Equations on Timed Languages *

Eugene ASARIN

Institute for Information Transmission Problems
19 Bolshoï Karetnyï lane, 101447, Moscow, Russia
asarin@ippi.ras.ru

Abstract. We continue investigation of languages, accepted by timed automata of Alur and Dill. In [ACM97] timed regular expressions equivalent to timed automata were introduced. Here we introduce quasilinear equations over timed languages with regular coefficients. We prove that the minimal solution of such an equation is regular and give an algorithm to calculate this solution. This result is used to obtain a new proof of Kleene theorem ([ACM97]) for timed automata. Equations over timed languages can be also considered as an alternative way of specifying these languages.

1 Introduction

Timed automata ([AD94]) form the best investigated class of hybrid systems. It is known which problems about these automata are decidable and which are not, and there are tools for testing emptiness, evaluating reachable states etc. ([DOTY96]). However some theoretical aspects and parallels with ordinary finite automata are still not clear. This paper may be considered as a continuation of ([ACM97]) where timed languages were analyzed from the traditional linguistic viewpoint — and timed regular expression capable to specify exactly the same languages as timed automata were introduced.

We take for a model following classical (forty years old) results about finite automata, regular languages and linear equations (see e.g. [Brz62]).

Any system of linear equations in the form

$$X_i = \alpha_i + \sum_{j=1}^{n} \beta_{ij} X_j \qquad i = 1, \ldots, n, \tag{1}$$

where X_i stand for unknown languages and α_i, β_{ij} — for given regular coefficients, has a regular minimal solution. The regular expression for this solution can be found effectively from the coefficients.

For any finite automaton a system (1) can be easily constructed, each unknown X_i of the system corresponding to a state q_i of the automaton. In the

* This research was supported in part by the Russian Foundation for Basic Research under the grants 97-01-00692 and 96-15-96048; and by the International Association for the Promotion of Cooperation with Scientists from the Independent States of the Former Soviet Union (INTAS) under the grant 94-697.

minimal solution, the language X_i is exactly the language accepted by the automaton starting from the state q_i.

As a corollary these two classical results imply Kleene theorem ([Kle56]) about regularity of languages accepted by finite automata.

Our aim is to port these results to timed automata and to introduce a class of equations over timed languages capable to specify languages of one-clock timed automata. These equations are similar to classical linear equations (1). However the following example shows that a straightforward timed adaptation of linear equations cannot work.

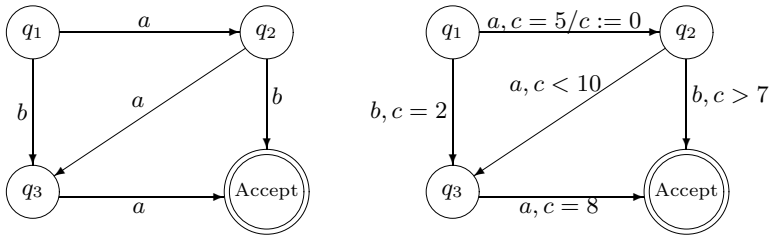

Fig. 1. Two automata

Example 1. The language of the first (untimed) automaton on Figure 1 can be represented by the following equations:

$$\begin{cases} X_1 = aX_2 + bX_3 \\ X_2 = b + aX_3 \\ X_3 = a \end{cases}$$

X_i here stands for the language accepted from the state q_i and each transition from q_i to q_j labeled with a can be represented by a term aX_j in the equation for X_i. Roughly speaking, such a transition corresponds to concatenating its label a to the language.

The case of the second (timed) automaton is more complicated because now there are two kinds of transitions. Some of them reset the clock and in this case they also can be represented by concatenation of the label (with time restriction) to the language. However some transitions do not reset the clock. We cannot write an equation like $X_1 = aX_2 + bX_3$ with a constraint on the sojourn time in state q_1, because after completing action b the automaton enters the state q_3 with a modified clock value.

To deal with this problem we introduce another composition operation on timed languages ($\circ$ operation) which corresponds to non-resetting transitions. We introduce quasilinear equations on timed languages which use both kinds

of concatenation ($\cdot$ and $\circ$) and are strong enough to represent one-clock timed automata.

Our main result is that any system of equations of this class with regular coefficients has a regular minimal solution. We give an algorithm to find out this solution.

The paper is motivated by the theory of timed automata, however the major part of it (sections 3–4) contains an automata-free theory of timed languages, regular timed expressions and quasilinear equations on timed languages. At our opinion, this linguistic approach could be useful for other classes of hybrid systems as well.

The outline of the paper is as follows. In section 2 we recall the definition of timed regular languages from [ACM97]. In section 3 the new operation $\circ$ over languages is formally introduced. This operation is crucial for representing timed automata by equations. We investigate algebraic properties of this operation and show, that $\circ$ can be eliminated in a sense. In section 4 quasilinear equations are introduced and solved. The possibility to solve this kind of equations is the main result of the paper. In section 5 we recall the definition of timed automata and apply our main result to languages of these automata. For any one-clock automaton we construct a quasilinear system, which represents the language of this automaton. This provides an alternative proof of expressive equivalence of timed automata and timed regular expressions from ([ACM97]). In the last section further work is discussed.

2 Timed Regular Languages

We reproduce in a slightly modified form the basic definitions of timed languages and timed regular equations from [ACM97]. Let Σ be a finite *alphabet* and let $\mathbb{R}_+$ denote the set of positive real numbers. A *signal* over Σ is a timed sequence of elements of Σ, i.e. a finite sequence $w = ((a_1, t_1), \ldots, (a_n, t_n))$ with $a_i \in \Sigma$ and $t_i \in \mathbb{R}_+$, such that $0 < t_1 < \ldots < t_n$. We will also write this signal as

$$w = a_1^{r_1} a_2^{r_2} \cdots a_n^{r_n},$$

where $r_1 = t_1$, and $r_{i+1} = t_{i+1} - t_i$, i.e. r_i are relative delays between a_i occurrences. We call t_n the *length* of w and denote it by $|w|$. The empty signal is denoted by ε. Its length equals 0. The set of all signals is denoted by $\mathcal{S}(\Sigma)$. Subsets of $\mathcal{S}(\Sigma)$ are referred to as *(timed) languages*. For every $w_1, w_2 \in \mathcal{S}(\Sigma)$ such that $w_1 = a_1^{r_1} a_2^{r_2} \cdots a_n^{r_n}$ and $w_2 = b_1^{s_1} b_2^{s_2} \cdots b_n^{s_n}$ we define their concatenation as $w = w_1 w_2 = a_1^{r_1} \cdots a_n^{r_n} b_1^{s_1} \cdots b_n^{s_n}$. This notion can be extended naturally to concatenation of languages by letting

$$L_1 L_2 = \{w_1 w_2 : w_1 \in L_1 \wedge w_2 \in L_2\}.$$

An *integer-bounded interval* is either $[l, u]$, $(l, u]$, $[l, u)$, or (l, u) where $l \in \mathbb{N}$ and $u \in \mathbb{N} \cup \{\infty\}$ such that $l \leq u$. We exclude $\infty]$ and use l for $[l, l]$.

Definition 1 ((Timed Regular Expressions)). *The set $\mathcal{E}(\Sigma)$ of timed regular expressions over an alphabet Σ,* (expressions, for short) *is defined recursively as either a, $\alpha_1 \cdot \alpha_2$, $\alpha_1 + \alpha_2$, α^* or $\langle\alpha\rangle_I$ where $a \in \Sigma$, $\alpha, \alpha_1, \alpha_2 \in \mathcal{E}(\Sigma)$ and I is an integer-bounded interval.*

The semantics of timed regular expressions, $[\![]\!] : \mathcal{E}(\Sigma) \to 2^{\mathcal{S}(\Sigma)}$, is given by:

$$\begin{array}{ll} [\![a]\!] & = \{a^r : r \in \mathbb{R}_+\} \\ [\![\alpha_1 + \alpha_2]\!] & = [\![\alpha_1]\!] \cup [\![\alpha_2]\!] \\ [\![\alpha_1 \cdot \alpha_2]\!] & = [\![\alpha_1]\!][\![\alpha_2]\!] \\ [\![\alpha^*]\!] & = \bigcup_{i=0}^{\infty}([\![\alpha^i]\!]) \\ [\![\langle\alpha\rangle_I]\!] & = [\![\alpha]\!] \cap \{w : |w| \in I\} \end{array}$$

Some comments should be given here. First, the semantics of a is not a singleton, but a non-countable language. The intuitive meaning of this expression is that some unknown time passes and then event a happens. Operations $+$, $\cdot$ and * are the same as for untimed languages. The only operation which introduces time explicitly is "time restriction" $\langle\rangle_I$ which chooses only those signals in the language, whose lengths belong to the constraining interval I.

Example 2.

$$[\![\langle\langle ab\rangle_{(2;3)}c\rangle_{100}]\!] = \{a^x b^y c^z | 2 < x + y < 3; x + y + z = 100\}.$$

To simplify notation we write ε for the following regular expression $\langle a^*\rangle_0$, whose semantics is exactly ε.

Expressions introduced here form a proper subclass of those introduced in [ACM97], because here intersection is not allowed in the syntax. This change explains the difference between the formulation of Theorem 2 below from that of the same theorem in [ACM97].

3 Operation $\circ$

Begin with the following *shift operation* over signals, which just delays the beginning by t and preserves relative delays between events.

Definition 2. *For a signal $w = a_1^{t_1} a_2^{t_2} \dots a_n^{t_n}$ let $S^t w = a_1^{t_1+t} a_2^{t_2} \dots a_n^{t_n}$*

We say that a language is *shift-invariant*, if $S^{-t}L = L$ for any $t > 0$, i.e. any signal w belongs (or does not belong) to L simultaneously with $S^t w$. The following condition is sufficient for shift invariance — the regular expression should not begin with something in $\langle\rangle$. Formally speaking

Lemma 1. *If a regular expression has a form $\sum_i \alpha_i\beta_i$ where $\alpha_i \not\ni \varepsilon$ and α_i does not contain $\langle\rangle$, then its language is shift-invariant. We call this type of regular expressions* dull.

Now we can define a new composition operation over timed languages which is crucial for describing timed automata.

Definition 3. *Let L_1 and L_2 be timed languages. Then*

$$L_1 \circ L_2 = \{w_1 w_2 | w_1 \in L_1 \text{ and } S^{|w_1|} w_2 \in L_2\}.$$

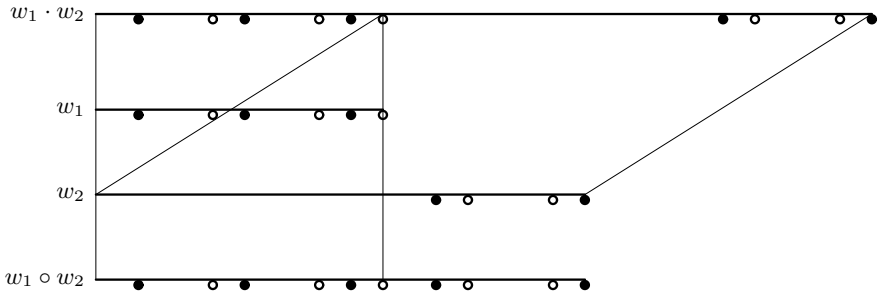

Fig. 2. Two compositions

In other words, for two signals $w_1 = ((a_1, t_1), \ldots, (a_n, t_n)) \in L_1$ and $w_2 = ((b_1, s_1), \ldots, (b_m, s_m)) \in L_2$ such that $t_n < s_1$ we include the signal $((a_1, t_1), \ldots, (a_n, t_n), (b_1, s_1), \ldots, (b_m, s_m))$ into $L_1 \circ L_2$. Figure 2 illustrates $\circ$-composition in comparison with concatenation.

First of all, state some simple algebraic properties of this composition operation.

Proposition 1 ((Algebraic properties of circle)).

- *operation $\circ$ is $+$-distributive:* $(\alpha + \beta) \circ \gamma = \alpha \circ \gamma + \beta \circ \gamma$ *and* $\alpha \circ (\beta + \gamma) = \alpha \circ \beta + \alpha \circ \gamma$
- *operation $\circ$ is associative:* $(\alpha \circ \beta) \circ \gamma = \alpha \circ (\beta \circ \gamma)$
- $\alpha \circ (\beta\gamma) = (\alpha \circ \beta)\gamma$ *if* $\varepsilon \notin \beta$

We suppose that $\circ$ cannot be expressed in terms of other operations. However, it can be eliminated for regular languages.

Proposition 2 ((Circle elimination)). *If L_1 and L_2 are regular, then $L_1 \circ L_2$ is regular. The regular expression for it can be obtained algorithmically.*

Circle elimination is easy with the following prefix form of regular expressions

Lemma 2. *Any regular expression can be effectively transformed to the form:*

$$\gamma + \sum_{k=1}^{n} \langle \alpha_k \rangle_{I_k} \beta_k \tag{2}$$

or

$$\varepsilon + \gamma + \sum_{k=1}^{n} \langle \alpha_k \rangle_{I_k} \beta_k, \tag{3}$$

where γ is dull and $\alpha_k \not\ni \varepsilon$.

The proof is by induction over the structure of regular expression. The only bad operation is Kleene star — all others are trivial. To deal with Kleene star suppose that δ is already in the prefix form (2) $\delta = (\gamma + \sum_k \langle \alpha_k \rangle_{I_k} \beta_k)$ and transform the expression δ^* to the form $\delta\delta^* + \varepsilon$ and open the parentheses:

$$\delta^*\delta\delta^* + \varepsilon = (\gamma + \sum_k \langle \alpha_k \rangle_{I_k} \beta_k)\delta^* + \varepsilon = \gamma\delta^* + \sum_k \langle \alpha_k \rangle_{I_k} \beta_k \delta^*.$$

which is already in the required form (3). The case when δ is in the form (3) is considered similarly.

It is easy to calculate $\circ$-composition with terms of (2) or (3):

- If γ is dull then $\delta \circ \gamma = \delta\gamma$;
- $\delta \circ \langle \alpha \rangle_I \beta = \langle \delta\alpha \rangle_I \beta$ if $\alpha \not\ni \varepsilon$;
- $\delta\varepsilon = \delta$.

Proposition 2 is now immediate.

We illustrate Proposition 2 by the following example.

Example 3. Let us eliminate $\circ$ from $\delta = \langle d \rangle_3 \circ (\langle ab \rangle_8 c)^*$. First transform the second term to the prefix form: $(\langle ab \rangle_8 c)^* = \langle ab \rangle_8 c(\langle ab \rangle_8 c)^* + \varepsilon$, and second calculate $\delta = \langle \langle d \rangle_3 ab \rangle_8 c(\langle ab \rangle_8 c)^* + \langle d \rangle_3$.

We can introduce the following analogue of Kleene star for $\circ$-composition.

Definition 4. $L^{\circledast} = \varepsilon \cup L \cup L \circ L \cup L \circ L \circ L \cup \ldots$

This operation can also be eliminated for regular languages. However this result is less straightforward.

Proposition 3 ((Circled star elimination)). *If L is regular, then $L^{\circledast}$ is regular. The regular expression for it can be obtained algorithmically.*

Notice that this is easy for terms of (2). In fact, if γ is dull then $\gamma^{\circledast} = \gamma^*$, and

$$(\langle \alpha \rangle_I \beta)^{\circledast} = \langle \langle \alpha \rangle_I (\beta\alpha)^* \rangle_I \beta + \varepsilon.$$

The general case is more difficult. We give only a sketch of proof. First of all, transform the expression to the prefix form (2). Let $0 = \tau_0 < \tau_1 < \tau_2 < \cdots < \tau_n = \infty$ be all the endpoints of intervals I_k. For each values of i and k either $(\tau_i, \tau_{i+1}) \subseteq I_k$ (in this case we say that α_k is *active* on (τ_i, τ_{i+1})), or $I_k \cap (\tau_i, \tau_{i+1}) = \emptyset$. If α_k is *active*, it means that it is allowed to terminate anywhere inside the interval (τ_i, τ_{i+1}). Otherwise it is not allowed to terminate in (τ_i, τ_{i+1}). Let $A_i = \{k | \alpha_k \text{ active on } (\tau_i, \tau_{i+1})\}$. γ is allowed everywhere, and β_k should happen after the corresponding active α_k. For each i we define a regular

expression $\mathcal{A}_i = (\gamma + \sum_{k\in A_i} \alpha_k\beta_k)^*$. Its language contains concatenations of words, active on (τ_i, τ_{i+1}). Any word from $\mathcal{A}_i$ if it fits into (τ_i, τ_{i+1}) may occur during this time interval.

Let w be a signal from $L^{\circledast}$. It can be parsed as follows:

$$w = w_0\delta_0 w_1\delta_1 \dots w_m\delta_m, \tag{4}$$

where $m \le n$, τ_i occurs during w_i, shifts of w_i belong to some $\langle\alpha_k\rangle_{I_k}\beta_k$ or to γ and $\delta_i \in \mathcal{A}_i$. The idea behind this parsing is to see what happens at finitely many critical times τ_i and to allow any number of γ and $\alpha_k\beta_k$, where $k \in A_i$ to happen on the interval (τ_i, τ_{i+1}) (see Fig. 3).

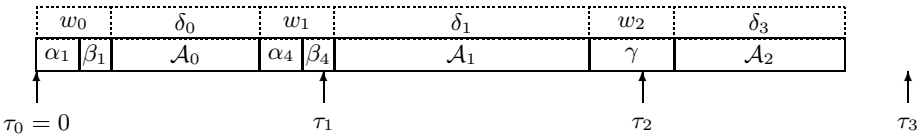

Fig. 3. Parsing a signal from $L^{\circledast}$

All these requirements can be written as a regular expression. For sake of simplicity we ignore the case when some w_i boundary is exactly at τ_i or if some w_i covers several consecutive τ_i.

For any τ_i find out what happens in w at τ_i, i.e to which term $\langle\alpha_k\rangle_{I_k}\beta_k$ or to γ belongs (the shift of) w_i. We also find when τ_i occurs: during α or β. All this information for all the τ_i forms the *pattern* of the word w. Notice that there are finitely many possible patterns. An example pattern P (corresponding to Fig. 3) is as follows: "α_3 at τ_0, β_4 at τ_1, γ at τ_2 and the signal is finished before τ_3" (for this pattern to be *valid*, α_3 and α_4 should be active at (τ_0, τ_1)). Now consider each pattern separately. For any pattern, using parsing (4) and expressions $\mathcal{A}_i$ we can write a regular expression which defines the set of all the words in $L^{\circledast}$ having this pattern. Instead of a heavy general formula consider only the expression corresponding to our sample pattern P:

$$\mathcal{E}_P = \langle\langle\langle\langle\alpha_1\beta_1\mathcal{A}_0\alpha_4\rangle_{(0,\tau_1)}\beta_4\rangle_{(\tau_1,\tau_2)}\mathcal{A}_1\rangle_{(\tau_1,\tau_2)}\gamma\rangle_{(\tau_2,\tau_3)}\mathcal{A}_2\rangle_{(\tau_2,\tau_3)}.$$

Last, to obtain the final regular expression we sum expressions $\mathcal{E}_P$ over all valid patterns P.

4 Quasilinear Equations

Definition 5. *A* system of quasilinear equations *has the following form:*

$$X_i = \alpha_i + \sum_{j=1}^{n} \beta_{ij}X_j + \sum_{j=1}^{n} \gamma_{ij} \circ X_j, \qquad i = 1, \dots, n, \tag{5}$$

where X_i stand for unknown timed languages and $\alpha_i, \beta_{ij}, \gamma_{ij}$ — for given regular coefficients.

We can now formulate the main result of the paper.

Theorem 1. *The minimal solution of a system of quasilinear equations is regular. Its regular expression can be obtained algorithmically from expressions for the coefficients.*

The rest of this section is devoted to the sketch of proof of this theorem and algorithm description. Without loss of generality suppose that β_{ij} do not contain the empty signal ε. Otherwise we can move this empty signal from β_{ij} to γ_{ij}.

The first thing to do is to separate unknowns to which concatenation is applied from those to which $\circ$ is applied. To achieve this aim we create another copy of each unknown.

Lemma 3. *The following system:*

$$\begin{cases} X_i = \alpha_i + \sum\limits_{j=1}^{n} \beta_{ij} Y_j + \sum\limits_{j=1}^{n} \gamma_{ij} \circ X_j, \\ Y_i = X_i \end{cases} \tag{6}$$

has the same solutions as the original system (5). Formally $X_1 = L_1, \ldots, X_n = L_n$ is a solution to (5) iff $X_1 = Y_1 = L_1, \ldots, X_n = Y_n = L_n$ is a solution to (6) and all the solutions to the latter system have this form.

The following lemma gives a solution to a single equation with only one operation. Its proof is fully similar to the proof of the same result for discrete equations.

Lemma 4. – *The minimal solution to $X = \alpha + \gamma \circ X$ is $X = \gamma^{\circledast} \circ \alpha$;*
– *The minimal solution to $Y = \alpha + \beta Y$ is $Y = \beta^* \alpha$;*

The algorithm of solving the system (6) is similar to the classical algorithm for discrete languages and consists in iterated application of Lemma 4 together with circle elimination from Section 3. At the first stage we begin with the first equation and express X_1 from it as

$$X_1 = \gamma_{11}^{\circledast} \circ (\alpha_1 + \sum_{j=1}^{n} \beta_{1j} Y_1 + \sum_{j=2}^{n} \gamma_{1j} \circ X_j).$$

Eliminating circles, this equation can be transformed to the form

$$X_1 = \alpha'_1 + \sum_{j=1}^{n} \beta'_{1j} Y_1 + \sum_{j=2}^{n} \gamma'_{1j} \circ X_j.$$

We put this expression into the second equation and solve it for X_2. And we continue till X_n for which we find an expression that contains only Ys and not X

unknowns. Then the second stage begins. We go backwards putting this expression for X_n into equation number $n-1$. This allows to find X_n-free expression for X_{n-1} and so on until we reach X_1 once again. Now the system has the form

$$\begin{cases} X_i = \alpha_i'' + \sum_{j=1}^{n} \beta_{ij}'' Y_j; \\ Y_i = X_i. \end{cases} \tag{7}$$

Replacing Y_i by X_i we obtain the $\circ$-free system

$$X_i = \alpha_i'' + \sum_{j=1}^{n} \beta_{ij}'' X_j;$$

and we express again

$$X_1 = \beta_{11}^{''*}(\alpha_1'' + \sum_{j=2}^{n} \beta_{1j}'' X_j),$$

put the result into the second equation, find X_2 and so on. This is the third stage of the algorithm. The fourth (and the last) stage consists in going backwards putting the regular expression for X_n into equation $n-1$ and so on. This ends up with finding regular expressions for all the X_i. This concludes the algorithm and the proof of Theorem 1.

5 Applying Equations to Timed Automata

First recall shortly the definition of timed automata and their languages.

Definition 6 ((Timed Automaton, [AD94])). *A timed automaton is a tuple $\mathfrak{A} = (Q, C, \Delta, \Sigma, S, F)$ where Q is a finite set of states, C is a finite state of clocks, Σ is an output alphabet, Δ is a transition relation (see below), $S \subseteq Q$ an initial set and $F \subseteq Q$ an accepting set. An element of the transition relation is of the form (q, ϕ, ρ, q', a) where q and q' are states, $a \in \Sigma$ -an output symbol, $\rho \subseteq C$ and ϕ (the transition guard) is a boolean combination of formulae of the form $(c \in I)$ for some clock c and some integer-bounded interval I.*

A *clock valuation* is a function $\mathbf{v} : C \to \mathbb{R}_+ \cup \{0\}$ (which is the same a vector $\mathbf{v} \in (\mathbb{R}_+ \cup \{0\})^{|C|}$). We denote the set of all clock valuations by $\mathcal{H}$. For a clock valuation $\mathbf{v}$ and a set $\rho \subseteq C$ we put for any clock variable $c \in C$

$$\mathrm{Reset}_\rho\, \mathbf{v}(c) = \begin{cases} 0 & \text{if } c \in \rho \\ \mathbf{v}(c) & \text{if } c \notin \rho \end{cases}$$

That is, Reset_ρ resets to zero all the clocks in ρ and leaves the other clocks unchanged. We use $\mathbf{1}$ to denote the unit vector $(1, \ldots, 1)$.

A *finite run* of the automaton is a sequence

$$(q_0, \mathbf{v}_0) \xrightarrow[t_1]{\delta_1} (q_1, \mathbf{v}_1) \xrightarrow[t_2]{\delta_2} \ldots \xrightarrow[t_n]{\delta_n} (q_n, \mathbf{v}_n),$$

where $q_i \in Q, \mathbf{v}_i \in \mathcal{H}, \delta_i \in \Delta, t_i \in \mathbb{R}_+$, and which satisfies the following conditions:

Time progress: $0 < t_1 < \ldots < t_n$ (for convenience we put $t_0 = 0$);
Succession: If $\delta_i = (q, \phi, \rho, q', a_i)$ then $q_{i-1} = q, q_i = q'$, the condition $\phi(\mathbf{v}_{i-1} + (t_i - t_{i-1})\mathbf{1})$ holds and $\mathbf{v}_i = Reset_\rho(\mathbf{v}_{i-1} + (t_i - t_{i-1})\mathbf{1})$.

An *accepting* run is a run satisfying the additional conditions:

Initialization: $q_0 \in S; \mathbf{v}_0 = \mathbf{0}$;
Termination: $q_n \in F$.

The *trace* of such a run is the signal

$$a_1^{t_1} a_2^{t_2 - t_1} \cdots a_n^{t_n - t_{n-1}},$$

whose length is t_n. The *language of a timed automaton*, $L(\mathfrak{A})$, consists of all the traces of its accepting runs.

Now recall the main result of [ACM97].

Theorem 2 ([ACM97]). *A timed language L can be accepted by a timed automaton of Alur and Dill iff it can be represented in the form*

$$L = \varphi\left(\bigcap_{i=1}^{n} L_i\right),$$

where L_i are regular languages and φ — a homomorphism.

(The terminology of [ACM97] is slightly different.)

The difficult direction is of course to find regular expressions for a given automaton. This operation in ([ACM97]) is split into 2 parts. The first one consists in reduction to one-clock automata.

Lemma 5 ([ACM97]). *Any timed language L accepted by a timed automaton can be represented in the form*

$$L = \varphi\left(\bigcap_{i=1}^{n} L_i\right),$$

where L_i are languages accepted by one-clock automata and φ — a homomorphism.

Our equation techniques is of no help here. However our result can simplify the proof of the second part.

Lemma 6 ([ACM97]). *Any timed language L accepted by a one-clock timed automaton is regular.*

Given a one-clock timed automaton it is easy to construct an equivalent system of quasilinear equations.

In order to do it, for any control state of the automaton q_i introduce an unknown X_i. For a transition from q_i to the accepting state with the label a and the guard $(c \in I)$ put $\alpha_i = \langle a \rangle_I$. For a transition from q_i to q_j with label a, guard $(c \in I)$ and no reset put $\gamma_{ij} = \langle a \rangle_I$. For a transition q_i to q_j with label a, guard $(c \in I)$ and reset $(c := 0)$ put $\beta_{ij} = \langle a \rangle_I$. Finally write the system of equations

$$X_i = \alpha_i + \sum_{j=1}^{n} \beta_{ij} X_j + \sum_{j=1}^{n} \gamma_{ij} \circ X_j, \qquad i = 1, \ldots, n, \tag{8}$$

of the form (5).

The quasilinear system obtained in such a straightforward way from the one-clock automaton can be solved using the algorithm of the previous section. The following lemma concludes the new proof of Lemma 6 and Theorem 2.

Lemma 7. *X_i in the minimal solution of equations (8) is the language accepted by the automaton from the state q_i with initial value of the clock $c = 0$.*

Example 4. Consider the second (timed) automaton on the Figure 1. According to the general construction corresponding quasilinear equations are like this:

$$\begin{cases} X_1 = & \langle a \rangle_5 X_2 + \langle b \rangle_2 \circ X_3 \\ X_2 = \langle b \rangle_{(7,\infty)} & + \langle a \rangle_{(0,10)} \circ X_3 \\ X_3 = \langle a \rangle_8 & \end{cases} \tag{9}$$

For the system 9 the procedure of section 4 gives the solution

$$\begin{cases} X_1 = \langle a \rangle_5 \langle b \rangle_{(7,\infty)} + \langle a \rangle_5 \langle \langle a \rangle_{(0,10)} a \rangle_8 + \langle \langle b \rangle_2 a \rangle_8 \\ X_2 = \langle b \rangle_{(7,\infty)} + \langle \langle a \rangle_{(0,10)} a \rangle_8 \\ X_3 = \langle a \rangle_8 \end{cases}$$

6 Conclusions and Further Work

In this paper a new linguistic formalism for timed languages is proposed. This formalism is adequate for timed automata. However there are still many questions to investigate.

- Which is the complexity of the algorithms?
- Is it possible to apply this approach directly to multi-clock timed automata?
- Which are other possible applications of this formalism? In particular, is it convenient for specification of timed systems?
- What can be done for more complicated equations?

References

ACM97. Eugene Asarin, Paul Caspi, and Oded Maler. A Kleene theorem for timed automata. In *Proc. 12th Annual IEEE Symposium on Logic in Computer Science*, pages 160–171, Warsaw, June 1997. IEEE Computer Society.

AD94. Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.

Brz62. Janusz A. Brzozowski. A survey of regular expressions and their applications. *IRE Trans. on Electronic Computers*, EC-11(3):324–335, 1962.

DOTY96. Conrado Daws, Alfredo Olivero, Stavros Tripakis, and Sergio Yovine. The tool KRONOS. In Rajeev Alur, Thomas A. Henzinger, and Eduardo D. Sontag, editors, *Hybrid Systems III, Verification and Control*, number 1066 in Lecture Notes in Computer Science, pages 208–219. Springer-Verlag, 1996.

Kle56. S.C. Kleene. Representations of events in nerve nets and finite automata. In R. McNaughton and H. Yamada, editors, *Automata Studies*, pages 3–42. Princeton University Press, 1956.