Verifiers for the detection of timed patterns in discrete event systems

Dimitri Lefebvre, Zhiwu Li, Ye Liang

Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnline CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 55-28 (2022) 264–269

Verifiers for the detection of timed patterns in discrete event systems

Dimitri Lefebvre * **Zhiwu Li** ** **Ye Liang** ***

* *GREAH - Université Le Havre Normandie (e-mail: dimitri.lefebvre@ univ-lehavre.fr).*
** *School of Electro-Mechanical Engineering, Xidian University, Xi'an 710071, China and Institute of Systems Engineering, Macau University of Science and Technology, Taipa, Macau (e-mail: zhwli@xidian.edu.cn)*
*** *School of Electro-Mechanical Engineering, Xidian University, Xi'an 710071, China, (e-mail: liangye@stu.xidian.edu.cn)*

Abstract: This paper is about the detection of timed patterns for discrete event systems (DESs) that are modeled by a particular class of timed automata. A timed pattern is a set of behaviours characterized by a sequence of events, occurring in a given order and in a given time domain. The problem in which we are interested is to detect the occurrence of such timed patterns thanks to the analysis of the timed observations that is captured from the productions of the system. For this purpose we define a timed composition and propose two verifiers, a timed one and a logical one, that transform the detection problem of timed patterns into a state isolation problem. Safety and security applications are finally discussed.

Keywords: Timed automaton, Timed Pattern, Timed composition, Timed verifier, Logical verifier.

1. INTRODUCTION

The design of verifiers for discrete event systems (DESs) is helpful to solve a large variety of problems where one is interested in characterizing and detecting some specific behaviours based on the analysis of incomplete (maybe parcimonial) information. The main idea of a verifier is to transform the search of the behaviours of interest into a state isolation problem. For this purpose, the verifier aims to add some tags to the states of a system, which are assigned depending when the behaviours of interest have occurred. Various verifiers have been proposed for DESs and a detailed description of state isolation techniques with automata can be found in Hadjicostis (2019). Basically, a verifier results from the completely synchronous composition of an automaton that models the system by a second automaton that models the information of interest. Such techniques have been already used with success, in particular, for diagnosis issues. In such a case, the verifier is composed by two copies of the system that embed respectively the behaviours before and after the fault has occurred. From this structure it becomes possible to derive a logical observer that can be interpreted as a diagnoser, Cassandras and Lafortune (2008). As far as complexity issues are concerned, it is also possible to develop more efficient tests by using directly pair verifiers of polynomial complexity instead of observers that are of exponential complexity, Yoo and Lafortune (2002).

However, in numerous systems, the behaviours of interest are characterized not only by sequences of logical events but also by timing aspects. In order to discuss such timing aspects with DESs, timed automata, Alur and Dill (1994), or timed Petri nets, Cassandras and Lafortune (2008), have been studied. In particular, timed automata are finite state automata endowed with a time structure. Such timed models become necessary to deal with real-time applications, in particular, for scheduling or other control issues, Md T. Bin Waez (2013). As far as the timing aspects improve the information that one can capture from the execution of a system, timed models are also interesting for analysis and diagnosis purposes, Bouyer et al. (2005), Cassez and Tripakis (2009), Lin et al. (2018). This paper touches upon the design of verifiers that may lead to the detection of timed patterns for DESs, modeled by labeled timed automata. More precisely, we consider labeled timed automata with a single clock and right semi-open time intervals that are associated to the transitions, where the lower and upper bounds are multiples of a given time unit. The timed pattern detection is formulated as an isolation problem that is solved by designing first a timed verifier and second a pure logical verifier, both being helpful to decide whether a given timed pattern has occurred. The use of time stamps improves the detection of patterns. More precisely, patterns may exist that cannot be detected in a logical sense and that become diagnosable thank to the time stamps. In addition, adding timing specifications to the patterns enlarges the application fields. Such verifiers are obtained by introducing a timed composition. Different from the timed synchronous product already defined for various classes of timed automata, D'Souza and Thiagarajan (2002), Komenda et al. (2010), Lin et al. (2019), the proposed composition is inspired from the logical completely synchronous composition. In Section 2, the modeling aspects are detailed and a timed composition of automata is introduced. Based on this timed product, timed and logical verifiers are detailed in Section 3. Potential applications for safety, diagnosis, security and privacy issues are described in Section 4. Section 5 concludes the paper.

2. PRELIMINARIES

In this paper, we use $\mathbf{R}_+$ and $\mathbf{N}$ to denote the sets of non-negative real numbers and non-negative integers, respectively.

10.1016/j.ifacol.2022.10.352

Time is measured in time units (TUs). A right semi-open time interval is denoted by $I = [\tau_{min}, \tau_{max})$ and we refer to $\tau_{min}(I)$ and $\tau_{max}(I)$ as the left and right bounds of I. We denote a set of right semi-open time intervals over $\mathbf{R}_+$ as $\mathbf{I}$ and we will discuss in detail the particular case where $\tau_{min}(I)$, $\tau_{max}(I) \in \mathbf{N}$.

2.1 Timed automata

In the next, we consider a particular class of timed automata with a single clock that is reset at each event occurrence.

Definition 1. A timed automaton with a single clock (TFA) is a six-tuple $A = (X, E, \mathbf{I}, \Delta, \mathbf{S}, x_0)$, where X is a finite set of states, E is a finite set of events, $\mathbf{I}$ is a set of time intervals, $\Delta \subseteq X \times E \times \mathbf{I} \times X$ is a timed transition relation, $\mathbf{S}$ is the time semantics, and x_0 is an initial state.

In simple words, a TFA A is a finite automaton [1] endowed with a time structure that associates with each transition a time interval. More precisely, we say that (x, e, I, x') is defined (i.e., $(x, e, I, x') \in \Delta$) if the state x' is reachable from the state x by the occurrence of event e at any value of time $t \in I$ with $I \in \mathbf{I}$, where t is counted from the previous event occurrence. In this case, the transition from x to x' is said to be an (e, I)-transition.

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:

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

where $x_{j_i} \in X, e_i \in E, t_i \in \mathbf{R}_+, (x_{j_{i-1}}, e_i, I_i, x_{j_i}) \in \Delta$, $t_i - t_{i-1} \in I_i$ and $I_i \in \mathbf{I}$, $i = 1, \dots, n$ (with $t_0 = 0$). 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: $\mathcal{L}(A) = \{w(\rho) \mid \rho \in \mathcal{R}(A, x_0)\}$.

In order to handle the timing aspects, let us define $\Lambda(x) = \{(e, I) \in E \times \mathbf{I} \mid \exists x' \in X : (x, e, I, x') \in \Delta\}$ as the set of enabled (e, I)-transitions at x, $E(x) = \{e \in E \mid (e, I) \in \Lambda(x)\}$ as the set of events that are activated at x (for some values of time), $\mathbf{I}(x, e) = \{I \in \mathbf{I} \mid (e, I) \in \Lambda(x)\}$ as the set of time intervals associated to the event e at x and $I(x, e) = \{t \in \mathbf{R}_+ \mid \exists I \in \mathbf{I}(x, e) : t \in I\}$ as the corresponding set of time values. Similarly, $\mathbf{I}(x) = \cup_{e \in E(x)} \mathbf{I}(x, e)$ is the set of time intervals associated to state x and $I(x) = \{t \in \mathbf{R}_+ \mid \exists I \in \mathbf{I}(x) : t \in I\}$ is the corresponding set of time values. Each state $x \in X$ of a TFA may have several different sojourn times and $\tau_{min}(x)$ and $\tau_{max}(x)$ stand for the minimal and maximal sojourn times, respectively. In this work, we use the time semantics $\mathbf{S}$ defined as follows.

(1) The minimal sojourn time at state $x \in X$ is defined by $\tau_{min}(x) = \min(\tau_{min}(I(x)))$, if $\Lambda(x) \neq \emptyset$, and $\tau_{min}(x) = +\infty$, otherwise.
(2) The maximal sojourn time at state $x \in X$ is defined by $\tau_{max}(x) = \max(\tau_{max}(I(x)))$, if $\Lambda(x) \neq \emptyset$, and $\tau_{max}(x) = +\infty$, otherwise.
(3) A transition has to fire once the clock at x reaches $\tau_{max}(x)$.
(4) The clock is reset once a transition fires.

Definition 2. A TFA is said to be *complete* from the timed-logical perspective if for all $x \in X$, for all $e \in E$, we have $E(x) = E$ and $I(x, e) = \mathbf{R}_+$.

Completeness means that each event $e \in E$ and each value of time $t \in \mathbf{R}_+$ cause a state switch (including possible self-loops) at any $x \in X$. Observe that the completeness from a timed-logical perspective implies that from a logical perspective.

Definition 3. A TFA is said to be *deterministic* from a timed-logical perspective if the timed transition function satisfies the following property: $(x, e, I, x') \in \Delta$ and $(x, e', I', x'') \in \Delta$ imply, at least, one of the three cases: (i) $e \neq e'$; (ii) $I \cap I' = \emptyset$; (iii) $x' = x''$. For a deterministic TFA, we write $x' = \delta(x, e, I)$ if $(x, e, I, x') \in \Delta$.

Determinism from a timed-logical perspective means that, at any $x \in X$, each pair (e, t) with $e \in E$ and $t \in \mathbf{R}_+$ causes an unambiguous timed transition (or causes no transition at all). Determinism in the logical perspective implies determinism from a timed-logical perspective.

2.2 Labeled timed automata

In order to model the non deterministic aspects of a TFA from the perspective of observation, labeled TFAs are introduced. For this purpose, the set E is partitioned into two disjoint subsets $E = E_o \cup E_u$, where E_o and E_u represent the set of observable events and the set of unobservable events, respectively. A special label, namely empty label, denoted by ε, is used when unobservable events occur. Let Q be the set of observable labels and P be a labeling function that associates each observable event $e \in E_o$ to a label $q \in Q$ and each silent event $e \in E_u$ to ε. Note that one or more events can have the same label.

A timed projection $\mathcal{P} : (E \times \mathbf{R}_+)^* \to (Q \times \mathbf{R}_+)^*$ is defined as follows. Given $w \in (E \times \mathbf{R}_+)^*$, $\mathcal{P}(w(e, t)) = \mathcal{P}(w)(P(e), t)$ if $e \in E_o$; $\mathcal{P}(w(e, t)) = \mathcal{P}(w)$ if $e \in E_u$; $\mathcal{P}(\lambda) = \lambda$. The projection $w_q = \mathcal{P}(w)$ is the timed sequence of observations associated to w. We refer to $\mathcal{P}^{-1}(w_q)$ as the set of timed strings w in $\mathcal{L}(A)$ that satisfy $\mathcal{P}(w) = w_q$, i.e., $\mathcal{P}^{-1}(w_q) = \{w \in \mathcal{L}(A) \mid \mathcal{P}(w) = w_q\}$.

Definition 4. A labeled timed automaton (LTFA) is a three-tuple (A, Q_ε, P), where A is a TFA, $Q_\varepsilon = Q \cup \{\varepsilon\}$ is a set of output labels including the empty label, and P is a labeling function.

Example: The automaton in Fig. 1 illustrates an LTFA with three states: $X = \{x_1, x_3, x_3\}$, three events: $E = \{e_1, e_2, e_3\}$, and a single label a. The timed transition function and the observed labels are visualized near the edges. For example, we have $x_2 = \delta(x_1, e_1, [0, 2))$ and $P(e_1) = \varepsilon$. In addition, $\Lambda(x_2) = \{(e_2, [1, 3))\}$, $E(x_2) = \{e_2\}$, $\mathbf{I}(x_2, e_2) = \mathbf{I}(x_2) = \{[1, 3)\}$ whereas $I(x_2, e_2) = I(x_2) = [1, 3)$.

In the rest of this paper, we consider LTFA (A, Q_ε, P) with the following assumptions: (i) A is a deterministic TFA; (ii) each time interval $I \in \mathbf{I}$ is a right semi-open time interval that satisfies $\tau_{min}(I)$, $\tau_{max}(I) \in \mathbf{N}$; (iii) the time semantics is $\mathbf{S}$; (iv) the language $\mathcal{L}(A)$ and observed timed language are

[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).

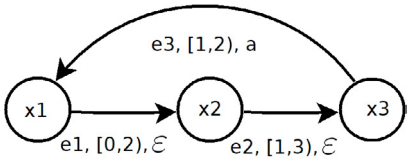

Fig. 1. An example of labeled timed finite automaton.

both live. Observe that the use of right semi-open time intervals I imposes $\tau_{max}(I) > \tau_{min}(I)$. Such models exclude any cyclic production [2] of duration equal to 0, and consequently the considered TFAs have no Zeno [3].

2.3 Timed product

Definitions 5 provides the notion of the timed product, i.e., completely synchronous composition, of two TFAs with the same set of event E.

Definition 5. Let $A_1 = (X_1, E, \mathbf{I}_1, \Delta_1, \mathbf{S}, x_{01})$ and $A_2 = (X_2, E, \mathbf{I}_2, \Delta_2, \mathbf{S}, x_{02})$ be two TFAs. The timed product $A_1 \times A_2$ is defined as a TFA $M = (X, E, \mathbf{I}, \Delta, \mathbf{S}, x_0)$ where

- $X \subseteq X_1 \times X_2$ is a finite set of states $x = (x_1, x_2)$ with $x_1 \in X_1$ and $x_2 \in X_2$;
- $\mathbf{I}$ is a set of intervals obtained as the intersections of the intervals $I_1 \in \mathbf{I}_1(x_1, e)$ and $I_2 \in \mathbf{I}_2(x_2, e)$ for all $(x_1, x_2) \in X$, and $e \in E$:
$$\mathbf{I} = \cup_{(x_1,x_2)\in X} \cup_{e\in E} \mathbf{I}(x_1, x_2, e),$$
$$\mathbf{I}(x_1, x_2, e) = \{I_1 \cap I_2 | I_1 \in \mathbf{I}_1(x_1, e), I_2 \in \mathbf{I}_2(x_2, e)\};$$
- Δ is the timed transition relation defined by $((x_1, x_2), e, I_1 \cap I_2, (x_1', x_2')) \in \Delta$ if there exist $e \in E$, $I_1 \in \mathbf{I}_1(x_1, e)$ and $I_2 \in \mathbf{I}_2(x_2, e)$ such that (i) $(x_1, e, I_1, x_1') \in \Delta_1$; (ii) $(x_2, e, I_2, x_2') \in \Delta_2$; (iii) $I_1 \cap I_2 \neq \emptyset$. Otherwise $((x_1, x_2), e, I_1 \cap I_2, (x_1', x_2'))$ is not defined;
- $x_0 = (x_{01}, x_{02})$ is the initial state.

Proposition 1. The timed product of two deterministic TFAs (in the timed-logical setting) is a deterministic TFA (in the timed-logical setting).

Proof: By contrapositive, consider two deterministic TFAs A_1 and A_2 and suppose that $M = A_1 \times A_2$ is not deterministic. Then, there exist $e \in E$, $I_1 \in \mathbf{I}_1(x_1, e)$ and $I_2 \in \mathbf{I}_2(x_2, e)$ such that $((x_1, x_2), e, I_1 \cap I_2, (x_1', x_2')) \in \Delta$ and $((x_1, x_2), e, I_1 \cap I_2, (x_1'', x_2'')) \in \Delta$ with $(x_1', x_2') \neq (x_1'', x_2'')$. So we have $(x_1, e, I_1, x_1') \in \Delta_1$, $(x_1, e, I_1, x_1'') \in \Delta_1$ and also $(x_2, e, I_2, x_2') \in \Delta_2$, $(x_2, e, I_2, x_2'') \in \Delta_2$. By $x_1' \neq x_1''$ or $x_2' \neq x_2''$, A_1 or A_2 is not deterministic. □

Given two deterministic TFAs A_1 and A_2, and their timed product M, we write $(x_1', x_2') = \delta((x_1, x_2), e, I_1 \cap I_2)$ if $((x_1, x_2), e, I_1 \cap I_2, (x_1', x_2')) \in \Delta$. Obviously, the timed language of M also satisfies $\mathcal{L}(M) = \mathcal{L}(A_1) \cap \mathcal{L}(A_2)$.

2.4 Timed pattern

Given a set of events E, a timed pattern $\Sigma_T(E)$ is defined as a set of timed strings: $\Sigma_T(E) \subseteq (E \times \mathbf{R}_+)^*$. Here, we restrict the discussion to timed patterns $\Sigma_T(E)$ that can be represented with a deterministic and complete TFA called a *timed pattern automaton* (TPA).

[2] A cyclic production - a cycle for short - in A is a production such that the initial and final states are identical and a mininal cycle is a cycle composed of distinct states (excepted the initial and final states that are identical).

[3] A Zeno is a behaviour where an infinite number of events occur in finite time.

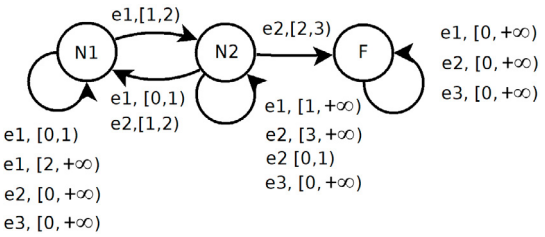

Fig. 2. An example of timed pattern.

Definition 6. Given a timed pattern $\Sigma_T(E)$ with E being a set of events, a timed pattern automaton (TPA) is a deterministic and complete timed finite automaton $TP = (X_F, E, \mathbf{I}_F, \delta_F, \mathbf{S}, x_{F0}, F)$ with timed language $\mathcal{L}(TP) = (E \times \mathbf{R}_+)^*$ and *accepted* timed language $\mathcal{L}_F(TP) = \Sigma_T(E)$, satisfying

- X_F is a finite set of states such that $X_F = N \cup \{F\}$, where $N = \{N_1, N_2, \ldots\}$ is a subset of one or more states and F is a unique final state indicating that the pattern of interest has occurred;
- $\mathbf{I}_F$ is a set of time intervals I_F with $\tau_{min}(I_F) \in \mathbf{N}$ and $\tau_{max}(I_F) \in \mathbf{N} \cup \{+\infty\}$;
- δ_F is a timed transition function such that for each $x_F \in X_F$ and $e \in E$, $E(x_F) = E$ and $I(x_F, e) = \mathbf{R}_+$ hold;
- $x_{F0} = N_1$ is the initial state;

Once the TPA reaches the final state F, it will stay in F eternally. This property will be used in the verifiers to be defined next, to tag with F some timed productions (the ones that have experienced the pattern of interest at least once). Observe that the proposed TPAs are suitable to define not only time patterns with (e, I)-transitions that are identical to the (e, I)-transitions of A, but also (e, I')-transitions with $I' \neq I$. Consequently, they are suitable to represent a large variety of patterns with timing specifications that are independant from the timing specifications of a system.

Example: The automaton in Fig. 2 illustrates a TPA for a set of events $E = \{e_1, e_2, e_3\}$. Here the pattern of interest is composed by the occurrence of the event e_1 in time interval $[1, 2)$ followed immediatelly, i.e., without any occurrence of e_3, by the occurrence of the event e_2 in time interval $[2, 3)$. All other behaviours are considered as normal behaviours. The set of states is $X_F = \{N_1, N_2, F\}$. Observe that the TPA is complete in the timed-logical setting.

3. TIMED AND LOGICAL VERIFIERS

In this section we detail successively the design of a timed verifier and a logical verifier for a labeled TFA A. Such a verifier encodes a given timed pattern.

3.1 Timed verifier

Definition 7. Let (A, Q_ε, P) be a LTFA that satisfies the assumptions detailed in Section 2.2, with $A = (X, E, \mathbf{I}, \delta, \mathbf{S}, x_0)$ and $\Sigma_T(E)$ be a timed pattern, the timed verifier $V = (X_V, E, \mathbf{I}_V, \delta_V, \mathbf{S}, x_{0V})$ of (A, Q_ε, P) with respect to $\Sigma_T(E)$ is defined by the timed product $V = A \times TP$ where TP is the TPA of $\Sigma_T(E)$.

Such a verifier is useful to transform the pattern detection problem into a state isolation problem: V explicitly specifies

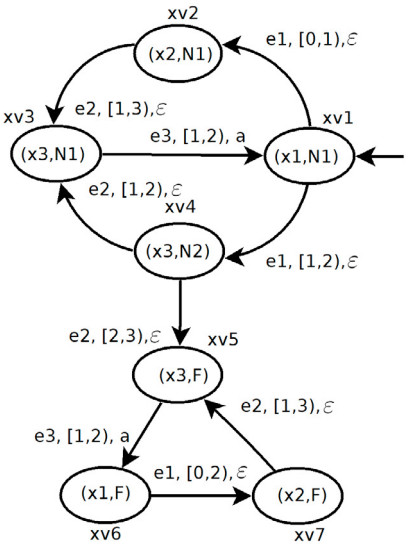

Fig. 3. Timed verifier of the LTFA in Fig. 1 and timed pattern of Fig. 2.

the pattern occurrences because each verifier state that results from, at least, one occurrence of the timed pattern is tagged with F. This method is inspired from the design of logical verifiers, Hadjicostis (2019). The space complexity of the timed verifier is $O(\mid X \mid \times \mid X_F \mid)$.

Proposition 2. Given an TFA A, a timed pattern automaton TP, and their timed verifier V, we have $\mathcal{L}(V) = \mathcal{L}(A)$.

Proof: The TFAs A, TP and V satisfy $\mathcal{L}(V) = \mathcal{L}(A) \cap \mathcal{L}(TP)$ and $\mathcal{L}(TP) = (E \times \mathbf{R}_+)^*$. Consequently, $\mathcal{L}(V) = \mathcal{L}(A)$ holds. □

Example: The automaton in Fig. 3 is the timed verifier V resulting from the timed product of the TFA in Fig. 1 by the TPA TP in Fig. 2. The set of verifier states $\{x_{V1}, ..., x_{V7}\}$ can be partitioned into a subset of states $\{x_{V1}, x_{V2}, x_{V3}, x_{V4}\}$ that result from the non-occurrence of the timed pattern $\Sigma_T(E)$ and a subset of states $\{x_{V5}, x_{V6}, x_{V7}\}$ that result from the occurrence of $\Sigma_T(E)$, i.e., tagged with F and taking the form $(\bullet, F)$.

3.2 Logical verifier

For the purpose of abstracting the timing aspects of the timed verifier, a logical verifier is also proposed. Let us first define for each $x_V \in X_V$, the set of integer values $TOI(x_V) = \{0, ..., \tau_{max}(x_V) - 1\}$. Each $j \in TOI(x_V)$ means that the system may stay at x_V within the time interval $[j,\ j+1)$ of width 1.

Definition 8. The logical verifier of V is defined as a DFA $V_Y = (Y, Q_Y, \delta_Y, y_0)$ where:

- $Y = \{y_{i,j}\}$ is a set of states where each state $y_{i,j}$ is associated to a verifier state $x_{Vi} \in X_V$ and a timing information variable $j \in TOI(x_{Vi})$ that indicates that the system may stay at x_{Vi} during the time interval $[j,\ j+1)$;
- $Q_Y = Q \cup \{⓪, ①\}$ is an alphabet, where Q is the set of labels of the considered LTFA: ⓪ and ① are two additional labels related to time: ⓪ means that time is freezing and ① means that time is moving from one TU;
- δ_Y is a transition function defined by

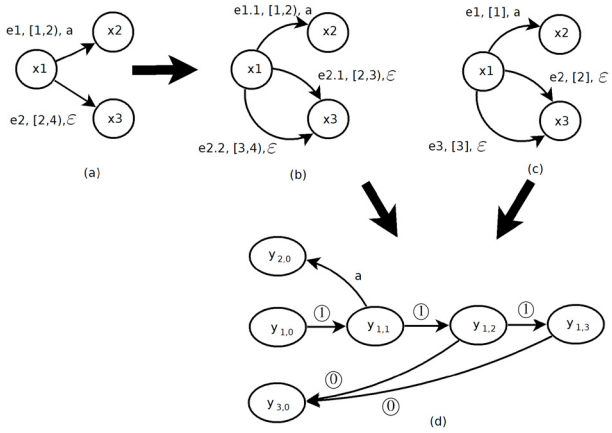

Fig. 4. Construction of the extended automaton.

(1) $y_{i,j+1} = \delta_Y(y_{i,j}, ①)$ for $x_{Vi} \in X_V$ and $j \in TOI(x_{Vi})$, $j < \tau_{max}(x_V) - 1$. This indicates that a switch from $y_{i,j}$ to $y_{i,j+1}$ occurs when the verifier stays at state x_{Vi} during the time interval $[j,\ j+1)$,
(2) $y_{i',0} = \delta_Y(y_{i,j}, ⓪)$ for $x_{Vi}, x_{Vi'} \in X_V$ and $j \in TOI(x_{Vi})$ if there exist $e \in E$ and $I \in \mathbf{I}_V$ such that $x_{Vi'} = \delta_V(x_{Vi}, e, I)$, $[j,\ j+1) \subseteq I$ and $P(e) = \varepsilon$. This indicates that a switch from $y_{i,j}$ to $y_{i',0}$ occurs when the verifier switches silently from state x_{Vi} to state $x_{Vi'}$ during the time interval $[j,\ j+1)$,
(3) $y_{i',0} = \delta_Y(y_{i,j}, q)$ for $x_{Vi}, x_{Vi'} \in X_V$, $j \in TOI(x_{Vi})$ and $q \in Q$ if there exists $e \in E$ such that $x_{Vi'} = \delta_V(x_{Vi}, e, I)$, $[j,\ j+1) \subseteq I$ and $P(e) = q$. This indicates that a switch from $y_{i,j}$ to $y_{i',0}$ occurs when the verifier switches from state x_{Vi} to state $x_{Vi'}$ during the time interval $[j,\ j+1)$ by delivering the label q.

- $y_0 = y_{i,0}$, where x_{Vi} is the initial state of the verifier.

The space complexity of the logical verifier is $O(\mid X \mid \times \mid X_F \mid \times \max\{\tau_{max}(x_V) : x_V \in X_V\})$.

Proposition 3. The extended automaton V_Y encodes all timed productions of V.

Proof: As far as each time interval $I_V \in \mathbf{I}_V$ manipulated by the verifier is a right semi-open interval $I_V = [\tau_{min}, \tau_{max})$, with $\tau_{min}(I_V)$, $\tau_{max}(I_V) \in \mathbf{N}$, it is possible to decompose I_V as $I_V = [\tau_{min}, \tau_{min}+1) \cup [\tau_{min}+1, \tau_{min}+2) \cup ... \cup [\tau_{max}-1, \tau_{max})$ and to duplicate each timed transition in order to transform V in a TFA that has only timed transitions with intervals of the form $[j, j+1)$ (see Figs. 4a and 4b). The timing information of such transitions can be represented implicitly by j and consequently the interval $I(x_V)$ can be abstracted by the set $TOI(x_V)$. The resulting extended automaton V_Y (see Fig. 4d) is similar to the extended automaton that has been obtained from TFA with single values of time Li et al. (2021) (see Fig. 4c). Consequently, the reasoning to prove Proposition 3 is the same as the one detailed in Li et al. (2021). □

Example: The automaton in Fig. 5 is the logical verifier V_Y of the TFA in Fig. 1 and TP in Fig. 2. This verifier has 17 states and each state is of the form $y_{i,j}$ that corresponds to a given timed verifier state x_{Vi} and a given time of interest j. It is worth noting that this verifier can be viewed as a logical structure with two particular events ⓪ and ①.

Algorithm 1. Design of the logical verifier V_Y

Require: An LTFA (A, Q_ε, P) and its timed verifier $V = (X_V, E, \mathbf{I}_V, \delta_V, \mathbf{S}, x_{V0})$

Ensure: A logical verifier $V_Y = (Y, Q_Y, \delta_Y, y_0)$

$Y \leftarrow \emptyset$, $\delta_Y \leftarrow \emptyset$, $Q_Y \leftarrow Q \cup \{⓪, ①\}$
for each $x_{Vi} \in X_V$ **do**
  for each $j \in TOI(x_{Vi})$ **do**
    $Y \leftarrow Y \cup \{y_{i,j}\}$
    if $j > 0$ **then**
      $\delta_Y(y_{i,j-1}, ①) = y_{i,j}$
    end if
  end for
end for
for each $y_{i,j} \in Y$ **do**
  for each $y_{i',0} \in Y$ **do**
    if $(\exists e \in E) \wedge (\exists I \in \mathbf{I}_V)$ **with** $x_{Vi'} = \delta_V(x_{Vi}, e, I)$, $[j,\ j+1) \subseteq I$, $P(e) = \varepsilon$ **then**
      $\delta_Y(y_{i,j}, ⓪) \leftarrow y_{i',0}$
    end if
    if $(\exists e \in E) \wedge (\exists I \in \mathbf{I}_V)$ **with** $x_{Vi'} = \delta_V(x_{Vi}, e, I)$, $[j,\ j+1) \subseteq I$, $P(e) = q$ **then**
      $\delta_Y(y_{i,j}, q) \leftarrow y_{i',0}$
    end if
  end for
end for
$y_0 \leftarrow y_{i,0}$ **for** $x_{Vi} = x_{V0}$

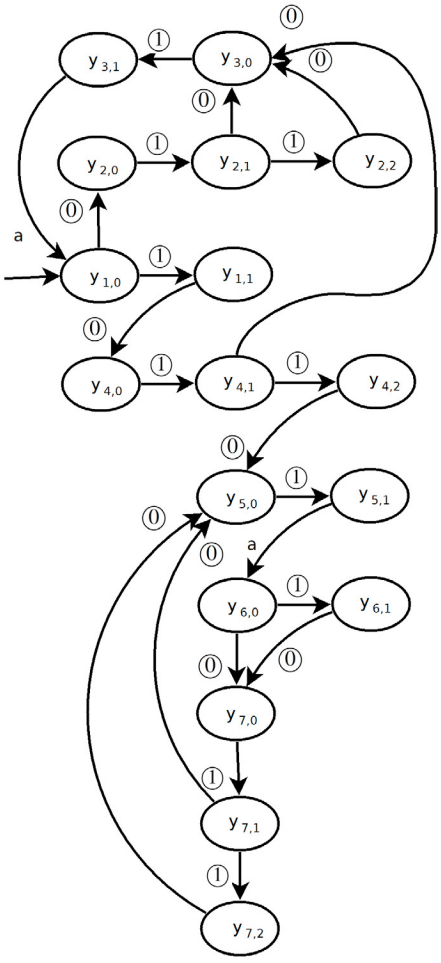

Fig. 5. Logical verifier of the LTFA in Fig. 1 and timed pattern of Fig. 2.

4. APPLICATIONS

The systematic design of verifiers for timed patterns opens a large variety of applications in the domains of safety and security.

4.1 Safety and diagnosis applications

In our opinion the obtained verifiers are helpful for the diagnosis of fault patterns that include timing aspects. The problem of fault diagnosis for DESs aims to detect whether faults have occurred in a given system, based on the observation of the system execution. Numerous results have been developed based on automata, or Petri nets, Cassandras and Lafortune (2008), Sampath et al. (1995). In particular, labeled finite state automata have been introduced in order to capture exlicitly the observed information, Zhang et al. (2018). In addition, diagnosability checking focuses on determinating whether the faults can be detected and distinguished with certainly, Hadjicostis (2019). Fault patterns have been introduced to extend the notion of simple faults and to describe complex faults that cannot be represented by single events, Jeron et al. (2006). Fault patterns are suitable to represent a broad class of undesirable behaviours as multiple faults, intermittent faults, and so on. Pattern diagnosability analysis has been also investigated in Ye and Daguet (2012), Yan et al. (2010), Yoo and Lafortune (2002), Hadjicostis (2019). It is worth noting that the former works are based on a logical analysis of the sequences of observations and ignore the timing aspects. In Tripakis (2002), Cassez and Tripakis (2009), some results have been proposed in order to extend fault diagnosis issues with timed automata. Unfortunately the results are reserved for simple faults represented by single events.

The proposed verifiers can be used for the diagnosis of timed patterns. In particular, the logical verifiers previously detailed are very similar to the extended automata that have been obtained from timed automata with single values of time, Li et al. (2021). In this former work, the authors have proved that timed observers are computable for such structures by a slight adaptation of standard methods, Cassandras and Lafortune (2008). The advantages of such timed observers is to refine the state estimation, compared to the estimation obtained with a pure logical observer, i.e., an observer that ignores the timing aspects. Consequently, one possible extension of this contribution is to turn out the logical verifier into a timed observer and then a timed diagnoser useful to detect directly the timed patterns of interest.

4.2 Security and privacy applications

Motivated by security and privacy considerations, there exist secret behaviors of a system that should be protected from intruders. The notion of opacity has been introduced for that purpose. A system is said to be *opaque* if no intruder is able to infer the truth of a predicate representing the secret behavior. DESs, in particular automata, have been intensively used for opacity verification, Lin (2011), Saboori and Hadjicostis (2013), Tong et al. (2017), Ma et al. (2021). There exist diverse notions of opacity according to the characteristics of the DES models, the types of secrets, the privacy requirements, and the intruder ability to collect and process observation information. In the context of finite state automata, two important notions are *language-based opacity*, including strong and weak opacity, Lin (2011) and *state-based opacity*, including current-state

opacity, initial-state opacity, K-step and infinite-step opacity Saboori and Hadjicostis (2013), Tong et al. (2017), Ma et al. (2021). In the first case, the secret, is described as a sub-language and in the second case it is described as a subset of states. As far as timing aspects are considered, opacity notions become more complicated. In particular, Cassez (2009) proves the language-based opacity problem is undecidable for timed automata defined according to the general framework introduced in Alur and Dill (1994). Wang et al. (2018) verify the notion of language-based opacity for real-time automata with a single clock and whose transitions are associated with time intervals. Gao et al. (2020) propose an algorithm of state estimation for timed automata where the lower and upper bounds of all time intervals are integers and Li et al. (2021) study a similar problem for timed automata with single values of time. Both contributions can be extended to study opacity notions in a timed setting.

The proposed verifiers can be used to study language-based opacity notions in a timed setting. Two problems will be studied in our future works. The first problem is to define and characterize some secret behaviours as timed patterns. Then, the logical verifiers previously detailed make it possible to analyse the conditions in which the behaviours of interest remain secret. The second problem is to define the secret behaviours by logical patterns and to relaxe the opacity conditions by introducing some timing aspects. In particular, it could be helpful to compute the time intervals during which one can be certain that the secret is kept safe to intruders. Computing the time intervals during which the secret is revealed is also interesting.

5. CONCLUSION

This paper has detailed the design of two verifiers (a timed one and a logical one) that can be used to check the occurrence of a large variety of timed patterns for DESs. For this purpose, the considered systems and the timed patterns have been described with labeled timed automata with a single clock and right semi-open time intervals that are associated to the transitions, where the lower and upper bounds are multiples of a given time unit. The design of such verifiers is a preliminary work to explore promising research directions in the field of diagnosis and privacy including timing aspects. Such applications are our future directions of research. Improving the numerical complexity of the approach and considering a more realistic case study also belong to our perspectives.

REFERENCES

Alur, R. and Dill, D. (1994). A theory of timed automata. *Theoretical Computer Science*, 126(2), 183–235.

Bouyer, P., Chevalier, F., and D'Souza, D. (2005). Fault diagnosis using timed automata. *Lecture Notes in Computer Science*, 3441, 219–233.

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

Cassez, F. (2009). The dark side of timed opacity. In *Proc. the 3rd Int. Conf. and Workshops on Advances in Information Security and Assurance; Seoul, Korea*, 21–30.

Cassez, F. and Tripakis, S. (2009). Fault diagnosis of timed systems. In *Proc. of Communicating Embedded Systems – Software and Design*.

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

Gao, C., Lefebvre, D., Seatzu, C., Li, Z., and Giua, A. (2020). A region-based approach for state estimation of timed automata under no event observation. In *Proc. of IEEE Int. Conf. on Emerging Technologies and Factory Automation.*, volume 1, 799–804.

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

Jeron, T., Marchand, H., Pinchinat, S., and Cordier, M.O. (2006). Supervision patterns in discrete event systems diagnosis. In *Proc. of 8th Int. Workshop on Discrete Event Systems*.

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

Li, J., Lefebvre, D., Hadjicostis, C.N., and Li, Z.W. (2021). Observers for a class of timed automata based on elapsed time graphs. *IEEE Trans. on Automatic Control*.

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 *Proc. of ICCA*, 578–583.

Lin, L., Wonham, W., and Su, R. (2018). Timed discrete-event systems are synchronous product structures. *ArXiv*.

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

Md T. Bin Waez, J. Dingel, K.R. (2013). A survey of timed automata for the development of real-time systems. *Computer Science Review*, 9, 1–26.

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 Trans. 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.

Tripakis, S. (2002). Fault diagnosis for timed systems. In *Proc. of the 7th Int. Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems*, 205–224.

Wang, L.T., Zhan, N.J., and An, J. (2018). The opacity of real-time automata. *IEEE Trans. on Computer Aided Design of Integrated Circuits and Systems*, 11(37), 2845–2856.

Yan, Y., Ye, L., and Dague, P. (2010). Diagnosability for patterns in distributed discrete event systems. In *Proc. of the 21st Int. Workshop on Principles of Diagnosis DX'10, Portland, OR Etats-Unis*, 345–352.

Ye, L. and Daguet, P. (2012). A general algorithm for pattern diagnosability of distributed discrete event systems. In *Proc. of the 24th IEEE Int. Conf. on Tools with Artificial Intelligence*, 130–137.

Yoo, T. and Lafortune, S. (2002). Polynomial-time verification of diagnosability of partially observed discrete-event systems. *IEEE Trans. on Automatic Control*, 47(9), 1491–1495.

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