. To summarize, we are interested in computing $f(z)$, but this computation will be based on $f^1(z, \mathbf{x})$, and sometimes on $f^2(z, \mathbf{x}, \mathbf{x}')$.

Given a timed automaton, timed and clock languages, and thus generating functions are naturally associated to its states, for example $f^1_q(z, \mathbf{x}) = \sum_k z^k V_k(\mathcal{L}_q(\mathbf{x})) = f_L(z)$, with $L = \mathcal{L}_q(\mathbf{x})$. Taken for all states, functions f_q and f^1_q form $|Q|$-dimensional vector functions $\mathbf{f}(z, \mathbf{x}), \mathbf{f}^1(z, \mathbf{x})$, while functions $f^2_{q,q'}$ form a $Q \times Q$-matrix function $\mathbb{f}^2(z, \mathbf{x}, \mathbf{x}')$.

3.2 Analytic Characterization

Elementary Properties. First, let us state the relations between the three kinds of generating functions:

Proposition 4. *The functions f, f^1, f^2 are related as follows: $f(z) = f^1(z, \mathbf{0})$; $f^1(z, \mathbf{x}) = f^2(z, \mathbf{x}, \boldsymbol{M})$.*

By definition, f^2, f^1 and f are analytic functions of z. Since we consider timed automata with guards bounded by some constant M, all the volumes V_k (with any initial or final conditions) can be upper bounded by $(M|\Sigma|)^k$. This implies that convergence radius of series for f^2, f^1 and f is at least $(M|\Sigma|)^{-1} > 0$. More precisely, the radius of convergence of f is $1/\limsup_{k\to\infty}(V_k(L))^{1/k} = 2^{-\mathcal{H}(L)}$, where $\mathcal{H}(L)$ is called the volumetric entropy of L (see [3]).

For generating functions associated to timed automata, the following result is a straightforward corollary of Prop. 3:

Proposition 5. *Within its convergence radius, the generating function $\mathbf{f}^1(z, \mathbf{x})$ associated to a timed automaton $\mathcal{A}$ is continuous wrt the initial clock vector $\mathbf{x}$.*

Integral Equation for Generating Functions. Consider a timed automaton. Using formula (2), its generating function can be computed as follows: $\mathbf{f}^1(z, \mathbf{x}) = \sum_k z^k \mathbf{V}_k(\mathbf{x}) = \sum_k z^k \mathbb{V}^{\star k} \star \mathbf{V}_F$, which implies our first main result.

Theorem 1 (Integral equation). *In the interior of its convergence circle, the generating function $\mathbf{f}^1$ is the unique solution of the integral equation*

$$\mathbf{f}^1 - z\mathbb{V} \star \mathbf{f}^1 = \mathbf{V}_F. \tag{3}$$

Example (1, continued). For the automaton $\mathcal{A}_3$ (using the notation $x \dot{-} y$ for $\max(x-y,0)$):

$$\mathbb{V} = \begin{pmatrix} \min(x',1) \dot{-} x\mathbb{1}_{y'\geq 0} & \min(y',1) \dot{-} y\mathbb{1}_{x'\geq 0} \\ \min(x',1) \dot{-} x\mathbb{1}_{y'\geq 0} & \min(y',1) \dot{-} y\mathbb{1}_{x'\geq 0} \end{pmatrix};\ \mathbf{V}_F = \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

Equation (3) gives: $f^1_p(z,x,y) = f^1_q(z,x,y) = 1 + z\int_x^1 f^1_p(z,x',0)dx' + z\int_y^1 f^1_q(z,0,y')dy'$. In Section 4.1 below we develop a technique for solving such equations (for a subclass of automata including this one), and compute the generating function for this language.

3.3 Volumes, Generating Functions and Functional Analysis

In this section (which can be skipped by a reader not interested in functional analysis), similarly to [3], we rephrase previous results in terms of the spectral theory of linear operators.

Given a timed automaton, consider the Banach space $\mathcal{F}$ of Q-vectors of continuous functions on clock valuations. Thus an element of $\mathcal{F}$ is a vector $\mathbf{v}$ whose components are continuous $v_q : [0;M]^d \to \mathbb{R}$, and $\mathcal{F} = C([0;M]^d)^Q$. The matrix $\mathbb{V}$ corresponds to an operator $\Psi : \mathcal{F} \to \mathcal{F}$ defined by $\Psi(\mathbf{v}) = \mathbb{V} \star \mathbf{v}$ (a variant of this operator plays the central role in [3]). In terms of this operator, Prop. 3 and equation (2) can be rephrased as follows:

Proposition 6 ([3]). *Ψ is a bounded linear operator on $\mathcal{F}$ (represented by a matrix of integral operators). The volume vector can be obtained by iteration of this operator: $\mathbf{V}_n = \Psi^n(\mathbf{V}_F)$.*

Recall that, by definition, the *resolvent* of an operator A is $R(\lambda, A) = (A - \lambda I)^{-1}$; it is well defined when λ does not belong to the spectrum of A, in particular for $|\lambda| > \rho(A)$, where ρ denotes the spectral radius. We obtain as a consequence of Thm. 1 another characterization of the generating function:

Proposition 7 (Generating function and resolvent). *The generating function $\mathbf{f}^1$ satisfies the formula: $\mathbf{f}^1 = -z^{-1}R(z^{-1},\Psi)\mathbf{V}_F$, which holds in the interior of the circle $|z| < \rho(\Psi)^{-1}$.*

3.4 Inductive Characterization of Generating Functions

The form of generating functions of finite triplet languages follows from Prop. 1:

Proposition 8. *For a finite triplet language L with maximal word length ℓ, the generating functions f^2, f^1 are piecewise polynomial in $z, \mathbf{x}, \mathbf{x}'$ (pieces are polytopes in $\mathbf{x}, \mathbf{x}'$) of degree $\leq \ell$ wrt z and wrt $\mathbf{x}$ and $\mathbf{x}'$.*

More complex languages can be obtained from finite ones using Kleene algebra operations. As usual in the context of generating functions, we suppose that the operations are unambiguous. A language operation is *ambiguous* if a word of the resulting language can be obtained in several ways by composing different words from the operands. We consider first the simple case of timed languages.

Proposition 9. *Generating functions behave well for unambiguous operations on measurable timed languages:* $f_{L_1 \cup L_2} = f_{L_1} + f_{L_2}$; $f_{L_1 \cdot L_2} = f_{L_1} f_{L_2}$; $f_{L^*} = 1 + f_L f_{L^*}$ *provided* $\epsilon \notin L$.

However, in order to obtain general timed regular languages we need operations on clock languages, which are more involved.

Proposition 10. *Generating functions* f^2 *behave well for unambiguous operations on deterministic measurable clock languages (whenever the resulting language is also deterministic):* $f^2_{\mathcal{L}_1 \cup \mathcal{L}_2} = f^2_{\mathcal{L}_1} + f^2_{\mathcal{L}_2}$; $f^2_{\mathcal{L}_1 \cdot \mathcal{L}_2} = f^2_{\mathcal{L}_1} \star f^2_{\mathcal{L}_2}$; $f^2_{\mathcal{L}^*} = \mathbb{1}_{\mathbf{x} \le \mathbf{x}'} + f^2_{\mathcal{L}} \star f^2_{\mathcal{L}^*}$ *provided* $\mathcal{E} \cap \mathcal{L} = \emptyset$.

Corollary 1. *Generating function* f^1 *for unambiguous compositions of clock languages (under the same hypotheses) can be computed as follows:* $f^1_{\mathcal{L}_1 + \mathcal{L}_2} = f^1_{\mathcal{L}_1} + f^1_{\mathcal{L}_2}$; $f^1_{\mathcal{L}_1 \cdot \mathcal{L}_2} = f^2_{\mathcal{L}_1} \star f^1_{\mathcal{L}_2}$; $f^1_{\mathcal{L}^*} = 1 + f^2_{\mathcal{L}} \star f^1_{\mathcal{L}^*}$ *provided* $\mathcal{E} \cap \mathcal{L} = \emptyset$.

4 Computing Generating Functions

The generating function of a timed language represented by an automaton is characterized by a system of integral equations (3). The generating function of a timed language represented by a regular expression can be found recursively from piecewise polynomial functions using operations $+, \star$ and solving fixpoint integral equations of Prop. 10 and Cor. 1. Unfortunately, both procedures involve computation of integrals, and solution of integral equations, for this reason, the result cannot be always presented by an explicit formula. Below we consider several subclasses of timed automata, for which generating functions can be obtained in closed form, or at least admit a simpler characterization.

4.1 Generating Functions for Particular Classes of Automata

System of Equations. Our closed-form solutions for subclasses of timed automata will be obtained using a variant of language equations.

Let $Q = G \cup B$ be a disjoint partition of the states of a timed automaton $\mathcal{A}$ into *good* and *bad*. We want to describe the vector $\overline{\mathbf{L}}$ of triplet languages L_q recognized from good states $q \in G$ only. This vector satisfies the equation:

$$\overline{\mathbf{L}} = \overline{\mathbb{T}} \cdot \overline{\mathbf{L}} + \overline{\mathbf{F}}, \tag{4}$$

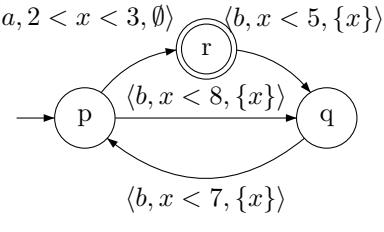

Fig. 2. A regenerating automaton

where $\overline{\mathbb{T}}$ is a $G \times G$-matrix and $\overline{\mathbf{F}}$ is a G-vector of triplet languages. Their elements are defined as follows: $\overline{T}_{pq}$ consists of all words leading from p to q via bad states only; $\overline{F}_p$ consists of all words leading from p to a final state via bad states only[6].

[6] If this final state is good the word should be ϵ.

Automata with Regeneration. Following [10], we call an automaton *regenerating* if there exists a partition $Q = G \cup B$ having two properties: (a) every cycle in the automaton contains a state in G (good); (b) all the transitions coming into a good state reset all clocks.

W.l.o.g. we suppose that the initial state is good (this can be achieved by adding a new initial state). Condition (a) implies that no cycle is possible within bad states, and thus all the elements of $\overline{\mathbb{T}}$ and $\overline{\mathbf{F}}$ are finite triplet languages (with maximal word length $\leq |B| + 1$). Condition (b) means that (4) can be rewritten in timed languages (instead of clock languages), since when entering in a good state all clocks are reset. This gives

$$\overline{\mathbf{L}}_{\text{timed}} = \overline{\mathbb{T}}_{\text{timed}} \cdot \overline{\mathbf{L}}_{\text{timed}} + \overline{\mathbf{F}}_{\text{timed}}. \tag{5}$$

Applying simple compositionality conditions for generating functions for timed languages (Prop. 9) we obtain that $\overline{\mathbf{f}} = \overline{\mathbb{f}}\ \overline{\mathbf{f}} + \overline{\mathbf{f}}_F$. Due to Prop. 8 all the coefficients (elements of matrix $\overline{\mathbb{f}}$ and vector $\overline{\mathbf{f}}_F$) are polynomials of z. Solving this linear $|G|$-dimensional system we express $\overline{\mathbf{f}}$ as a vector of rational functions of z: $\overline{\mathbf{f}} = \left(I - \overline{\mathbb{f}}\right)^{-1} \overline{\mathbf{f}}_F$. The generating function f of the timed language accepted by the automaton is just one element of this vector $\overline{\mathbf{f}}$. We conclude.

Theorem 2. *For a regenerating automaton the generating function $f(z)$ is a rational function.*

Example 2. Consider a regenerating automaton on Fig. 2. We choose good and bad states as follows: $G = \{p, q\}; B = \{r\}$. The system of equations on timed languages of good states takes the form

$$\begin{pmatrix} L_p \\ L_q \end{pmatrix} = \begin{pmatrix} \emptyset & T_{pq} \\ T_{qp} & \emptyset \end{pmatrix} \cdot \begin{pmatrix} L_p \\ L_q \end{pmatrix} + \begin{pmatrix} F_p \\ \emptyset \end{pmatrix} \text{ with} \tag{6}$$

$$\begin{aligned} T_{pq} &= \{(t_1, a)(t_2, b) | 2 < t_1 < 3 \wedge t_1 + t_2 < 5\} \cup \{(t, b) | t < 8\}; \\ T_{qp} &= \{(t, b) | t < 7\}; \\ F_p &= \{(t, a) | 2 < t < 3\}. \end{aligned}$$

For generating functions this yields: $\begin{pmatrix} f_p \\ f_q \end{pmatrix} = \begin{pmatrix} 0 & 2.5z^2 + 8z \\ 7z & 0 \end{pmatrix} \cdot \begin{pmatrix} f_p \\ f_q \end{pmatrix} + \begin{pmatrix} z \\ 0 \end{pmatrix}$. Solving this linear system we find the required $f_p(z) = 2z/\left(2 - 35z^3 - 112z^2\right)$. It converges for $|z| < 0.1309$, its Taylor coefficients (i.e. volumes V_n for $n = 0..11$) are $0; 1; 0; 56; 17\frac{1}{2}; 3136; 1960; 175922\frac{1}{4}; 164640; 9885946; 12298479\frac{3}{8}; 556494176$.

$1\frac{3}{4}$-Clocks Automata. We call an automaton *$1\frac{3}{4}$-clocks* if there exists a partition of $Q = G \cup B$ into good and bad states having three properties:(a) every cycle in the automaton contains a good state; (b) the initial state is a good one; (c) for each good state p there is at most one clock $x_{i(p)}$ not reset by incoming transitions.

Similarly to regenerating automata, we apply equations (4), and observe that all the coefficients are finite triplet languages. Unfortunately, since some clocks

are not reset, we cannot write an equation on timed languages similar to (5). Instead, we pass to clock languages and their generating functions, as in the general case. This gives:

$$\overline{\mathbf{f}^1} = \overline{\mathbb{f}^2} \star \overline{\mathbf{f}^1} + \overline{\mathbf{f}^1}_F, \tag{7}$$

an integral equation with piecewise polynomial coefficients. We notice that functions in the last equation depend on the clock vector $\mathbf{x} \in \mathbb{R}^d$ (or on two clock vectors $\mathbf{x}$,$\mathbf{x}'$), but in fact for any good state $p \in G$ only one clock $x_{i(p)}$ matters. This allows extracting simpler integral equations from (7), involving only functions of scalar argument.

We proceed as follows: given a G-vector $\mathbf{v}$ whose elements v_p are functions on $\mathbb{R}^d$, we define reduced functions on $\mathbb{R}$: $\widetilde{v}_p(x) = v_p(0, \ldots, 0, x, 0, \ldots, 0)$, with the argument x at position $i(p)$. Reduced G-vector $\widetilde{\mathbf{v}}$ consists of reduced elements $\widetilde{v}_p$. Reduced versions of matrices are defined similarly.

The following identity is based on the requirement of clock resets:

Lemma 1. *For a $1\frac{3}{4}$-clocks automaton the following holds:* $\widetilde{\overline{\mathbb{f}^2} \star \overline{\mathbf{f}^1}} = \widetilde{\overline{\mathbb{f}^2}} \star \widetilde{\overline{\mathbf{f}^1}}$.

Equation (7), reduced to $\widetilde{\overline{\mathbf{f}^1}} = \widetilde{\overline{\mathbb{f}^2}} \star \widetilde{\overline{\mathbf{f}^1}} + \widetilde{\overline{\mathbf{f}^1}_F}$, implies that the reduced vector of generating functions is a solution of equations of the form:

$$\mathbf{f}(z, x) = (\mathbb{A} \star \mathbf{f})(z, x) + \mathbf{b}(z, x), \tag{8}$$

where all the coefficients are piecewise polynomial functions of z and a scalar argument x.

Lemma 2. *An integral equation of the form (8) can be transformed into a system of linear ordinary differential equation with piecewise polynomial coefficients (depending on x and z).*

Theorem 3. *For a $1\frac{3}{4}$-clocks automaton the generating function f can be obtained by solving a system of linear ordinary differential equations with piecewise polynomial coefficients.*

We notice that the theorem gives a rather explicit characterization of f, but not always a closed-form expression.

Example (1, completed). $\mathcal{A}_3$ is $1\frac{3}{4}$-clocks[7] with good states $G = \{p, q\}$ and no bad state $B = \emptyset$. The matrix $\mathbb{A}$ and the vector $\mathbf{b}$ are

$$\mathbb{A} = \begin{pmatrix} z(\min(x', 1) \dot{-} x) & z \min(x', 1) \\ z \min(x', 1) & z(\min(x', 1) \dot{-} x) \end{pmatrix}; \ \mathbf{b} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

We use equation (8) and remark that by symmetry of $\mathbb{A}$, the two generating functions $\widetilde{f^1_p}$ and $\widetilde{f^1_q}$ are equal to a unique function f^1 which satisfies $f^1(z, x) = z \int_x^1 f^1(z, x')\, dx' + z \int_0^1 f^1(z, x')\, dx' + 1$. Differentiating it one time w.r.t x we obtain: $\frac{\partial f^1}{\partial x}(z, x) = -z f^1(z, x)$. The solution has the form $f^1(z, x) = A(z) e^{-zx}$. Remark that $f^1(z, 0) - 1 = 2z \int_0^1 f^1(z, x')\, dx' = 2(f^1(z, 1) - 1)$. We are done since $f(z) = f^1(z, 0) = A(z) = 1/(2e^{-z} - 1)$.

[7] $\mathcal{A}_3$ can be seen as $\mathcal{A}_2$ whose state is split to make it a $1\frac{3}{4}$-clock automaton.

Example 3. Automata $\mathcal{A}_1$ and $\mathcal{A}_4$ are also $1\frac{3}{4}$-clocks, and their generating functions are $\tan z + \sec z$ and $4/\left(\pi\left(\mathrm{Bi}'(0)\mathrm{Ai}(-z) - \mathrm{Ai}'(0)\mathrm{Bi}(-z)\right)\right)$, where Ai and Bi stand for Airy functions.

5 Conclusions

In this article, we have introduced generating functions of timed languages, explored their properties and characterized them by integral equations. For subclasses of timed regular languages we have presented closed-form expressions or simpler characterization of generating functions. Generating functions describe with a high precision the quantitative behavior of timed languages.

At the current stage of research, the computation of generating functions is a semi-manual task and restrictions are imposed to the automata. We are planning to explore theoretical and practical algorithmics of timed generating functions, and to implement the algorithm. On the other hand, we want to see whether closed form solutions are possible beyond the class of $1\frac{3}{4}$-clocks languages.

We hope that this approach will lead to new combinatorial results for timed regular languages and sequences of polytopes, better quantitative characterization of such languages with applications to information theory and verification of real-time systems. Also, the approach can be extended to timed formal series, non-regular timed languages, or to richer models such as hybrid automata.

References

1. Alur, R., Dill, D.L.: A theory of timed automata. Theoretical Computer Science 126, 183–235 (1994)
2. Asarin, E., Basset, N., Degorre, A., Perrin, D.: Generating functions of timed languages. Tech. rep (2012), `http://hal.archives-ouvertes.fr/hal-00678443`
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. Basset, N., Asarin, E.: Thin and Thick Timed Regular Languages. In: Fahrenberg, U., Tripakis, S. (eds.) FORMATS 2011. LNCS, vol. 6919, pp. 113–128. Springer, Heidelberg (2011)
6. Berstel, J., Reutenauer, C.: Noncommutative rational series with applications. Enc. of Math. and Appl., vol. 137. Cambridge University Press (2011)
7. Bouyer, P., Petit, A.: A Kleene/Büchi-like theorem for clock languages. Journal of Automata, Languages and Combinatorics 7(2), 167–186 (2002)
8. Flajolet, P., Sedgewick, R.: Analytic combinatorics. Cambridge University Press (2009)
9. Salomaa, A., Soittola, M.: Automata-theoretic aspects of formal power series. Springer (1978)
10. Sassoli, L., Vicario, E.: Close form derivation of state-density functions over DBM domains in the analysis of non-Markovian models. In: QEST 2007, pp. 59–68. IEEE Computer Society (2007)
11. Stanley, R.P.: A survey of alternating permutations. In: Combinatorics and graphs. Contemp. Math., vol. 531, pp. 165–196. Amer. Math. Soc., Providence (2010)