Distributed Time-Asynchronous Automata

Cătălin Dima[1] and Ruggero Lanotte[2]

[1] LACL, Université Paris 12, 61 av. du Général de Gaulle, 94010 Créteil Cedex, France
[2] Università dell'Insubria, Via Valleggio 11, 22100, Como, Italy

Abstract. We show that the class of distributed time-asynchronous automata is more expressive than timed automata, has a decidable emptiness problem, is closed under union, concatenation, star, shuffle and renaming, but not under intersection. The closure results are obtained by showing that distributed time-asynchronous automata are equivalent with a subclass of shuffle regular expressions and its related class of stopwatch automata.

1 Introduction

The theory of timed systems has reached a certain level of sofistication, proved by the various results in decidability for automata with dense timing [2,9,11], logical characterizations [8], regular expressions [4,7], monoidal characterizations [5]. However, as E. Asarin has noted in [3], the picture is still not that "nice" in the timed setting as it is in the untimed setting, and [3] states a number of challenges for enriching this picture, among which the first challenge is "to complete [...] a theory of timed systems and timed languages". The study of new classes of automata that have different expressive power than existing ones, though being decidable, can be a means to complete the timed languages picture.

We investigate here the class of *distributed time-asynchronous automata*, which are tuples of timed automata synchronized on input symbols and whose time-passage transitions are asynchronous (i.e. time is local to each automaton). Each automaton owns a set of clocks which only the owner can reset, but everyone may check the value of everyone's clocks. Synchronizations take place by jointly accepting an input symbol while testing global clock constraints (but note that we do not consider here distributed alphabets, in the sense of [13]). As we show, this class of automata has a decidable emptiness problem and are strictly more expressive than timed automata and incomparable with the rectangular automata of [9]. They are inspired from [10], being an intermediary step between the distributed timed automata and the interleaved timed automata of [10].

We investigate here the closure properties of this class, by comparing it with the class of *timed shuffle expressions*, which were suggested in [3] and studied in [7], where they were showed to be equivalent with stopwatch automata. Shuffle and stopwatches, as noted in [7], model preemptive scheduling, and their study would help understanding the benefits of using automata theory in solving scheduling problems [1].

Clearly, distributed time-asynchronous automata are strictly less expressive than stopwatch automata and timed shuffle expressions. However we may provide a subclass of shuffle expressions, called here *fair shuffle expressions* that are equivalent with distributed time-asynchronous automata. Fair shuffle expressions are only allowed to intersect with untimed regular expressions. This amounts to the fact that distributed timed automata are closed under union, concatenation, star, shuffle and renaming, but not under intersection. The equivalence result is proved by showing that our automata are equivalent with a subclass of stopwatch automata [9], called here *partitioned stopwatch automata* in which the set of stopwatches is partitioned into disjoint classes. Then, at each location, stopwatches in only one class are allowed to be active. The interesting fact about this equivalence is that global clock constraints suffice for simulating the centralized control in a partitioned stopwatch automaton – there is no need to be able to reset, in a component of a distributed time-asynchronous automaton, some clocks that are not owned by that component.

Finally, the result on nonclosure under intersection is proved by reducing the problem to checking that, when considering only *private clocksets*, that is, clock constraints in component i only refer to clocks owned by component i, then a certain language accepted by a (general) stopwatch automaton cannot be accepted by a distributed time-asynchronous automaton *modulo renaming*.

The paper goes on as follows: in the next section we introduce our class of distributed time-asynchronous automata and recall the definition of stopwatch automata. In the third section, we recall the timed shuffle expressions and introduce the fair shuffle expressions, then show the equivalence between distributed time-asynchronous automata, partitioned stopwatch automata and fair shuffle expressions. The fourth section shows the nonclosure under intersection of distributed time-asynchronous automata. We end with a section of conclusions.

2 Basic Notions

A *timed word*, also called *timed event sequence*, 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 length $\ell(w)$ of a timed word w is the sum of all the reals in it, e.g. $\ell(1.2\,a\,1.3\,b) = 1.2+1.3 = 2.5$. *Timed (event) languages* are then sets of timed words.

Several operations on timed words will be used in this paper. The first is concatenation, which extends the classic concatenation of untimed words by considering that concatenation of two reals amounts to summation of the reals. Hence, $a\,1.3{\cdot}1.7\,b\,c\,0.4 = a(1.3{+}1.7)b = a\,3\,b\,c\,0.4$. The second operation on timed words is *shuffle*, which is formally defined as follows: for each $w_1, w_2 \in \mathsf{TW}(\Sigma)$, $w_1 \sqcup\!\sqcup w_2 = \big\{u_1v_1 \dots u_nv_n \mid w_1 = u_1 \dots u_n, w_2 = v_1 \dots v_n\big\}$.

Concatenation and shuffle can be straightforwardly extended to languages, so, given $L_1, L_2 \subseteq \mathsf{TW}(\Sigma)$, we will denote $L_1 \cdot L_2 = \big\{w_1 \cdot w_2 \mid w_1 \in L_1, w_2 \in L_2\big\}$ and $L_1 \sqcup\!\sqcup L_2 = \bigcup \big\{w_1 \sqcup\!\sqcup w_2 \mid w_1 \in L_1, w_2 \in L_2\big\}$.

Another useful operation on timed languages is *renaming*: it replaces syntactically symbols with other symbols, while keeping durations the same. The renaming of $a \in \Sigma$ with $b \in \Sigma$ is denoted $[a/b]$. We also use *deletion*, which removes a symbol from a timed word, and consider it as a special case of renaming. The deletion of a symbol $a \in \Sigma$ is denoted $[a/\varepsilon]$. For example, $[a/c, b/\varepsilon]\big(1.3\, a\, 1.2\, b\, 0.1\, a\big) = 1.3\, c\, 1.3\, c$.

All our automata use nonnegative real-valued variables that are called *clocks* when used in timed automata, resp. *stopwatches* when used in stopwatch automata. The values of such variables may inhibit or allow taking some transition. Formally, transitions are enabled by *clock constraints* which are positive boolean combinations of elementary constraints of the type $x \in I$, with x being a (clock or stopwatch) variable and $I \subseteq \mathbb{R}_{\geq 0}$ an interval with bounds in $\mathbb{Z} \cup \{\infty\}$. The set of constraints with variables in a given set X is denoted $\mathsf{Constr}(X)$.

Given $v : X \to \mathbb{R}_{\geq 0}$ and $C \in \mathsf{Constr}(X)$, we denote as usual $v \models C$ if C holds when all occurrences of each $x \in X$ are replaced with $v(x)$. We also denote $v + t$ the valuation $(v + t) : X \to \mathbb{R}_{\geq 0}$ defined by $(v + t)(x) = v(x) + t$ for all $x \in X$. Further, for $Y \subseteq X$, we denote $v\big|_Y$ the valuation $v\big|_Y : Y \to \mathbb{R}_{\geq 0}$ defined by $v\big|_Y(x) = v(x)$ for all $x \in Y$.

A final notion to be employed is *clock* (or *stopwatch*) *reset*: given a valuation $v : X \to \mathbb{R}_{\geq 0}$ and a subset $Y \subseteq X$, we denote $v[Y := 0]$ the clock valuation defined by $v[Y := 0](x) = 0$ when $x \in Y$ and $v[Y := 0](x) = v(x)$ when $x \notin Y$.

2.1 Distributed Timed Automata

Definition 1. *A* ***distributed time-asynchronous automaton*** *is a tuple* $\mathcal{A} = (Q_1, \ldots, Q_n, \Sigma, X_1, \ldots, X_n, \delta_1, \ldots, \delta_n, q_1^0, \ldots, q_n^0, F_1, \ldots F_n)$ *where* Σ *is a finite set of* symbols*,* $X_1, \ldots, X_n$ *are* n *finite, pairwise-disjoint sets of* clocks*,* $Q_1, \ldots, Q_n$ *are* n *finite sets of* locations*,* $q_i^0 \in Q_i$ *are* initial locations*,* $F_i \subseteq Q_i$ *are subsets of* final *locations, and* $\delta_1, \ldots, \delta_n$ *are* transition relations*, with* $\delta_i \subseteq \big\{q \xrightarrow{C,a,Y} r \mid q, r \in Q_i, a \in \Sigma \cup \{\varepsilon\}, C \in \mathsf{Constr}(X), Y \subseteq X_i\big\}$.

A ***timed automaton*** *is a distributed time-asynchronous automaton with only one component.*

A ***distributed time-asynchronous automaton with private clocksets*** *is a distributed time-asynchronous automaton in which each component can test only clocks in* X_i*, i.e., for all* $1 \leq i \leq n$*, if* $q \xrightarrow{C,a,Y} q' \in \delta_i$ *then* $C \in \mathsf{Constr}(X_i)$.

We denote in the sequel $X = \bigcup_{1 \leq i \leq n} X_i$.

Intuitively, an distributed time-asynchronous automaton can make time-passage transitions in which clocks in different components evolve independently, and discrete transitions in which components may synchronize. Synchronizations take place by jointly accepting an input symbol while testing global clock constraints, and then reseting some clocks. This amounts to the fact that each component can see the value of every clock.

Formally, the semantics of a distributed time-asynchronous automaton is given as a *timed transition system* $\mathcal{T}(\mathcal{A}) = (\mathcal{Q}, \theta, \mathcal{Q}_0, \mathcal{Q}_f)$ where $\mathcal{Q} = Q_1 \times \ldots \times Q_n \times$

$\mathbb{R}^n_{\geq 0}$ represents the set of system states, $\mathcal{Q}_0 = \{(q_0^1, \ldots, q_0^n)\} \times \{\mathbf{0}_n\}$ is the initial state, $\mathcal{F} = F_1 \times \ldots \times F_n \times \mathbb{R}^n_{\geq 0}$ are the final states, while θ is the set of transitions:

$$\begin{aligned}\theta = &\big\{(q_1, \ldots, q_n, v) \xrightarrow{t} (q_1, \ldots, q_n, v') \mid \forall 1 \leq i \leq n, q_i \in Q_i \text{ and} \\ &\qquad \exists t_i \in \mathbb{R}_{\geq 0} \text{ with } v'\big|_{X_i} = v\big|_{X_i} + t_i \text{ and } t = t_1 + \ldots + t_n\big\} \\ &\cup \big\{(q_1, \ldots, q_n, v) \xrightarrow{a} (q_1', \ldots, q_n', v') \mid \forall 1 \leq i \leq n \; \exists C_i \in \mathsf{Constr}(X) \text{ with } v \models C_i \\ &\qquad \text{and } \exists Y_i \subseteq X_i \text{ with } q_i \xrightarrow{C_i, a, Y_i} q_i' \in \delta_i \text{ and } v' = v[Y_1 \cup \ldots \cup Y_n := 0]\big\} \\ &\cup \big\{(q_1, \ldots, q_n, v) \xrightarrow{\varepsilon} (q_1', \ldots, q_n', v') \mid \exists 1 \leq i \leq n \; \exists C_i \in \mathsf{Constr}(X), \text{ with } v \models C_i \\ &\qquad \text{and } \exists Y_i \subseteq X_i \text{ with } q_i \xrightarrow{C_i, \varepsilon, Y_i} q_i' \in \delta_i, v' = v[Y_i := 0] \text{ and } \forall j \neq i, q_j' = q_j\big\}\end{aligned}$$

Informally, distributed time-asynchronous automata can make time passage transitions or discrete transitions. In time passage transitions, all components evolve independently one of the other, the local times being incremented independently, which will increment the global time with the sum of local increments. Discrete transitions are of two types: synchronizing transitions and asynchronous, silent transitions. In synchronizing transitions, each component checks the validity of a clock constraint and, upon validity, all agree on the same symbol $a \in \Sigma$ and reset some clocks while changing location. Only clocks "owned by" component i (i.e. $x \in X_i$) can be reset by component i, but any component may read clocks not owned by it. Finally, in silent transitions, a specified component checks for the validity of a clock constraint, then resets some clocks it owns and changes location "silently", i.e. on an ε-transition.

A *trajectory* in $\mathcal{A}$ is a sequence of transitions in θ, $\tau = \big((q_1^{j-1}, \ldots, q_n^{j-1}, v_{j-1}) \xrightarrow{\xi_j} (q_1^j, \ldots, q_n^j, v_j)\big)_{1 \leq j \leq m}$, where $\xi_j \in \Sigma \cup \{\epsilon\} \cup \mathbb{R}_{\geq 0}$ for all $1 \leq j \leq m$. Note that each trajectory starts in the initial state of $\mathcal{T}(\mathcal{A})$. The trajectory τ is *accepting* if it ends in $\mathcal{Q}_f$ and *does not end with a time passage transition.* The *timed word accepted by* τ is $\mathsf{acc}(\tau) = \xi_1 \ldots, \xi_m$. The *timed language accepted by* $\mathcal{A}$ is $L(\mathcal{A}) = \{acc(r) \mid r \text{ is accepted by } A\}$. An example of a distributed time-asynchronous automaton is given in Figure 1 below. Note that this language is not timed regular, that is, cannot be accepted by a timed automaton – see [7].

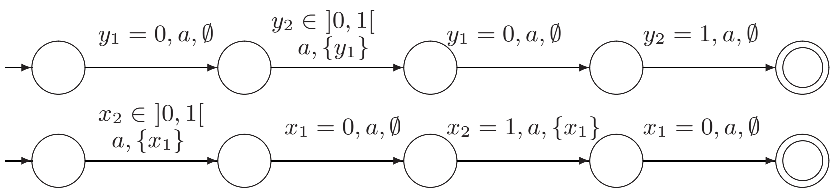

Fig. 1. The two components of a distributed time-asynchronous automaton recognizing the language $\{t_1 \; a \; t_2 \; a \; t_3 \; a \; t_4 \; a \mid t_1, t_2, t_3, t_4 \in]0,1[\; , t_1 + t_3 = 1 \land t_2 + t_4 = 1\}$

2.2 Stopwatch Automata

Definition 2. *A* ***stopwatch automaton*** *is a tuple $\mathcal{A} = (Q, \Sigma, X, \gamma, \delta, q_0, F)$ where Q is a finite set of* locations, *Σ is a finite set of symbols and X is a finite set of* stopwatches, *$\gamma : Q \to \mathcal{P}(X)$ is a mapping assigning to each location the set of stopwatches that are* active *in that location, $q_0 \in Q$ is the* initial location, *$F \subseteq Q$ the set of* final locations, *and δ is a finite set of* transitions *with*

$$\delta \subseteq \big\{ q \xrightarrow{C,a,Y} r \mid q, r \in Q, a \in \Sigma \cup \{\epsilon\}, C \in \mathsf{Constr}(X), Y \subseteq X) \big\}$$

A ***partitioned stopwatch automaton*** *is a stopwatch automaton in which*

$$\forall q, q' \in Q,\ \gamma(q) \neq \gamma(q') \Rightarrow\ \gamma(q) \cap \gamma(q') = \emptyset$$

Similarly to distributed time-asynchronous automata, we give the semantics of a stopwatch automaton as a timed transition system $\mathcal{T}(\mathcal{A}) = (\mathcal{Q}, \theta, \mathcal{Q}_0, \mathcal{Q}_f)$ where $\mathcal{Q} = Q \times \mathbb{R}^n_{\geq 0}$ is the set of *states*, $\mathcal{Q}_0 = \{(q_0, \mathbf{0}_n)\}$ is the (singleton) set of *initial states*, $\mathcal{Q}_f = Q_f \times \mathbb{R}^n_{\geq 0}$ is the set of *final states* and

$$\begin{aligned} \theta = \big\{ (q, v) \xrightarrow{t} (q, v') \mid t \in \mathbb{R}_{\geq 0}, v'\big|_{\gamma(q)} = v\big|_{\gamma(q)} + t, v'\big|_{X \setminus \gamma(q)} = v\big|_{X \setminus \gamma(q)} \big\} \\ \cup \big\{ (q, v) \xrightarrow{a} (q', v') \mid a \in \Sigma \cup \{\varepsilon\},\ \exists (q, C, a, Y, q') \in \delta \text{ such that } v \models C \\ \text{and } v' = v[Y := 0] \big\} \end{aligned}$$

Informally, the automaton can make time passage transitions in which all stopwatches that are active in some location advance by τ, and discrete transitions, in which location changes. The discrete transitions are enabled when the current stopwatch valuation v satisfies the guard C of a certain transition $q \xrightarrow{C,a,X} q' \in \delta$, and when they are executed, the stopwatches in the reset component X are set to zero.

Formally, a *trajectory* in $\mathcal{T}(\mathcal{A})$ is a chain $\rho = \Big((q_{i-1}, v_{i-1}) \xrightarrow{\xi_i} (q_i, v_i) \Big)_{1 \leq i \leq k}$ of transitions from θ that starts in $\mathcal{Q}_0$. An *accepting trajectory* is a trajectory which ends in $\mathcal{Q}_f$ and does not end with a time passage transition. The accepting trajectory $\rho = \Big((q_{i-1}, v_{i-1}) \xrightarrow{\xi_i} (q_i, v_i) \Big)_{1 \leq i \leq k}$ *accepts* the timed word $\mathsf{acc}(\rho) = \xi_1 \xi_2 \ldots \xi_k$. The *language accepted by* $\mathcal{A}$ is then $L(\mathcal{A}) = \big\{ \mathsf{acc}(\rho) \mid \rho \text{ is an accepting trajectory } \big\}$.

An example of a stopwatch automaton is given in Figure 2, and an example of a partitioned stopwatch automaton is given in Figure 3 below. Note that the timed language from Figure 1 is the same as for the automaton in Figure 3. Note also that the timed language in Figure 2 is not timed regular either.

Before ending this section, we state some useful properties of partitioned stopwatch automata.

Proposition 1. *For each partitioned stopwatch automaton $\mathcal{A}$, there exists an equivalent partitioned stopwatch automaton $\mathcal{B}$ satisfying the following properties:*

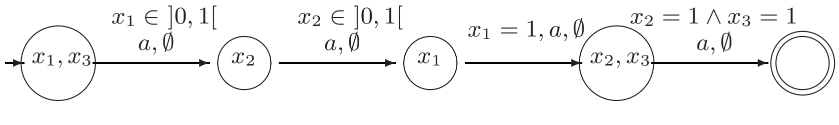

Fig. 2. A stopwatch automaton recognizing the language $\{t_1 \ a \ t_2 \ a \ t_3 \ a \ t_4 \ a \mid t_1, t_2, t_3, t_4 \in]0,1[\ , t_1 + t_3 = t_2 + t_4 = t_1 + t_4 = 1\}$

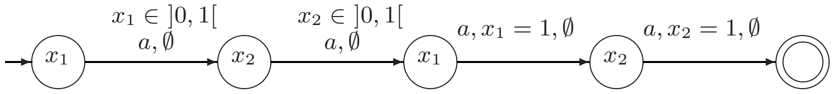

Fig. 3. A partitioned stopwatch automaton recognizing the language $\{t_1 \ a \ t_2 \ a \ t_3 \ a \ t_4 \ a \mid t_1, t_2, t_3, t_4 \in]0,1[\ , t_1 + t_3 = 1 \wedge t_2 + t_4 = 1\}$

1. *Between any two locations q, r there exists at most one transition, and there are no self-loops (i.e. no transitions $q \xrightarrow{C,a,Y} q$).*
2. *Exactly one of the following conditions is satisfied:*
   (a) *Either for each transition $q \xrightarrow{C,a,Y} r$, we have that $Y \subseteq \gamma(q)$ and $C \in \mathsf{Constr}(\gamma(q))$.*
   (b) *Or for each transition $q \xrightarrow{C,a,Y} r$, we have that $Y \subseteq \gamma(r)$ and $C \in \mathsf{Constr}(\gamma(r))$.*
3. *If $q \in Q$ is the target of a visible (i.e. non-ε) transition, then any transition leading to q is visible.*
4. *$\mathcal{A}$ must spend a non-zero amount of time in each location which is the source or the target of some ε-transition: there exists a stopwatch $\xi_i \in X_i$ which is reset when entering each location q with $\gamma(q) = X_i$. Moreover, on each transition $q \xrightarrow{C,\varepsilon,Y} q'$ or on each transition $r \xrightarrow{C,a,Y} r'$ in which r is the target of ε-transitions, then C contains also a constraint of the form $\xi_i \in I$ with $I \subseteq]0,\infty[$.*

These results rely on straightforward state-splitting constructions.

2.3 Decidability of the Reachability Problem for Partitioned Stopwatch Automata

A *zone* (see [12,6]) is a nonempty n-dimensional convex set of points characterized by a constraint of the form $C_Z = \bigwedge_{0 \leq i,j \leq n} x_i - x_j \in I_{ij}$, where $x_0 = 0$ and I_{ij} are non-negative intervals with integer bounds. In the sequel, we consider only zones that use variables from a set of stopwatches (or clocks) X.

An *M-region*, with $M \in \mathbb{R}_{\geq 0}$, is a zone R for which there exists $Y \subseteq X$ such that some constraint characterizing R can be put in the following format:

$$C_R = \bigwedge\nolimits_{x \in Y} (x \in I_x) \wedge \bigwedge\nolimits_{x,y \in Y, x \neq y} x - y \in I_{xy} \wedge \bigwedge\nolimits_{x \in X \setminus Y} x \in]M, \infty[$$

where for each $x, y \in X, x \neq y$:

– Either $I_x = \{\alpha\}$ with $\alpha \in \mathbb{N}$, $\alpha \leq M$, or $I_x =]\alpha, \alpha+1[$ with $\alpha \in \mathbb{N}$, $\alpha \leq M-1$,
– Either $I_{xy} = \{\alpha\}$ with $\alpha \in \mathbb{N}$, $-M \leq \alpha \leq M$, or $I_x =]\alpha, \alpha+1[$ with $\alpha \in \mathbb{N}$, $-M \leq \alpha \leq M-1$,

We denote $\mathsf{Reg}_{\mathcal{A}}$ the set of regions for the automaton $\mathcal{A}$.

The following theorem adapts the well-known region construction of [2] for partitioned stopwatch automata:

Theorem 1. *The reachability problem for the class of partitioned stopwatch automata is decidable.*

Proof. Consider a partitioned stopwatch automaton $\mathcal{A} = (Q, \Sigma, X, \gamma, \delta, q_0, F)$ and denote $n = card\{\gamma(q) \mid q \in Q\}$ the number of partitions of the set of stopwatches X, and denote these partitions as $X_1, \ldots, X_n$. The *region automaton* corresponding to $\mathcal{A}$ will then be the following: $\mathcal{R}_{\mathcal{A}} = \big(Q \times \mathsf{Reg}^n_{\mathcal{A}}, \delta_{\mathcal{R}}, r_0, \mathcal{R}_f\big)$ where $r_0 = (q_0, \mathbf{0}_{X_1}, \ldots, \mathbf{0}_{X_n})$, $\mathcal{R}_f = \big\{(q, R_1, \ldots, R_n) \mid q \in Q_F\big\}$ and

$$\delta_{\mathcal{R}} = \big\{(q, R_1, \ldots, R_n) \to (q, R'_1, \ldots, R'_n) \mid \exists (q,v) \xrightarrow{\xi} (q',v') \in \theta, \xi \in \mathbb{R}_{\geq 0} \cup \Sigma \cup \{\varepsilon\}$$
$$\text{such that } \forall 1 \leq i \leq n,\ R_i, R'_i \in \mathsf{Reg}_{\mathcal{A}} \text{ and } v\big|_{X_i} \in R_i, v'\big|_{X_i} \in R'_i\big\}$$

It is easy to see that $L(\mathcal{A})$ is not empty if and only if $\mathcal{R}_{\mathcal{A}}$ has at least one reachable final configuration. □

3 Shuffle Regular Expressions

In this section we recall the notion of *timed shuffle expression* and its relationship with stopwatch autoamta. We then define the subclass of *fair shuffle expressions*, which will be proved to be equivalent with distributed time-asynchronous automata.

Definition 3. *The set of* ***timed shuffle expressions*** *over a set of symbols Σ is recursively defined as follows:*

$$E ::= a \mid \underline{\mathsf{t}} \mid \langle E \rangle_I \mid f(E) \mid E_1 ⧢ E_2 \mid E_1 + E_2 \mid E_1 \wedge E_2 \mid (E)^*$$

where $a \in \Sigma \cup \{\epsilon\}$, $I \subseteq \mathbb{R}_{\geq 0}$ is an interval with integer and nonnegative bounds, or infinite bounds. and $f : \Sigma \to \Sigma \cup \{\epsilon\}$ is a renaming *function.*

A ***timed regular expression*** *is a timed shuffle expression constructed without the operator ⧢ and an* ***untimed shuffle expression*** *is a shuffled timed expression constructed without the operator $\langle \_ \rangle_I$.*

The *semantics* of a timed shuffle expression E is denoted $\|E\|$ and is given by the following rules:

$$\begin{aligned}
\|a\| &= \{a\} & \|\underline{\mathbf{t}}\| &= \{t \mid t \in \mathbb{R}_{\geq 0}\} \\
\|E_1 + E_2\| &= \|E_1\| \cup \|E_2\| & \|E^*\| &= \|E\|^* \\
\|E_1 \wedge E_2\| &= \|E_1\| \cap \|E_2\| & \|\langle E\rangle_I\| &= \big\{w \in \|E\| \mid \ell(w) \in I\big\} \\
\|E_1 \cdot E_2\| &= \|E_1\| \cdot \|E_2\| & \|E_1 \sqcup\!\sqcup E_2\| &= \|E_1\| \sqcup\!\sqcup \|E_2\| \\
\|f(E)\| &= \big\{f(w) \mid w \in \|E\|\big\}
\end{aligned}$$

Theorem 2 ([4,7]). *Timed regular expressions are equivalent with timed automata and timed shuffle expressions are equivalent with stopwatch automata.*

The following expression is equivalent with the automaton in Figure 2:

$$[z_1/\varepsilon, z_2/\varepsilon, z_3/\varepsilon, z_4/\varepsilon]\big((\langle z_1 \langle\underline{\mathbf{t}}\rangle_{]0,1[} a z_3 \langle\underline{\mathbf{t}}\rangle_{]0,1[} a\rangle_1) \sqcup\!\sqcup (\langle z_2 \langle\underline{\mathbf{t}}\rangle_{]0,1[} z_4 \langle\underline{\mathbf{t}}\rangle_{]0,1[} a\rangle_1)\big) \\ \wedge \big(((z_2\underline{\mathbf{t}}az_3\underline{\mathbf{t}}a) \sqcup\!\sqcup (\langle z_1\underline{\mathbf{t}}az_4\underline{\mathbf{t}}a\rangle_1)\big)$$

Definition 4. *The set of* ***fair shuffle expressions*** *is the subset of timed shuffle expressions defined recursively as follows:*

$$F ::= U \mid f(F) \mid F_1 + F_2 \mid F \wedge T \mid (F)^* \mid F_1 \sqcup\!\sqcup F_2$$

where T is a timed expression, U is an untimed expression and $f : \Sigma \to \Sigma \cup \{\varepsilon\}$ is a renaming.

The following expression is a fair shuffle expression which is equivalent with the partitioned stopwatch automaton in Figure 3:

$$[z_1/\varepsilon, z_2/\varepsilon]\big((z_1\underline{\mathbf{t}}az_2\underline{\mathbf{t}}az_1\underline{\mathbf{t}}az_2\underline{\mathbf{t}}a) \wedge (\langle z_1 \langle\underline{\mathbf{t}}\rangle_{]0,1[} a z_1\underline{\mathbf{t}}a\rangle_1) \sqcup\!\sqcup (\langle z_2 \langle\underline{\mathbf{t}}\rangle_{]0,1[} z_2\underline{\mathbf{t}}a\rangle_1)\big)$$

3.1 Relations Between Partitioned Stopwatch Automata and Fair Shuffle Expressions

Theorem 3. *Partitioned stopwatch automata are equivalent with fair shuffle expressions, and the equivalence is effective.*

Proof. For the left-to-right inclusion, consider a partitioned stopwatch automaton $\mathcal{A} = (Q, \Sigma, X, \gamma, \delta, q_0, F)$. The set of states Q can be partitioned into $S_1, \ldots, S_n$ states such that $\gamma(q) = \gamma(q')$ iff $q, q' \in S_i$, for some i. By means of Proposition 1, we can assume that if $q \xrightarrow{C,a,Y} q' \in \delta$, then $Y \subseteq \gamma(q)$ and $C \in \mathsf{Constr}(\gamma(q))$.

The idea is to construct n timed automata $\mathcal{A}_1, \ldots, \mathcal{A}_n$ such that each $\mathcal{A}_i$ performs all the actions of $\mathcal{A}$ while letting time pass only in states from S_i. The languages of all $\mathcal{A}_i$ will be shuffled, and intersected with the language of an untimed automaton that will ensure proper interleaving. This proper interleaving is also ensured by considering distinct labeling for each transition and by introducing a new symbol ζ, to be performed instantaneously in each $\mathcal{A}_i$ before each step that simulates a step of $\mathcal{A}$. This is needed for keeping from mixing time elapses within $\mathcal{A}_i$ with time elapses within $\mathcal{A}_j$ when they are shuffled. This technique is a variation of the one used in [7]. Finally, each timed automaton will reference a new clock x_i which is needed for ensuring that time passage is 0 in each automaton $\mathcal{A}_i$ while passing through a state q with $\gamma(q) \neq X_i$.

Formally $\mathcal{A}_i = (Q \times \{1,2\}, \Sigma_i, X_i, \delta_i, q_0^i = (q_0, 1), F \times \{1,2\})$ with $\Sigma_i = \{\zeta\} \cup \{q \xrightarrow{C,a,Y} q' \mid q \xrightarrow{C,a,Y} q' \in \delta \text{ and } q \in S_i\}$, $X_i = \gamma(S_i) \cup \{x_i\}$ and:

$$\begin{aligned}
\delta = &\left\{(q,1) \xrightarrow{x_i=0,\zeta,\emptyset} (q,2) \mid q \in Q\right\}\\
&\cup \left\{(q,2) \xrightarrow{C,(q,C,a,X,q'),Y\cup\{(x_i,0)\}} (r,1) \mid q \xrightarrow{C,a,Y} r \in \delta \text{ and } q \in S_i\right\}\\
&\cup \left\{(q,2) \xrightarrow{(x_i=0),\varepsilon,\{(x_i,0)\}} (r,1)) \mid q \xrightarrow{C,a,Y} r \in \delta \text{ and } q \notin S_i\right\}
\end{aligned}$$

As a consequence of Theorem 2, we can construct timed expressions $T_1, \ldots, T_n$ such that $\|T_i\| = L(\mathcal{A}_i)$ for all $1 \leq i \leq n$. Note that T_i expresses the accepting trajectories of $\mathcal{A}$ projected to states in S_i and with the introduction of the symbol ζ at the beginning of each step. More precisely, for any i,

$$\rho = (q_1, v_1) \xrightarrow{t_1} (q_1, \hat{v}_1) \xrightarrow{a_1} \ldots (q_m, v_m) \xrightarrow{t_m} (q_m, \hat{v}_m) \ldots \xrightarrow{a_m} (q_{m+1}, v_{m+1})$$

is an accepting trajectory of $\mathcal{A}$ iff, for any i, there exists $\zeta t_1' a_1' \zeta \cdot \ldots \cdot \zeta t_m' a_m' \in \|T_i\|$ s.t. for all j, $t_j' = \begin{cases} t_j & \text{if } q_j \in S_i \\ 0 & \text{otherwise} \end{cases}$, $a_j = \begin{cases} (q_j, a_j, q_{j+1}) & \text{if } q_j \in S_i \\ \epsilon & \text{otherwise} \end{cases}$. Let then f be the renaming $f : \bigcup_{i\in[1,n]} \Sigma_i \to \Sigma$ defined by $f(q,C,a,Y,q') = a$ and $f(\zeta) = \epsilon$.

Observe that not all timed words specified by $T = f(T_1 ⧢ \ldots ⧢ T_n)$ might be accepted by trajectories of $\mathcal{A}$ since the projected trajectories could be combined in a non-coherent way. Formally, from $\|T\|$ we must only keep the strings of the form $(q_0, C_1, a_1, Y_1, q_1)(q_1, C_2, a_2, Y_2, q_2) \ldots$ of $(T_1 ⧢ \ldots ⧢ T_n)$. But T might also contain strings of the form $(q_0, C_1, a_1, Y_1, r_1)(q_1, C_2, a_2, Y_2, r_2) \ldots$ with $q_1 \neq r_1$.

Hence we need an extra expression that checks the right sequencing of states. Consider then the timed automaton $\mathcal{A}' = (Q \times \{1,2\}, \bigcup_{i\in[1,n]} \Sigma, \emptyset, \delta', (q_0, 1), F \times \{1,2\})$ in which:

$$\delta' = \left\{(q,1) \xrightarrow{\mathsf{true},\zeta,\emptyset} (q,2)) \mid q \in Q\right\} \cup \left\{(q,2) \xrightarrow{\mathsf{true},(q,C,a,Y,q'),\emptyset} (q',1) \mid (q,C,a,Y,q') \in \delta\right\}$$

Since $\mathcal{A}'$ has no clock constraint, we can construct an untimed expression ϑ such that $\|\vartheta\| = \mathcal{L}(\mathcal{A}')$. Finally, we have that $\mathcal{L}(\mathcal{A}) = ((\tau_1 ⧢ \ldots ⧢ \tau_n) \wedge \vartheta)[f]$. This proves the left-to-right inclusion in Theorem 3.

For the right-to-left inclusion, we proceed by structural induction on the given regular expression T. The case when T is a timed regular expression is already covered by Theorem 2 while the cases of union, concatenation, star, shuffle and renaming can be treated exactly as in [7].

For the intersection case, suppose $T = T' \wedge U$, with T' a fair shuffle expression and U an untimed expression. Then, by induction, there exist $\mathcal{A}_1$ a partitioned stopwatch automaton and $\mathcal{A}_2$ a timed automaton such that $\mathcal{L}(\mathcal{A}_1) = \|T\|$ and $\mathcal{L}(\mathcal{A}_2) = \|E\|$. Let $\mathcal{A}_i = (Q_i, \Sigma, X_i, \gamma_i, \delta_i, q_0^i, F_i)$, for $i = 1, 2$.

Note that in U, between two symbols a, b we can have times equal to 0 (when ab is a subexpression of U) or all possible times (when $a\underline{\mathbf{t}}b$ is a subexpression of U). Therefore, we can assume that $\mathcal{A}_2$ is such that $X_2 = \{x_q | q \in Q_2\}$ and each transition of $\mathcal{A}_2$ is of the form $q \xrightarrow{C,a,\{x_{q'}\}} q'$ with $C \in \{true, x_q = 0\}$. We then construct a partitioned stopwatch automaton $\mathcal{A} = (Q_1 \times Q_2, \Sigma, X_1 \cup X_2, \gamma, \delta, (q_0^1, q_0^2), F_1 \times F_2)$ in which $\gamma(q, q') = \gamma_1(q) \cup \{x_{q'}\}$ and

$$\delta = \{(q_1, q_2) \xrightarrow{C_1 \wedge C_2, a, Y_1 \cup Y_2} (q_1', q_2') \mid q_i \xrightarrow{C_i, a, Y_i} q_i' \in \delta_i, \;\; i = 1, 2\}$$

Note that the construction would not work with only one extra clock, since that clock would have to belong to each $\gamma(q)$, which would mean that the automaton $\mathcal{A}$ is not partitioned. □

Proposition 2. *The class of partitioned stopwatch automata is closed under union, renaming, shuffle, Kleene star but not under intersection.*

Proof. Closure under union, renaming, shuffle, Kleene star is obvious since fair shuffle expressions are.

For the proof of non-closure under intersection we rely on several results on distributed time-asynchronous automata from the following section. □

Theorem 4. *The classes of distributed time-asynchronous automata and partitioned stopwatch automata are equivalent.*

Proof. For the left-to-right inclusion, consider first a distributed time-asynchronous automaton $\mathcal{A} = (Q_1, \ldots, Q_n, \Sigma, X_1, \ldots, X_n, \delta_1, \ldots, \delta_n, q_1^0, \ldots, q_n^0, F_1, \ldots F_n)$.

The language $L(\mathcal{A})$ is then straightforwardly equal with the language of the partitioned stopwatch automaton $\mathcal{A}' = (Q, \Sigma, X, \gamma, \delta, q_0, F)$ in which:

- $Q = Q_1 \times \ldots \times Q_n \times \{1, \ldots, n\}$ where index j represents the sequential component for which the time elapses.
- The set of clocks is $X = \bigcup_{i \in [1,n]} X_i$ and γ is such that $\gamma(q_1, \ldots, q_n, j) = X_j$.
- δ is the following set of transitions:

$$\begin{aligned}\delta = \Big\{&(q_1, \ldots, q_n, j) \xrightarrow{C,a,Y} (q_1', \ldots, q_n', j) \mid \forall 1 \le i \le n \; \exists q_i \xrightarrow{C_i, a, Y_i} q_i' \in \delta_i \\ &\text{with } Y = \bigcup\nolimits_{i \in [1,n]} Y_i \text{ and } C = \bigwedge\nolimits_{1 \le i \le n} C_i \Big\} \\ \cup \Big\{&(q_1, \ldots, q_i, \ldots, q_n, j) \xrightarrow{C,\epsilon,Y} (q_1, \ldots, q_i', \ldots, q_n, j) \mid q_i \xrightarrow{C,\epsilon,Y} q_i' \in \delta_i \Big\} \\ \cup \Big\{&(q_1, \ldots, q_n, j) \xrightarrow{\mathsf{true},\epsilon,\emptyset} (q_1, \ldots, q_n, j+1)\big) \mid 1 \le j < n \Big\} \\ \cup \Big\{&(q_1, \ldots, q_n, n) \xrightarrow{\mathsf{true},\epsilon,\emptyset} (q_1, \ldots, q_n, 1) \Big\}\end{aligned}$$

- $q_0 = (q_0^1, \ldots, q_n^0)$ and $F = F_1 \times \ldots \times F_n \times \{1, \ldots, n\}$.

The reverse proof requires a slightly more involved construction, since this time we need to simulate some centralized control (the location of a partitioned stopwatch automaton) by a distributed control, via synchronous read of input and checking global constraints. The main problem here is posed by ε-transitions.

Consider then a partitioned stopwatch automaton $\mathcal{A} = (Q, \Sigma, X, \gamma, \delta, q_0, F)$. We will assume $\mathcal{A}$ satisfies conditions from Proposition 1. The main idea is to construct a distributed time-asynchronous automaton with n components, where $n = card\{\gamma(q) \mid q \in Q\}$ is the number of partitions of X. X_i will be (part of) the set of clocks that can be reset by component i. Each component will then basically behave like $\mathcal{A}$ but keeping from reseting *and* updating clocks that are not in X_i. A first (essential, but not sufficient) means of synchronization is the use of Σ as a common input alphabet.

Additionnaly, we will employ a second set of clocks $\overline{X}$ whose elements will be indexed by the set $Q \times \{1, \ldots, n\}$, which will be used to forbid local time to advance in component i, when it reaches a location q with $\gamma(q) \neq X_i$, that is, a component j that does not own the set of clocks X_i. Hence, each clock $x_{(q,i)}$ will be reset in each component $j \neq i$ when entering location q, and then tested for 0 when j leaves q.

But we need a supplimentary mechanism to ensure full synchronization in the presence of ε-transitions in the original automaton $\mathcal{A}$. This mecanisms employs an extra set of clocks $Z = \{z_{(q,r,i)} \mid 1 \leq i \leq n \text{ and } \exists q \xrightarrow{C,a,Y} r \in \delta\}$. Each clock $z_{(q,r,i)}$ will be reset in each component exactly when the (unique) transition between q and r is taken by the appropriate component.

The actual idea is that each component i *guesses* that the next transition is $q \to r$, by resetting the clock $z_{(q,r,i)}$; at the same time, it resets $x_{(q,i)}$. Immediately after (fact which can be checked by $x_{(q,i)} = 0$), each component checks that everybody has guessed the same transition, by checking that $\bigwedge_{1 \leq j \leq n} z_{(q,r,j)} = 0$. Here we rely on assumption 4 in Proposition 1, which ensures that $z_{(q,r,i)} \neq 0$ when $\mathcal{A}$ crosses a ε-transition different from that connecting q and r. The same reasoning will be employed when the transition between q and r is in Σ, as this will ensure the components that they all agree on the transition that links q to r, and do not employ some other transition that has the same input label.

Formally, the distributed time-asynchronous automaton that is equivalent with $\mathcal{A}$ is $\mathcal{B} = (Q_1, \ldots, Q_n, \Sigma, X_1', \ldots, X_n', \delta_1, \ldots, \delta_n, q_0^1, \ldots q_n^0, F_1, \ldots, F_n)$ where

1. $Q_i = (Q \cup (Q \times Q)) \times \{i\}$, $q_0^i = (q_0, i)$ and $F_i = F \times \{i\}$.
2. $X_i' = X_i \cup \{x_{(q,i)} \mid \gamma(q) \neq X_i\} \cup \{z_{(q,r,i)} \mid 1 \leq i \leq n\}$.
3. δ_i consists of transitions

$$\begin{aligned}\delta = \big\{(q,i) \xrightarrow{C,a,Y} (q,r,i) \mid \gamma(q) = X_i \text{ and } \exists q \xrightarrow{C,a,Y'} r \in \delta, Y = \{x_{(q,i)}, z_{(q,r,i)}\}\big\} \\ \cup \big\{(q,i) \xrightarrow{C,a,Y} (q,r,i) \mid \gamma(q) \neq X_i \text{ and } \exists q \xrightarrow{C',a,Y'} r \in \delta, C = (x_{(q,i)} = 0) \\ \text{and } Y = \{x_{(q,i)}, z_{(q,r,i)}\}\big\} \\ \cup \big\{(q,r,i) \xrightarrow{C,a,Y} (r,i) \mid \gamma(q) = X_i \text{ and } \exists q \xrightarrow{C',a,Y'} r \in \delta, \\ C = C' \wedge (x_{(q,i)} = 0) \wedge \bigwedge_{1 \le j \le n} (z_{(q,r,j)} = 0) \text{ and } Y = Y'\big\} \\ \cup \big\{(q,r,i) \xrightarrow{C,a,Y} (r,i) \mid \gamma(q) \neq X_i \text{ and } \exists q \xrightarrow{C',a,Y'} r \in \delta, \\ C = C' \wedge (x_{(q,i)} = 0) \wedge \bigwedge_{1 \le j \le n} (z_{(q,r,j)} = 0) \text{ and } Y = \{x_{(r,i)}\}\big\}\end{aligned}$$

Note first that, in any accepting trajectory, in locations of the type (q,r,i) time cannot pass since they are needed for synchronization purposes, which is achieved by resetting $z_{(q,r,i)}$ before entering that location, and then checking, when leaving (q,r,i), that $z_{(q,r,j)} = 0$ for all j.

We may then show that in any trajectory $\rho = \big((\overline{q}_1^{j-1}, \ldots, \overline{q}_n^{j-1}, v_{j-1}) \xrightarrow{\xi_j} (\overline{q}_1^j, \ldots, \overline{q}_n^j, v_j)\big)_{1 \le j \le m}$, each state $(\overline{q}_1^{j-1}, \ldots, \overline{q}_n^{j-1}, v_{j-1})$ has the following properties

1. There exists a (possibly empty) subset $Y_j \subseteq \{1, \ldots, n\}$ such that for each $i \in Y_j$ there exist $q, r \in Q$ such that $\overline{q}_i^{j-1} = (q,r,i)$.
2. If $Y_j = \emptyset$ then there exists $q \in Q$ such that for all $1 \le i \le n$, $\overline{q}_i^j = (q,i)$.
3. If $Y_j \neq \emptyset$ then and exactly one of the two following properties holds:
   (a) Either for all $1 \le i \le n, i \notin Y_j$, $\overline{q}_i^j = (q,i)$;
   (b) Or for all $1 \le i \le n, i \notin Y$, $\overline{q}_i^j = (r,i)$.

The correctness of this construction can then be proved by induction on the length of an accepting trajectory in $\mathcal{B}$, $\rho = \big((\overline{q}_1^{j-1}, \ldots, \overline{q}_n^{j-1}, v_{j-1}) \xrightarrow{\xi_j} (\overline{q}_1^j, \ldots, \overline{q}_n^j, v_j)\big)_{1 \le j \le m}$ by showing that there exists a corresponding accepting trajectory in $\mathcal{A}$, $\overline{\rho} = \big((r_{l-1}, u_{l-1}) \xrightarrow{\zeta_l} (r_j, u_l)\big)_{1 \le l \le p}$, which accepts the same timed word. The choice of r_j can be done as follows:

1. If $Y_j \neq \emptyset$ and hence $\overline{q}_i^j = (q,r,i)$ for all $i \in Y_j$, then we put $r_j = r$.
2. If $Y_j = \emptyset$ and hence $\overline{q}_i^j = (q,i)$ for some $q \in Q$ and all $1 \le i \le n$, then we put $r_j = q$.

$\square$

4 Nonclosure Under Intersection of Partitioned Stopwatch Automata

In this subsection we prove the non-closure result from Proposition 2:

Proposition 3. *The class of languages accepted by distributed time-asynchronous automata is not closed under intersection.*

The proof of result relies on a series of additional properties, starting with the following:

Proposition 4. *For each distributed time-asynchronous automaton* $\mathcal{A} = (Q_1, \dots, Q_n, \Sigma, X_1, \dots, X_n, \delta_1, \dots, \delta_n, q_1^0, \dots, q_n^0, F_1, \dots F_n)$ *there exists a distributed time-asynchronous automaton with private clocksets* $\mathcal{B} = (\overline{Q}_1, \dots, \overline{Q}_n, \overline{\Sigma}, X_1, \dots, X_n, \overline{\delta}_1, \dots, \overline{\delta}_n, \overline{q}_1^0, \dots, \overline{q}_n^0, \overline{F}_1, \dots \overline{F}_n)$ *and a renaming* $f : \overline{\Sigma} \to \Sigma \cup \{\varepsilon\}$ *such that* $L(\mathcal{A}) = f\big(L(\mathcal{B})\big)$.

Moreover, if $\mathcal{A}$ *contains no* ε*-transition, then* f *contains no symbol deletion.*

Proof. The idea in the construction of $\mathcal{B}$ is to no longer have ε-transitions and thus have only synchronizing discrete transitions. Then, synchronization done by global constraints in $\mathcal{A}$ will be simulated by unique labels of the transitions.

If we recall that $Q = Q_1 \times \ldots \times Q_n$, $F = F_1 \times \ldots \times F_n$ and $q_0 = (q_0^1, \dots, q_0^n)$, then, formally, the components of $\mathcal{B}$ are:

- The set of locations is $\overline{Q}_i = Q \times \{i\}$, with $\overline{q}_0^i = (q_0, i)$ and $F_i = F \times \{i\}$.
- The set of inputs is

$$\overline{\Sigma} = \big\{(tr_1, \dots, tr_n) \mid \forall 1 \le i \le n, tr_i = q \xrightarrow{C,a,Y} r \in \delta_i\big\} \\ \cup \big\{q \xrightarrow{C,\varepsilon,Y} r \mid q \xrightarrow{C,\varepsilon,Y} r \in \delta_i \text{ for some } 1 \le i \le n\big\}\big\}$$

- The i-th transition relation is:

$$\begin{aligned} \overline{\delta}_i = \big\{&(\overline{q}, i) \xrightarrow{\overline{C}_i,\xi,Y_i} (\overline{r}, i) \mid \overline{q} = (q_1, \dots, q_n), \overline{r} = (r_1, \dots, r_n), \\ &\forall 1 \le j \le n,\ \exists tr_j = q_j \xrightarrow{C_j,a,Y_j} r_j \in \delta_j, \xi = (tr_1, \dots, tr_n), \overline{C}_i = C_i\big|_{X_i}\big\}\big\} \\ \cup \big\{&(\overline{q}, i) \xrightarrow{\overline{C}_i,\xi,\overline{Y}_i} (\overline{r}, i) \mid \overline{q} = (q_1, \dots, q_n), \overline{r} = (r_1, \dots, r_n), \text{ and } \exists j, \\ &1 \le j \le n \text{ for which } \exists tr_j = q_j \xrightarrow{C_j,\varepsilon,Y_j} r_j \in \delta_j \text{ such that } \xi = tr_j, \\ &\overline{C}_i = C_j\big|_{X_i}, \overline{Y}_i = Y_j \cap X_i, \text{ and } \forall l \ne j, q_l = r_l\big\} \end{aligned}$$

For each constraint C, we have denoted here $C\big|_Y$ the sub-constraint of C which uses only clocks in Y. □

The following result is a generalization of Proposition 12 of [11] to the case of distributed time-asynchronous automata with private clocksets:

Lemma 1. *Given a distributed time-asynchronous automaton with private clockset* $\mathcal{A}$ *suppose that all timed words in* $L(\mathcal{A})$ *contain the same sequence of symbols,* $L(\mathcal{A}) = \big\{t_1 a_1 \dots t_n a_n t_{n+1} \mid t_i \in \mathbb{R}_{\ge 0}\big\}$, *for some* $a_1, \dots, a_n \in \Sigma$.

Then for each $t_1 a_1 \dots t_n a_n t_{n+1} \in L(\mathcal{A})$ *there exists some* $k \in \mathbb{N}$ *and* k *sequences of time points* $(t_i^j)_{1 \le i \le n+1}$ $(1 \le j \le k)$ *such that*

– $t_i = \sum_{1\le j\le k} t_i^j$ *for all* $1 \le i \le n+1$.
– *If, for each* $1 \le j \le k$, *we denote* $R_j \subseteq \mathbb{R}_{\ge 0}{}^{n+1}$ *the region to which belongs each* $(n+1)$*-dimensional point* v^j *defined by* $v_i^j = \sum_{1\le l\le i} t_l^j$ $(1 \le i \le n+1)$*, then for any other point* $u^j \in R$*, if we define* $s_i^j = u_i^j - u_{i-1}^j$ *(with* $u_0^j = 0$*), then*

$$\Big(\sum\nolimits_{1\le j\le k} s_1^j\Big) a_1 \ldots \Big(\sum\nolimits_{1\le j\le k} s_n^j\Big) a_n \Big(\sum\nolimits_{1\le j\le k} s_{n+1}^j\Big) \in L(\mathcal{A})$$

The last result that needed in the proof of Proposition 4 is the following:

Lemma 2 ([6]). *For any region* $R \subseteq \mathbb{R}^n_{\ge 0}$ *and subset of indices* $Y \subseteq \{1,\ldots,n\}$*, if* $R\big|_Y$ *denotes the restriction (or projection) of* R *onto points whose coordinates belong to* Y*, then for any* $v \in R\big|_Y$*, there exist* $v' \in R$ *with* $v'\big|_Y = v$.

Proof (of Proposition 4). Consider the timed language

$$L = \big\{ t_1 a t_2 a t_3 a t_4 a \mid t_1, t_2, t_3, t_4 \in \,]0,1[\,, t_1 + t_3 = t_2 + t_4 = t_1 + t_4 = 1 \big\}$$

L is the intersection of the language of the distributed time-asynchronous automaton in Figure 1 with the language of the distributed time-asynchronous automaton in Figure 4, and is the language in Figure 2, hence is accepted by a stopwatch automaton.

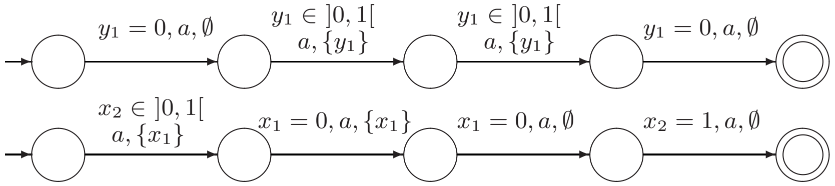

Fig. 4. The two components of a distributed time-asynchronous automaton recognizing the language $\{t_1\ a\ t_2\ a\ t_3\ a\ t_4\ a \mid t_1, t_2, t_3, t_4 \in \,]0,1[\,, t_1 + t_4 = 1\}$

Suppose there exists a distributed time-asynchronous automaton with private clocks $\mathcal{A}$ and a renaming f such that $f(L(\mathcal{A})) = L$. In the sequel, we will present all elements of $L(\mathcal{A})$ in the form $t_1 a_1 (t_2 - t_1) a_2 \ldots (t_n - t_{n-1}) a_n$, for some $(t_1, \ldots, t_n) \in \mathbb{R}_{\ge 0}{}^n$.

Take the timed word $w = 0.3\,a\,0.3\,a\,0.7\,a\,0.7\,a \in L$, hence there exists a timed word $w' \in L(\mathcal{A})$ with $f(w') = w$. Suppose $w' = t_1 a_1 (t_2 - t_1) a_2 \ldots (t_n - t_{n-1}) a_n$. Then, by Lemma 1, there exist k sequences of time points $(t_i^j)_{1\le i\le n}$ $(1 \le j \le k)$ such that $t_i = \sum_{1\le j\le k} t_i^j$, and regions $R_j \ni t_i^j$ which fulfill the properties in Lemma 1.

Consider now the renaming f and suppose that $f(a_l) = a$ for $l \in \mathcal{I} = \{i_1, i_2, i_3, i_4\}$ and $f(a_l) = \varepsilon$ otherwise. Hence,

$$\sum\nolimits_{1\le i\le i_1} t_i = \sum\nolimits_{i_1 < i \le i_2} t_i = 0.3 \qquad \sum\nolimits_{i_2 < i \le i_3} t_i = \sum\nolimits_{i_3 < i \le i_4} t_i = 0.7$$

By Lemma 2, if we consider the restriction of each region R_j to the set of indices $\mathcal{I}$, then for any $u^j \in R_j\big|_{\mathcal{I}}$ $(1 \leq j \leq k)$ there exists $v^j \in R_j$ with $u^j = v^j\big|_{\mathcal{I}}$, But this means that for any timed word

$$w_1 = \Big(\sum_{1\leq j\leq k} s_1^j\Big)a\Big(\sum_{1\leq j\leq k}(s_2^j - s_1^j)\Big)a\Big(\sum_{1\leq j\leq k}(s_3^j - s_2^j)\Big)a\Big(\sum_{1\leq j\leq k}(s_4^j - s_3^j)\Big)a$$

there exists a timed word $w_2 = \Big(\sum_{1\leq j\leq k} t_1^j\Big)a_1 \dots \Big(\sum_{1\leq j\leq k}(t_n^j - t_{n-1}^j)\Big)a_n \in L(\mathcal{A})$ with $f(w_2) = w_1$ and $w_2 \in L(\mathcal{A})$, and hence $w_1 \in L$.

Therefore, we only need to consider what are the possible 4-dimensional regions R_j that may compose timed words in L of the form $t_1a(t_2 - t_1)a(t_3 - t_2)a(t_4 - t_3)a$, for which there exists s_l^j $(1 \leq j \leq k, 1 \leq l \leq 4)$ such that

- $\sum s_l^j = t_l$ and $(s_1^j, s_2^j, s_3^j, s_4^j) \in R_j$,
- And the same holds also for $t_1 = t_2 = 0.3$ and $t_3 = t_4 = 0.7$.

The first observation to make on R_j is that in its normal form C_{R_j}, the constraint for $s_l^j - s_{j-1}^j$ is of the form $s_l^j - s_{l-1}^j \in [l-1, l[$ $(1 \leq l \leq 4$, and we consider $s_0^j = 0)$. This follows since $s_l^j - s_{l-1}^j$ must belong to the same interval as the l-th time passage in w. For example, s_4^j must belong to the same unit length interval as 0.7.

We may then see that there exists only one region R_j whose constraint on s_1^j is not of the form $s_1^j = 0$. This follows by contradiction, since if we suppose that this holds for two indices j_1, j_2, that is, that $C_{R_{j_1}}$ and $C_{R_{j_2}}$ both contain $s_1^j \in]0, 1[$, then, by lemma 2, we may construct s^{j_1}, s^{j_2} such that $s_1^{j_1} = s_1^{j_2} = 0.8$, which would mean that a timed word of the type $1.6\, a\, t_2\, a\, t_3\, a\, t_4 a$ would have to be in L, which is obviously false.

This observation can be further generalized to all l and all constraints $s_l^j - s_{l-1}^j$. As a consequence, we only have to consider that $k \leq 4$. This gives only finitely many (actually only 4) cases to check, and all lead to the possibility to construct a timed word as in Lemma 2, but which is not in L. □

5 Conclusions

We have introduced the class of distributed time-asynchronous automata, that correspond to asynchronous compositions of timed automata, in which time is allowed to progress independently between components, and resynchronizations are done with the aid of global clock constraints and input symbols. We have proved that this class is equivalent with fair shuffle expressions, which are timed shuffle expressions which allow intersection only with untimed expressions. We have also proved nonclosure under intersection for distributed time-asynchronous automata.

An interesting question concerns the study of closure properties for distributed time-asynchronous automata with private clocksets.

References

1. Abdeddaïm, Y., Maler, O.: Preemptive job-shop scheduling using stopwatch automata. In: Katoen, J.-P., Stevens, P. (eds.) ETAPS 2002 and TACAS 2002. LNCS, vol. 2280, pp. 113–126. Springer, Heidelberg (2002)
2. Alur, R., Dill, D.: A theory of timed automata. Theoretical Computer Science 126, 183–235 (1994)
3. Asarin, E.: Challenges in timed languages. Bulletin of EATCS 83 (2004)
4. Asarin, E., Caspi, P., Maler, O.: Timed regular expressions. Journal of ACM 49, 172–206 (2002)
5. Bouyer, P., Petit, A., Thérien, D.: An algebraic approach to data languages and timed languages. Inf. Comput. 182(2), 137–162 (2003)
6. Dima, C.: Computing reachability relations in timed automata. In: Proceedings of LICS'02, pp. 177–186 (2002)
7. Dima, C.: Timed shuffle expressions. In: Abadi, M., de Alfaro, L. (eds.) CONCUR 2005. LNCS, vol. 3653, pp. 95–109. Springer, Heidelberg (2005)
8. Henzinger, T., Raskin, J.-F., Schobbens, P.-Y.: The regular real-time languages. In: Larsen, K.G., Skyum, S., Winskel, G. (eds.) ICALP 1998. LNCS, vol. 1443, pp. 580–591. Springer, Heidelberg (1998)
9. Henzinger, T.A., Kopke, P.W., Puri, A., Varaiya, P.: What's decidable about hybrid automata. J. Comput. Syst. Sci. 57, 94–124 (1998)
10. Krishnan, P.: Distributed timed automata. Electr. Notes Theor. Comput. Sci. 28 (1999)
11. Ouaknine, J., Worrell, J.: Revisiting digitization, robustness, and decidability for timed automata. In: Proceedings of LICS'03, pp. 198–207. IEEE Computer Society Press, Los Alamitos (2003)
12. Yovine, S.: Model-checking timed automata. In: Rozenberg, G. (ed.) Lectures on Embedded Systems. LNCS, vol. 1494, pp. 114–152. Springer, Heidelberg (1998)
13. Zielonka, W.: Notes on finite asynchronous automata. Informatique Théorique et Applications 21(2), 99–135 (1987)