Journal of Automata, Languages and Combinatorics **7** (2002) 2, 167–186

A KLEENE/BÜCHI-LIKE THEOREM FOR CLOCK LANGUAGES [1]

Patricia Bouyer [2] and Antoine Petit [3]
LSV, CNRS UMR 8643, ENS de Cachan
61, Avenue du Prisident Wilson, F-94230 Cachan Cedex, France
e-mail: {bouyer,petit}@lsv.ens-cachan.fr

ABSTRACT

We propose in this paper a generalization of the famous Kleene/Büchi's theorem on formal languages, one of the cornerstones of theoretical computer science, to the timed model of clock languages. These languages extend the now classical timed languages introduced by Alur and Dill as a suitable model for real-time systems. As a corollary of our main result, we get a simple algebraic characterization of timed languages recognized by (updatable) timed automata.

Keywords: Kleene theorem, timed languages

1. Introduction

The notion of timed automata has been introduced in a seminal paper by Alur and Dill [3, 4], in order to provide a suitable model for timed systems. Since its introduction, this model has been largely studied under several aspects: determinization [5], definitions of interesting subclasses [23], minimization [1], power of clocks [2, 21], of ε-transitions [10], extensions of the model [6, 9, 19, 16] and logical characterization [28] have been considered in particular. Moreover this model has been successfully used for verification and specification of real-time systems [18, 29, 22, 26, 30, 20, 17].

But there is a lack of algebraic characterizations of timed languages or timed automata. Even the famous Kleene's theorem [25] and its extension to ω-languages by Büchi [15] has only partial counterparts in the framework of timed languages. An interesting result, due to Asarin, Caspi and Maler [8], proposes such a Kleene theorem for a slightly different notion of timed languages. But the main drawback of their characterization is that it uses in a crucial way the operations of intersection and renaming [24].

In [13], we also proposed a Kleene/Büchi theorem for timed automata but without the need of intersection and renaming. To this purpose we defined composition

[1] Full version of a submission presented at the workshop *Logic and Algebra for Concurrency* held at Dresden University of Technology (Germany), September 13 – 16, 2000.

[2] With the partial support of the French project RNRT "Calife"

[3] With the partial support of the French-India project CEPIPRA no. 2102-1

operations on timed automata, indexed by a subset of clocks, and the corresponding finite and infinite iterations. We then introduced the notion of constrained generator as semantics of a timed automaton in order to get our main result.

The present paper aims at giving a new presentation, much more simple and elegant, of the previous result which can be considered as one of the simplest algebraic characterization of timed automata. In fact, we even propose here a generalization of this characterization to updatable timed automata, an extension of timed automata, introduced and studied in [11, 12]. To this end, we extend the classical notion of timed languages to clock languages. We then define a natural concatenation between clock languages (note that this operation is unique: we do not need anymore to index it with a set of clocks or an update) and the corresponding finite and infinite iterations.

We thus define the notion of rational clock expressions. Even if the expressions have to be constructed from an infinite number of basic objects with the operations of union, concatenation, finite and infinite iterations, any rational clock expression is defined from a finite number of such objects. We therefore keep the spirit of rational expressions in formal language theory. Rational expressions lead naturally to the definition of rational clock languages.

The notion of recognizable clock languages is defined through updatable timed automata. We then prove that any rational clock language is recognizable. As in the formal language case, the key point is to show that recognizable clock languages are closed under the operations of union, concatenation, finite and infinite iterations. To this purpose, we prove that, essentially, the constructions on classical untimed finite automata can be also used in the framework of clock languages. We think that the simplicity of the technique illustrates the power and interest of clock languages.

To show that, conversely, any recognizable clock language is rational, we generalize one of the proof of Kleene's theorem based on the resolution of equations on set of words, see e. g. [14]. The idea to solve equations on timed languages – in the simple case of automata with a unique clock – was already proposed by Asarin in [7] who gave a new proof of the main result of [8]. Here, we solve roughly equations on clock languages and thus on arbitrary updatable timed automata.

We thus prove a Kleene/Büchi-like theorem for languages of finite and infinite clock words. We also show how this fundamental result can be interpreted in terms of timed languages, both in the general framework of updatable timed automata and on the restricted one of classical timed automata.

2. Clock Languages

2.1. Clocks, Guards and Updates

Let Z be any set. We write Z^* (resp. Z^ω) the set of finite (resp. infinite) sequences of elements in Z, and $Z^\infty = Z^* \cup Z^\omega$ the set of all sequences of elements in Z.

Throughout this paper, we consider a finite set of actions Σ and a time domain $\mathbb{T}$. This time domain can be either the set of non-negative integers $\mathbb{N}$, the set of non-negative rationals $\mathbb{Q}^+$ or the set of non-negative reals $\mathbb{R}^+$. A *time sequence* over $\mathbb{T}$ is a finite or infinite divergent non decreasing sequence $(t_i)_{i\geq 0}$ of elements in $\mathbb{T}$.

We also use a countable set of clocks $X = \{x_1, x_2, \dots\}$. A clock is simply a variable with values in $\mathbb{T}$. If n is an integer, the set of the first n clocks in X, $\{x_1, \dots, x_n\}$, is denoted by X_n.

The *guards* or *clock constraints* over X are the formulas defined by the grammar

$$g \ ::= \ \top \ | \ x \sim c \ | \ x - y \sim c \ | \ g \vee g \ | \ g \wedge g \ | \ \neg g \tag{1}$$

where x and y are clocks, $c \in \mathbb{Q}$ and $\sim$ is a binary operator in $\{<, \leq, =, \geq, >, \neq\}$. The set of all guards over X is denoted by $\texttt{Guards}(X)$. The subset of $\texttt{Guards}(X)$ corresponding to the guards using clocks in X_n only is denoted by $\texttt{Guards}(X_n)$.

Let Y be a subset of clocks, $Y \subseteq X$. A *clock valuation* on Y is a function $v : Y \to \mathbb{T}$ that assigns to each clock $y \in Y$ a time value $v(y) \in \mathbb{T}$. The set of all clock valuations on Y is denoted $\mathbb{T}^Y$. If t is some element in $\mathbb{T}$ and v a valuation, the valuation $v + t$ is defined by, for any clock x, $(v + t)(x) = v(x) + t$. By language misuse, we say that a tuple α of $\mathbb{T}^n$ is a valuation by merging it with the valuation $v \in \mathbb{T}^{X_n}$ defined by $v(x_i) = \alpha_i$ for each $1 \leq i \leq n$. We write $v \models g$ when the clock valuation v satisfies the clock constraint g.

An *update* up on a set of clocks $Y \subseteq X$ associates with each valuation on Y a set of valuations on Y ($up : \mathbb{T}^Y \longrightarrow 2^{\mathbb{T}^Y}$). The set of updates on the set Y is denoted by $\mathcal{U}p(Y)$.

Example 1 The clock valuation v on X_2 defined by $v(x_1) = 7.5$ and $v(x_2) = 5.6$ verifies the clock constraint $x_1 > 5 \wedge x_2 < 8 \wedge x_1 - x_2 \neq 2$.

Consider the update up on X_2 defined by

$$up : v \longmapsto \{v' \mid v'(x_1) \geq 4 \text{ and } v'(x_2) = v(x_1) + 1\}.$$

The valuation $(4.3, 2.2)$ is then in $up((1.2, 0.5))$.

A *reset update* is an update up such that for each valuation v, $up(v) = \{v'\}$ where for each clock x, either $v'(x) = v(x)$ or $v'(x) = 0$.

2.2. Clock Words

Let n be an integer, a *n-clock word* is an element

$$w = (t_0, \tau_0)(a_1, t_1, \tau_1)(a_2, t_2, \tau_2) \dots \in (\mathbb{T} \times \mathbb{T}^n) \times (\Sigma \times \mathbb{T} \times \mathbb{T}^n)^\infty$$

such that $t_0, t_1, t_2, \dots$ is a time sequence over $\mathbb{T}$. Hence in such a clock word w, the sequence $(a_i)_i$ represents the actions which are performed (or the signals which are received) by the system. The dates $(t_i)_i$ correspond to the absolute times where these actions occur and finally the valuations $(\tau_i)_i$ denote the values of the clocks just after the computations of the actions (or the receipts of the signals).

The n-clock word w is said finite of length p (the length of w is then denoted by $|w|$) if $w \in (\mathbb{T} \times \mathbb{T}^n) \times (\Sigma \times \mathbb{T} \times \mathbb{T}^n)^p$, infinite if $w \in (\mathbb{T} \times \mathbb{T}^n) \times (\Sigma \times \mathbb{T} \times \mathbb{T}^n)^\omega$. A *clock word* is simply a n-clock word for some n. The set of all n-clock languages (i.e. of sets of n-clock words) is denoted by $\mathcal{C}l\mathcal{W}_n(\Sigma, \mathbb{T})$. We define also the set of all clock languages by

$$\mathcal{C}l\mathcal{W}(\Sigma, \mathbb{T}) = \bigcup_{n \geq 0} \mathcal{C}l\mathcal{W}_n(\Sigma, \mathbb{T}).$$

We denote the set of empty n-clock words by $\Gamma_n (= \mathbb{T} \times \mathbb{T}^n)$.

Two clock words can be concatenated if they use the same number of clocks and if the absolute time and the values of the clocks at the end of the first word coincide with those at the beginning of the second. Formally a finite n-clock word $w = (t_0, \tau_0)(a_1, t_1, \tau_1) \dots (a_p, t_p, \tau_p)$ is *compatible* with a (finite or infinite) n-clock word $w' = (t'_0, \tau'_0)(a'_1, t'_1, \tau'_1)(a'_2, t'_2, \tau'_2) \dots$ whenever $(t'_0, \tau'_0) = (t_p, \tau_p)$.

The *concatenation* of w and w', denoted by $w \bullet w'$, is then the n-clock word $(t''_0, \tau''_0)(a''_i, t''_i, \tau''_i)_{i \geq 1}$ where

- $(t''_0, \tau''_0) = (t_0, \tau_0)$,
- for $1 \leq i \leq p$, $a''_i = a_i$ and $(t''_i, \tau''_i) = (t_i, \tau_i)$,
- for $i > p$, $a''_i = a'_{i-p}$ and $(t''_i, \tau''_i) = (t'_{i-p}, \tau'_{i-p})$.

The following proposition on the associativity of the concatenation follows directly from the definition above.

Proposition 1 *Let w, w' and w'' be three n-clock words such that w is compatible with w' and w' is compatible with w''. Then it holds*

- *$(w \bullet w')$ is compatible with w'',*
- *w is compatible with $(w' \bullet w'')$,*
- *$(w \bullet w') \bullet w'' = w \bullet (w' \bullet w'')$.* □

2.3. Operations on Clock Languages

The concatenation between clock words extend naturally in a classical way to clock languages. If L, L' are two n-clock languages, then we set

$$\begin{aligned} L \bullet L' = &\{w \bullet w' \mid w \in L,\ w' \in L' \text{ and } w \text{ is compatible with } w'\} \\ &\cup \{w \mid w \text{ is infinite and } w \in L\}. \end{aligned}$$

Thanks to Proposition 1, this concatenation is of course associative. Moreover, it holds that

$$\forall L \in \mathcal{ClW}_n(\Sigma, \mathbb{T}), \quad L \bullet \Gamma_n = \Gamma_n \bullet L = L.$$

From this concatenation between clock languages, we can now define finite and infinite iterations as follows.

$$\begin{aligned} L^* = \Gamma_n \ \cup \ &\{w_1 \bullet w_2 \bullet \ \cdots \ \bullet w_h \mid \forall 1 \leq i \leq h,\ w_i \in L \text{ and} \\ &\quad \forall 1 \leq i \leq h-1,\ w_i \text{ is compatible with } w_{i+1}\} \\ L^\omega = &\{w_1 \bullet w_2 \bullet \ \cdots \ \bullet w_h \mid \forall i \geq 1 \leq h,\ w_i \in L \text{ and} \\ &\quad \forall 1 \leq i \leq h-1,\ w_i \text{ is compatible with } w_{i+1} \text{ and} \\ &\quad w_h \text{ is an infinite clock word}\} \\ &\cup \{w_1 \bullet w_2 \bullet \ \cdots \ \mid \forall 1 \leq i,\ w_i \in L \setminus \Gamma_n \text{ and} \\ &\quad \forall 1 \leq i,\ w_i \text{ is compatible with } w_{i+1}\}. \end{aligned}$$

The following proposition summarizes some of the first immediate properties of these operations on clock languages. If L and L' are two n-clock languages, $L + L'$ denotes simply the union of these languages.

Proposition 2 *Let L, L' and L'' be three n-clock languages. Then it holds*

1. $L \bullet (L' + L'') = (L \bullet L') + (L \bullet L'')$,
2. $(L + L') \bullet L'' = (L \bullet L'') + (L' \bullet L'')$,
3. $(L \bullet L') \bullet L'' = L \bullet (L' \bullet L'')$,
4. $L^* = \bigcup_{i \geq 0} L^i$ *where* $L^0 = \Gamma_n$ *and* $\forall i \geq 1, L^{i+1} = L^i \bullet L$,
5. $L^* \bullet L' = \sum_{i \geq 0} \left(L^i \bullet L'\right)$.

2.4. A Fundamental Equation

The following result on equations on clock languages will play a crucial role in the sequel. It can be seen as an extension of Asarin's result on timed languages [7]. It is also very similar to the result proposed in [13] in the more technical framework of constrained generators of timed languages.

Lemma 3 *Let L_1 and L_2 be two n-clock languages such that $L_1 \cap \Gamma_n = \emptyset$. Then the equation on n-clock languages of* finite *clock words*

$$L = L_1 \bullet L + L_2 \tag{2}$$

has for unique solution the n-clock language $L_1^* \bullet L_2$.

Proof. First let us show that $K = L_1^* \bullet L_2$ is indeed a solution. Using Proposition 2, it holds

$$\begin{aligned} L_1 \bullet K + L_2 &= L_1 \bullet (L_1^* \bullet L_2) + L_2 \\ &= (L_1 \bullet L_1^*) \bullet L_2 + L_2 \\ &= (L_1 \bullet L_1^* + \Gamma_n) \bullet L_2 \\ &= L_1^* \bullet L_2 \\ &= K. \end{aligned}$$

Now let L be a solution, then $L_2 \subseteq L$. And thus $L_1 \bullet L_2 \subseteq L$ and by an immediate induction, for any $i \geq 0$, $L_1^i \bullet L_2 \subseteq L$. Therefore $\sum_{i \geq 0}(L_1^i \bullet L_2) \subseteq L$. Hence, using Proposition 2, we deduce that $L_1^* \bullet L_2 \subseteq L$. Hence, any solution contains K.

Conversely assume that L is a solution which contains strictly K and let w be a clock word in $L \setminus K$. Moreover, assume that the length of w is minimal among the words of $L \setminus K$. Two possible cases arise:

1. either w is in L_2: this is not possible since $L_2 \subseteq K$,
2. or w is in $L_1 \bullet L$. As w is a finite clock word, there exists $w_1 \in L_1$ and $w' \in L$ such that $w = w_1 \bullet w'$. If $w' \in K$, then $w \in L_1 \bullet K \subseteq K$ which contradicts the hypothesis. Otherwise we get $w' \in L \setminus K$ and, since $L_1 \cap \Gamma_n = \emptyset$, $|w'| < |w|$ which again contradicts the hypothesis on w.

Therefore $L = K$ and the equation (2) has a unique solution. □

3. Rationality

In the formal language framework, rational expressions are built from a finite number of basic objects the semantics of which are:

- $\emptyset$, the empty set,
- $\{\varepsilon\}$, the language restricted to the empty word,
- $\{a\}$, the language reduced to a single word of length 1.

Consider now the n-clock language

$$L_1 = \{(t_0, \tau_0)(a_1, t_1, \tau_1) \mid t_0, t_1 \in \mathbb{T},\ \tau_0, \tau_1 \in \mathbb{T}^n\}$$

and assume that we want some kind of "rational clock expression" defining L_1. If, as above, our basic objects correspond to languages reduced to a single word of length at most 1, L_1 has to be written as an uncountable union of languages defined by basic objects:

$$L_1 = \bigcup_{(t_0,\tau_0),(t_1,\tau_1)\in\mathbb{T}\times\mathbb{T}^n} \{(t_0, \tau_0)(a_1, t_1, \tau_1)\}$$

which is not very suitable. Indeed, the main idea of rational expressions is to use a finite number of basic objects to define, with the operations of union, concatenation and iterations, more complicated objects.

Following this idea, we propose to define rational clock expressions in such a way that, as in the formal language case, any given clock language is defined from a *finite* number of basic objects. Note that nevertheless, in order to get interesting classes of clock languages, the total number of basic objects has to be infinite.

3.1. Rational Clock Expressions

The set of *rational clock expressions* over the set of actions Σ and the time domain $\mathbb{T}$ is defined by the following grammar.

$$\begin{aligned} E ::= {} & \emptyset \\ & \mid \varepsilon_n \quad \text{with } n \in \mathbb{N} \\ & \mid \langle g, a, up\rangle \quad \text{with } a \in \Sigma,\ g \in \texttt{Guards}(X_n),\ up \in \mathcal{U}p(X_n) \text{ for some } n \in \mathbb{N} \\ & \mid E + E \\ & \mid E \bullet E \\ & \mid E^* \\ & \mid E^\omega \end{aligned}$$

The semantics of these rational clock expressions associates with each rational clock expression E a clock language $[\![E]\!]$ defined inductively in the following manner.

(i) *The trivial expressions*:

$$\begin{aligned} [\![\emptyset]\!] &= \emptyset, \\ [\![\varepsilon_n]\!] &= \Gamma_n = \mathbb{T} \times \mathbb{T}^n. \end{aligned}$$

(ii) *The basic rational clock expressions*:

$$\llbracket\langle g, a, up\rangle\rrbracket = \{(t,\tau)(a, t+t_0, \tau') \in (\mathbb{T}\times\mathbb{T}^n)\times(\Sigma\times\mathbb{T}\times\mathbb{T}^n) \mid \tau + t_0 \models g \text{ and } \tau' \in up(\tau+t_0)\}.$$

(iii) *Operations on rational clock expressions*:

$$\begin{array}{lll}
\llbracket E + E'\rrbracket = \llbracket E\rrbracket \cup \llbracket E'\rrbracket & & \text{(Union)},\\
\llbracket E \bullet E'\rrbracket = \llbracket E\rrbracket \bullet \llbracket E'\rrbracket & & \text{(Concatenation)},\\
\llbracket E^*\rrbracket = \llbracket E\rrbracket^* & & \text{(Finite iteration)},\\
\llbracket E^\omega\rrbracket = \llbracket E\rrbracket^\omega & & \text{(Infinite iteration)}.
\end{array}$$

3.2. Rational Clock Languages

A n-clock language $L \subseteq \mathcal{C}l\mathcal{W}_n(\Sigma,\mathbb{T})$ is said *rational* whenever there exists a rational clock expression E over X such that $L = \llbracket E\rrbracket$. The set of all rational n-clock languages is denoted by $\mathrm{Rat}[\mathcal{C}l\mathcal{W}_n(\Sigma,\mathbb{T})]$. We also define the set of rational clock languages by

$$\mathrm{Rat}[\mathcal{C}l\mathcal{W}(\Sigma,\mathbb{T})] = \cup_{n\geq 0}\mathrm{Rat}[\mathcal{C}l\mathcal{W}_n(\Sigma,\mathbb{T})].$$

4. Recognizability

In order to define a suitable notion of recognizable clock languages, we introduce now the notion of updatable timed automata. These automata, which are an extension of Alur and Dill's timed automata (see [3, 4]), have been proposed and studied in [11, 12].

4.1. Updatable Timed Automata

An *updatable timed automaton* over Σ and $\mathbb{T}$ is a tuple $\mathcal{A} = (Q, n, T, I, F, R)$, where

- Q is a finite set of states,
- $n \in \mathbb{N}$ is the number of clocks,
- $T \subseteq Q \times [\mathtt{Guards}(X_n) \times \Sigma \times \mathcal{U}p(X_n)] \times Q$ is the set of transitions,
- $I \subseteq Q$ is the subset of initial states,
- $F \subseteq Q$ is the subset of final states,
- $R \subseteq Q$ is a subset of repeated states.

Thus, a transition has the form (q, g, a, up, q'), where g is a guard, a is an action in Σ and up is an update.

Usually, the semantics of an (updatable) timed automaton is given by a timed language. We define here the semantics of an updatable timed automaton by a clock language. This idea, simple, will be proved very powerful in the following.

In order to define how a clock word is recognized by a timed automaton, we define the notions of path and of timed run through a path. A *path* in $\mathcal{A}$ is a non empty finite or infinite sequence of consecutive transitions:

$$P = q_0 \xrightarrow{g_1,a_1,up_1} q_1 \xrightarrow{g_2,a_2,up_2} q_2 \ \cdots$$

where $(q_{i-1}, g_i, a_i, up_i, q_i) \in T$ for all $i > 0$. The path P is accepting, if it starts in an initial state and either it is finite ending in a final state or it is infinite and a Büchi condition holds: $\mathit{inf}(P) \cap R$ is non empty, where $\mathit{inf}(P)$ is the set of states which occur infinitely often in P.

A *run* of the automaton through the path P and from the initial conditions $(t_0, v_0) \in \mathbb{T} \times \mathbb{T}^n$ is a sequence

$$R = \langle q_0, t_0, v_0\rangle \xrightarrow{g_1,a_1,up_1} \langle q_1, t_1, v_1\rangle \xrightarrow{g_2,a_2,up_2} \ \cdots$$

where $\forall i \geq 1, v_{i-1} + (t_i - t_{i-1}) \models g_i$ and $v_i \in up_i(v_{i-1} + (t_i - t_{i-1}))$. With every finite (resp. infinite) run R, we associate the finite (resp. infinite) clock word which is the label of R:

$$\ell(R) = (t_0, v_0)(a_1, t_1, v_1)(a_2, t_2, v_2) \cdots \in (\mathbb{T} \times \mathbb{T}^n) \times (\Sigma \times (\mathbb{T} \times \mathbb{T}^n))^\infty.$$

Note that in particular, if the path P reduces to a single state q_0, the finite clock word associated with a run through P and from the initial conditions $(t_0, v_0) \in \mathbb{T} \times \mathbb{T}^n$ is simply (t_0, v_0).

The clock language recognized by the updatable automaton $\mathcal{A} = (Q, n, T, I, F, R)$ is the set of all such labels :

$$\begin{aligned} L(\mathcal{A}) = \{\ell(R) \mid\ & R \text{ is a run through some accepting path } P \text{ of } \mathcal{A}, \\ & \text{from the initial conditions } (t_0, v_0)\}. \end{aligned}$$

The set of all updatable timed automata over the set of actions Σ, the time domain $\mathbb{T}$ and using n clocks is denoted by $\mathrm{UTA}_n(\Sigma, \mathbb{T})$. We also define the set of all updatable timed automata

$$\mathrm{UTA}(\Sigma, \mathbb{T}) = \bigcup_{n \geq 0} \mathrm{UTA}_n(\Sigma, \mathbb{T}).$$

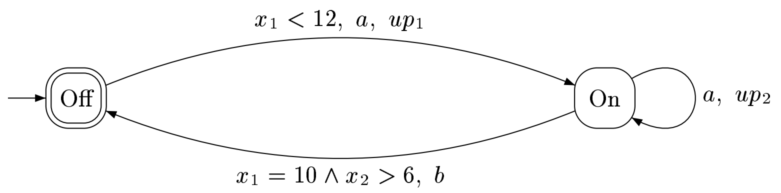
A n-clock language $L \in \mathcal{ClW}_n(\Sigma, \mathbb{T})$ is said *recognizable* whenever there exists an updatable timed automaton $\mathcal{A} \in \mathrm{UTA}_n(\Sigma, \mathbb{T})$ such that $L = L(\mathcal{A})$. The set of all recognizable n-clock languages is denoted by $\mathrm{Rec}[\mathcal{ClW}_n(\Sigma, \mathbb{T})]$. We denote the set of recognizable clock languages by

$$\mathrm{Rec}[\mathcal{ClW}(\Sigma, \mathbb{T})] = \bigcup_{n \geq 0} \mathrm{Rec}[\mathcal{ClW}_n(\Sigma, \mathbb{T})].$$

Example 2 Consider the updatable timed automaton of Figure 1.

A possible run of this updatable timed automaton is:

$$\begin{aligned} \langle \mathrm{Off}, 0, (0,0)\rangle & \xrightarrow{x_1<12,\ a,\ up_1} \langle \mathrm{On}, 3.1, (8, 6.1)\rangle \xrightarrow{a,\ up_2} \langle \mathrm{On}, 5.7, (9.1, 8.7)\rangle \\ & \xrightarrow{x_1=10 \wedge x_2>6,\ b} \langle \mathrm{Off}, 6.6, (10, 9.6)\rangle \ \cdots \end{aligned}$$



$$up_1(v) = \{v' \in \mathbb{T}^2 \mid 0 < v'(x_1) < 10 \text{ and } v'(x_2) = v(x_1) + 3\}$$
$$up_2(v) = \{v' \in \mathbb{T}^2 \mid v'(x_1) > v(x_2) - 2\}$$

Figure 1: An updatable timed automaton

The label of this run is the clock word

$$(0,(0,0))(a,3.1,(8,6.1))(a,5.7,(9.1,8.7))(b,6.6,(10,9.6))\ldots$$

4.2. Closure Properties

We study in this section the closure properties of recognizable clock languages under the operations on clock languages defined in Section 2.3. To this purpose, we prove that, essentially, the constructions on classical untimed finite automata can be also used in the framework of clock languages.

The case of union is immediate.

Proposition 4 *Let $\mathcal{A}$ and $\mathcal{B}$ be two updatable timed automata in $UTA_n(\Sigma, \mathbb{T})$. Then there exists an updatable timed automaton $\mathcal{A} + \mathcal{B} \in UTA_n(\Sigma, \mathbb{T})$ such that*

$$L(\mathcal{A} + \mathcal{B}) = L(\mathcal{A}) + L(\mathcal{B})$$

Proof. We assume that $\mathcal{A} = (Q_1, n, T_1, I_1, F_1, R_1)$ and $\mathcal{B} = (Q_2, n, T_2, I_2, F_2, R_2)$ have disjoint sets of states. Then $\mathcal{A} + \mathcal{B}$ is simply the updatable timed automaton defined by the tuple

$$(Q_1 \cup Q_2, n, T_1 \cup T_2, I_1 \cup I_2, F_1 \cup F_2, R_1 \cup R_2). \qquad \square$$

The case of concatenation is the key point where the power of clock languages appears clearly. We therefore treat this operation in details.

Proposition 5 *Let $\mathcal{A}$ and $\mathcal{B}$ be two updatable timed automata in $UTA_n(\Sigma, \mathbb{T})$. Then there exists an updatable timed automaton $\mathcal{A} \bullet \mathcal{B} \in UTA_n(\Sigma, \mathbb{T})$ such that*

$$L(\mathcal{A} \bullet \mathcal{B}) = L(\mathcal{A}) \bullet L(\mathcal{B}).$$

Proof. We assume that $\mathcal{A} = (Q_1, n, T_1, I_1, F_1, R_1)$ and $\mathcal{B} = (Q_2, n, T_2, I_2, F_2, R_2)$ have disjoint sets of states. The automaton $\mathcal{A} \bullet \mathcal{B}$ is defined as the updatable timed automaton

$$(Q_1 \cup Q_2, n, T_1 \cup T_2 \cup T, I, F, R_1 \cup R_2)$$

where (see Figure 2)

$$T = \{(q_1, g_1, a_1, up_1, i_2) \mid i_2 \in I_2 \text{ and } \exists f_1 \in F_1 \text{ s.t. } (q_1, g_1, a_1, up_1, f_1) \in T_1\},$$

$$F = F_2,$$

$$I = \begin{cases} I_1 & \text{if } I_1 \cap F_1 = \emptyset, \\ I_1 \cup I_2 & \text{if } I_1 \cap F_1 \neq \emptyset. \end{cases}$$

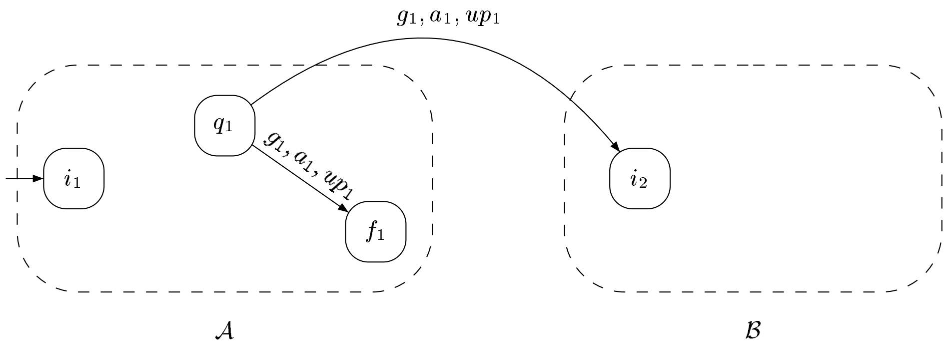


Figure 2: Construction of $\mathcal{A} \bullet \mathcal{B}$

We now prove successively the two inclusions between $L(\mathcal{A} \bullet \mathcal{B})$ and $L(\mathcal{A}) \bullet L(\mathcal{B})$.

First inclusion: $L(\mathcal{A}) \bullet L(\mathcal{B}) \subseteq L(\mathcal{A} \bullet \mathcal{B})$.

If w is an infinite clock word of $L(\mathcal{A})$, w is also in $L(\mathcal{A}) \bullet L(\mathcal{B})$ by definition of the concatenation. And from the definition of $\mathcal{A} \bullet \mathcal{B}$, it is immediate that $w \in L(\mathcal{A} \bullet \mathcal{B})$ too.

Now, let w be a finite clock word of $L(\mathcal{A})$ and w' be a clock word of $L(\mathcal{B})$ such that w is compatible with w'. Then

$$w = (t_0, \tau_0)(a_1, t_1, \tau_1) \dots (a_p, t_p, \tau_p)$$

and there exists some accepting path P in $\mathcal{A}$

$$P = q_0 \xrightarrow{g_1, a_1, up_1} q_1 \ \cdots \ \xrightarrow{g_p, a_p, up_p} q_p$$

and a run R through P and from the initial conditions (t_0, τ_0) such that $w = \ell(R)$.

In the same way, w' can be written as

$$w' = (t'_0, \tau'_0)(a'_1, t'_1, \tau'_1)(a'_2, t'_2, \tau'_2) \dots$$

and there exists some accepting path P' in $\mathcal{B}$

$$P' = q'_0 \xrightarrow{g'_1, a'_1, up'_1} q'_1 \xrightarrow{g'_2, a'_2, up'_2} q'_2 \ \cdots$$

and a run R' through P' and from the initial conditions (t'_0, τ'_0) such that $w' = \ell(R')$.

We distinguish now two cases:

Case 1: $w \notin \Gamma_n$.

By construction of the automaton $\mathcal{A} \bullet \mathcal{B}$, the sequence

$$P'' = q_0 \xrightarrow{g_1, a_1, up_1} q_1 \ \cdots \ q_{p-1} \xrightarrow{g_p, a_p, up_p} q'_0 \xrightarrow{g'_1, a'_1, up'_1} q'_1 \ \cdots$$

is an accepting path in $\mathcal{A} \bullet \mathcal{B}$. Moreover, from the compatibility of w and w', it holds $(t_p, \tau_p) = (t'_0, \tau'_0)$. Therefore,

$$R'' = \langle q_0, t_0, \tau_0 \rangle \xrightarrow{g_1, a_1, up_1} \cdots \langle q_{p-1}, t_{p-1}, \tau_{p-1} \rangle \xrightarrow{g_p, a_p, up_p} \langle q'_0, t'_0, \tau'_0 \rangle \xrightarrow{g'_1, a'_1, up'_1} \cdots$$

is a run of $\mathcal{A} \bullet \mathcal{B}$ through the path P'' and from the initial conditions (t_0, τ_0). Hence $\ell(R'') = w \bullet w' \in L(\mathcal{A} \bullet \mathcal{B})$.

Case 2: $w \in \Gamma_n$.

Hence it holds $I_1 \cap F_1 \neq \emptyset$ and thus, from the definition of the initial states in $\mathcal{A} \bullet \mathcal{B}$, P' is also an accepting path in $\mathcal{A} \bullet \mathcal{B}$. Therefore $\ell(R) = w' = w \bullet w' \in L(\mathcal{A} \bullet \mathcal{B})$.

Second inclusion: $L(\mathcal{A} \bullet \mathcal{B}) \subseteq L(\mathcal{A}) \bullet L(\mathcal{B})$.

Let $w'' \in L(\mathcal{A} \bullet \mathcal{B})$,

$$w" = (t''_0, \tau''_0)(a''_1, t''_1, \tau''_1)(a''_2, t''_2, \tau''_2) \ldots$$

and there exists some accepting path P'' in $\mathcal{A} \bullet \mathcal{B}$

$$P'' = q''_0 \xrightarrow{g''_1, a''_1, up''_1} q''_1 \xrightarrow{g''_2, a''_2, up''_2} q''_2 \ \cdots$$

and a run R'' through P'' and from the initial conditions (t''_0, τ''_0) such that $w'' = \ell(R'')$.

From the definition of $\mathcal{A} \bullet \mathcal{B}$, three cases arise.

Case 1: P'' contains states of Q_1, only.

Then P'' is an accepting path of $\mathcal{A}$ and $w'' \in L(\mathcal{A})$. As $F \cap F_1 = \emptyset$, w'' is infinite and hence $w'' \in L(\mathcal{A}) \bullet L(\mathcal{B})$ from the definition of the concatenation.

Case 2: P'' contains states of Q_2, only.

Then P'' is an accepting path of $\mathcal{B}$ and $I_1 \cap F_1 \neq \emptyset$. In particular, $(t''_0, \tau''_0) \in \Gamma_n \subseteq L(\mathcal{A})$ and hence $w'' = (t''_0, \tau''_0) \bullet w'' \in L(\mathcal{A}) \bullet L(\mathcal{B})$.

Case 3: P'' contains at least one state of Q_1 and one state of Q_2.

From the definition of T, this implies that there exists some integer p such that

$$P'' = q''_0 \xrightarrow{g''_1, a''_1, up''_1} q''_1 \ \cdots \xrightarrow{g''_p, a''_p, up''_p} q''_p \xrightarrow{g''_{p+1}, a''_{p+1}, up''_{p+1}} q''_{p+1} \ \cdots$$

with

$$P' = q''_p \xrightarrow{g''_{p+1}, a''_{p+1}, up''_{p+1}} q''_{p+1} \ \cdots$$

is a (finite or infinite) accepting path of $\mathcal{A}$ and there exists some state $f_1 \in F_1$ such that

$$P = q''_0 \xrightarrow{g''_1, a''_1, up''_1} q''_1 \ \cdots \xrightarrow{g''_p, a''_p, up''_p} f_1$$

is a finite accepting path of $\mathcal{B}$.

Now, since R'' is a run of $\mathcal{A} \bullet \mathcal{B}$ through P'' and from the initial conditions (t''_0, τ''_0), it is immediate that

$$R = \langle q''_0, t''_0, \tau''_0 \rangle \xrightarrow{g''_1, a''_1, up''_1} \cdots \xrightarrow{g''_p, a''_p, up''_p} \langle f_1, t''_p, \tau''_p \rangle$$

is a run of $\mathcal{A}$ through P and from the initial conditions (t''_0, τ''_0). In the same way,

$$R' = \langle q''_p, t''_p, \tau''_p \rangle \xrightarrow{g''_{p+1}, a''_{p+1}, up''_{p+1}} \langle q''_{p+1}, t''_{p+1}, \tau''_{p+1} \rangle \cdots$$

is a run of $\mathcal{B}$ through P' and from the initial conditions (t''_p, τ''_p).

Therefore $w = \ell(R) \in L(\mathcal{A})$ and $w' = \ell(R') \in L(\mathcal{B})$. Hence $w = w' \bullet w'' \in L(\mathcal{A}) \bullet L(\mathcal{B})$ and the proof is achieved. □

We can now treat more briefly the cases of finite and infinite iterations.

Proposition 6 *Let $\mathcal{A}$ and $\mathcal{B}$ be two updatable timed automata in $UTA_n(\Sigma, \mathbb{T})$. Then there exist*

- *an updatable timed automaton $\mathcal{A}^* \in UTA_n(\Sigma, \mathbb{T})$ such that*

  $$L(\mathcal{A}^*) = L(\mathcal{A})^*,$$

- *an updatable timed automaton $\mathcal{A}^\omega \in UTA_n(\Sigma, \mathbb{T})$ such that*

  $$L(\mathcal{A}^\omega) = L(\mathcal{A})^\omega.$$

Proof. Assume that $\mathcal{A} = (Q, n, T, I, F, R)$.

Finite iteration: The automaton $\mathcal{A}^*$ is defined as the updatable timed automaton

$$(Q \cup \{q_0\}, n, T \cup T' \cup T'', \{q_0\}, \{q_0\}, R)$$

where

q_0 is a new state not in Q,
$T' = \{(q_0, g, a, up, q) \mid q \in Q \text{ and } \exists i \in I \text{ s.t. } (i, g, a, up, q) \in T\}$,
$T'' = \{(q, g, a, up, q_0) \mid q \in Q \text{ and } \exists f \in F \text{ s.t. } (q, g, a, up, f) \in T\}$.

Infinite iteration: The automaton $\mathcal{A}^\omega$ is defined as the updatable timed automaton

$$(Q \cup \{q_0\}, n, T \cup T' \cup T'', \{q_0\}, \emptyset, R \cup \{q_0\})$$

where

q_0 is a new state not in Q,
$T' = \{(q_0, g, a, up, q) \mid q \in Q \text{ and } \exists i \in I \text{ s.t. } (i, g, a, up, q) \in T\}$,
$T'' = \{(q, g, a, up, q_0) \mid q \in Q \text{ and } \exists f \in F \text{ s.t. } (q, g, a, up, f) \in T\}$.

The proofs that $L(\mathcal{A}^*) = L(\mathcal{A})^*$ and $L(\mathcal{A}^\omega) = L(\mathcal{A})^\omega$ are technical and very similar to the proof of Proposition 5. They are therefore left to the reader. □

5. A Kleene/Büchi-like Theorem

In the formal language theory, Kleene's theorem [25] (and its extension to ω-languages by Büchi [15]) is considered as one of the cornerstones in theoretical computer science. It claims roughly that to specify a formal language through a rational expression or to recognize a formal language by a finite automaton are two equivalently powerful procedures. Note that this theoretical result is also the basis to more applied researches, for instance for verification purposes [27].

In this section, we generalize the Kleene/Büchi's theorem to clock languages, assessing thus the pertinence of our definitions of rational and recognizable clock languages. More precisely, we will prove the following result:

Proposition 7 *Let $n \in \mathbb{N}$. Then the set of rational n-clock languages coincide with the set of recognizable n-clock languages:*

$$Rat[\mathcal{ClW}_n(\Sigma, \mathbb{T})] = Rec[\mathcal{ClW}_n(\Sigma, \mathbb{T})].$$

As an immediate consequence, we get a Kleene/Büchi-like theorem for clock languages:

Theorem 8 *The set of rational clock languages coincide with the set of recognizable clock languages:*

$$Rat[\mathcal{ClW}(\Sigma, \mathbb{T})] = Rec[\mathcal{ClW}(\Sigma, \mathbb{T})].$$

Using the results presented in the previous sections, we will now prove the two inclusions of the Proposition (and the Theorem) just above.

5.1. Rational Clock Languages are Recognizable

We have already seen that the recognizable n-clock languages are closed under the operations of union, concatenation, finite and infinite iterations (see Propositions 4, 5 and 6). Now it is immediate to verify that the languages (see Section 3.1)

$$[\![\emptyset]\!], \qquad [\![\varepsilon_n]\!], \quad \text{and} \quad [\![\langle g, a, up\rangle]\!]$$

are recognizable n-clock languages. Indeed, they are respectively recognized by the updatable timed automata represented in Figures 3. Hence, we have proved that any rational clock language is recognizable.

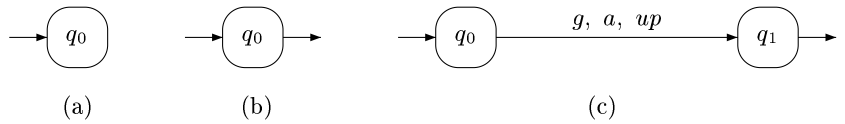


Figure 3: (a) Automaton $\mathcal{A}_\emptyset$, (b) Automaton $\mathcal{A}_{\varepsilon_n}$ and (c) Basic Automaton $\mathcal{A}_{\langle g,a,up\rangle}$

5.2. Recognizable Clock Languages are Rational

We will prove this result by extending one of the proof of Kleene's theorem for formal languages which use the resolution of equations on set of words (see e.g. [14]). The key point to solve equivalent equations on clock languages has already been given through Lemma 3.

So let L be a recognizable clock language. There exists an updatable timed automaton $\mathcal{A} = (Q, n, T, I, F, R)$ such that $L = L(\mathcal{A})$.

For any states q, $q' \in Q$, we set $\mathcal{A}_{q,q'} = (Q, n, T, \{q\}, \{q'\}, \emptyset)$ and we consider the associated clock language $L_{q,q'} = L(\mathcal{A}_{q,q'})$. From the definition of a run in an updatable timed automaton (see Section 4.1), the following equation holds.

$$L(\mathcal{A}) = \sum_{(i,f)\in I\times F} L_{i,f} + \sum_{(i,r)\in I\times R} L_{i,r} \bullet\ L^{\omega}_{r,r}. \tag{3}$$

Therefore, in order to find a rational clock expression for $L(\mathcal{A})$, it suffices to find rational clock expressions for $L_{q,q'}$ (for q, $q' \in Q$).

Assume now that $f \in Q$ is fixed, then the family of clock languages $(L_{q,f})_{q\in Q}$ verifies the system of equations on clock languages

$$L_{q,f} = \begin{cases} \displaystyle\sum_{(q,g,a,up,q')\in T} L(\mathcal{A}_{\langle g,a,up\rangle}) \bullet L_{q',f} + \Gamma_n & \text{if } q = f, \\ \displaystyle\sum_{(q,g,a,up,q')\in T} L(\mathcal{A}_{\langle g,a,up\rangle}) \bullet L_{q',f} & \text{otherwise.} \end{cases}$$

Consider now an arbitrary order $q_1 < q_2 < \cdots < q_n$ on the elements of Q. Then the equation with left member $L_{q_n,f}$ is solved, with $L_{q_n,f}$ as unknown, using the fundamental Lemma 3:

$$L_{q_n,f} = \left(\sum_{(q_n,g,a,up,q_n)\in T} [\![\langle g,a,up\rangle]\!] \right)^{*} \bullet \left(\sum_{\substack{(q_n,g,a,up,q')\in T \\ q'\neq q_n}} [\![\langle g,a,up\rangle]\!] \bullet L_{q',f}\ (+\Gamma_n) \right)$$

We thus replace $L_{q_n,f}$ by this formula in the $n-1$ other equations. Step by step, we solve the system using the fundamental Lemma 3 (at each step we are indeed in the conditions of application).

The last step proves that the clock language $L_{q_1,f}$ can be expressed using the elementary clock languages $[\![\varepsilon_n]\!]$ and $[\![\langle g,a,up\rangle]\!]$, and the composition operators.

We thus deduce that $L_{q_1,f}$ is a rational clock language and, by induction, so are $L_{q_2,f}, \ldots, L_{q_n,f}$.

Thus, Proposition 7 and Theorem 8 are therefore proved.

Example 3 Consider the updatable timed automaton $\mathcal{A}$ of Figure 1. We will compute a rational clock expression E such that $[\![E]\!] = L(\mathcal{A})$. We have that

$$L(\mathcal{A}) = L_{\mathrm{Off},\mathrm{Off}} \bullet L^{\omega}_{\mathrm{Off},\mathrm{Off}} = L^{\omega}_{\mathrm{Off},\mathrm{Off}}.$$

$\mathcal{A}$ defines the system of equations

$$\begin{aligned} L_{\text{Off,Off}} &= [\![\langle x_1 < 12, a, up_1\rangle]\!] \bullet L_{\text{On,Off}} + \Gamma_2, \\ L_{\text{On,Off}} &= [\![\langle a, up_2\rangle]\!] \bullet L_{\text{On,Off}} + [\![\langle x_1 = 10 \wedge x_2 > 6, b\rangle]\!] \bullet L_{\text{Off,Off}}. \end{aligned}$$

We thus get

$$\begin{aligned} L(\mathcal{A}) &= (([\![\langle x_1 < 12, a, up_1\rangle]\!] \bullet [\![\langle a, up_2\rangle]\!]^* \bullet [\![\langle x_1 = 10 \wedge x_2 > 6, b\rangle]\!])^*)^\omega \\ &= [\![(\langle x_1 < 12, a, up_1\rangle \bullet \langle a, up_2\rangle^* \bullet \langle x_1 = 10 \wedge x_2 > 6, b\rangle)^\omega]\!]. \end{aligned}$$

Hence a rational clock expression that describes $L(\mathcal{A})$ is

$$(\langle x_1 < 12, a, up_1\rangle \bullet \langle a, up_2\rangle^* \bullet \langle x_1 = 10 \wedge x_2 > 6, b\rangle)^\omega.$$

We must notice that the procedure to compute rational clock expressions is not more complicated than the one to compute rational expressions in formal language theory.

6. Applications

6.1. Restrictions on the Guards and on the Updates

For the while, we do not make any assumption on the set of guards (except that they are given by the grammar presented in Equation 1) nor on the set of updates. Nevertheless, a lot of work on timed automata consider restricted set of guards (for instance, diagonal-free guards, where the comparisons between the difference of two clocks and a constant are not allowed) or a restricted set of updates (for instance, Alur and Dill's timed automata use only reset updates and subclasses of updates for which emptiness is decidable have been proposed in [11]). Therefore it is worth to study the impact of such restrictions on our main result, Theorem 8. As we show below, we can handle without problem with such restrictions, keeping similar results. Once again, we think that this robustness shows the interest of our approach.

Precisely, let n be an integer, let $\mathcal{G}$ be a subset of $\mathtt{Guards}(X_n)$ and let $\mathcal{U}$ be a subset of $\mathcal{U}p(X_n)$. Hence we define the set of $(\mathcal{G},\mathcal{U})$-rational clock expressions as those defined by the following grammar:

$$\begin{aligned} E ::= {} & \emptyset \\ & \mid \varepsilon_n \text{ with } n \in \mathbb{N} \\ & \mid \langle g, a, up\rangle \text{ with } a \in \Sigma, g \in \mathcal{G}, up \in \mathcal{U} \\ & \mid E + E \\ & \mid E \bullet E \\ & \mid E^* \\ & \mid E^\omega \end{aligned}$$

We now say that a n-clock language L is $(\mathcal{G},\mathcal{U})$-rational if there exists some $(\mathcal{G},\mathcal{U})$-rational expression E such that $L = [\![E]\!]$. We denote by $\mathrm{Rat}[\mathcal{C}l\mathcal{W}(\Sigma, \mathbb{T}, \mathcal{G}, \mathcal{U})]$ the set of $(\mathcal{G},\mathcal{U})$-rational clock languages.

In the same spirit, we denote by $\mathrm{UTA}(\Sigma, \mathbb{T}, \mathcal{G}, \mathcal{U})$ the set of updatable timed automata $\mathcal{A}$ in which all the transitions belong to $Q \times \mathcal{G} \times \Sigma \times \mathcal{U} \times Q$ (where Q is the set of states of $\mathcal{A}$). A n-clock language L is said $(\mathcal{G}, \mathcal{U})$-recognizable if there exists an updatable timed automaton $\mathcal{A} \in \mathrm{UTA}(\Sigma, \mathbb{T}, \mathcal{G}, \mathcal{U})$ such that $L = L(\mathcal{A})$. The set of all $(\mathcal{G}, \mathcal{U})$-recognizable n-clock languages is denoted by $\mathrm{Rec}[\mathcal{C}l\mathcal{W}(\Sigma, \mathbb{T}, \mathcal{G}, \mathcal{U})]$.

It is now without difficulty to verify that the proof of Theorem 8 can be adapted to these families of languages in order to prove the following result.

Proposition 9 *Let $n \in \mathbb{N}$. Let $\mathcal{G} \subseteq \mathtt{Guards}(X_n)$ be a set of guards and let $\mathcal{U} \subseteq \mathcal{U}p(X_n)$ be a subset of updates. Then*

$$Rat[\mathcal{C}l\mathcal{W}(\Sigma, \mathbb{T}, \mathcal{G}, \mathcal{U})] = Rec[\mathcal{C}l\mathcal{W}(\Sigma, \mathbb{T}, \mathcal{G}, \mathcal{U})]. \qquad \square$$

In [11], we presented decidability results for subclasses of updatable timed automata. In that work, we restricted to updates on X_n of the form:

$$up \ ::= \ up_{x:\sim c} \ \mid \ up_{x:\sim y+c} \ \mid \ up \wedge up \tag{4}$$

where $x, y \in X_n$, $\sim \in \{<, \leq, =, \geq, >\}$ and $c \in \mathbb{Q}$. We denote by $\mathcal{U}p'(X_n)$ the set of updates on X_n defined by (4).

Let v, v' be valuations on X_n. The semantics of the updates in $\mathcal{U}p'(X_n)$ is

$$\begin{aligned}
v' \in up_{x:\sim c}(v) \quad &\iff v'(x) \sim c \text{ and for each } y \in X_n \setminus \{x\},\ v'(y) = v(y),\\
v' \in up_{x:\sim y+c}(v) \quad &\iff v'(x) \sim v(y) + c\\
&\qquad \text{and for each } y \in X_n \setminus \{x\},\ v'(y) = v(y),\\
v' \in (up_1 \wedge up_2)(v) \quad &\iff v' \in up_1(v) \cap up_2(v).
\end{aligned}$$

For example, if we consider the updatable timed automaton of Figure 1, the update up_1 is in fact $up_{x_1:>0} \wedge up_{x_1:<10} \wedge up_{x_2:=x_1+3}$.

We brought out classes of guards $\mathcal{G} \subseteq \mathtt{Guards}(X_n)$ and of updates $\mathcal{U} \subseteq \mathcal{U}p'(X_n)$ such that the emptiness problem for $\mathrm{UTA}(\Sigma, \mathbb{T}, \mathcal{G}, \mathcal{U})$ is decidable. Thus, as an immediate application of Proposition 9, we can test for emptiness of the rational clock languages in $\mathrm{Rat}[\mathcal{C}l\mathcal{W}(\Sigma, \mathbb{T}, \mathcal{G}, \mathcal{U})]$.

6.2. Timed Languages

Formally, timed languages as defined by Alur and Dill and studied in numerous papers since (see the Introduction) correspond simply to 0-clock languages. But the notions of rational and recognizable timed languages can not be defined as the corresponding classes of 0-clock languages. Indeed, a rational or recognizable 0-clock language does not carry any information about dates i. e. a timed word $(a_1, t_1)(a_2, t_2) \ldots$ is in such a language if and only if for any time sequence $(t'_i)_{i \geq 1}$, the timed word $(a_1, t'_1)(a_2, t'_2) \ldots$ is also in the language.

Hence we have to define rational and recognizable timed languages in a more elaborate way. The key point is that a timed language can also be defined as the projection of a n-clock language on its two first components. Formally, let us define the partial function

$$\Pi : \bigcup_{n\geq 0} [(\mathbb{T} \times \mathbb{T}^n) \times (\Sigma \times \mathbb{T} \times \mathbb{T}^n)^\infty] \longrightarrow (\Sigma \times \mathbb{T})^\infty$$

by

$$(t_0, \tau_0)(a_i, t_i, \tau_i)_{i\geq 1} \longmapsto \begin{cases} (a_i, t_i)_{i\geq 1} & \text{if } (t_0, \tau_0) = (0, 0^n), \\ \perp & \text{otherwise,} \end{cases}$$

where $\perp$ denotes the undefined value.

We can thus define a timed language (rational timed language, recognizable timed language resp.) as a language $\Pi(L)$ for some n-clock language (rational n-clock language, recognizable n-clock language resp.) L. If $\mathcal{A}$ is an updatable timed automaton, the timed language accepted by $\mathcal{A}$ is $\Pi(L(\mathcal{A}))$. As an immediate consequence of Theorem 8, we get:

Proposition 10 *The class of rational timed languages coincide with the class of recognizable timed languages.* □

Note that on the contrary to the result of [8], presented in the slightly different framework of signals, we do not need here the intersection to define rational timed languages. But note that we both need to use projections.

6.3. Timed Automata

We end this paper by showing how the result of the previous paper [13] about composition and decomposition of timed automata can be nicely presented, and generalized to updatable timed automata, in the framework of clock languages.

Let $\mathcal{A}$ and $\mathcal{B}$ be two updatable timed automata. We say that $\mathcal{A}$ and $\mathcal{B}$ are strongly equivalent (equivalent resp.) if they accept the same clock language (the same timed language resp.).

Now let Mod-UTA$(\Sigma, \mathbb{T})$ be the set of updatable timed automata that can be constructed in a modular way from the automata $\mathcal{A}_{\varepsilon_n}$ and $\mathcal{A}_{\langle g,a,up\rangle}$ (see Figures 3 (b) and 3 (c)) using the operations $+, \bullet, ^*, ^\omega$ on updatable timed automata defined in Propositions 4, 5 and 6. It holds obviously that Mod-UTA$(\Sigma, \mathbb{T}) \subseteq$ UTA$(\Sigma, \mathbb{T})$. The proof of Theorem 8 implies that a "converse" result can be obtained, if we consider the clock languages recognized by the automata.

Proposition 11 *For any updatable timed automaton $\mathcal{A} \in$ UTA$(\Sigma, \mathbb{T})$, there exists an updatable timed automaton $\mathcal{B} \in$ Mod-UTA$(\Sigma, \mathbb{T})$ such that $\mathcal{A}$ and $\mathcal{B}$ are strongly equivalent.* □

Since equivalence is obviously weaker than strong equivalence, we get in particular that any updatable timed automaton can be decomposed in a modular way into an automaton of Mod-UTA$(\Sigma, \mathbb{T})$ which accepts exactly the same timed language.

7. Conclusion

In this paper, we defined a new semantics of timed systems, called clock languages, which is much convenient for representing executions of (updatable) timed automata. Moreover, using this semantics, we presented a Kleene/Büchi-like theorem which is very close to the theorem in the framework of formal languages. This enforces the power and the interest of clock languages. More precisely we defined rational clock expressions from basic objects, union, concatenation and iterations that define exactly the clock languages accepted by some updatable timed automaton. Both transformations (from updatable timed automata to rational clock expressions and vice versa) can be computed similarly as in the theory of formal languages.

References

[1] R. Alur, C. Courcoubetis, D. Dill, N. Halbwachs, H. Wong-Toi, An Implementation of Three Algorithms for Timing Verification Based on Automata Emptiness. In: *Proc. 13th IEEE Real-Time Systems Symp.* (*RTSS '92*). IEEE Computer Society Press, 1992, 157–166.

[2] R. Alur, C. Courcoubetis, T. A. Henzinger, The Observational Power of Clocks. In: *Proc. 5th Int. Conf. Theory of Concurrency* (*CONCUR '94*). LNCS **836**, Springer-Verlag, 1994, 162–177.

[3] R. Alur, D. Dill, Automata for Modeling Real-Time Systems. In: *Proc. 17th Int. Coll. Automata, Languages, and Programming* (*ICALP '90*). LNCS **443**, Springer-Verlag, 1990, 322–335.

[4] R. Alur, D. Dill, A Theory of Timed Automata. *Theoretical Computer Science* **126** (1994) 2, 183–235.

[5] R. Alur, L. Fix, T. A. Henzinger, Event-Clock Automata: A Determinizable Class of Timed Automata. In: *Proc. 6th Int. Conf. Computer Aided Verification* (*CAV '94*). LNCS **818**, Springer-Verlag, 1994, 1–13.

[6] R. Alur, T. A. Henzinger, Back to the Future: Towards a Theory of Timed Regular Languages. In: *33rd Annual IEEE Symp. Foundations of Computer Science* (*FOCS '92*). IEEE Computer Society Press, 1992, 177–186.

[7] E. Asarin, Equations on Timed Languages. In: *Hybrid Systems: Computation and Control.* LNCS **1386**, Springer-Verlag, 1998, 1–12.

[8] E. Asarin, P. Caspi, O. Maler, A Kleene Theorem for Timed Automata. In: *Proc. 12th IEEE Symp. Logic in Computer Science* (*LICS '97*). IEEE Computer Society Press, 1997, 160–171.

[9] B. Bérard, Untiming Timed Languages. *Information Processing Letters* **55** (1995), 129–135.

[10] B. Bérard, V. Diekert, P. Gastin, A. Petit, Characterization of the Expressive Power of Silent Transitions in Timed Automata. *Fundamenta Informaticae* **36** (1998) 2, 145–182.

[11] P. BOUYER, C. DUFOURD, E. FLEURY, A. PETIT, Are Timed Automata Updatable? In: *Proc. 12th Int. Conf. Computer Aided Verification* (*CAV 2000*). LNCS **1855**, Springer-Verlag, 2000, 464–479.

[12] P. BOUYER, C. DUFOURD, E. FLEURY, A. PETIT, Expressiveness of Updatable Timed Automata. In: *Proc. 25th Int. Symp. Mathematical Foundations of Computer Science* (*MFCS 2000*). LNCS **1893**, Springer-Verlag, 2000, 232–242.

[13] P. BOUYER, A. PETIT, Decomposition and Composition of Timed Automata. In: *Proc. 26th Int. Coll. Automata, Languages, and Programming* (*ICALP '99*). LNCS **1644**, Springer-Verlag, 1999, 210–219.

[14] J. A. BRZOZOWSKI, A Survey of Regular Expressions and their Applications. *IRE transactions on Electronic Computers* **3** (1962), 324–335.

[15] J. R. BÜCHI, On a Decision Method in Restricted Second Order Arithmetic. In: *Proc. Int. Congr. Logic, Methodology and Philosophy of Science, 1960.* Stanford University Press, 1962, 1–11.

[16] C. CHOFFRUT, M. GOLDWURM, Timed Automata with Periodic Clock Constraints. *Journal of Automata, Languages and Combinatorics* **5** (2000) 4, 371–404.

[17] E. CLARKE, O. GRUMBERG, D. PELED, *Model Checking.* The MIT Press, Cambridge, Massachusetts, 1999.

[18] C. COURCOUBETIS, M. YANNAKAKIS, Minimum and Maximum Delay Problems in Real-Time Systems. In: *Proc. 3rd Int. Conf. Computer Aided Verification* (*CAV '91*). LNCS **575**, Springer-Verlag, 1991, 399–409.

[19] F. DEMICHELIS, W. ZIELONKA, Controlled Timed Automata. In: *Proc. 9th Int. Conf. Concurrency Theory* (*CONCUR '98*). LNCS **1466**, Springer-Verlag, 1998, 455–469.

[20] T. HENZINGER, P.-H. HO, H. WONG-TOI, HYTECH: A Model Checker for Hybrid Systems. *Journal of Software Tools for Technology Transfer* **1** (1997) 1/2, 110–122.

[21] T. A. HENZINGER, P. W. KOPKE, H. WONG-TOI, The Expressive Power of Clocks. In: *Proc. 22nd Int. Coll. Automata, Languages, and Programming* (*ICALP '95*). LNCS **944**, Springer-Verlag, 1995, 335–346.

[22] T. A. HENZINGER, X. NICOLLIN, J. SIFAKIS, S. YOVINE, Symbolic Model-Checking for Real-Time Systems. *Information and Computation* **111** (1994), 193–244.

[23] T. A. HENZINGER, J.-F. RASKIN, P.-Y. SCHOBBENS, The Regular Real-Time Languages. In: *Proc. 25th Int. Coll. Automata, Languages, and Programming* (*ICALP '98*). LNCS **1443**, Springer-Verlag, 1998, 580–591.

[24] P. HERRMANN, Renaming is Necessary in Timed Regular Expressions. In: *Proc. 19th Conf. Foundation of Software Technology and Theoretical Computer Science* (*FST&TCS '99*). LNCS **1738**, Springer-Verlag, 1999, 47–59.

[25] S. C. Kleene, Representation of Events in Nerve Nets and Finite Automata. In: *Automata Studies.* Annals of Math. Studies **34**, Princeton, 1956, 3–42.

[26] K. G. Larsen, P. Pettersson, W. Yi, Uppaal in a Nutshell. *Journal of Software Tools for Technology Transfer* **1** (1997) 1/2, 134–152.

[27] P. Schnoebelen, B. Bérard, M. Bidoit, F. Laroussinie, A. Petit, *Systems and Software Verification – Model-Checking Techniques and Tools.* Springer-Verlag, 2001.

[28] T. Wilke, Specifying Timed State Sequences in Powerful Decidable Logics and Timed Automata. In: *Proc. 3rd Int. Symp. Formal Techniques in Real-Time and Fault-Tolerant Systems* (*FTRTFT '94*). LNCS **863**, Springer-Verlag, 1994, 694–715.

[29] H. Wong-Toi, G. Hoffmann, The Control of Dense Real-Time Discrete Event Systems. In: *Proc. 30th IEEE Conf. Decision and Control.* 1991, 1527–1528.

[30] S. Yovine, KRONOS: A Verification Tool for Real-Time Systems. *Journal of Software Tools for Technology Transfer*, **1** (1997) 1/2, 123–133.

(Received: February 22, 2001)