On tick automata for distributed timed DESs with synchronisations and minimal time constraints

Jan Komenda, Dimitri Lefebvre

Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 56-2 (2023) 8635–8640

On tick automata for distributed timed DESs with synchronisations and minimal time constraints*

Jan Komenda * **Dimitri Lefebvre** **

* *Institute of Mathematics, Czech Academy of Sciences, Prague*
** *Université Le Havre Normandie, GREAH, France.*

Abstract: This paper is about the representations of distributed timed discrete event systems (DESs) with synchronisation events and minimal time constraints. A subclass of Timed Petri nets with specific time semantics is first recalled as a reference model for the considered systems. Such a model is reformulated according to modular tick automata with minimal times that behave like logical automata. A synchronous composition of such automata is then defined and the result of this composition is a new tick automaton that has the same timed language as the original Petri net, which is not the case for the earlier model of tick automata with constant (exact) times.

Keywords: timed discrete event systems, distributed discrete event systems, tick automata, synchronous composition, timed Petri nets.

1. INTRODUCTION

In numerous systems, complex behaviors exist that are characterized not only by sequences of logical events but also by timing aspects. In order to discuss such aspects from the perspective of discrete event systems (DESs), various formalisms have been studied (Brandin and Wonham (1994), Berard et al. (2005)) as weighted automata endowed with a time structure given in terms of transition weights (Lai et al. (2022)), temporal extensions of Petri nets (Ramchandani (1973), Merlin (1974), Boyer and Roux (2008)). One motivation for this research effort is that timed DESs are interesting to refine analysis and verification techniques by improving the information that one can capture from the execution of a system (Li et al. (2022)). Domains of application range from cyber physical systems to logistic, robotics and other industrial systems.

This paper concerns the abstraction of a class of distributed timed DESs. Our reference model for such systems is an extension of PNs with time parameters interpreted as minimal firing durations associated with transitions. The main contribution of the paper is to prove that such systems can be represented as a set of modular local tick automata synchronized with respect to some particular events. In detail, we define a synchronous composition of the tick automata and show that this composition is suitable to represent transition-timed Petri nets with synchronisation transitions. The proposed composition is much simpler than other existing timed products like the product interval automata (D'Souza and Thiagarajan (2002)) that is based on parallel composition, the compositions of interval weighted automata with a single clock (Komenda et al. (2010)), or the interval synchronous product defined for a class of timed automata with time intervals (Lin et al. (2019)). The advantage of the proposed approach is that we do not work with explicit intervals that count the number of ticks, but rather work with tick automata that can be manipulated as logical (i.e., untimed) structures. In particular, the important concept of synchronous product that enables to build more complex systems (e.g., multi-clock timed automata) out of smaller ones (e.g., single-clock timed automata) works well with the tick automata corresponding to automata with minimal time (as opposed to constant time) that are used in this paper. In detail, our motivation for this work is as follows. Synchronous products of automata with exact transition times modeled by previous approaches lead to non deterministic weighted automata over semirings where most of fundamental problems (such as equivalence and comparison of behaviors) are undecidable. The tick automata are certainly not optimal from complexity viewpoint, but their behaviors are just rational languages over alphabet of discrete events augmented by the special tick event corresponding to elapsing of one unit of time. This class of automata with timing of transitions is stable by synchronous products. This makes a difference with automata with exact transition times, where synchronous product of their tick automata does not work, because it typically leads to deadlocks due to impossibility to letting the time pass in order to wait for a correct synchronization between the components.

The paper is organized as follows. Section 2 is about the background, where T-time Petri nets as the reference model for timed DES and automata with minimal time are recalled. Section 3 introduces the tick automata and their synchronous product. Section 4 is reserved for the conclusion and perspectives.

* This work was supported by the French National Research Agency under grant agreement ANR-22-CE10-0002, by RVO 67985840, and by GAČR 19-06175J

10.1016/j.ifacol.2023.10.039

2. PRELIMINARIES

2.1 Petri nets

Definition: An ordinary Petri net (PN) is a 4-tuple $G = (\mathcal{P}, \mathcal{T}, \mathcal{F}, M_0)$, where $\mathcal{P}$ is a finite set of places, $\mathcal{T}$ is a finite set of transitions, $\mathcal{F} \subseteq (\mathcal{P} \times \mathcal{T}) \cup (\mathcal{T} \times \mathcal{P})$ is a relation between places and transitions and M is a map $\mathcal{P} \to \mathbf{N}$ with $\mathbf{N}$ the set of integer numbers and $M_0(p)$ is the initial marking of place $p \in \mathcal{P}$. □

For transition $a \in \mathcal{T}$, ${}^\bullet a$ (resp. $a^\bullet$) denotes the set of its input (resp. output) places as given by $\mathcal{F}$. When $|{}^\bullet a| > 1$, a is called a synchronisation transition and we note $\mathcal{T}_s$ the set of synchronisation transitions of the net. The marking of a net evolves according to usual firing rules of transitions:

- A transition a is *enabled* at M if there exists at least one token in each of its input places.
- An enabled transition $t \in \mathcal{T}$ can fire. The firing of a transforms M into M' by removing one token from each of the input places ($p \in {}^\bullet a$) and adding one token in each of the output places of a ($p \in a^\bullet$).

We say that a word $w = a_1 a_2 \dots a_n \in \mathcal{T}^*$ is a *firing sequence* starting from marking M_0 if there is a sequence of markings $M_1 M_2 \dots M_n$ such that transition a_i is enabled at M_{i-1} and its firing transforms M_{i-1} into M_i. We call *language* of the PN the set $L \subset \mathcal{T}^*$ of firing sequences starting from initial marking M_0. A PN is said to be *safe* (resp. *m-bounded*) if for all accessible marking each place contains at most one token (resp. m tokens).

A PN is a bipartite directed graph represented by a directed graphs with two types of nodes: places in $\mathcal{P}$ drawn as circles and transitions in $\mathcal{T}$ drawn as bars. The incidence matrix given by $\mathcal{F}$ is represented by arcs going from a place to a transition or from a transition to a place. The marking in place p is drawn as $M(p)$ tokens inside this place.

2.2 T-timed Petri nets with minimal time constraints

One reference model for timed DES is the extension of PNs with time parameters interpreted as minimal firing durations associated with transitions. This model meets the standard definition of transition-timed PNs (T-Timed PNs) (Ramchandani (1973)) when no earliest firing policy is considered and firings can be delayed, i.e., a transition can fire when it has been enabled during a time equal or larger than its time parameter. In detail, for each transition $a \in \mathcal{T}$ its parameter $\tau(a)$ is interpreted as the minimal time that must elapse, starting from the time at which a is enabled, until this transition can fire. In other words, one can associate a time interval $[\tau(a), +\infty)$ to each transition a that represents the range of times when the firing of a can occur (from the time at which a is enabled).

Definition: An ordinary T-timed PN with minimal time constraints is a pair $(\mathcal{G}, \tau)$, where $\mathcal{G} = (\mathcal{P}, \mathcal{T}, \mathcal{F}, M_0)$ is an ordinary PN and τ is a map $\mathcal{T} \to \mathbf{N}$ with $\tau(a)$ the minimal time constraint of transition $a \in \mathcal{T}$, i.e., the minimal time before a can fire. □

The following functioning rules on TPN are adopted:

- for each transition $a \in \mathcal{T}$, $\tau(a) \in \mathbf{N}$ (observe that we can relax this assumption and use real values of time that are all multiples of a common increment);
- it is assumed that tokens of the initial marking M_0 have arrived in the T-timed PN at time instant $t = 0$;
- in case a place has several output transitions, which corresponds to a *conflict*, there exist several semantics to decide which of the downstream transitions are to be fired. In this paper we consider all logically feasible choices, i.e., timing aspects are not taken into account. This is known as *preselection semantics*. There exist other semantics, e.g., *race semantics* or open-loop race semantics, where time considerations are important for choosing transitions are to be fired.

A timed trajectory of length n that begins from the marking M_{j_0} is a sequence of markings and pairs formed by transitions and their firing times:

$$\rho = M_{j_0} \xrightarrow{(a_1,t_1)} M_{j_1} \xrightarrow{(a_2,t_2)} \dots \xrightarrow{(a_n,t_n)} M_{j_n}, \quad (1)$$

where for $i = 1, \dots n$, M_{j_i} are markings, $a_i \in \mathcal{T}$ are transitions enabled at $M_{j_{i-1}}$, and $t_{i-1}, t_i \in \mathbf{R}$ are values of time that satisfy $t_0 = 0$ and $\tau(a_i) \leq t_i - t_{i-1}$.

Example : An example of a safe ordinary T-timed PN with minimal time constraints $\mathcal{G}_1$ is reported in Figure 1. Observe that a is a synchronisation transition. The minimal time constraints are reported near the transitions: $\tau(a) = 1$, $\tau(a_1) = 2$, $\tau(a_2) = 1$ and $\tau(a_3) = 3$.

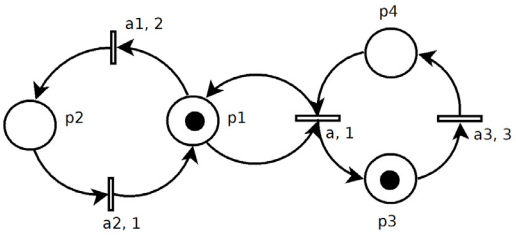

Fig. 1. A T-timed PN with minimal time constraints $\mathcal{G}_1$.

2.3 Automata with minimal times

The time behaviour of a given T-timed PN with minimal time constraints can be represented by a set of modular timed automata and a synchronous product that tracks the synchronisations and other timing aspects. A particular class of timed automata is considered for that purpose that is characterized by a single clock, minimal values of time and synchronisation events. Such automata are referred to as automata with minimal times in the following.

Definition: An Automaton with Minimal Times (AMT) is a four-tuple $A = (X, E, \Delta, x_0)$, where X is a finite set of states, E is a finite set of events, $\Delta \subseteq X \times E \times \mathbf{N} \times X$ is a timed transition relation, and x_0 is an initial state. □

In simple words, an AMT A is a finite automaton [1] endowed with a time structure that associates with each transition a time parameter interpreted as the minimal duration required to fire this transition. More precisely, we say that (x, e, τ, x') is defined (i.e., $(x, e, \tau, x') \in \Delta$) if the state x' is reachable from the state x by the occurrence of event e after an amount of time that equals at least τ time units (TUs), where τ is count from the previous event occurrence. In this case, the transition from x to x' is said to be an (e, τ)-transition. In other word, we consider systems with a single clock that is reset once an event occurs. An AMT is either nondeterministic or deterministic (in the usual logical sense) depending if there can be two transitions associated with the same event out of a given state.

A timed production of length n that begins from the state x_{j_0} is a sequence of states and pairs formed by events and their time stamps:

$$\rho = x_{j_0} \xrightarrow{(e_1,t_1)} x_{j_1} \xrightarrow{(e_2,t_2)} x_{j_2} \dots x_{j_{n-1}} \xrightarrow{(e_n,t_n)} x_{j_n}, \quad (2)$$

where $x_{j_i} \in X, e_i \in E, t_0 = 0, t_i \in \mathbf{R}, (x_{j_{i-1}}, e_i, \tau_i, x_{j_i}) \in \Delta$, with $\tau_i \leq t_i - t_{i-1}$, τ_i being the minimal time from which e_i may occur.

Let us define $\mathcal{R}(A, x_0)$ as the set of timed productions starting from state x_0 generated by A. The timed string associated to ρ is $w(\rho) = (e_1, t_1) \dots (e_n, t_n)$ (w for short when no confusion holds) and λ denotes the empty string. The timed language of A, denoted by $\mathcal{L}(A)$, is composed by the time strings associated to the executable productions in A starting from x_0: $\mathcal{L}(A) = \{w(\rho) \mid \rho \in \mathcal{R}(A, x_0)\}$.

Two AMTs $A_i = (Xi_i, E_i, \Delta_i, x_{0i})$, $i = 1, 2$ can share some events in $E_s = E_1 \cap E_2$ that synchronize the timed behaviours of the system. Such automata are referred to as modular automata.

Example : An example of two modular AMTs $A_1 = (X_1, E_1, \Delta_1, x_{10})$ and $A_2 = (X_2, E_2, \Delta_2, x_{20})$ is presented in Figure 2 where $E_1 = \{e_1, e_2, e\}$, $E_2 = \{e_3, e\}$ and $E_s = E_1 \cap E_2 = \{e\}$, e being the single synchronisation event.

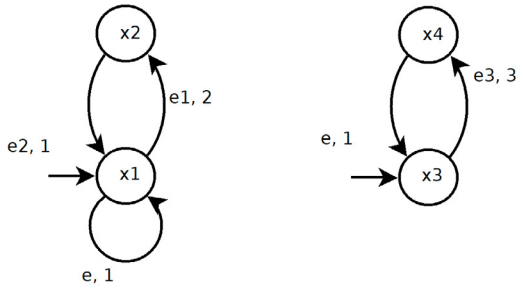

Fig. 2. Example of two modular AMTs

[1] A finite automaton (FA) is a four-tuple $A = (X, E, \Delta, x_0)$, where X is the set of states, E is the set of events, x_0 is the initial state, $\Delta \subseteq X \times E \times X$ is the transition relation and x_0 is the initial state (Cassandras and Lafortune (2008)).

3. TICK AUTOMATA

3.1 Tick automata for AMTs

In this section we transform an AMT into a tick automaton where the time is moving according to an arbitrary number of time increment of exactly one TU. For the purpose of abstracting the timing aspects, let us first introduce the following notations for each $x_i \in X$ and $e \in E$.

- $E(x_i) = \{e \in E \mid (x_i, e, \bullet, \bullet) \in \Delta\}$ is the set of events that are activated at x_i, i.e., that may occur at x_i after a certain amount of time.
- $\tau_{min}(x_i, e) = \min\{\tau \in \mathbf{N} \mid (x_i, e, \tau, \bullet) \in \Delta\}$ is the minimal value of time when e can occur at state x_i and $\tau_{max}(x_i) = \max\{\tau_{min}(x_i, e) : e \in E(x_i)\}$ if $E(x_i) \neq \emptyset$ and $\tau_{max}(x_i) = 0$, otherwise. Observe that $\tau_{max}(x_i)$ is the time from which any of the events in $E(x_i)$ may occur. But, according to the considered time semantic, the system may stay for ever at x_i and $\tau_{max}(x_i)$ is not the maximal sojourn time at x_i;
- $Y(x_i) = \{y_{i,0}, ..., y_{i,\tau_{max}(x_i)}\}$ is the set of extended states associated to x_i. For each extended state $y_{i,j}$, $i = 0, ..., \tau_{max}(x_i)$, we define an elementary time interval $I_j = [j, j+1)$ when $j < \tau_{max}(x_i)$ and $I_{\tau max}(x_i) = [\tau_{max}(x_i), +\infty)$. Observe that the extended state $y_{i,j}$ can be written as the pair (x_i, I_j);
- $Y(x_i, e) = \{y_{i,\tau_{min}(x_i,e)}, ..., y_{i,\tau_{max}(x_i)}\}$, $Y(x_i, e) \subseteq Y(x_i)$ is the subset of extended states when event e may occur at x_i of the underlying AMT.

Definition: The tick automaton of $A = (X, E, \Delta, x_{i_0})$ is defined as a finite state automaton $A_Y = (Y, E_Y, \Delta_Y, y_0)$ where:

- $Y = \bigcup_{x_i \in X} Y(x_i)$ is the set of extended states where each state $y_{i,j} \in Y(x_i)$ is associated to the system state $x_i \in X$ and to a timing information $j \in \{0, ..., \tau_{max}(x_i)\}$. When $j < \tau_{max}(x_i)$, $y_{i,j}$ indicates that the system may stay at x_i during the time interval $[j,\ j+1)$. When $j = \tau_{max}(x_i)$, $y_{i,\tau_{max}(x_i)}$ indicates that it may stay at x_i during any time interval $[k,\ k+1), k \in \mathbf{N}, k \geq \tau_{max}(x_i)$;
- $E_Y = E \cup \{ \textcircled{1} \}$ where $\textcircled{1}$ is an additional event associated to the clock and indicating that the time is moving from one TU;
- Δ_Y is the transition relation defined by
  (1) $(y_{i,j}, \textcircled{1}, y_{i,j+1}) \in \Delta_Y$ for $y_{i,j} \in Y(x_i)$ and $j < \tau_{max}(x_i)$. This indicates that a transition from $y_{i,j}$ to $y_{i,j+1}$ occurs when the system stays at state x_i during the time interval $[j,\ j+1)$. The transition then simply means that one unit of time has passed;
  (2) $(y_{i,\tau_{max}(x_i)}, \textcircled{1}, y_{i,\tau_{max}(x_i)}) \in \Delta_Y$ for $x_i \in X$. This indicates that the self-loop $\textcircled{1}$-transition from $y_{i,\tau_{max}(x_i)}$ to $y_{i,\tau_{max}(x_i)}$ occurs at each tick when the system stays at state x_i during $[\tau_{max}(x_i), +\infty)$;
  (3) $(y_{i,j}, e, y_{i',0}) \in \Delta_Y$ for $e \in E(x_i)$, $y_{i,j} \in Y(x_i, e)$, $x_{i'} \in X$. This indicates that a transition from $y_{i,j}$ to $y_{i',0}$ occurs when the system switches from state x_i to state $x_{i'}$ during the time interval $[j,\ j+1)$ if $\tau_{min}(x_i, e) \leq j < \tau_{max}(x_i)$ or during $[\tau_{max}(x_i, e), +\infty)$.

- $y_0 = y_{i_0,0}$, x_{i_0} being the initial state of the system. □

Observe that the event ① associated to the clock does not add nondeterminism in the tick automaton, and the tick automaton of a deterministic AMT is also a deterministic automaton. Observe also that the previous definition can be obviously extended to AMT with non negative real time parameters as far as this parameters are all multiples of a common timed increment $\delta t \in \mathbf{R}$. In our setting, the number of states of A_Y depends on both the number of states of A and the time parameters of A and $|Y| \leq |X| \times \max\{\tau_{min}(x_i, e) + 1 : e \in E(x_i), x_i \in X\}$.

Proposition 1. *To each timed production ρ of the AMT $A = (X, E, \Delta, x_{i_0})$ of the form of equation (2) corresponds exactly one (logical) production ρ_Y of the tick automaton $A_Y = (Y, E_Y, \Delta_Y, y_0)$.*

Proof: The proof is quite similar to the proof proposed in (Li et al. (2022)) for timed automata with constant times. In brief, let introduce the general form of a production ρ_Y in A_Y that begins from the state $y_{j_0,1}$:

$$\begin{aligned}\rho_Y : y_{j_0,0} \underbrace{\xrightarrow{①} y_{j_0,1} \cdots \xrightarrow{①} y_{j_0,d_1}}_{d_1} \xrightarrow{e_1} y_{j_1,0} \\ \xrightarrow{①} \cdots \to \ldots \xrightarrow{e_{n-1}} y_{j_{n-1},0} \\ \underbrace{\xrightarrow{①} y_{j_{n-1},1} \xrightarrow{①} \cdots \xrightarrow{①} y_{j_{n-1},d_n}}_{d_n} \xrightarrow{e_n} y_{j_n,0}\end{aligned} \tag{3}$$

where $y_{j_i,k} \in Y(x_{j_i}), e_i \in E, d_i \in \mathbf{N}, (y_{j_{i-1},k_i-1}, ①, y_{j_{i-1},k_i}) \in \Delta$ and $(y_{j_{i-1},d_i}, e_i, y_{j_i,0}) \in \Delta$, $k_i = 1, ...d_i$, $i = 1, ..., n$. Observe, according to the rule 2 in the definition of the transition relation Δ_Y that d_i can take arbitrary large values. Then, consider a timed production in A of the form of equation (2). By defining $d_i = \lfloor t_i - t_{i-1} \rfloor$, the production ρ_Y detailed in (3) becomes equivalent to (2). Note that the language of A_Y can be defined as $\mathcal{L}(A_Y) = \{w(\rho_Y) \mid \rho_Y$ is of the form of equation (3) and there exists $w = (e_1, t_1) \ldots (e_n, t_n) \in \mathcal{L}(A)$ with $d_i = \lfloor t_i - t_{i-1} \rfloor, i = 1, .., n\}$. In addition, the logical languages (ignoring the timing aspects) of A and A_Y are obviously the same. □

Example : The tick automata of the modular AMTs in Figure 2 are reported in Figure 3. In detail, the tick automaton of the top (resp. bottom) of Figure 3 is equivalent to the AMT on the left (resp. right) of Figure 2. Observe that an event e may occur at state x_1, from $y_{1,1}$, i.e., during time interval $[1, 2)$ or from $y_{1,2}$, during time interval $[2, +\infty)$, that is to say at any time value at least equal to 1 at x_1.

3.2 Synchronous composition of tick automata

In this section we introduce a synchronous composition for two tick automata that represent modular AMTs We know already that in a distributed system the only external events that can be synchronized are the events in set $E_s = E_1 \cap E_2$. In addition, there exists one internal event, generated by the clock, namely the event ①, that should also be used for synchronisation. When time is moving

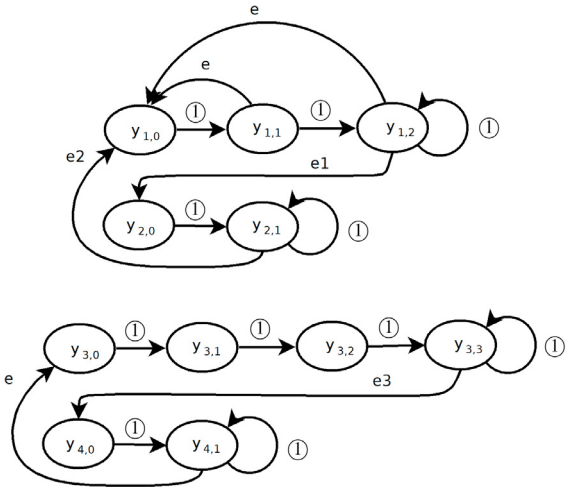

Fig. 3. Example of tick automata for AMTs in Figure 2

ahead in a given timed automaton, it should also move ahead in all automata that model parts of the system. The synchronous composition defined below acts as an automata product for non synchronisation events and as a parallel composition for events in set $E_s \cup \{①\}$.

Definition: Let $A_{Y1} = (Y_1, E_1 \cup \{①\}, \Delta_{Y1}, y_{01})$ and $A_{Y2} = (Y_2, E_2 \cup \{①\}, \Delta_{Y2}, y_{02})$ be two tick automata. The synchronous composition of A_{Y1} by A_{Y2} is defined as the tick automaton $S = A_{Y1} \otimes A_{Y2} = (Y_S, E_1 \cup E_2 \cup \{①\}, \Delta_S, (y_{01}, y_{02}))$ where $Y_S \subseteq Y_1 \times Y_2$ and the transition relation satisfies:

- $((y_1, y_2), e, (y_1', y_2)) \in \Delta_S$ if $e \in E_1 \setminus E_s$ and $(y_1, e, y_1') \in \Delta_{Y1}$;
- $((y_1, y_2), e, (y_1, y_2')) \in \Delta_S$ if $e \in E_2 \setminus E_s$ and $(y_2, e, y_2') \in \Delta_{Y2}$;
- $((y_1, y_2), e, (y_1', y_2')) \in \Delta_S$ if $e \in E_s \cup \{①\}$, $(y_1, e, y_1') \in \Delta_{Y1}$ and $(y_2, e, y_2') \in \Delta_{Y2}$. □

Observe that the synchronous composition can be trivially extended to three or more tick automata. Lemma 1 states that the synchronous product is conservative with respect to determinism.

Lemma 1. *The synchronous composition of two deterministic tick automata $A_{Y1} = (Y_1, E_1 \cup \{①\}, \Delta_{Y1}, y_{01})$ and $A_{Y2} = (Y_2, E_2 \cup \{①\}, \Delta_{Y2}, y_{02})$ is also as a deterministic tick automaton $S = (Y_S, E_1 \cup E_2 \cup \{①\}, \Delta_S, (y_{01}, y_{02}))$.*

Proof: Let us introduce the notation $Out(y)$ to refer to the set of events out of the extended state y. Consider a given state $(y_1, y_2) \in Y_S$ and the four following cases.

- For $e \in (E_1 \setminus E_s) \cap Out(y_1)$, (i) there exists a unique state $y_1' \in Y_1$ with $(y_1, e, y_1') \in \Delta_{Y1}$ (because A_{Y1} is deterministic); (ii) $e \notin Out(y_2)$ (because e is not a synchronization event). Consequently, only the first condition of the synchronous composition definition is satisfied and their exists a unique state $(y_1', y_2) \in Y_S$ such that $((y_1, y_2), e, (y_1', y_2)) \in \Delta_S$;
- For $e \in (E_2 \setminus E_s) \cap Out(y_2)$, (i) there exists a unique state $y_2' \in Y_2$ with $(y_2, e, y_2') \in \Delta_{Y2}$ (because

A_{Y2} is deterministic); (ii) $e \notin Out(y_1)$ (because e is not a synchronization event). Consequently, only the second condition of the synchronous composition definition is satisfied and their exists a unique state $(y_1, y_2') \in Y_S$ such that $((y_1, y_2), e, (y_1, y_2')) \in \Delta_S$;

- For $e = ①$, (i) there exists a unique state $y_1' \in Y_1$ with $(y_1, ①, y_1') \in \Delta_{Y1}$; (ii) there exists also a unique state $y_2' \in Y_2$ with $(y_2, ①, y_2') \in \Delta_{Y2}$. Consequently, only the third condition of the synchronous composition definition is satisfied and their exists a unique state $(y_1', y_2') \in Y_S$ such that $((y_1, y_2), ①, (y_1', y_2')) \in \Delta_S$;
- For $e \in E_s$, (i) there exists a unique state $y_1' \in Y_1$ with $(y_1, e, y_1') \in \Delta_{Y1}$ (because A_{Y1} is deterministic); (ii) there exists also a unique state $y_2' \in Y_2$ with $(y_2, e, y_2') \in \Delta_{Y2}$ (because A_{Y2} is deterministic). Consequently, only the third condition of the synchronous composition definition is satisfied and their exists a unique state $(y_1', y_2') \in Y_S$ such that $((y_1, y_2), e, (y_1', y_2')) \in \Delta_S$; □

Now we will show that safe (i.e. 1-bounded) T-time Petri nets with minimal firing times nets can be represented by synchronous compositions of AMTs. We will use state machine covering of safe TPNs, which is a well known result stating that every safe TPN is behaviorally equivalent to a composition of its sequential components (state machines). There are many algorithms for finding the state machine covering of a safe Petri net in the literature. A polynomial-time algorithm for finding minimal state machine cover of a safe Petri net has been proposed in (Karatkevich and Wisniewski (2020)). Another recent reference on decomposition of safe Petri nets into state machines components is (Bouvier et al. (2020)).

Note that state machine components are in fact finite automata (with current state corresponding to the place containing its unique marking) and we can endow them with timing to be AMT by simply taking the same minimal timing of a transition between two states in AMT as the one for the transition between the two places in T-time PN that correspond to these two states in the underlying AMT.

Proposition 2. *Let $(\mathcal{G}, \tau)$ be a T-timed PN with minimal time constraints and let $\mathcal{G}_1, \ldots, \mathcal{G}_K$ be the state machines that form the state machine cover of $\mathcal{G}$. Let $A_1 = (X_1, E_1, \Delta_1, x_{10})$,..., $A_K = (X_K, E_K, \Delta_K, x_{K0})$ be K AMTs that correspond to state machines $\mathcal{G}_1, \ldots, \mathcal{G}_K$, respectively and let $A_{Y,i} = (Y_i, E_i \cup \{①\}, \Delta_{Yi}, y_{0i})$, $i = 1, \ldots, K$ be their corresponding tick automata.*

Then, the synchronous composition of tick automata $S = A_{Y,1} \otimes ... \otimes A_{Y,K}$ encodes all timed productions of $(\mathcal{G}, \tau)$.

Proof: We first claim that every local tick automaton $A_{Y,i} = (Y_i, E_i \cup \{①\}, \Delta_{Yi}, y_{0i})$, $i = 1, \ldots, K$ corresponding to the AMT of the state machine $\mathcal{G}_i$ encodes all timed productions of the subnet $(\mathcal{G}_i, \tau)$. Note that the timed state machines are sequential timed systems, which can be viewed as a class of single clock timed automata with discrete time, hence the underlying AMT A_i and subsequently its tick automaton $A_{Y,i}$ preserve the timed behavior of the subnet $(\mathcal{G}_i, \tau)$. This first claim follows after transformation of the time behavior of state machine components of the T- Petri net, defined in the common semantic model for timed systems, timed transition systems (TTSs), into their corresponding tick automata as presented in Proposition 1, where every continuous transition (time elapsing) labeled by $\ell > 1$ can be replaced by the sequence of $\ell > 1$ consecutive tick events. Since state machines are simple (sequential) models, the resulting TTSs are in fact finite and deterministic (as we do not use non injective labeling) AMTs. Consequently, Proposition 1 can be applied. Note, finally, that the product of timed state machines has the same timed behavior as the product of tick automata $A_{Y,i}$. Since local timed behaviors of tick automata are the same as the timed behaviours of the corresponding TTSs and the synchronous product of tick automata preserves this timed behavior equivalence, we conclude that the product of the local tick automata is the same as the product of tick automata corresponding to TTSs, which is the timed behavior of the whole net. Notice that the synchronous product of tick automata corresponding to AMTs preserves their property that at each state of these tick automata the tick event can fire. This simply follows from the fact that the tick event acts as a shared (synchronization) event in the definition of synchronous product of tick automata.

Example : The synchronous composition of the two tick automata in Figure 3 is reported in Figure 4. The resulting structure has 19 extended states and one can notice that this size does not exceed $|Y_1| \times |Y_2| = 30$.

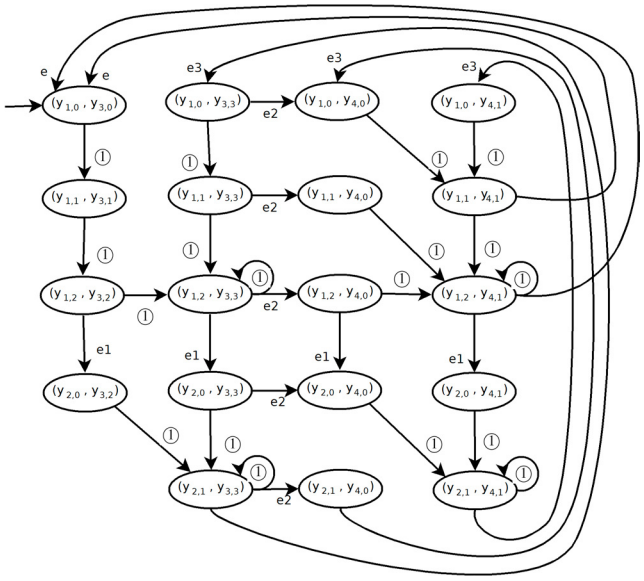

Fig. 4. Synchronous composition of A_{Y1} and A_{Y2} detailed in Figure 3.

4. CONCLUSION AND PERSPECTIVES

This paper has shown that a specific subclass of timed DESs with synchronisation events and minimal time constraints can be exactly represented as the synchronous product of modular tick automata with minimal times. This modular modeling abstracts the timing aspects and takes advantages of existing literature about logical automata.

First, our plan is to extend this work to automata with time intervals, where not only lower bounds but also upper bounds on the firing times are imposed. It is to be noted that proposing their synchronous product to be consistent with weak semantics of underlying safe T-time Petri nets as composition of sequential components (time state machines) associated with these automata, will be a challenging task. Nevertheless, we believe that at least for some subclasses of systems this will be possible.

It appears also that the proposed transformation is suitable to design observers in a distributed setting when some events are unobservable or when their occurrences deliver identical labels. For this purpose, we guess that modular observers can be designed and composed to obtain a global structure. We believe that modular observers preserve some (but not all) observation properties. The main advantages of such modular observers is also to prevent the combinatorial increase of the complexity resulting from the time abstraction and determinisation principle. One can also observe that tick automata contain some additional information that are not used to build the standard logical observers. In particular, each extended state of a tick automaton has a time domain that gives information about how long the system stays in a given state. Such information may be helpful for some applications.

In a general perspective, this contribution opens the way to modular verification and control of timed systems, especially for systems with different time scales, i.e., a large differences between the smallest and the highest minimal firing times involved in AMT. Designing modular logical or timed observers can facilitate the diagnosis of faults (Sampath et al. (1995), Zhang et al. (2018), Hadjicostis (2019)) in a non centralized setting. Consequently, one possible extension of this contribution is to develop modular diagnosers of timed systems. Another promising application is to study state-based and language-based opacity notions (Saboori and Hadjicostis (2013), Lin (2011), Tong et al. (2017), Ma et al. (2021)) in a distributed setting for timed DESs. For this purpose, the standard notion of opacity will be alleviated. In particular, it could be helpful to compute the time intervals during which one can be certain that a given secret is kept safe to intruders. Computing the time intervals during which the secret is revealed is also interesting. Finally, we aim to investigate modular supervisory control of certain classes of distributed timed DESs.

REFERENCES

Berard, B., Cassez, F., Haddad, S., Lime, D., and Roux, O.H. (2005). Comparison of the expressiveness of timed automata and time Petri nets. In *FORMATS 2005 - 3rd International Conference on Formal Modeling and Analysis of Timed Systems*, volume 3829, 211–225. Springer-Verlag.

Bouvier, P., Garavel, H., and Ponce-de León, H. (2020). Automatic decomposition of Petri nets into automata networks – a synthetic account. In R. Janicki, N. Sidorova, and T. Chatain (eds.), *Application and Theory of Petri Nets and Concurrency*, 3–23. Springer International Publishing, Cham.

Boyer, M. and Roux, O. (2008). On the compared expressiveness of arc, place and transition time Petri nets. *Fundamenta Informaticae*, 88, 225–249.

Brandin, B. and Wonham, W. (1994). Supervisory control of timed discrete-event systems. *IEEE Transactions on Automatic Control*, 39(2), 329–342.

Cassandras, C.G. and Lafortune, S. (2008). *Introduction to discrete event systems*. Springer, Boston, MA, USA.

D'Souza, D. and Thiagarajan, P.S. (2002). Product interval automata: a subclass of timed automata. In *Proceedings of FSTTCS*, 60–71.

Hadjicostis, C.N. (2019). *Estimation and Inference in Discrete Event Systems: A Model-Based Approach with Finite Automata*. Springer, Cham.

Karatkevich, A.G. and Wisniewski, R. (2020). A polynomial-time algorithm to obtain state machine cover of live and safe Petri nets. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 50(10), 3592–3597.

Komenda, J., Lahaye, S., and Boimond, J.L. (2010). Synchronous composition of interval weighted automata using tensor algebra of product semirings. In *IFAC Proceedings Volumes (IFAC-PapersOnline)*, volume 10, 318–323.

Lai, A., Lahaye, S., and Komenda, J. (2022). Observer construction for polynomially ambiguous max-plus automata. *IEEE Transactions on Automatic Control*, 67(3), 1582–1588.

Li, J., Lefebvre, D., Hadjicostis, C.N., and Li, Z.W. (2022). Observers for a class of timed automata based on elapsed time graphs. *IEEE Transactions on Automatic Control*, 67(2), 767–779.

Lin, F. (2011). Opacity of discrete event systems and its applications. *Automatica*, 47(3), 496–503.

Lin, L., Su, R., Brandin, B.A., Ware, S., Zhu, Y., and Sun, Y. (2019). Synchronous composition of finite interval automata. In *Proceedings of ICCA*, 578–583.

Ma, Z., Yin, X., , and Li, Z.W. (2021). Verification and enforcement of strong infinite- and k-step opacity using state recognizers. *Automatica*, 133.

Merlin, P. (1974). A study of the recoverability of computer systems. *Ph. D. Thesis, Computer Science Dept., University of California*.

Ramchandani, C. (1973). Analysis of asynchronous concurrent systems by timed Petri nets. *Ph. D. Thesis, Massachusetts Institute of technology*.

Saboori, A. and Hadjicostis, C.N. (2013). Verification of initial-state opacity in security applications of discrete event systems. *Information Sciences*, 246, 115–132.

Sampath, M., Sengupta, R., Lafortune, S., Sinnamohideen, K., and Teneketzis, D. (1995). Diagnosability of discrete-event systems. *IEEE Transactions on Automatic Control*, 40(9), 1555–1575.

Tong, Y., Li, Z.W., Seatzu, C., and Giua, A. (2017). Verification of state-based opacity using Petri nets. *IEEE Trans. on Automatic Control*, 62(6), 2823–2837.

Zhang, K., Liu, T., and Cheng, D. (2018). Observability of finite labeled transition systems. *IEEE Transactions on Automatic Control*, 63(6), 1591–1602.