Regular Expressions with Timed Dominoes: Extended Abstract

Cătălin Dima
ENSEIRB & LaBRI, 1, avenue du dr. Albert Schweitzer,
Domaine Universitaire - BP 99, 33402 Talence Cedex, France

Abstract. We give a class of timed regular expressions that involve the use of colored parentheses for specifying timing constraints. The expressions are given in a matricial format, and their semantics is based upon an "overlapping concatenation" of timed words. We then give a calculus for emptiness checking of a regular expression, that does not go through translating expressions into timed automata. To this end we use the class of $2n$-automata, studied in a parallel paper [Dim02] in connection with the problem of representing timing constraints.

1 Introduction

Timed automata [AD94] are a successful and widely used extension of finite automata for modeling real-time systems. They are finite automata enhanced with the possibility to record time passage, by means of real-valued clocks. They accept *timed words*, which are sequences of action symbols and real numbers, denoting time passage.

One of the main features in the theory of finite automata is their equivalence to *regular expressions*. Recently, several papers have addressed the problem of giving regular expressions for timed automata [ACM97,BP99], or for subclasses [Dim99] or superclasses [BP02] of timed automata. Timed regular expressions [ACM97,ACM02] are a very convenient specification language for timed systems. They are regular expressions enhanced with the possibility to express intervals between two moments, by the use of a new operator, the *time binding operator* $\langle E \rangle_I$, where I is an interval. Here, $\langle$ corresponds to resetting a clock and the beginning, while $\rangle_I$ corresponds to checking whether the clock value is in the interval I at the end of the behavior.

In spite of their elegance in use, timed regular expressions bear some expressiveness problems: intersection and renaming are essential when embedding timed automata into timed regular expressions. The first solution proposed was the use of "clocked regular expressions" [BP99,BP02], in which atoms contain clock constraint and clock reset informations. However this solution uses an "implementation" detail (clocks) at the "specification" level. Many of the specifications of real-time systems talk about intervals separating actions, which is more appropriate to natural language, their description using clocks is more targeted to verification purposes.

Another solution is the use of *colored parentheses* – one color for each clock – mentioned in [ACM02] and studied in part in [AD02]. The idea is that "interleaved" timing constraints, which are the main source of the need for intersection and renaming, are handled by parentheses of different colors. This solution has its problems too: the language of "well-balanced" colored parentheses language is context sensitive.

We investigate here a different approach, which can be seen as a combination of the "colored parentheses" approach and the "clocked" approach. We still use colored parentheses, but with an "overlapping concatenation", that allows timed words to concatenate only if they match on a common subword. These timed words with "distinguished subwords" can be also seen as n-dimensional dominoes of timed words, with one dimension for each color. Unlike Post dominoes, in our dominoes the components are "tied" to one another.

We represent these dominoes in a matricial format: an n-domino is represented by a $2n \times 2n$ matrix in which the $(i, n+1)$ entry holds the i-th component of the domino, while the (i, j) entry $(i, j \leq n)$ holds the part of the timed word that lies in between the beginning of the i-th component and the beginning of the j-th component of the domino; similarly for all the other combinations of i and j. Our atomic regular expressions are then matrices whose entries are timed regular expressions of the type $\langle E \rangle_I$, where E does not contain any other time binding operator.

The use of matrices is due to the following considerations: if we had opted for atoms containing n colored regular expressions of the kind $\langle^{color} E \rangle_I^{color}$, then some timing constraints would be hidden in such an atom. E.g., the fact that $\langle^{red}$ is always at the left of $\langle^{blue}$ would require a new pair of parentheses between $\langle^{red}$ and $\langle^{blue}$, with time label $[0, \infty[$. But what should be the color of this pair of parentheses? We rather notice that the timings between the $2n$ parentheses can be represented by a $2n \times 2n$ *difference bound matrix* (DBM) [Bel57], in which the (i, j) entry is the timing constraint between $\langle^i$ and $\langle^j$ (i, j being colors), while the $(i, n+i)$ entry is I, the label of $\rangle_I^i$. In [Dim02], we have presented an elegant calculus with unions, concatenations and star on DBMs. We generalize this for timed regular expressions, and the matrix presentation of atomic expressions is then the generalization of DBMs to timed languages.

We also give a method for checking whether the semantics of a given regular expression is empty, method that does not require the construction of a timed automaton. The method uses n-*automata*, a concept introduced in [Dim02] for the representation of DBMs. Roughly, our idea is to represent each set of timed language with a set of "discretized" timed words, that approximate each non-integer duration d to either $\lfloor d \rfloor$ or $\lceil d \rceil$. The discretizations also carry some information regarding the "direction of approximation" of each timed word, which assures exact representation of timed languages accepted by timed automata. At the basis of this discretization lies a variant of the region equivalence [AD94] that timed automata induce on the n-dimensional space of clock values. Our method for emptiness checking is not compositional: the set of representations of the semantics of a concatenation of two regular expressions is smaller than the concatenation of the respective sets of representations. But one of the contributions of this paper is that we never obtain "false positive" results, and is based upon the fact that discretized timed words induce an equivalence relation on timed languages.

The paper is organized as follows: the second section reviews the notions of timed words and automata, their timed regular expressions, and presents intuitively the overlapping concatenation. The third section explores this overlapping concatenation on the semantic domain of matrices of timed words and our regular expressions that generate them. The fourth section reminds the use of n-automata for computing timing relations and adapts them for the untiming of our regular expressions. The fifth section presents the technique for checking regular expressions for emptiness. We end with a section with conclusions and comments.

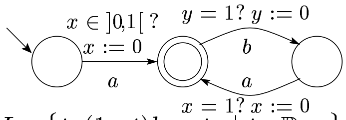
2 Timed automata and timed regular expressions

Behaviors of timed systems can be modeled by **timed words** over a set of symbols Σ. A timed word is a finite sequence of nonnegative numbers and symbols from Σ. For example, the sequence $1.2\,a\,1.3\,b$ denotes a behavior in which an action a occurs 1.2 time units after the beginning of the observation, and after another 1.3 time units action b occurs. The set of timed words over Σ can be organized as a monoid, and be represented as the *direct sum* $\mathbb{R}_{\geq 0} \oplus \Sigma^*$ [ACM02,Dim01]. We denote this monoid as $\mathsf{TW}(\Sigma)$. Note that in this monoid, concatenation of two reals amounts to summation of the reals. Thence, $a\,1.3 \cdot 1.7\,b = a(1.3+1.7)b = a\,3\,b$.

The *untiming* of a timed word σ, $\mathcal{U}(\sigma)$, is the sequence of actions in it, and the *length* of a timed word, $\ell(\sigma)$, is the sum of its delays. E.g., $\mathcal{U}(1.2\,a\,1.3\,b) = ab$ and $\ell(1.2\,a\,1.3\,b) = 2.5$. *Renamings* can be defined on timed words in the straightforward way: for example, $[a \mapsto b](a\,3\,b) = b\,3\,b$. Only action symbols can be renamed, not durations. For $m \leq n, m, n \in \mathbb{N}$, we denote by $[m \ldots n]$ the set $\{m, \ldots, n\}$.

A **timed automaton** [AD94] with n clocks is a tuple $\mathcal{A} = (Q, \mathcal{X}, \Sigma, \delta, Q_0, Q_f)$ where Q is a finite set of *states*, $\mathcal{X} = \{x_1, \ldots, x_n\}$ is the set of *clocks*, Σ is a finite set of *action symbols*, $Q_0, Q_f \subseteq Q$ are sets of *initial*, resp. *final* states, and δ is a finite set of *transitions* (q, C, a, X, r) where $q, r \in Q$, $X \subseteq [1 \ldots n]$, $a \in \Sigma \cup \{\varepsilon\}$ and C is a finite conjunction of *clock constraints* of the form $x \in I$, where $x \in \mathcal{X}$ and $I \subseteq [0, \infty[$ is an interval with integer (or infinite) bounds. For the formal definition of the semantics and the language of a timed automaton, the reader is referred to [AD94,Dim02].

An example of a timed automaton is given in the figure sideways. The accepted language is the language of timed words containing only as and bs, with the property that any two consecutive as are separated by one time unit, and similarly for bs:



$L = \{ta(1-t)b \ldots ta \mid t \in \mathbb{R}_{\geq 0}\}$

The class of **timed regular expressions** is built using the following grammar:

$$E ::= 0 \mid \varepsilon \mid \underline{\mathbf{t}}z \mid E + E \mid E \cdot E \mid E^* \mid \langle E \rangle_I \mid E \wedge E \mid [a \mapsto z]E, \qquad (1)$$

where $z \in \Sigma \cup \{\varepsilon\}$ and I is an interval. Their semantics is defined as follows:

$$\begin{array}{lll} \|\underline{\mathbf{t}}z\| = \{tz \mid t \in \mathbb{R}_{\geq 0}\} & \|E_1 + E_2\| = \|E_1\| \cup \|E_2\| & \|E^*\| = \|E\|^* \\ \|E_1 \cdot E_2\| = \|E_1\| \cdot \|E_2\| & \|[a \mapsto z]E\| = \{w[a \mapsto z] \mid w \in \|E\|\} & \|0\| = \emptyset \\ \|E_1 \wedge E_2\| = \|E_1\| \cap \|E_2\| & \|\langle E \rangle\|_I = \{\sigma \in \|E\| \mid \ell(\sigma) \in I\} & \|\varepsilon\| = \{\varepsilon\} \end{array}$$

The *untiming* of a timed regular expression E is the (classical) regular expression $\mathcal{U}(E)$ which is obtained by removing all symbols not in Σ. Note that $\|\mathcal{U}(E)\| = \mathcal{U}(\|E\|)$.

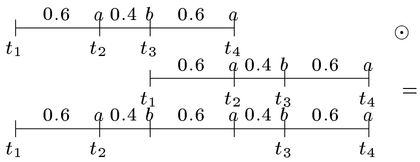
Theorem 1 ([ACM02]). *The class of timed languages accepted by timed automata equals the class of timed languages accepted by timed regular expressions.*

As an example, the timed regular expression $\langle \underline{\mathbf{t}}a \rangle_{]0,1[} \langle \underline{\mathbf{t}}b\underline{\mathbf{t}}a \rangle_1^* \wedge \langle \underline{\mathbf{t}}a\underline{\mathbf{t}}b \rangle_1^* \underline{\mathbf{t}}a$ generates the language of the timed automaton shown in the above picture.

In [AD02] we have proposed to code this timed language with the following expression: $E_1 = \lfloor \underline{\mathbf{t}}a \rfloor_{]0,1[} + \langle \lfloor \underline{\mathbf{t}}a \rfloor_{]0,1[} \big(\lfloor \underline{\mathbf{t}}b \rangle_1 \langle \underline{\mathbf{t}}a \rfloor_1 \big)^* \lfloor \underline{\mathbf{t}}b \rangle_1 \underline{\mathbf{t}}a \rfloor_1$, in which $\langle$ and $\rangle$ are used for coding the constraints on clock y, while $\lfloor$ and $\rfloor$ are used for coding the constraints on clock x. Note that in the subexpression $\big(\lfloor \underline{\mathbf{t}} b\rangle_1 \langle \underline{\mathbf{t}} a\rfloor_1\big)$, the angular parentheses are not balanced. This fact creates some syntactic problems on checking whether some expression really represents a timed automaton, see [AD02] for details.

We will employ here an "overlapping" concatenation, along the following idea: consider the expression $E_2 = \lfloor \underline{\mathbf{t}} a \,\langle \underline{\mathbf{t}} b\rfloor_1 \lfloor \underline{\mathbf{t}} a\rangle_1 \langle \underline{\mathbf{t}} b\rfloor_1 \, \underline{\mathbf{t}} a\rangle_1$, which can be seen as "one iteration" of the automaton on the previous page. Its generated language will be represented by $\lfloor \underline{\mathbf{t}} a \,\langle \underline{\mathbf{t}} b\rfloor_1 \, \underline{\mathbf{t}} a\rangle_1 \odot \lfloor \underline{\mathbf{t}} a \,\langle \underline{\mathbf{t}} b\rfloor_1 \, \underline{\mathbf{t}} a\rangle_1$. Then, with a suitable definition of "overlapping star" $\circledast$, the expression E_1 could be rewritten as $\big(\lfloor \underline{\mathbf{t}} a \,\langle \underline{\mathbf{t}} b\rfloor_1 \, \underline{\mathbf{t}} a\rangle_1\big)^{\circledast}$.

At the semantic level, we would then need to work with timed words with "distinguished points". An example of the semantic concatenation that is needed is given here on the right:



3 Timed words with distinguished points = timed n-words

We will use a matricial presentation of "timed words with distinguished points", in which the (i, j)-th component in the matrix holds the timed word in between the i-th and the j-th distinguished point. When the j-th point is before the i-th point, we put as the (i, j)-component the *inverse* of the respective timed word – an object that we will call an *timed antiword*. E.g., the inverse of the timed word $1a2b$ is the timed antiword $b^{-1}(-2)a^{-1}(-1)$. The monoid of timed antiwords is simply the set $\mathsf{TW}^{-1}(\Sigma) = \mathbb{R}_{\le 0} \oplus (\Sigma^{-1})^*$, endowed with the same concatenation as $\mathsf{TW}(\Sigma)$. For example, any of the two factors in the above figure is represented by the following 4×4 matrix:

$$W = \begin{pmatrix} \varepsilon & 0.6\,a & 0.6\,a\,0.4\,b & 0.6\,a\,0.4\,b\,0.6\,a \\ a^{-1}(-0.6) & \varepsilon & 0.4\,b & 0.4\,b\,0.6\,a \\ b^{-1}(-0.4)a^{-1}(-0.6) & b^{-1}(-0.4) & \varepsilon & 0.6\,a \\ a^{-1}(-0.6)b^{-1}(-0.4)a^{-1}(-0.6) & a^{-1}(-0.6)b^{-1}(-0.4) & a^{-1}(-0.6) & \varepsilon \end{pmatrix}$$

Observe that, in this matrix, $W_{ij} \cdot W_{jk} = W_{ik}$ for all $i, j, k \in [1 \ldots 4]$.

Formally, we work in the monoid $\mathbb{R} \oplus (\Sigma \cup \Sigma^{-1})^*$, factored by the congruence generated by the identities $a \cdot a^{-1} = a^{-1} \cdot a = \varepsilon$. This structure is a *group*, denoted $\mathsf{BiTW}(\Sigma)$, in which the inverse of a timed word w is denoted w^{-1}. In $\mathsf{BiTW}(\Sigma)$, timed words and timed antiwords may mix. We do not have a natural interpretation of some "mixed" timed words like $2a(-3)b$, but their use is unavoidable since the set $\mathsf{TW}(\Sigma) \cup \mathsf{TW}^{-1}(\Sigma)$ is not closed under concatenation.

Definition 1. *A **timed** n**-domino** over Σ is a matrix $w = (w_{ij})_{i,j \in [1 \ldots n]}$ with $w_{ij} \in \mathsf{BiTW}(\Sigma)$, satisfying the **triangle identity**: for each $i, j, k \in [1 \ldots n]$, $w_{ij} \cdot w_{jk} = w_{ik}$. When $w_{ij} \in \mathsf{TW}(\Sigma) \cup \mathsf{TW}^{-1}(\Sigma)\ \forall i, j \in [1 \ldots n]$, we say that w is a **timed** n**-word**.*

We denote by $\mathcal{D}_n(\Sigma)$ the class of timed n-dominoes over Σ and by $\mathsf{TW}_n(\Sigma)$ the class of timed n-words over Σ. Sets of timed n-dominoes will be called **timed n-domino languages**, and sets of timed n-words will be called **timed n-word languages**.

We actually decompose concatenation into two operations: a *juxtaposition* operation that "fuses" two timed words with distinguished points along a certain subset of points, and a *projection* operation, that forgets the points that have "actively" participated to the concatenation. Translated to matrices, projection removes some lines and columns in a matrix, while juxtaposition "fuses" two matrices along a common submatrix.

The X**-projection** of $w \in \mathcal{D}_n(\Sigma)$, is the timed $card(X)$-domino $w\big|_X$ whose (i,j)-component is: $\left(w\big|_X\right)_{ij} = w_{l_X^{-1}(i) l_X^{-1}(j)}$ for all $i,j \in [1\dots card(X)]$, where $l_X : X \to [1\dots card(X)]$ is the bijection defined by $l_X(i) = card\{j \in X \mid j \leq i\}$.

Given $w \in \mathcal{D}_m(\Sigma)$, $w' \in \mathcal{D}_n(\Sigma)$, $p \leq \min(m,n)$, the p**-indexed juxtaposition** of w and w' is defined iff $w\big|_{[m-p+1\dots m]} = w'\big|_{[1\dots p]}$ and is the timed $(m+n-p)$-domino $w \,\square_p\, w'$ for which $\forall i,j \in [1\dots m]$, $\left(w \,\square_p\, w'\right)_{ij} = w_{ij}$, $\left(w \,\square_p\, w'\right)_{m-p+i,m-p+j} = w'_{i-m+p,j-m+p}$, $\left(w \,\square_p\, w'\right)_{i,m+p-j} = w_{ik} \cdot w'_{k-m+p,j-m+p}$, where $k \in [m-p\dots m]$.

The **concatenation** of $w, w' \in \mathcal{D}_{2n}(\Sigma)$, is $w \odot w' = \left(w \,\square_n\, w'\right)\big|_{[1\dots n]\cup[2n+1\dots 3n]}$.

Observe that in the definition of $\left(w \,\square_p\, w'\right)_{i,m+p-j}$, the assumption $w\big|_{[m-p+1\dots m]} = w'\big|_{[1\dots p]}$ assures that any choice of k gives the same result for $(w \,\square_p\, w')_{ij}$ when $i \in [1\dots m], j \in [m-p+1\dots m+n-p]$.

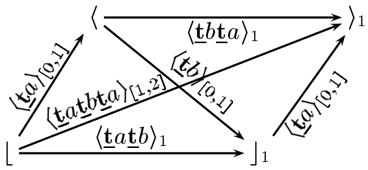
Proposition 1. *For each triplet of timed $2n$-dominoes $w, w', w'' \in \mathcal{D}_{2n}(\Sigma)$, $(w \odot w') \odot w''$ is defined iff $w \odot (w' \odot w'')$ is, and then $(w \odot w') \odot w'' = w \odot (w' \odot w'')$.*

For each $w \in \mathcal{D}_{2n}(\Sigma)$, consider the timed $2n$-dominoes $\mathbf{1}^l_w$, $\mathbf{1}^r_w$ in which, for each $i,j \in [1\dots n]$, $(\mathbf{1}^l_w)_{ij} = (\mathbf{1}^l_w)_{n+i,j} = (\mathbf{1}^l_w)_{i,n+j} = (\mathbf{1}^l_w)_{n+i,n+j} = w_{ij}$ and $(\mathbf{1}^r_w)_{ij} = (\mathbf{1}^r_w)_{n+i,j} = (\mathbf{1}^r_w)_{i,n+j} = (\mathbf{1}^r_w)_{n+i,n+j} = w_{n+i,n+j}$. Then $\mathbf{1}^l_w \odot w = w \odot \mathbf{1}^r_w = w$.

Concatenation can be extended to timed $2n$-domino languages: given two timed $2n$-domino languages $L, L' \subseteq \mathcal{D}_{2n}(\Sigma)$, we define $L \odot L' = \{w \odot w' \mid w \in L, w' \in L'\}$. The unit for concatenation on timed $2n$-word languages is $\mathbf{1}_{2n} = \big\{w \in \mathcal{D}_{2n}(\Sigma) \mid \forall i \in [1\dots n], w_{i,n+i} = \varepsilon\big\}$. The *star* is defined as follows: for each $L \subseteq \mathcal{D}_{2n}(\Sigma)$, $L^{\circledast} = \bigcup_{k\geq 0} L^{k\odot}$, where $L^{0\odot} = \mathbf{1}_{2n}$ and $L^{(k+1)\odot} = L^{k\odot} \odot L$ for all $k \in \mathbb{N}$. It is easy to see that the structure $\big(\mathcal{P}(\mathcal{D}_{2n}(\Sigma)), \cup, \odot, \circledast, \emptyset, \mathbf{1}_{2n}\big)$ is a Kleene algebra [Con71,Koz94].

3.1 Timed regminoes, timed regwords, and regular expressions over them

Observe that in the atom $\lfloor \underline{\mathbf{t}} a \langle \underline{\mathbf{t}} b \rfloor_1 \underline{\mathbf{t}} a \rangle_1$, there exist some implicit timing constraints limiting the duration of each state: neither of the time passages may last more than 1 time unit. A graphical presentation of the resulting object is enclosed here:



We put all this information into matricial presentation: the atoms of our calculus are matrices of *timed regular expressions*. As we have to specify anti-words, we employ *anti-timed regular expressions*, whose class is denoted $\mathsf{TREx}^{-1}(\Sigma)$. They are generated by the grammar:

$$E ::= 0 \mid \varepsilon \mid a^{-1}\underline{\mathbf{t}}^{-1} \mid E + E \mid E \cdot E \mid E^* \mid \langle E \rangle_I$$

When a(n) (anti-)timed regular expression does not utilize the parenthesis operator we say it is a(n) (anti-)untimed regular expression.

Definition 2. *A **timed** n**-regword** is a matrix $R = (R_{ij})_{i,j\in[1\dots n]}$ with $R_{ij} = \langle E + E' \rangle_I$ for some $E \in \mathsf{TREx}(\Sigma)$, $E' \in \mathsf{TREx}^{-1}(\Sigma)$ and some interval I with integer bounds.*

The set of timed n-regwords is denoted $\mathcal{TRW}_n(\Sigma)$. The **semantics** of timed n-regwords is the mapping $\|\cdot\| : \mathcal{TRW}_n(\Sigma) \to \mathcal{P}(\mathsf{TW}_n(\Sigma))$ defined as follows:

$$\|R\| = \{w \in \mathsf{TW}_n(\Sigma) \mid \forall i,j \in [1\dots n], w_{ij} \in \|R_{ij}\|\}, \text{ for each } R \in \mathcal{TRW}_n(\Sigma)$$

We denote by $C(R)$ the timed n-regword for which $C(R)_{ij}$ is R_{ij} without the parentheses, and $I(R)$ for the matrix of intervals used in R. That is, $R_{ij} = \langle C(R)_{ij} \rangle_{I(R)_{ij}}$.

Definition 3. *The class of* $2n$***-timed regular expressions****, denoted* $\mathsf{TREx}_{2n}(\Sigma)$*, is generated by the grammar:*

$$E ::= R \mid E + E \mid E \odot E \mid E^{\circledast} \text{ where } R \in \mathcal{TRW}_{2n}(\Sigma)$$

The semantics of $2n$-timed regular expressions is in terms of *timed* $2n$*-dominoes*, due to the non-closure of $\mathsf{TW}_{2n}(\Sigma)$ under concatenation, and is based on the operations on timed $2n$-words: $\|E+E'\| = \|E\| \cup \|E'\|$, $\|E \odot E'\| = \|E\| \odot \|E'\|$, $\|E^{\circledast}\| = \|E\|^{\circledast}$.

Theorem 2 ([Dim02]). *The emptiness problem for* $2n$*-timed regular expressions is undecidable.*

The next result, shows that timed automata with n clocks can be simulated by $2n$-timed regular expressions. To this end, we define the **language associated to** a $2n$-timed regular expression E as the set of "largest components" that occur in some timed $2n$-word w which belongs to the semantics of the given $2n$-timed regular expression:

$$L(E) = \{\sigma \in \mathsf{TW}(\Sigma) \mid \exists w \in \|E\|, i,j \in [1\dots 2n] \text{ s.t. } w_{ij} = \sigma, \text{ and for all } k \in [1\dots 2n], \ell(w_{ik}), \ell(w_{kj}) \geq 0\}$$

Theorem 3 ([Dim01]). *The class of languages accepted by timed automata with* n *clocks is included in the class of languages associated to* $2n$*-timed regular expression.*

4 Checking $2n$-timed regular expressions for emptiness

Our aim is to identify a class of $2n$-timed regular expressions whose emptiness problem is decidable. To this end, we represent each timed $2n$-regword by a special kind of automata, and identify some property that assures closure of this class of automata under union, concatenation and star. We utilize n*-automata*, introduced in [Dim02] for the computation of the reachability relation in timed automata.

The untiming of a timed n-word w is a matrix of elements from $\Sigma^* \cup (\Sigma^{-1})^*$ that satisfies the triangle identity $w_{ij} \cdot w_{jk} = w_{ik}$ $\forall i,j,k \in [1\dots n]$. We will call such a matrix as a n**-word**. The set of n-words over Σ is denoted $\mathcal{W}_n(\Sigma)$ Projection, juxtaposition and concatenation have the same definitions as for timed n-words – in fact, they might have been introduced for n-words first, and then derived for timed n-words.

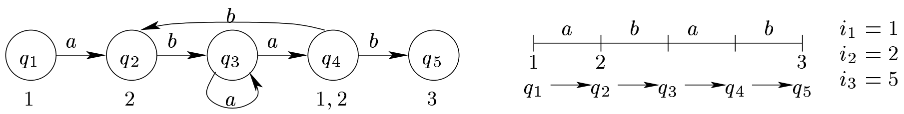
"Regular" sets of n-words can be generated with n**-regwords**, which are $n \times n$ matrices whose components are classical regular expressions. The class of n-regwords over Σ is denoted $\mathcal{RW}_n(\Sigma)$. We denote the semantics of a classical regular expression E as $\|E\|$ and the semantics of a n-regword R as $\|R\|$.

Definition 4. *An* n***-word automaton*** *with alphabet* Σ *is a tuple* $\mathcal{A} = (Q, \delta, Q_1, \dots, Q_n)$ *in which* Q *is a finite set of* states*,* $\delta \subseteq Q \times \Sigma \times Q$ *is a set of* transitions*, and for each* $i \in [1\dots n]$*,* $Q_i \subseteq Q$ *is a set of* accepting states for index i.

$\mathcal{A}$ accepts an n-word w in the following way: the automaton starts in an arbitrary state and tries to build a run that passes through all the sets $(Q_i)_{i \in [1 \dots n]}$ (perhaps several times through each set). For each $i, j \in [1 \dots n]$, the sequence of transitions between one of the moments when the run passes through Q_i and one of the moments when the run passes through Q_j be labeled with w_{ij}.

More formally, given a run $\rho = (q_j)_{j \in [1 \dots k]}$ (that is, $(q_i, q_{i+1}) \in \delta$ for all $i \in [1 \dots k-1]$), we call ρ as accepting iff we may choose a sequence of integers $\Delta = (j_i)_{i \in [1 \dots n]}$ with $j_i \in [1 \dots k]$, such that $q_{j_i} \in Q_i$ for all $i \in [1 \dots n]$. Or, in other words, ρ is accepting iff $\{q_1, \dots, q_k\} \cap Q_i \neq \emptyset$ for all $i \in [1 \dots n]$. We then say that the tuple (ρ, Δ) **accepts** a n-word w iff for all $i_1, i_2 \in [1 \dots n]$, if $i_1 \leq i_2$ then the piece of run between $q_{j_{i_1}}$ and $q_{j_{i_2}}$ is labeled by $w_{i_1 i_2}$. Note that if $i_2 \leq i_1$ then $w_{i_1 i_2}$ is an antiword and $w_{i_1 i_2}^{-1} = w_{i_2 i_1}$.

An example of a 3-automaton is provided in the figure below; there, the accepting sets are $Q_1 = \{q_1, q_4\}$, $Q_2 = \{q_2, q_4\}$ and $Q_3 = \{q_5\}$. We also present a 3-word accepted by this 3-automaton, together with its accepting run.



Proposition 2. *1. Emptiness checking for n-word automata is co-NP-complete.*
2. *The class of languages accepted by n-word automata equals the class of n-word languages which are the union of the semantics of a finite set of n-regwords.*
3. *The class of languages of n-word automata is closed under union, intersection, while the class of languages of $2n$-word automata is closed under concatenation.*

Definition 5. *A $2n$-word w is called **non-elastic** if for each $i, j \in [1 \dots n]$, if $w_{i,n+i} \neq \varepsilon$ and $w_{j,n+j} \neq \varepsilon$ then $w_{i,n+j} \in \Sigma^*$ and $w_{j,n+i} \in \Sigma^*$.*

Theorem 4. *Given a $2n$-automaton $\mathcal{A}$, suppose that, for any $k \in \mathbb{N}$, $L(\mathcal{A})^{k \odot}$ contains only non-elastic $2n$-words. Then $L(\mathcal{A})^{\circledast}$ is accepted by a $2n$-automaton.*

Representing pure timing information with n-automata [Dim02]. We sketch here the use of n-automata in representing the regular expressions over DBMs, that is, over the matrices of intervals which result by deleting all action symbols in a regular expression over regminoes. The idea is to decompose each interval into point intervals, or open unit intervals, and then to code each matrix composed only of such intervals with a n-word in $\mathcal{W}(\{1\})$. Or, in other words, to approximate each interval to an integer, and to represent integers in unary.

For example, the matrix $R = \begin{pmatrix} 0 &]2,3[&]1,2[\\ ]-3,-2[& 0 &]-1,0[\\ ]-2,-1[&]0,1[& 0 \end{pmatrix}$ is represented as $R' = \begin{pmatrix} \varepsilon & 111 & 11 \\ 1^{-1}1^{-1}1^{-1} & \varepsilon & 1^{-1} \\ 1^{-1}1^{-1} & 1 & \varepsilon \end{pmatrix}$. Note that each component of R' is one of the margins of the respective interval in R, but represented in unary. Note also that R' is a n-word, and that not all the combinations of margins of the intervals from R would be n-words We also need some information regarding the positioning of R' relative to R. This information is is given in the form of a *relational matrix* $M = \begin{pmatrix} `=` & `<` & `<` \\ `>` & `=` & `>` \\ `>` & `<` & `=` \end{pmatrix}$. The connection is the following: M_{12} is ' $<$ ' because $R_{12} =]2,3[< R'_{12} = 11 = 2$, and similarly for the other components of M. Observe that M satisfies a sort of "triangle identity" too: M_{ij}, M_{jk} and M_{ik} must not be "contradictory", like is the case with some M' in which $M'_{ij} =$ ' $<$ ', $M'_{jk} =$ ' $<'$ but $M'_{ik} =$ ' $>$ '.

Given a n-word R and a correct matrix of relation symbols M, the matrix of intervals which they code is denoted $[R, M]$. Its components are recovered as follows: if $M_{ij} =$ ' $<$ ', then $[R.M]_{ij} =]R_{ij} - 1, R_{ij}[$, and similarly for the other cases for M_{ij}.

5 Combining the timing and the untimed representations

By this point, we know how to analyze separately the timing information and the untimed information in a $2n$-timed regular expression. But studying them separately might prove an incomplete method, since we might miss interconnections which limit the behaviors. Our solution is the following: we decompose each timed $2n$-regword into the untimed and the timing part, then we recombine the $2n$-regword in the untimed part with the $2n$-regword over a one-letter alphabet from the $2n$-word representation of the timing part. The recombination we want is *shuffle* [BE97]. Remind that, for any $u, v \in \Sigma^*$, $u \sqcup\!\sqcup v = \{x_1y_1 \ldots x_ky_k \mid u = x_1 \ldots x_k, v = y_1 \ldots y_k, x_i, y_i \in \Omega^* \ \forall i \in [1 \ldots k]\}$.

Then the n-word lifting of this operation is the following: for each $w, w' \in \mathcal{W}_n(\Omega)$,

$$w \sqcup\!\sqcup w' = \{w'' \in \mathcal{W}_n(\Omega) \mid \forall i, j \in [1 \ldots n], w''_{ij} \in w_{ij} \sqcup\!\sqcup w'_{ij}\}$$

Observe that the shuffle of two n-words is more than the matrix lifting of shuffle: we do not only require that each component be shuffled, but the resulting matrix must also satisfy the triangle identity. Moreover, we do not allow shuffling w with w' if w_{ij} is a word and w'_{ij} is an antiword, because the resulting w''s would not be n-words.

For example, take the timed 2-regword $E = \begin{pmatrix} \varepsilon & \langle \underline{\mathbf{t}} a \underline{\mathbf{t}} \rangle_{]2,3[} \\ \langle \underline{\mathbf{t}}^{-1} a^{-1} \underline{\mathbf{t}}^{-1} \rangle_{]-3,-2[} & \varepsilon \end{pmatrix}$, which is the matricial representation of the timed regular expression $\langle \underline{\mathbf{t}} a \underline{\mathbf{t}} \rangle_{]2,3[}$. Its untiming is the 2-word $w = \begin{pmatrix} \varepsilon & \underline{\mathbf{t}} a \underline{\mathbf{t}} \\ \underline{\mathbf{t}}^{-1} a^{-1} \underline{\mathbf{t}}^{-1} & \varepsilon \end{pmatrix}$, while its timing is the matrix $R = \begin{pmatrix} \{0\} &]2,3[\\ ]-3,-2[& \{0\} \end{pmatrix}$, which is represented by the pair $T = \left(\begin{pmatrix} \varepsilon & 11 \\ 1^{-1}1^{-1} & \varepsilon \end{pmatrix}, \begin{pmatrix} `=` & `>` \\ `<` & `=` \end{pmatrix} \right)$. Then E can be represented by one of the shuffles of w and the 2-word in T, an example being the tuple $\left(\begin{pmatrix} \varepsilon & 1a1 \\ 1^{-1}a^{-1}1^{-1} & \varepsilon \end{pmatrix}, \begin{pmatrix} `=` & `>` \\ `<` & `=` \end{pmatrix} \right)$.

Formally, our representations are pairs (w, M) in which $w \in \mathcal{W}_n(\Sigma \cup \{1\})$ and M is a matrix of relation symbols. Such pairs are called **shuffled n-words**. The set of shuffled n-words is denoted $\mathsf{SW}_n(\Sigma)$. The *semantics* of a shuffled n-word (w, M) is the mapping $[\![\cdot]\!] : \mathsf{SW}_n(\Sigma) \to \mathsf{TW}_n(\Sigma)$ defined as follows:

$$[\![w, M]\!] = \{\sigma \in \mathsf{TW}(\Sigma) \mid \mathcal{U}(\sigma) = p_\Sigma(w) \text{ and } \ell(\sigma) \in [p_1(w), M]\}$$

Here, p_Σ denotes the deletion of 1 (or the projection onto Σ), while p_1 denotes the deletion of all symbols from Σ (the projection on $\{1\}$).

We may define projection and juxtaposition (and hence concatenation) on shuffled $2n$-words componentwise:

1. For $X \subseteq [1 \dots n]$, the X-projection of a shuffled n-word (w, M) is $(w\big|_X, M\big|_X)$.
2. The p-juxtaposition of a shuffled m-word (w, M) with a shuffled n-word (w', M') is $(w, M) \square_p (w', M') = \big\{ (w \square_p w', M'') \mid M'' \in M \square_p M' \big\}$.

Our strategy for checking emptiness of $2n$-timed regular expressions E is the following:

1. For each atom R of the given $2n$-timed regular expression E, we consider the untimed and the timing projection, denoted respectively $\mathcal{U}(R)$ and $t(R)$, and defined as follows: for all $i, j \in [1 \dots 2n]$, $\mathcal{U}(R)_{ij} = \mathcal{U}(R_{ij})$ (the untiming of R_{ij} as a timed regular expression), and $t(R)_{ij} = I(R_{ij})$ (the interval labeling R_{ij}).
2. We construct a $2n$-automaton $\mathcal{B}(R)$ for $\mathcal{U}(R)$.
3. We construct a $2n$-automaton for the set of $2n$-regions included in the timing projection $t(R)$.
4. We then produce the shuffle of (the language of) $\mathcal{B}(R)$ and $\mathcal{C}(R)$ into a $2n$-automaton $\mathcal{D}(R)$. Hence this automaton accepts shuffled $2n$-words (w, M) with the property that $p_\Sigma(w) \in L(\mathcal{B}(R))$ and $[p_1(w), M] \in L_r(\mathcal{C}(R))$. We will say then that (w, M) **represents** R.
5. With the resulting automata $\mathcal{D}(R)$ (one for each atom R), we compute unions, concatenations and stars, and get a $2n$-automaton for the whole expression E.
6. Finally, we check whether this resulting automaton has a nonempty language.

For this strategy to work, we need the following conditions to hold:

1. $2n$-automata must be closed under shuffle.
2. The representation of timed $2n$-regwords by shuffled $2n$-words must be faithful, that is, $\|R\|$ must be equal to the union of the semantics of all shuffled $2n$-words accepted by $\mathcal{D}(R)$.
3. The operations on shuffled $2n$-words might not be compositional w.r.t. semantics – this is quite expected since we lose some information (i.e., parentheses!) by concatenation. But we must assure that each shuffled $2n$-word has a nonempty semantics and if a concatenation $(w, M) = (w_1, M_1) \odot (w_2, M_2)$ is nonempty then we can pick up some timed $2n$-words in the semantics of each term such that their concatenation is in $[\![w, M]\!]$.
4. The requirements for star closure must be satisfied for each resulting $2n$-automaton.

The **first requirement** holds by a straightforward adaptation of the "shuffling" construction for finite automata – see [Dim01] for details.

For the **second requirement**, we may observe that our representation is *not faithful*: the timed 2-regword $R = \begin{pmatrix} \varepsilon & \langle \underline{\mathbf{t}} a \rangle_1 \\ \langle a^{-1} \underline{\mathbf{t}}^{-1} \rangle_{-1} & \varepsilon \end{pmatrix}$, which is nothing else but the matricial representation of the timed regular expression $\langle \underline{\mathbf{t}} a \rangle_1$, is represented, in our approach, by the shuffled 2-word $S = \left(\begin{pmatrix} \varepsilon & 1a \\ a^{-1}1^{-1} & \varepsilon \end{pmatrix} , \begin{pmatrix} `=` & `=` \\ `=` & `=` \end{pmatrix} \right)$. But $[\![S]\!] \neq \|R\|$, since, intuitively, it does not impose that the a action must be "bound" at the end of the behavior. E.g., $\sigma = \begin{pmatrix} \varepsilon & \alpha a (1-\alpha) \\ (\alpha-1) a^{-1} (-\alpha) & \varepsilon \end{pmatrix}$ belongs to $[\![S]\!]$ but not to $\|R\|$.

The only way we may "bind" an action to a certain point is to require that the respective action is happening in zero time just before (or after) that point. Following this idea, we replace R with the timed 3-regword $\tilde{R} = \begin{pmatrix} \varepsilon & \langle \underline{\mathbf{t}} \rangle_1 & \langle \underline{\mathbf{t}} a \rangle_1 \\ \langle \underline{\mathbf{t}}^{-1} \rangle_{-1} & \varepsilon & \langle \underline{\mathbf{t}} a \rangle_0 \\ \langle a^{-1} \underline{\mathbf{t}}^{-1} \rangle_{-1} & \langle a^{-1} \underline{\mathbf{t}}^{-1} \rangle_0 & \varepsilon \end{pmatrix}$, which is the matricial representation of the expression $\big\langle \langle \underline{\mathbf{t}} \rangle_1 \langle \underline{\mathbf{t}} a \rangle_0 \big\rangle_1$. Then, the semantics of each shuffled 3-word that represents $\tilde{R}$ contains timed 3-words in which the action a is "pinned at the end": for example, $\tilde{S} = \left(\begin{pmatrix} \varepsilon & 1 & 1a \\ a^{-1}1^{-1} & \varepsilon & a \\ a^{-1}1^{-1} & a^{-1} & \varepsilon \end{pmatrix}, \begin{pmatrix} `=` & `=` & `=` \\ `=` & `=` & `=` \\ `=` & `=` & `=` \end{pmatrix} \right)$ represents $\tilde{R}$, and for all $\sigma \in [\![\tilde{S}]\!]$, $\sigma_{23} = a$, $\sigma_{12} = 1$, therefore, by means of the triangle identity, $\sigma_{13} = 1a$, and hence $\sigma\big|_{\{1,3\}} \in \|R\|$.

For the general construction we employ the *left* and *right derivatives* of a regular expression [Eil74]. The inductive definition for the left derivative is the following: $\partial^l_a(a) = \varepsilon$, $\partial^l_a(b) = 0$ for $a \neq b$ $\partial^l_a(E + E') = \partial^l_a(E) + \partial^r_a(E')$ $\partial^l_a(E^*) = \partial^l_a(E) \cdot E^*$, $\partial^l_a(E \cdot E') = \partial^r_a(E) \cdot E' + o(E) \cdot \partial^r_a(E')$, where $o(E) = 1$ iff $\varepsilon \in |E|$, otherwise $o(E) = 0$. The mapping o is also inductively defined as follows: $o(\varepsilon) = \varepsilon$, $o(a) = 0$ and $o(E + E') = o(E) + o(E')$, $o(E \cdot E') = o(E) \cdot o(E')$, $o(E^*) = \varepsilon$. Right derivatives are defined similarly. For the special case of the time passage symbol we will slightly transform the derivative $\partial^l_{\underline{\mathbf{t}}}$ such that it does not remove $\underline{\mathbf{t}}$ from the expression. This amounts to changing the inductive definition for concatenation and star as follows: $\partial^l_{\underline{\mathbf{t}}}(E \cdot E') = \underline{\mathbf{t}} \cdot \partial^l_{\underline{\mathbf{t}}}(E) \cdot E' + o(E) \cdot \underline{\mathbf{t}} \cdot \partial^r_{\underline{\mathbf{t}}}(E')$, $\partial^l_{\underline{\mathbf{t}}}(E^*) = \underline{\mathbf{t}} \cdot \partial^l_{\underline{\mathbf{t}}}(E) \cdot E^*$.

Consider now a timed n-regword R and associate to it two sets $\Psi(R)$ and $\Upsilon(R)$:

$$\Psi(R) = \big\{ i \in [1 \ldots n] \mid \exists j \in [1 \ldots n], \exists a \in \Sigma \text{ s.t. } \|\partial^r_a(C(R))\| \neq \emptyset \big\}$$
$$\Upsilon(R) = \big\{ i \in [1 \ldots n] \mid \exists j \in [1 \ldots n] \text{ s.t. } \|\partial^r_{\underline{\mathbf{t}}}(C(R))\| \neq \emptyset \big\}$$

(remind that $C(R)_{ij}$ is R_{ij} from which only the parentheses are removed). These sets have the following interpretation: if $i \notin \Psi(R)$ then in any timed n-word $w \in \|R\|$, at the distinguished point t_i no action happens. Also, in this case $i \in \Upsilon(R)$.

Our construction is then the following: we transform each timed n-regword R into a sum of timed n-regwords. Each term is associated to a subset X with $\Psi(R) \setminus \Upsilon(R) \subseteq X \subseteq \Psi(R)$, and, informally, requires that for each $i \in X$ the distinguished time point t_i be right after an action, not after a time passage. Formally, for each such X we build the X**-augmented timed** $2n$**-regword** $R^X \in \mathcal{TRW}_{2n}(\Sigma)$ in which

- For all $i, j \in [1 \ldots n]$, $R^X_{n+i,n+j} = R_{ij}$ and if $i \in X$, $R^X_{i,n+i} = \big\langle \sum_{a \in \Sigma} \underline{\mathbf{t}} a \big\rangle_0$.
- For all $i \notin X$, $R^X_{i,n+i} = \big\langle \underline{\mathbf{t}} \big\rangle_{]0,\infty[}$. and for all $j \in [1 \ldots n], j \neq i$, $R^X_{ji} = \partial^r_{\underline{\mathbf{t}}}\big(C(R)_{ji}\big) + \partial^l_{\underline{\mathbf{t}}^{-1}}\big(C(R)_{ji}\big)$.
- For the remaining cases, $R^X_{ij} = \langle \Big(\sum_{a \in \Sigma} \underline{\mathbf{t}} a \Big)^* + \Big(\sum_{a \in \Sigma} a^{-1} \underline{\mathbf{t}}^{-1} \Big)^* \rangle_{]-\infty,\infty[}$.

Proposition 3. $\big\| \sum_{\Psi(R) \setminus \Upsilon(R) \subseteq X \subseteq \Psi(R)} R^X \big\| \big|_{[n+1 \ldots 2n] \cup [3n+1 \ldots 4n]} = \|R\|$, *and for each* $\Psi(R) \setminus \Upsilon(R) \subseteq X \subseteq \Psi(R)$, $\big\| R^X \big\| = \bigcup \big\{ [\![w, M]\!] \mid \exists v \in \|\mathcal{U}(R^X)\|, \exists [\omega, M] \subseteq \|t(R^X)\|$ *such that* $w \in v \sqcup\!\sqcup \omega \big\}$.

Hence, any timed n-regword R is faithfully represented by the set of shuffled $2n$-words which are associated to R^X, for all $\Psi(R) \setminus \Upsilon(R) \subseteq X \subseteq \Psi(R)$.

This construction can be then adapted to timed $2n$-regwords, but in each R^X we must permute each row and column i with $i \in [n+1 \dots 2n]$ with the row, resp. column $n+i$. Formally, we consider the bijection $\phi : [1 \dots 4n] \to [1 \dots 4n]$ defined as follows: for all $i \in [1 \dots n]$, $\phi(i) = i$, $\phi(n+i) = 2n+i$, $\phi(2n+i) = n+i$ and $\phi(3n+i) = 3n+i$. Then for each timed $4n$-regword R, $\phi(R)$ is the timed $4n$-regword whose components are $\phi(R)_{ij} = R_{\phi^{-1}(i),\phi^{-1}(j)}$. Then, for any two timed $2n$-regwords R and R',

$$\|R \odot R'\| = \bigcup \{\phi(R^X) \odot \phi({R'}^Y) \mid \Psi(R) \setminus \Upsilon(R) \subseteq X \subseteq \Psi(R), \\ \Psi(R') \setminus \Upsilon(R') \subseteq Y \subseteq \Psi(R')\}$$

The **third requirement** brings in some problems too: observe first that there exist shuffled n-words whose semantics is *empty* – these are the shuffled n-words (w, M) in which the timing for component (i, j) gives a negative interval, but the untiming gives a word, not an antiword, like the shuffled 2-word $(w, M) = \left(\begin{pmatrix} \varepsilon & a \\ a^{-1} & \varepsilon \end{pmatrix}, \begin{pmatrix} `{=}` & `{<}` \\ `{<}` & `{=}` \end{pmatrix}\right)$.

We define a class of shuffled $2n$-words whose repeated concatenation always yields shuffled $2n$-words with nonempty semantics. This is the class of **TA-shuffled $2n$-words**, which are shuffled $2n$-words $S = (w, M)$, with the following properties:

1. For each $i \in [1 \dots 2n]$, $w_{i,n+i}$ is not an antiword and for each $j \in [1 \dots n]$, if $p_1(w_{ij}) = \varepsilon$ (i.e., w_{ij} contains no 1s), then $M_{ij} = `{<}`$ iff w_{ij} is an antiword.
2. For each $i, j \in [1 \dots n]$, if $w_{i,n+i} = \varepsilon$ and $w_{j,n+j} \neq \varepsilon$ then $w_{n+i,n+j}$ is not an antiword.

Proposition 4. *For each timed $2n$-regword R associated to a transition in a timed automaton, any shuffled $2n$-word which represents R is a TA-shuffled $2n$-word. Moreover, for each $\Psi(R) \setminus \Upsilon(R) \subseteq X \subseteq \Psi(R)$, the X-augmented timed $4n$-regword R^X is represented only by TA-shuffled $4n$-words.*

Every TA-shuffled $2n$-word has a nonempty semantics. Moreover, the concatenation of any two TA-shuffled $2n$-words yields a TA-shuffled $2n$-word.

The other problem brought in by the third requirement is that *projection is not compositional*: $[\![w, M]\!]\big|_X \subseteq [\![w\big|_X, M\big|_X]\!]$, but the reverse may fail. Contrary to it, juxtaposition is compositional: suppose $\mathcal{S}_1 \subseteq \mathsf{SW}_m(\Sigma)$ has the property that, if $(w, M) \in \mathcal{S}_1$ and $[\![w, M]\!] = [\![w', M']\!]$ for some $(w', M') \in \mathsf{SW}_m(\Sigma)$, then $(w', M') \in \mathcal{S}_1$. Suppose also that $\mathcal{S}_2 \subseteq \mathsf{SW}_n(\Sigma)$ has the same property. Then for any $p \le \min(m, n)$,

$$\bigcup \{ [\![w, M]\!] \mid (w, M) \in \mathcal{S}_1 \square_p \mathcal{S}_2 \} = \bigcup \{ [\![w_1, M_1]\!] \square_p [\![w_2, M_2]\!] \mid (w_i, M_i) \in \mathcal{S}_i \}$$

The noncompositionality of projection implies that, in general, our technique of replacing $2n$-timed regular expressions with sets of shuffled $2n$-words is not faithful, and that at the end we may get a set of shuffled $2n$-words whose semantics is larger than the semantics of the original $2n$-timed regular expression. The question is whether we would not get then "false positive" answers to the emptiness problem. The following proposition shows this is not the case:

Proposition 5. *Given k shuffled $2n$-words $(w_1, M_1), \dots, (w_k, M_k)$, suppose that $w_1 \odot \dots \odot w_k$ is defined and $M_1 \odot \dots \odot M_k \neq \emptyset$. Then $[\![w_1, M_1]\!] \odot \dots \odot [\![w_k, M_k]\!] \neq \emptyset$.*

Hence, our approach on emptiness checking works this way: suppose that at the end of our process the $2n$-automaton for the whole expression E has a nonempty language. Then each shuffled $2n$-word in its language is issued by a concatenation of shuffled $2n$-words that represent certain timed $2n$-regwords in E. By the above proposition, we may trace a timed $2n$-word σ_i in the semantics of each shuffled $2n$-word, such that $\sigma_1 \odot \ldots \odot \sigma_k$ is defined. But this assures that when we concatenate the respective regwords of the $2n$-timed regular expression, we get at least the concatenation of the σ_is – which proves that E has a nonempty semantics.

Finally, the **fourth requirement** is assured by the following property:

Proposition 6. *Denote $\mathcal{L}$ the union of the semantics of all TA-shuffled $2n$-words. Then $\mathcal{L}$ is closed under concatenation and is composed only of non-elastic timed $2n$-words.*

6 Conclusions

We have given a calculus with regular expressions for timed languages, calculus based on the idea of using colored parentheses. The main result is the possibility to do emptiness checking without translating the expressions into timed automata.

It can be argued that our regular expressions are rather complicated and not easy to use. But we believe they may prove useful as an intermediary language, to which the timed regular expressions of [ACM02] might translate – and hence one might check for emptiness timed regular expressions without passing through timed automata.

References

[ACM97] E. Asarin, P. Caspi, and O. Maler. A Kleene theorem for timed automata. In *Proceedings of LICS'97*, pages 160–171, 1997.

[ACM02] E. Asarin, P. Caspi, and O. Maler. Timed regular expressions. *Journal of ACM*, 49:172–206, 2002.

[AD94] R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.

[AD02] E. Asarin and C. Dima. Balanced timed regular expressions. In *Proceedings of MTCS'02*, 2002. extended version to be published in *ENTCS* vol. 68.

[BE97] S. L. Bloom and Z. Ésik. Axiomatizing shuffle and concatenation in languages. *Information and Computation*, 139:62–91, 1997.

[Bel57] R. Bellmann. *Dynamic Programming*. Princeton University Press, 1957.

[BP99] P. Bouyer and A. Petit. Decomposition and composition of timed automata. In *Proceedings of ICALP'99*, volume 1644 of *LNCS*, pages 210–219, 1999.

[BP02] P. Bouyer and A. Petit. A Kleene/Büchi-like theorem for clock languages. *Journal of Automata, Languages and Combinatorics*, 2002. To appear.

[Con71] J. H. Conway. *Regular Algebra and Finite Machines*. Chapman and Hall, 1971.

[Dim99] C. Dima. Kleene theorems for event-clock automata. In *Proceedings of FCT'99*, volume 1684 of *LNCS*, pages 215–225, 1999.

[Dim01] C. Dima. *An algebraic theory of real-time formal languages*. PhD thesis, Université Joseph Fourier Grenoble, France, 2001.

[Dim02] C. Dima. Computing reachability relations in timed automata. In *Proceedings of LICS'02*, pages 177–186, 2002.

[Eil74] S. Eilenberg. *Automata, Languages, and Machines*, volume A. Academic Press, 1974.

[Koz94] D. Kozen. A completeness theorem for Kleene algebras and the algebra of regular events. *Information and Computation*, 110:366–390, 1994.