Diagnosability of event patterns in safe labeled time Petri nets: a model-checking approach
Yannick Pencolé, Audine Subias

To cite this version:
Yannick Pencolé, Audine Subias. Diagnosability of event patterns in safe labeled time Petri nets: a model-checking approach. IEEE Transactions on Automation Science and Engineering, Institute of Electrical and Electronics Engineers, inPress, 10.1109/TASE.2020.3045565. hal-03139863

HAL Id: hal-03139863
https://hal.laas.fr/hal-03139863
Submitted on 12 Feb 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Abstract—Checking the diagnosability of a time discrete event system usually consists in determining whether a single fault event can always be identified with certainty after a finite amount of time. The aim of this paper is to extend this type of analysis to more complex behaviors, called event patterns, and to propose an effective method to check diagnosability with the use of model-checking techniques. To do so, we propose to convert the pattern diagnosability problem into checking a linear-time property over a specific time Petri net.

Note to Practitioners—This paper is motivated by the problem of improving the monitoring and the supervision of systems like automated and robotised manufacturing systems. Based on a model of the system, the paper proposes a method to assert with certainty whether the available set of sensors will always provide enough information to ensure that a complex and unexpected behavior has not happened in the system. The proposed method uses a publicly available model-checking tool to perform this analysis.

Index Terms—Fault Diagnosis, Diagnosability, Pattern, Time Petri nets, Discrete Event Systems (DES).

I. INTRODUCTION

Diagnosability is the property of a partially observed dynamical system that asserts whether it is always possible to diagnose with certainty the occurrence of an anticipated failure [1], [2]. This paper addresses the problem of checking diagnosability in timed discrete event systems such as automated and robotised manufacturing systems, communication protocols. As opposed to most of the previous work the anticipated type of failures are not represented as single fault events but as more complex event assemblings called event patterns [3], [4], [5]. This paper proposes a formal and effective method to check pattern diagnosability in timed systems that can be modeled as safe labeled time Petri nets. The first contribution of the paper is a formal characterization of the pattern diagnosis problem in timed system that extends the one presented in [6] for untimed labeled Petri nets. The second contribution is a formal extension of the single fault diagnosability problem as initially defined in [7] to pattern diagnosability. The proposed definition is language-based and does not rely on the actual formalism (time Petri net, time automata,...). We formally prove that this definition encompasses the single fault diagnosability definition of [7].

The third contribution is about a way to turn the pattern diagnosability problem into a model-checking problem [8], [9], [6], by building a specific twin-plant based on a set of products between the system’s model and the pattern. The paper proposes an effective solution that uses the off-the-shelf model-checker TINA [10] and presents experimental results.

Yannick Pencolé was with LAAS-CNRS, CNRS, Univ. Toulouse, France, Audine Subias was with LAAS-CNRS, INSA, Univ. Toulouse, France

II. RELATED WORK

The problem of fault diagnosis and diagnosability in DESs was introduced a few decades ago in [1], [2] and a survey can be found in [11]. For untimed systems and in the framework of automata, [2] proposed to check the diagnosability by detecting fault-indeterminate cycles in a global diagnoser. The algorithm is however exponential in the number of states of the global behavior model. Polynomial algorithms have then been proposed, directly based on a synchronisation of two copies of the behavioral model (twin-plant machine in [12] and verifier in [13]), that look for a pair of infinite behaviors of the system with the same observable projection, one faulty and one non-faulty, which proves the system is not diagnosable. More recently, this diagnosability problem has been revisited using Petri nets [14]. It was introduced in [15] for unbounded Petri nets by the use of a diagnoser built from a coverability graph. [16] proposes the definitions of weak and strong diagnosability and solve the problem by using net unfolding techniques. [17] considers a structural approach to give a sufficient condition for diagnosability relying on T-invariants and formulates the problem as a linear programming problem. [18] gives sufficient conditions for diagnosability and undiagnosability of a fault transition based on the notion of the so called g-markings [19], representing a set of inequalities to be solved. [20] updates the verifier to analyse diagnosability on bounded Petri nets based on minimal explanations to reduce the computational complexity. Similarly, minimal explanations are used in [21] for bounded nets and in [22] for unbounded nets, for the construction and the analysis of a compact state space representation called Basis Reachability Graph (BRG) [23]. The approach is extended to labeled Petri nets where different transitions can share the same label in [24]. More recently [25] proposes an approach for bounded and unbounded Petri nets from the coverability graph. [26] proposes some reduction rules to apply on the system model that preserve the diagnosability property and improve the computation efficiency.

For timed discrete event systems (TDES), the literature about fault diagnosis and diagnosability is far less developed. It has been introduced in [7] on timed automata. A fault is said to be diagnosable if there always exists a finite amount of time $\tau$ after the occurrence of the fault to assert its occurrence with certainty. If the diagnosability property holds for a fixed time $\tau$, then the system is $\tau$-diagnosable. In [27], another diagnosability property based on time ticks is analyzed: the method relies on a timed product of finite-state generators. In [28], the decidability and computational complexity of the problem is presented. The work of [9] reviews the algorithms for checking diagnosability in automata and timed automata.
and shows that both diagnosability problems can be reduced to a Büchi emptiness problem based on a twin-plant as defined in [7]. In the context of Petri nets, the work of [29] deals with P-time Petri nets under partial observation. The method is based on the synthesis of a modified state observer and on a schedulability analysis using linear programming techniques. Finally, [30] proposes a method for the diagnosability analysis of labeled T-time Petri nets under partial observation. The authors develop an approach based on the Modified State Class Graph proposed in [31] to perform \( \tau \)-diagnosability analysis in two steps: they search for ambiguous sequences and, if such sequences exist, they check for time ambiguities longer than \( \tau \). To the best of our knowledge, this last work is the closest to our proposal which consists in extending diagnosability analysis to behavioral event patterns as defined in [6]. The difference is that our proposal deals with the diagnosability problem (looking for a time \( \tau \) sequences exist, they check for time ambiguities longer than \( \tau \)).

The related works presented hereabove only deal with single faults. Diagnosability of more complex behaviors has also been investigated but only for untimed DES. In [32], a pattern is defined as a temporal logic specification over a finite-state automaton. A fault is then defined as a behavior that \textit{does not match} the pattern (that is called a specification in [32]). To check diagnosability, they propose an \textit{ad hoc} algorithm that builds the part of interest of the Kripke structure of the twin plant by synchronizing faulty and non-faulty behaviors. In [33], the patterns are defined as automata and they define the pattern diagnosability problem in automata: it is possible to always assert that the system \textit{matched} a pattern after a finite sequence of observations. As in [32] the authors propose an \textit{ad hoc} algorithm that builds the Kripke structure of the twin plant by successive synchronizations. In [33], the same problem is solved in a decentralized manner by aggregating pattern recognizers. In [34], patterns define some contexts represented as regular expressions and the authors propose a method to solve the pattern diagnosis problem on a class of untimed DES called active systems. [4] introduces the notion of patterns into labeled Petri nets. An unfolding approach is then applied on an updated twin-plant. More recently, [6] extends the work of [4] to more sophisticated patterns and proposes an effective method to automatically analyze the diagnosability.

III. BACKGROUND ON LABELED TIME PETRI NETS

A. Definition and Semantics

The labeled time Petri nets [35] constitute a specific class of Petri nets that maps any transition to a label and a static time interval that rules the transition firing. Let \( I_{\mathbb{Q}^+} \) be the set of nonempty real interval with nonnegative rational end-points.

**Definition 1** (LTPN). A Labeled Time Petri Net (LTPN for short) is a 6-tuple \( N = \langle P, T, A, E, \ell, I_\tau \rangle \) such that:

- \( P \) is a finite set of places;
- \( T \) is a finite set of transitions \((P \cap T = \emptyset)\);
- \( A \subseteq (P \times T) \cup (T \times P) \) is a binary relation that models the arcs between the places and the transitions;
- \( E \) is a finite set of transition labels;
- \( \ell : T \to E \) is a transition labeling function;
- \( I_\tau : T \to I_{\mathbb{Q}^+} \) is a static interval function.

Throughout this paper, the static interval \( I_\tau(t) \) of a transition \( t \) is either closed \((a, b]\) or semi-closed \((a, +\infty[\). The preset \( \text{pre}(t) \) of a transition \( t \) is the set of (input) places \( \text{pre}(t) = \{p \in P : (p, t) \in A\} \) and the postset \( \text{post}(t) \) of a transition \( t \) is the set of (output) places \( \text{post}(t) = \{p \in P : (t, p) \in A\} \).

A state of an LTPN is a couple \( S = (M, I) \). \( M \) is the marking of the net and is a function \((M : P \to \mathbb{N})\) that assigns to each place a number of tokens (or marks). A marking is usually denoted as a column vector with as many entries as the number of places. All along this paper, we will consider that any LTPN is safe i.e. \( M : P \to \{0, 1\} \). \( I \) is a partial firing interval application \( I : T \to I_{\mathbb{R}^+} \) that associates to any enabled transition \( t \) a time interval of \( \mathbb{R}^+ \) (\( I_{\mathbb{R}^+} \): intervals with bounds in \( \mathbb{R}^+ \)) in which transition \( t \) can be fired. The lower bound of \( I(t) \), also called the date of earlier firing, is denoted \([I(t)]\), and its upper bound, also called the date of later firing, is denoted \( \lceil I(t) \rceil \). A transition \( t \) is enabled by a given marking \( M \) (denoted \( t \in \text{enabled}(M) \)) if and only if \((\forall p \in \text{pre}(t), M(p) = 1)\). An LTPN with an initial marking \( M_0 \) is called a marked LTPN. The initial state of a marked LTPN is given by \( S_0 = (M_0, I_0) \) where \( I_0 \) is defined as follows: for any transition \( t \) enabled by \( M_0 \), \( I_0(t) = I_\tau(t) \).

Intuitively, in an LTPN a transition is fireable from a marking \( M \) if it is enabled for a sufficient amount of time i.e an amount of time within its own static time interval. Formally, a net transition \( t \) is fireable from a state \( S = (M, I) \) at a relative date \( \theta \) if and only if:

1) \( t \in \text{enabled}(M) \),
2) \( \theta \geq [I(t)] \) and for any transition \( t' \) enabled by \( M, \theta \) is not greater than the later firing date of \( t' : \theta \leq \lceil I(t') \rceil \) if \( I(t') \) is right-closed.

Firing a fireable transition \( t \) at the relative date \( \theta \) from the state \( S = (M, I) \) leads to a state \( S' = (M', I') \) such that:

- \( M' \) is such that \( \forall p \in \text{pre}(t) \setminus \text{post}(t), M'(p) = 0, \forall p \in \text{post}(t) \setminus \text{pre}(t), M'(p) = 1, \) else \( M'(p) = M(p) \);
- For any transition \( t' \in T (t' \neq t) \) enabled by \( M \) after the fire of \( t \) and enabled by \( M' \):
  - if \( I(t') = [a, b], I'(t') = [\max(0, a - \theta), b - \theta] \);
  - if \( I(t') = [a, +\infty[ , I'(t') = [\max(0, a - \theta), +\infty[ \)
  - else \( I'(t') = I_\tau(t') \).

In the following, the fire of a transition \( t \) at the relative date \( \theta \) is denoted: \( (M, I) \xrightarrow{\theta t} (M', I') \). A state \( S \) is reachable in a marked LTPN if it exists a sequence of fireable transitions \( S_0 \xrightarrow{\theta t_1} S_1 \xrightarrow{\theta t_2} \ldots \xrightarrow{\theta t_k} S \), also denoted \( S_0 \xrightarrow{\theta t_1, \ldots, \theta t_k} S \) where \( r \) is the timed transition sequence \( r = \theta t_1 \theta t_2 \ldots \theta t_k \). The sequence \( r \) will be called a \textit{run} of the net. Any run \( r' = \theta t_1 \ldots \theta t_k, i \leq k \) is a \textit{prefix} of \( r \). The set of reachable states from a state \( S \) in a marked LTPN \( N \) is denoted \( R(N, S) \).

A marking \( M' \) is reachable from a marking \( M \) (also denoted without ambiguity \( M' \in R(N, M) \)) if there exists a state \( S' \in R(N, \langle M, I \rangle) \) such that \( S' \in R(N, \langle M, I \rangle) \).

B. LTPN extensions and operations

We recall two classical LTPN extensions that will be also used throughout this paper: inhibitor arc and transition pri-
orities. An inhibitor arc modifies the enabling conditions of transition \( t \) that were detailed previously for the so-called regular arcs. With an inhibitor arc \((p, t)\), the input place \( p \) must be empty to enable \( t \). Formally, for an LTPN with both types of arcs (regular and inhibitor) the set of input places of a transition \( t \) \((\text{pre}(t))\) is then partitioned into \( \text{pre}^+(t) \) and \( \text{pre}^-(t) \). An arc \((p, t)\) of \( A \) is a regular arc if \( p \in \text{pre}^+(t) \) and an inhibitor arc if \( p \in \text{pre}^-(t) \). A transition \( t \) is enabled by a marking \( M \) \((t \in \text{enabled}(M))\) iff \( \forall p \in \text{pre}^+(t), M(p) = 1 \land \exists p \in \text{pre}^-(t), M(p) = 1 \).

A priority defines a binary relation between two transitions of an LTPN, denoted \( t_1 \triangleright t_2 \), that is transitive, not reflexive and not symmetrical. A firable transition is not allowed to fire if a transition with a higher priority is firable at the same date.

We finally introduce some further operations on LTPN.

Definition 2 (Restriction of an LTPN \( N \) to the transition \( t \)). The restriction of the LTPN \( N \) to the transition \( t \in T \) (inhibitor arcs included) is:

\[
N(t) = (\text{pre}(t) \cup \text{post}(t), \{t\}, A_t, \{\ell(t)\}, \ell_t, Ist_t)
\]

where \((n_1, n_2) \in A_t\) iff \((n_1, n_2) \in A \land (n_1 \in \text{pre}(t) \land n_2 = t) \lor (n_1 = t \land n_2 \in \text{post}(t))\).

Definition 3 (Union of LTPNs). Let \( N_1 = \langle P_1, T_1, A_1, E_1, \ell_1, I_1, s_1 \rangle, i = \{1, 2\}, \) the union of \( N_1 \) and \( N_2 \), denoted \( N_1 \cup N_2 \), is the LTPN

\[
N_1 \cup N_2 = \langle P_1 \cup P_2, T_1 \cup T_2, A_1 \cup A_2, E_1 \cup E_2, \ell_1, I_s \rangle
\]

such that \( \forall i \in \{1, 2\}: \forall t \in T_i, \ell(t) = \ell_i(t) \land \forall t \in T_1, I_s(t) = I_s(t) \). If \( N_1 \) and/or \( N_2 \) have inhibitor arcs, there are also in the union \( N_1 \cup N_2 \) similarly for priorities.

Definition 4 (Difference of LTPNs). Let \( N_1 = \langle P_1, T_1, A_1, E_1, \ell_1, I_1, s_1 \rangle, i = \{1, 2\}, \) the difference of \( N_1 \) by \( N_2 \), denoted \( N_1 \setminus N_2 \), is the LTPN

\[
N_1 \setminus N_2 = \langle P, T_1 \setminus T_2, A, E_1, \ell_1, I_1, s_1 \rangle
\]

such that:

- \( P = P_1 \setminus \{p \in P_2 : \exists t \in T_1 \setminus T_2, p \in \text{pre}(t) \cup \text{post}(t)\} \);
- \( A = A_1 \setminus \{(n_1, n_2) \in A_2 : n_1 \in T_2 \land n_2 \in T_2\} \);
- \( \forall t \in T_1 \setminus T_2, \ell(t) = \ell_1(t), I_1(t) = I_1(t) \).

Any inhibitor arc of \( A_1 \) in \( A \) is retained as an inhibitor arc in \( A \). Any priority in \( N_1 \) between two transitions of \( T_1 \setminus T_2 \) is also kept in \( N_1 \setminus N_2 \).

C. Timed languages

A timed sequence over a set of labels \( \Sigma \) is a sequence \( \rho \) of pairs \((s, d)\) where \( d \) is a duration and \( s \) is a symbol of \( \Sigma \cup \{\lambda\} \). Throughout this paper, symbol \( \lambda \) will represent the occurrence of time and will be always considered as a silent event (\( \lambda \) may be part of \( \Sigma \) or not). The set of timed sequences over \( \Sigma \) is denoted \( T(\Sigma) \) and \( T(\Sigma) \subseteq (\Sigma \cup \{\lambda\} \times \mathbb{R}^+)^* \). Each timed sequence \( \rho \) has a canonical representation denoted \( [\rho] \) with only one \( \lambda \) at the end. Suppose for instance that \( \Sigma = \{a, b, c\} \), then \( \rho = (a, 3, \lambda, 4, \lambda, 2, a, 1, b, 2, \lambda, 4) \) is a timed sequence of \( T(\Sigma) \) also denoted \( 3a4c2\lambda1b2b4\lambda \). The canonical representation of \( \rho \) is \( [\rho] = 3a4c3b2b4\lambda \). Throughout this paper, only canonical representations are considered.

Consider now a run \( r \) of an LTPN \( S_0 \xrightarrow{r} S = \langle M, I \rangle \), we can associate \( r \) with a set of canonical timed sequences, denoted \( \theta_{\ell}(r) \) as follows:

\[
\theta_{\ell}(r) = \{[\theta_{\ell}(r), d, \lambda], d \in [0, d_{\text{max}}]\}, \tag{2}
\]

where \( \theta_{\ell}(r) \) is the earliest timed sequence of \( r \):
- if \( r \) is the empty run (no transition) then \( \theta_{\ell}(r) = 0 \lambda \);
- if \( r = \ell_1t_1\ell_2t_2\ldots \ell_kt_k, k > 0 \) then \( \theta_{\ell}(r) = [\ell_1\ell(t_1)\ell_2(t_2)\ldots \ell_k(t_k)\lambda] \).

Date \( d_{\text{max}} \) is the latest possible duration stay in state \( S \) (i.e. the minimal upper bound of the current firing intervals \( d_{\text{max}} = \min_{t \in \text{enabled}(M)}([I(t)]) \)). Based on the sets \( \theta_{\ell}(r) \) we can define a timed language generated by an LTPN. Let \( Q \) be a subset of \( R(N, M_0) \), a timed sequence belongs to the language generated by the marked LTPN if there is a run of the LTPN that generates the sequence and that leads to a state whose marking is in \( Q \); such a run is called an admissible run.

Definition 5 (Language of an LTPN over \( Q \)). The language generated by a marked LTPN \( N \) over \( Q \) is

\[
L(N, Q) = \bigcup_{r : S_0 \xrightarrow{r} S = \langle M, I \rangle \land M \in Q} \theta_{\ell}(r)
\]

IV. PROBLEM STATEMENT

The fault diagnosis and diagnosability problems have been formally defined with the help of timed automata in [7]. We propose here to extend these definitions to deal with more complex behaviors than single faults.

A. System model

We consider throughout this paper that the supervised system is modelled as a safe LTPN denoted \( \Theta = \langle P_\Theta, T_\Theta, A_\Theta, E_\Theta, \ell_\Theta, I_\Theta, s_\Theta \rangle \) with an initial marking denoted \( M_0 \). \( A_\Theta \) represents the set of event types that are effectively generated by the system (\( \lambda \notin A_\Theta \)). In this paper, we assume that according to the instrumentation only a subset of transitions can be observed. These transitions are associated with a sensor that generates an output turned into an observable event by a measurement function. We consider also unobservable transitions that represent effective system’s event but not associated with any kind of sensors. Transition labels \( \Sigma_\Theta \) are then partitioned into two subsets: \( \Sigma_\Theta^O \) the set of observable events and \( \Sigma_\Theta^U \) the set of unobservable ones. Besides this definition, there are several assumptions about the system.

A0 The system has no deadlock.
A1 The system has no zeno runs.
A2 From any reachable state, there is no infinite sequence of firable transitions labeled with unobservable events.

A0 states that for any reachable state \( S \) there exists a transition \( t \) that is firable from \( S \) and such that \([I(t)] \neq \infty \). This classical assumption is from the seminal work [2]. Assumption A1 states that the model cannot represent unrealistic evolutions of the system by forbidding runs that have an infinite number of events in a finite amount of time [7], [30]. Typically, zeno...
runs are modeled by transition loops, each transition being associated with a static time interval whose lower bound is 0. Finally, Assumption A2 is also a classical assumption for diagnosability analyses [2], [7] without which the system is unlikely diagnosable due to the lack of observations.

Figure 1 presents the model of a system Θ. It consists of two concurrent processes synchronized to perform together a specific activity. The first process (on the left part of the figure) is composed of several activities represented by the places \( p_1^i, i \in \{1, \ldots, 5\} \) while the activities of the second process (on the right part of the figure) are modeled by the places \( p_2^i, i \in \{1, \ldots, 6\} \). Both processes are synchronized on transition \( t_1^{12} \) whose firing leads to the marking of a shared place \( p_1^2 \) representing the activity that requires the two processes. The end of this activity is modeled by the firing of transition \( t_2^{12} \) that leads to the marking of the places \( p_1^3 \) and \( p_2^6 \) so that each process can evolve independently to perform its own activities. The initial state of each process is given by the initial marking (places with black dots). Transition labels that are in bold represent observable events while the others are unobservable events. It must be noticed that Process 1 (resp. Process 2) only generates \( \alpha_1 \) (resp. \( \alpha_2 \)) events as observations. The timed language generated by the system is

\[
\mathcal{L}(\Theta) = \mathcal{L}(\Theta, R(\Theta, M_{0\Theta}))
\]

B. Event pattern model

Event patterns gather a set of specific combinations of un- and observable events whose occurrence in the system leads to a situation of interest (i.e. failure, critical/dangerous situation, safe/unsafe behavior,...). Patterns that are studied throughout this paper have been introduced in [6] and are defined as Labeled Petri Nets (LPN), i.e. LTPN such that \( I_s(t) = [0, +\infty) \) for any transition \( t \). As for the system model, transitions labels represent events of the system.

Definition 6 (Pattern). A pattern \( \Omega \) is an LPN \( \Omega = (P_\Omega, T_\Omega, A_\Omega, \ell_\Omega, \Sigma_\Omega, M_{0\Omega}) \) associated with a set of final markings \( Q_\Omega \subset R(\Omega, M_{0\Omega}) \) such that:

1) (unobservable) events of \( \Sigma_\Omega \) are unobservable (\( \Sigma_\Omega \subseteq \Sigma_{\Theta}^u \));
2) (initialized) \( M_{0\Omega} \notin Q_\Omega \);
3) (well-formed) from any reachable marking \( M \), there exists a run that leads to a marking \( M' \in Q_\Omega \);
4) (deterministic) from any reachable marking \( M \) in \( R(\Omega, M_{0\Omega}) \) there is no event \( e \in \Sigma_\Omega \) that labels more than one enabled transition;
5) (stable) from any reachable marking \( M \) of \( R(\Omega, M_{0\Omega}) \) such that \( M \in Q_\Omega \) any possible run leads to a marking \( M' \) such that \( M' \in Q_\Omega \).

For the sake of simplicity, we denote \( \mathcal{L}(\Omega) = \mathcal{L}(\Omega, Q_\Omega) \). Patterns are LPN that aim at representing succinct behaviors that are unobservable (Condition 1) (\( \mathcal{L}(\Omega) \subset \Sigma_{\Theta}^{u*} \)). Condition 2 ensures that the pattern does not represent an empty event sequence. Condition 3 guarantees that any run of the pattern is always a prefix of a matching run. Condition 4 avoid ambiguities within the pattern and is required to ensure the correctness of the combination of a pattern and a system as detailed later. Finally Condition 5 ensures that once a pattern has occurred, its effect is definitive (any system’s run that has a matching run as a prefix is a matching run).

Figure 2 illustrates different types of patterns of interest for the system of Figure 1. Figure 2a presents \( k \) occurrences of a single event \( b \) (pattern \( \Omega_1^b(k) \)). The marking of the final place \( pp_1^b \) indicates the pattern has occurred then the set of final markings is given by \( Q_{\Omega_1^b(k)} = \{ M : M(pp_1^b) = 1 \} \). This pattern extends the classical single fault event pattern that is usually studied in the literature. Throughout this paper, we will similarly denote \( \Omega_1^f(k) \) the pattern modeling \( k \) occurrences of the single event \( f \). Figure 2b presents a pattern that characterizes \( k \) occurrences of one event among a predefined set of events, here among \( \{b, f\} \). The structure of the pattern depends on the value of \( k \) required. If one wants to consider one occurrence of one event among \( \{b, f\} \), it is necessary to consider the marking of the place \( pp_1^b \) and then \( Q_{\Omega_2(b,k)} = \{ M : M(pp_1^b) = 1 \} \). The marking of the place \( pp_2^f \) indicates 2 occurrences (2 events \( b \) or 2 events \( f \) or 1 event \( b \) and 1 event \( f \), whatever the ordering) then the set of final markings is \( Q_{\Omega_2(b,f)} = \{ M : M(pp_2^f) = 1 \} \). The number of layers in the pattern depends on the \( k \) value. The \( k \) occurrences are obtained when the place \( pp_2^f \) is marked hence \( Q_{\Omega_2(b,f,k)} = \{ M : M(pp_2^f) = 1 \} \). The pattern \( \Omega_2(b,k) \) can be a base to extend the classical notion of fault class with \( k = 1 \) and the set of events is the set of \( n \) fault class events: \( \{f_1, f_2, \cdots, f_n\} \). In this case, the pattern has only one layer with as many transitions as fault class events. Finally, Figure 2c presents a pattern that models 3 consecutive occurrences of event \( b \) that are not interleaved with any occurrence of event \( f \). This pattern can also be extended into \( \Omega_3(b,f) \) to model \( k \) consecutive occurrences of an event without any occurrence of another type of event. This last pattern is interesting as it represents some fairness issues in the system that might be of interest to detect, \( Q_{\Omega_3(b,f)} = \{ M : M(pp_2^f) = 1 \} \).

C. Pattern diagnosis and diagnosability

This subsection introduces the problem of diagnosing patterns in timed discrete event systems and defines the diagnosability of a pattern in such a system. For a given pattern \( \Omega \), the diagnosis problem consists in defining an \( \Omega \)-diagonser function that takes as input a timed sequence of observations and checks for any run of the system that can generate these observations whether it matches the pattern \( \Omega \) or not. It results in the generation of three possible symbols: \( \Omega \)–faulty (all the possible runs match \( \Omega \)), \( \Omega \)–safe (none of them match \( \Omega \)), \( \Omega \)–ambiguous (some of them match \( \Omega \), some of them do not) [36], [6]. A run \( r \) of the system matches a pattern \( \Omega \) if one can find in a word \( \rho = \theta_1 e_1 \cdots \theta_n e_n \theta_{n+1}^\lambda \) of \( \Omega \) a sub-word \( \sum_{i=1}^{n} \theta_i \) \( e_{j_1} \sum_{i=j_1+1}^{n} \theta_i e_{j_2} \cdots \sum_{i=j_{k-1}+1}^{n} \theta_i e_{j_k} \) such that \( \theta_{j_1+1} \cdots \theta_{j_k+1} \) is a word of \( \mathcal{L}(\Omega) \); this will be denoted by \( \rho \Omega \) and more generally by \( \rho \circlearrowright \Omega \).

Checking whether a run of \( \Theta \) can generate the given observations (i.e. the run is consistent with the observations) relies on the timed language projection from the alphabet \( \Sigma_{\Theta} \) to the observable alphabet \( \Sigma_{\Theta}^o \). Given two alphabets \( \Sigma_1, \Sigma_2, \)
the projection \(P_{\Sigma_1 \to \Sigma_2} : T(\Sigma_1) \to T(\Sigma_2)\) of a canonical timed sequence is defined as follows.

- if \(e \in \Sigma_2\), \(P_{\Sigma_1 \to \Sigma_2}(\sigma e) = \sigma e\);
- if \(e \in \Sigma_1 \setminus \Sigma_2\), two cases hold:
  - \(P_{\Sigma_1 \to \Sigma_2}(\sigma e) = (\theta + \theta')\lambda\), if \(\theta' \lambda = P_{\Sigma_1 \to \Sigma_2}(\rho)\).
  - \(P_{\Sigma_1 \to \Sigma_2}(\sigma e) = (\theta + \theta')\rho'\), if \(\theta' \lambda = P_{\Sigma_1 \to \Sigma_2}(\rho)\).

The \(\Omega\)-diagnoser can then be formally defined as follows.

**Definition 7** (\(\Omega\)-diagnoser). An \(\Omega\)-diagnoser is a function 
\[\Delta_\Omega : T(\Sigma_0^\Theta) \to \{\Omega-faulty, \Omega-safe, \Omega-ambiguous\}\] with
- \(\Delta_\Omega(\sigma) = \Omega-faulty\) if for any \(\rho \in L(\Theta)\) that is consistent with \(\sigma\) (i.e., \(P_{\Sigma_0 \to \Sigma_0}(\rho) = \sigma\), \(\rho \equiv \Theta\));
- \(\Delta_\Omega(\sigma) = \Omega-safe\) if for any \(\rho \in L(\Theta)\) that is consistent with \(\sigma\), \(\rho \not\equiv \Theta\);
- \(\Delta_\Omega(\sigma) = \Omega-ambiguous\) otherwise.

It must be noticed that, even if Definition 7 uses Petri nets (\(\Theta\) and \(\Omega\) notations), this definition is purely language-based. The diagnosis problem is defined over the set of timed sequences \(\rho\) in \(L(\Theta)\), the matching relation is defined over event sequences and consistency is achieved by sequence projection. It follows that Definition 7 defines a problem independently from the underlying model structure but with respect to their language expressiveness (time Petri nets, timed Event Graphs, timed automata...). Considering now a set of patterns \(\Omega_1, \ldots, \Omega_n\), the diagnoser function of a system can be defined as \(\Delta: T(\Sigma_0^\Theta) \to \prod_{i=1}^n \{\Omega_i-faulty, \Omega_i-safe, \Omega_i-ambiguous\}\) such that \(\Delta(\sigma) = (\Delta_{\Omega_1}(\sigma), \ldots, \Delta_{\Omega_n}(\sigma))\).

Let us now illustrate the diagnosis problem with a few examples based on the system of Figure 1 and the family of patterns of Figure 2.

**Example 1.** Suppose that the timed sequence that is received by the diagnoser from the system of Figure 1 is \(\sigma_1 = 3o_1 2o_2 4\lambda\) (that is the reception of event \(o_1\) at date 3 then the reception of \(o_2\) at date 5 and then nothing till date 9). If the problem is to diagnose the occurrence of one event \(b\), one considers the pattern \(\Omega^b(1)\) with the observation sequence \(\sigma_1\). Then as the event \(o_2\) occurs at date 5, transition \(t_2^2\) is fired at date 5 which means that event \(b\) associated with transition \(t_2^2\) has certainly occurred and the result of the \(\Omega\)-diagnoser is: \(\Delta_{\Omega^b(1)}(\sigma_1) = \Omega^b(1)\)-faulty. The pattern \(\Omega^b(1)\) models one occurrence of an event among \(\{b, f\}\). In this case at date 9 places \(p_5^1\) and \(p_6^1\) are marked in the system model, so the event \(f\) has certainly occurred (firing of transition \(t_5^1\)) and \(\Delta_{\Omega^f(1)}(\sigma_1) = \Omega^f(1)\)-faulty. Finally, \(\Delta_{\Omega^f(2)}(\sigma_1) = \Omega^f(2)\)-safe (event \(b\) has necessarily occurred once, never twice otherwise a second event \(o_2\) would appear in \(\sigma_1\) after date 9 because of \(t_2^2\)).

Pattern diagnosability is the property of the system that ensures that, once a pattern \(\Omega\) has occurred in the system \(\Theta\), the \(\Omega\)-diagnoser is always able to return \(\Omega\)-faulty after a finite time. Let \(time(\rho)\) denote the global duration (i.e. the sum of all durations \(\sum_{i=1}^n \theta_i\)) of a timed sequence \(\rho\).

**Definition 8** (Pattern Diagnosability). A system \(\Theta\) is \(\Omega\)-diagnosable if

\[\exists \tau \in \mathbb{R}^+ : \forall \rho_1, \rho_2 \in L(\Theta) : \rho_1 = \rho'_1 \rho''_1, time(\rho''_1) \geq \tau, \rho'_1 \equiv \Omega \land P_{\Sigma_0 \to \Sigma_0}(\rho_1) = P_{\Sigma_0 \to \Sigma_0}(\rho_2) \Longrightarrow \rho_2 \equiv \Omega.\] (4)

**Proposition 1.** \(\Theta\) is \(\Omega^b(1)\)-diagnosable iff \(TA_\Theta\) is diagnosable with respect to the single fault event \(b\) in the sense of [7].

The diagnosability problem consists in finding a finite duration \(\tau\) after the occurrence of a fault or a pattern in the runs of \(\Theta\) such that the diagnoser can decide the system is faulty without any ambiguity. In other words, a system is diagnosable iff there exists \(\tau\) such that the system is \(\tau\)-diagnosable. [7] introduces such a notion of \(\tau\)-diagnosability that is also more recently investigated in [30]. We now extend it to pattern diagnosability.

**Definition 9** (Pattern \(\tau\)-Diagnosability). A system \(\Theta\) is \((\Omega, \tau)\)-diagnosable if

\[\forall \rho_1, \rho_2 \in L(\Theta) : \rho_1 = \rho'_1 \rho''_1, time(\rho''_1) \geq \tau, \rho'_1 \equiv \Omega \land P_{\Sigma_0 \to \Sigma_0}(\rho_1) = P_{\Sigma_0 \to \Sigma_0}(\rho_2) \Longrightarrow \rho_2 \equiv \Omega.\]
Looking back at the system of Figure 1, it can be noticed that each process emits only one type of observable events (process \( i \) emits \( \omega_1 \) events). Without observing the effective date of emission of these observations, none of the patterns described above would be diagnosable. Only the observation of time is discriminative here. The remaining question is now how to check that the system \( \Theta \) is \( \Omega \)-diagnosable.

V. PATTERN DIAGNOSABILITY ANALYSIS

Based on a model \( \Theta \) and a pattern \( \Omega \), this section describes a way to turn the \( \Omega \)-diagnosability checking problem over \( \Theta \) into checking a linear-time property over a specific LTPN called a twin-plant and denoted \( \Gamma \). Definition 8 asserts that checking whether a system \( \Theta \) is \( \Omega \)-diagnosable consists in determining that there are no infinite runs \( r \) and \( r' \) in \( \Theta \) with \( r \not\cong \Omega \) and \( r' \not\cong \Omega \) such that \( \mathbf{P}_{\Theta \rightarrow \Xi_{\omega}}(\theta \preceq^\tau(r)) = \mathbf{P}_{\Theta \rightarrow \Xi_{\omega}}(\theta \preceq^\tau(r')) \).

The twin plant gathers into a unique LTPN all the runs \( r \) and \( r' \) of \( \Theta \) that generate the same observation sequence. Any run of the twin-plant then characterizes two runs from \( \Theta \) with the same observations. Finally, checking \( \Omega \)-diagnosability simply consists in determining in the twin-plant that there is no infinite run that represents a couple of runs \( (r, r') \) such that \( r \not\cong \Omega \) and \( r' \not\cong \Omega \). To build the twin-plant, two problems need to be solved.

1) How to represent in an LTPN that any run of \( \Theta \) matches \( \Omega \) or not? This is solved by the pattern matching product \( \Theta \bowtie \Omega \) defined in Section V-A.

2) How to represent in an LTPN two runs of \( \Theta \bowtie \Omega \) that generate the same observations? This is solved by applying transition fusions of observable transitions from two copies \( \Pi_1 \) and \( \Pi_2 \) of \( \Theta \bowtie \Omega \) (Section V-B).

A. Pattern matching product

The first step of the computation of the twin-plant is to compute an LTPN that represents how a system \( \Theta \) is actually matching a pattern \( \Omega \); this step is the computation of an LTPN called the pattern matching product and denoted \( \Theta \bowtie \Omega \). This product relies on a specific product between the transitions of \( \Theta \) and \( \Omega \). Let \( t_\Theta \) be a transition of \( \Theta \) (with \( \ell_\Theta(t_\Theta) = a, I_{\Theta}(t_\Theta) = [\alpha, \beta], \alpha > 0 \)) with \( \text{pre}(t_\Theta) = \{P\Theta_1, \ldots, P\Theta_n\} \) and \( \text{post}(t_\Theta) = \{P\Theta_1, \ldots, P\Theta_n\} \).

Recall that Assumption A0 ensures \( [\alpha, \beta] \) is a closed interval and A1 ensures that \( \alpha > 0 \).
Without loss of generality, we assume that $t_\Theta$ is conflict-free (no other transition has the same preset).\footnote{Indeed, if $t_\Theta$ is in conflict with another transition $t_\Theta', \Theta$ can be transformed by adding two silent transitions $\lambda[0,0]$ enabled by the preset of $t_\Theta$ and $t_\Theta'$. Each silent transition has a postset with one new place that replaces the preset of either $t_\Theta$ or $t_\Theta'$. New versions of $t_\Theta$ and $t_\Theta'$ are then conflict-free while the generated language $L(\Theta)$ remains the same.} Let $t_\Omega$ be a transition of $\Omega$ labeled with $a$ with $\text{pre}(t_\Omega) = \{P_\Omega^1, \ldots, P_\Omega^{m_\Omega}\}$ and $\text{post}(t_\Omega) = \{P_\Omega^{m_\Omega+1}, \ldots, P_\Omega^{n_\Omega}\}$. The transition product $t_\Theta \times t_\Omega$ is then the LTPN defined by Figure 3.

![Pattern matching product between a transition $t$ of $\Theta$ and a transition of $\Omega$ labeled by $a$.](image)

In time Petri nets compositionality is kept only when synchronizations involve transitions with an interval $[0, +\infty]$ [37] and transitions with a time interval $[\alpha, \beta]$ cannot be synchronized as is. Indeed, synchronizing two transitions means that it is necessary to determine a clock enabled only when the transitions are both enabled and that stops as soon as one of the local clocks stops. This synchronization is then impossible by simply merging both transitions and assigning a time interval to it based on their respective time intervals as the stop of the synchronized clock depends on the enabling time of each transition and not on the enabling time of their synchronization.

It is then required to perform a time decomposition of transition $t_\Theta$ to ensure that the synchronization is performed only on transitions with $[0, +\infty]$ time intervals. The time decomposition requires the insertion of transitions with silent events $\lambda$ (see Section III-C). The result of this time decomposition is the places $T_i\Theta, Na\Theta$ and transitions $na\Theta, cs\Theta, to\Theta, e\Theta$ on the left. This time decomposition ensures that transition $e\Theta$ is firable exactly in the same conditions as the initial transition $t_\Theta$. Indeed, once $t_\Theta$ is enabled in $\Theta$ by the marking of $P_\Theta^1, \ldots, P_\Theta^{m_\Theta}$, transition $cs\Theta$ (i.e. clock start) is enabled in $\Theta \times \Omega$ and is silently fired exactly at time $\alpha > 0$ so transition $e\Theta$ starts being firable at time $\alpha$ till time $\beta$. At $\beta$, either $e\Theta$ or the silent transition $to\Theta$ (i.e. time out) is fired, so $e\Theta$ is firable in $[\alpha, \beta]$ as $t_\Theta$ and leads to the same postset. Now, as soon as $to\Theta$ is fired, it means that the behavior is actually not admissible (transition $t_\Theta$ must have been fired in $[\alpha, \beta]$ in $\Theta$ as it is conflict-free), the place $Na\Theta$ then permanently holds a token and inhibits any further fire of $cs\Theta$.

The use of $na\Theta$ is then to empty the preset of $cs\Theta$ in case of a non admissible behavior. The effective synchronization of $t_\Theta$ and $t_\Omega$ is then represented by transition $e\Theta\Omega$ resulting from the fusion of $t_\Theta$ and a copy of the transition $e\Theta$ but as opposed to the classical transition fusion, transition $e\Theta$ is kept to capture the evolutions of the system alone as a pattern evolution corresponds potentially (in case of matching) to a sub-word of the system. A transition priority $e\Theta\Omega \succ e\Theta$ is set between $e\Theta\Omega$ and $e\Theta$ which ensures that if both $e\Theta\Omega$ and $e\Theta$ are firable at a given time, only $e\Theta\Omega$ can be effectively fired. The global definition of the product between a system $\Theta$ and a pattern $\Omega$ consists then in applying the suitable transition product $t_\Theta \times t_\Omega$ for any couple of transitions $(t_\Theta, t_\Omega)$ in $\Theta$ and $\Omega$ that share the same event label. It is formally defined as follows where $\Theta(t_\Theta)$ denotes the restriction of $\Theta$ to the transition $t_\Theta$.

**Definition 10 (Pattern matching product).** The pattern matching product $\Theta \times \Omega$ is the LTPN such that

$$\Theta \times \Omega = \Theta \setminus \left( \bigcup_{(t_\Theta, t_\Omega) \in S} \{t_\Theta \times t_\Omega \} \right) \cup \bigcup_{(t_\Theta, t_\Omega) \in S} t_\Theta \times t_\Omega$$

where $S = \{(t_\Theta, t_\Omega) : t_\Theta(t_\Theta) = t_\Omega(t_\Omega)\}$.

We associate with $\Theta \times \Omega$, the following initial marking $M_{\Theta \times \Omega}^0$: for any place $p$ of $\Theta \times \Omega$, $M_{\Theta \times \Omega}^0(p) = M_\Theta^0(p)$ if $p$ is also a place of $\Theta$, $M_{\Theta \times \Omega}^0(p) = M_\Omega^0(p)$ if $p$ is also a place of $\Omega$ and $M_{\Theta \times \Omega}^0(p) = 0$ otherwise. We denote $Q_{\Theta \times \Omega}^{adm}$ the set of admissible markings, it consists of all the markings $M$ such that for any place of type $Na\Theta, M(Na\Theta) = 0$. Let $Q_{\Theta \times \Omega}^{match}$ denote the set of admissible markings $M$ such that the restriction of the marking $M$ to the places of $\Omega$ belongs to the final markings $Q_\Omega$ (see Definition 6). It follows that the set of runs in $\Theta \times \Omega$ that lead to a marking of $Q_{\Theta \times \Omega}^{match}$ generates exactly the sub-language of the system that matches the pattern $\Omega$. Formally, we have:

**Theorem 1.**

1. $L(\Theta \times \Omega, Q_{\Theta \times \Omega}^{adm}) = L(\Theta)$;
2. $L(\Theta \times \Omega, Q_{\Theta \times \Omega}^{match}) = \{\rho \in L(\Theta) : \rho \equiv \Omega\}$.

**Proof.** By construction of the time decomposition (Definition 10), for any transition product $t_\Theta \times t_\Omega$, suppose first that the priority does not exist then $e\Theta$ is firable in $\Theta \times \Omega$ in $[\alpha, \beta]$ from a given date $d$ iff $t_\Theta$ is firable in $[\alpha, \beta]$ from $d$ in $\Theta$. By adding the priority, the fire of $e\Theta$ can be possibly replaced by the fire of $e\Theta\Omega$ which generates the same label and only this one as the pattern is deterministic. It follows that $L(\Theta \times \Omega, Q_{\Theta \times \Omega}^{adm}) = L(\Theta)$, $Q_{\Theta \times \Omega}^{match}$ gathers the markings of $Q_{\Theta \times \Omega}^{adm}$ whose restrictions to the places of $\Omega$ are final markings of $\Omega$. If a run $r$ of $\Theta \times \Omega$ leads to a marking of $Q_{\Theta \times \Omega}^{match}$ it then means it is a run of $\Theta$ that generated a word $\rho$ that matches a word $\rho_\Omega$ of $\Omega$: $\rho \equiv \Omega$. Moreover as $\Omega$ is stable (Definition 6 condition 5), any continuation of $\rho$ also matches $\Omega$. \hfill $\square$

**B. Twin plant synthesis**

The overall principle is to confront two admissible runs from $\Theta \times \Omega$ that can produce the same observable timed sequence
by duplicating the two products \( \Theta \times \Omega \). In the following, we denote \( \Pi_i, i \in \{1, 2\} \), these two products. Let \( \Theta \times \Omega = \{P, T, A, \Sigma_0^T \cup \Sigma_0^A \cup \{\ell, I_s\} \) then \( \Pi_i = \langle P_i, T_i, A_i, \Sigma_i, \ell_i, I_{s_i} \rangle \) is defined as follows: \( P_i = P \times \{i\}, T_i = T \times \{i\}, (n, n') \in A_i \) iff \( (n, n') \in A \). \( \Sigma_i = \{(\Sigma_0^T \cup \{\ell\}) \times \{i\} \} \cup \Sigma_0^A \), \( \ell_i((t, i)) = \ell(t) \) if \( \ell(t) \in \Sigma_0^A \) otherwise \( \ell_i((t, i)) = \ell(t, i), I_{s_i}((t, i)) = I_s(t) \).

The twin plant is then obtained by applying a transition fusion [38] between a transition \( t_1 \) from \( \Pi_1 \) and a transition \( t_2 \) from \( \Pi_2 \) that share the same observable label \( \ell_1(t_1) = \ell_2(t_2) \) in \( \Sigma_0^A \). For \( i = 1, 2 \), let \( \text{pre}(t_i) = \{p_i^0\}, j \in \{1, \ldots, n_i\} \) and \( \text{post}(t_i) = \{p_i^m\}, j \in \{1, \ldots, m_j\} \). Suppose both transitions are labeled with event \( o = \ell_1(t_1) = \ell_2(t_2) \) and their respective static interval is \( [\alpha_i, \beta_i], i = 1, 2 \), the transition fusion denoted \( t_1[t_2] \) is the LTPN defined in Figure 4.

LTPN \( t_1[t_2] \) is obtained in two steps. The first intermediate step, similar to the pattern matching product (see Figure 3), is to compute the time decomposition of both transitions \( t_1 \) and \( t_2 \). Both decompositions independently lead to the creation of two transitions \( e_j \): such a transition \( e_j \), not shown in Figure 4, looks like transition \( e_0 \) in Figure 3, it is labeled with event \( o \) and associated with time interval \( [0, +\infty[ \) and \( \text{pre}(e_j) = \{T_i\} \) and \( \text{post}(e_j) = \{p_i^2, \ldots, p_i^m\} \). The second and final step is then to apply the classical transition fusion of \( e_1 \) and \( e_2 \) to obtain the transition \( e_{12} \) of Figure 4.

Theorem 2.
\[
\mathcal{L}(\Gamma, Q_{\Gamma}^{adm}) = \{\rho \in \mathcal{T}(S_T) : P_1^T(\rho) \in \mathcal{L}(\Theta) \land P_2^T(\rho) \in \mathcal{L}(\Theta) \land P_1^S(\rho) \in \mathcal{L}(\Omega) \land P_2^S(\rho) \in \mathcal{L}(\Omega) \}
\]

Proof. See appendix: we need to prove the equality of two languages. Lemma 1 proves the language inclusion \( \supseteq \) by the use of Definition 11 and Theorem 1. Lemma 2 proves \( \subseteq \) based on Definition 11 and transition fusions.

For any pair \( (\rho_1, \rho_2) \) of timed sequences generated by \( \Gamma \), as \( \rho_1 \) and \( \rho_2 \) belong to \( \mathcal{L}(\Theta) \), each timed sequence either matches pattern \( \Omega \) or not.

Definition 13. A run \( r \) of \( \Gamma \) is ambiguous if it is admissible and the pair \( (\rho_1, \rho_2) \) with \( \rho_i = P_i^T(\theta_{\omega_i}^{\epsilon i}(r)), i \in \{1, 2\} \) is such that \( \rho_1 \not\supseteq \Omega \land \rho_2 \not\supseteq \Omega \).

By Definition 13, we can always associate with an ambiguous run \( r \) a pair \( (\rho_1, \rho_2) \) such that \( \rho_1 \not\supseteq \Omega \land \rho_2 \not\supseteq \Omega \). But actually, any pair \( (\rho_1', \rho_2') \) generated by \( r \) holds the same property by Proposition 2.

Proposition 2. If run \( r \) is ambiguous then any pair \( (\rho_1, \rho_2) \) generated by \( r \) is such that \( \rho_1 \not\supseteq \Omega \land \rho_2 \not\supseteq \Omega \).

Proof. Pattern matching depends on the effective fire of transitions involving events from the pattern and not on time passing only. Timed sequences from \( \theta_{\omega_i}^{\epsilon i}(r) \) have the following form: there exists \( d \in [0, d_{\text{max}}] \) such that \( \rho = [\theta_{\omega_i}^{\epsilon i}(r) d] \times \) (see Section III-C), only the duration \( d \) between the last transition fire and the end of the timed sequence is different.
Definition 14. An infinite run $r_\infty$ of $\Gamma$ is an infinite set of finite runs of $\{r_i\}_{i \in \mathbb{N}^+}$ such that for any $i > 1$, $S_{i-1} \Rightarrow S_i$. By extension, we say that an infinite run is admissible if any of its finite run is admissible. Finally, an infinite run is ambiguous if there exists $k \in \mathbb{N}^+$ such that $\forall n \geq k$, run $r_n$ is ambiguous. Theorem 3 then holds the fundamental result of the proposed approach.

Theorem 3. A system $\Theta$ is $\Omega$-diagnosable if and only if $\Gamma$ does not contain ambiguous infinite runs.

Proof. To prove the theorem, we are going to actually prove that a system $\Theta$ is not $\Omega$-diagnosable if and only if $\Gamma$ contains an ambiguous infinite run.

$(\Rightarrow)$ First, let us remark that, Equation (4) from Definition 8 is equivalent to

$$\forall \rho_1, \rho_2 \in L(\Theta), \exists r \in \mathbb{R}^+: \rho_1 = \rho_1' \rho_1'', \text{time}(\rho_1') \geq \tau,$$

$$\rho_1' \ni \Omega \land P_0(\rho_1) = P_0(\rho_2) \Rightarrow \rho_2 \ni \Omega. \quad (6)$$

The fact that (4) $\Rightarrow$ (6) is straightforward. Then, from Equation (6), Equation (4) is obtained by simply selecting the maximal value $\tau$ amongst the possible values of $\tau$'s (therefore (4) $\Leftrightarrow$ (6)). Assuming $\Theta$ is not $\Omega$-diagnosable, then from (6):

$$\exists \rho_1, \rho_2 \in L(\Theta), \forall r \in \mathbb{R}^+: \rho_1 = \rho_1' \rho_1'', \text{time}(\rho_1') \geq \tau,$$

$$\rho_1' \ni \Omega \land P_0(\rho_1) = P_0(\rho_2) \land \rho_2 \ni \Omega. \quad (7)$$

Let $\rho_1, \rho_2 \in L(\Theta)$ be such that $\rho_1 = \rho_1' \rho_1''$, $\rho_1' \ni \Omega$, $P_0(\rho_1) = P_0(\rho_2)$, $\rho_2 \ni \Omega$ (such a pair exists as $\Theta$ is not $\Omega$-diagnosable). Any prefix $\sigma_1$ of $\rho_1$ that keeps $\rho_1'$ as a prefix is such that $\sigma_1 \ni \Omega$. Any prefix $\sigma_2$ of $\rho_2$ is such that $\sigma_2 \ni \Omega$. Consider now any pair $(\sigma_1, \sigma_2)$ such that $\text{time}(\sigma_1) = \text{time}(\sigma_2)$, as $P_0(\rho_1) = P_0(\rho_2)$, then $P_0(\rho_1) = P_0(\rho_2)$. By Theorem 2 and Proposition 2, there must exist an admissible and ambiguous run $r$ of $\Gamma$ that generates $(\sigma_1, \sigma_2)$. As $\Theta$ is not $\Omega$-diagnosable, $\rho_1$ and $\rho_2$ can be arbitrarily long (i.e $\forall r \in \mathbb{R}^+$ there exist $\rho_1$ and $\rho_2$ such that $\text{time}(\rho_1) \geq \tau$ and $\text{time}(\rho_2) \geq \tau$). Assumption A0 ensures that the number of events in $\rho_1$ and $\rho_2$ is not bounded. Moreover Assumption A2 ensures that the number of observable events in $\rho_1$ and $\rho_2$ is not bounded. It follows that it must exist at least an infinite run $r_\infty$ in $\Gamma$ that is ambiguous: $r_\infty$ is defined as the infinite set of ambiguous runs $r$ that generate the set of prefixes $(\sigma_1, \sigma_2)$.

$(\Leftarrow)$ Suppose that $\Gamma$ has an ambiguous infinite run $r_\infty$. By Assumption A1, $\lim_{i \to +\infty} \text{time}(\theta_{f_\Gamma}(r_i)) = +\infty$ and by assumption A2 $\lim_{i \to +\infty} |P_{\Sigma, R}(\theta_{f_\Gamma}(r_i))| = +\infty$. $N, Q$ can be arbitrarily long (i.e $\forall i > 1$, $S_{i-1} \Rightarrow S_i$). By Proposition 2, any $(\rho_1^{k+1} \rho_2^{k+1} \rho_1 \rho_2)$ generated from $\theta_{f_\Gamma}(r_{k+1})$ is also ambiguous and $\rho_1^{k+1} \rho_2^{k+1} \ni \Omega$. $r_{k+1}$ is also ambiguous, so there exists the ambiguous pair $(\rho_1^{k+1}, \rho_2^{k+1})$ generated from $\theta_{f_\Gamma}(r_{k+1})$. If $f_\Gamma(r_{k+1}) \neq \lambda$, $\rho_1^{k+1} = \rho_1^{k+1} \rho_2^{k+1} \Theta f_\Gamma(r_{k+1}) \lambda$, otherwise $\rho_1^{k+1} = \rho_1^{k+1} \lambda$. Similarly, if $f_\Gamma(r_{k+1}) \neq \lambda$, $\rho_2^{k+1} = \rho_2^{k+1} \rho_2^{k+1} \Theta f_\Gamma(r_{k+1}) \lambda$, otherwise $\rho_2^{k+1} = \rho_2^{k+1} \lambda$.

By Proposition 2, any $(\rho_1^{k+1} \rho_2^{k+1} \rho_1 \rho_2)$ generated from $\theta_{f_\Gamma}(r_{k+1})$ is also ambiguous and $\rho_1^{k+1} \rho_2^{k+1} \ni \Omega$. $r_{k+1}$ is also ambiguous, so there exists the ambiguous pair $(\rho_1^{k+1}, \rho_2^{k+1})$ generated from $\theta_{f_\Gamma}(r_{k+1})$. If $f_\Gamma(r_{k+1}) \neq \lambda$, $\rho_1^{k+1} = \rho_1^{k+1} \rho_2^{k+1} \Theta f_\Gamma(r_{k+1}) \lambda$, otherwise $\rho_1^{k+1} = \rho_1^{k+1} \lambda$. Similarly, if $f_\Gamma(r_{k+1}) \neq \lambda$, $\rho_2^{k+1} = \rho_2^{k+1} \rho_2^{k+1} \Theta f_\Gamma(r_{k+1}) \lambda$, otherwise $\rho_2^{k+1} = \rho_2^{k+1} \lambda$.

VI. A MODEL-CHECKING PROCEDURE FOR CHECKING
PATTERN DIAGNOSABILITY IN LTPN

A. Model-checking problem

Checking $\Omega$-diagnosability of the system $\Theta$ is equivalent to checking that there is no infinite run that is ambiguous in $\Gamma$. We propose to solve this problem by using the LTPN model checker TINA [10], [39]. TINA is able to edit LTPN and verify SE-LTL properties (State/Event Linear Temporal Logic) over LTPN. A formula $\psi$ is a SE-LTL formula if it is a universally quantified formula $\psi := \forall \varphi$ where $\varphi := \text{ct} r | \neg \varphi | \varphi \lor \varphi | \varphi \land \varphi | \Box \varphi | \Diamond \varphi | \varphi_U \varphi$. The constant $\text{ct}$ can be $\bot$ (false), $\top$ (true), $\text{dead}$ ( deadlock), $\text{div}$ (temporal divergence), $\text{sub}$ (partially known state). $r$ defines constraints $e \Delta e$ with $\Delta \in \{=,<,>,\leq,\geq\}$ between arithmetical expressions $e$ involving place and transition symbols from the underlying LTPN. The operators $\circ$ (next), $\lor$ (always), $\land$ (eventually) and $\lor$ (until) have their usual LTL semantics. Based on SE-LTL, it is in particular possible to implicitly characterize sets of markings in a given LTPN. For instance, suppose an LTPN $N$ composed of $n \geq 2$ places $p_1, p_2, \ldots, p_n$. The set of markings $M \subset N$ of $N$ such that $M(p_1) = 1$ and $M(p_2) = 1$ can be characterized by the SE-LTL property denoted $\text{MARKINGS}(N,Q)$:

$$\text{MARKINGS}(N,Q) \equiv (p_1 = 1 \land p_2 = 1). \quad (8)$$

To implement the resolution of the problem as a model-checking problem, it then suffices to design the SE-LTL formula $\text{DIAGNOSABLE}(\Theta, \Omega)$ that asserts there is no ambiguous infinite run in $\Gamma$. This formula is detailed below step by step. We recall here that the twin plant $\Gamma$ results from the synchronization of two copies $\Pi_i = \Theta_i \land \Omega_i$, $i \in \{1, 2\}$.

Diagnosability involves runs in $\Gamma$ only composed of admissible markings $M$. A marking $M_i$ from $\Pi_i$ is not admissible as soon as for one place of type $Na\Theta$ (see Section V-A), we have $M_i(Na\Theta) \neq 0$. Then the following SE-LTL properties $\text{ADM}(\Pi_i)$ (resp. $\text{ADM}(\Gamma)$) characterize the set of admissible markings of $\Pi_i$ (resp. $\Gamma$):

$$\text{ADM}(\Pi_i) \equiv \bigwedge_{Na\Theta \in \Pi_i} \text{ADM}(\Pi_i). \quad (9)$$

$$\text{ADM}(\Gamma) \equiv \bigwedge_{i \in \{1, 2\}} \text{ADM}(\Pi_i). \quad (10)$$

Now, $\text{MATCH}(\Pi_i)$ is the property of any marking of $M_i$ resulting from a pattern matching, firstly $M_i$ is admissible and secondly its restriction to the places of $\Omega_i$ should hold the properties of the admissible marking set $Q_{\Omega_i}$:

$$\text{MATCH}(\Pi_i) \equiv \text{ADM}(\Pi_i) \land \text{MARKINGS}(\Omega_i, Q_{\Omega_i}). \quad (11)$$
Similarly, the property NoMatch(Πi) asserts the admissible marking M of Πi is not a matching property, i.e. its restriction M[j], does not belong to the final markings of Ωi:

\[
\text{NoMatch}(\Pi_i) \equiv \text{ADM}(\Pi_i) \land \neg \text{MARKINGS}(\Omega_i, Q_{\Omega_i}).
\]  

(12)

Now, a marking M of Γ is said to be ambiguous, if its restriction M1 to Π1 is a matching marking but its restriction M2 to Π2 is not a matching property:

\[
\text{AMBIGUOUS}(\Gamma) \equiv \text{MATCH}(\Pi_1) \land \neg \text{NoMatch}(\Pi_2).
\]  

(13)

To finally setup Diagnosable(Θ, Ω), the idea is to write that any ambiguous run is finite. So we first need to logically characterize a finite ambiguous run. As soon as the run contains an ambiguous marking M, two cases hold. Either it will eventually reach (\(\square\)) an admissible marking \(M'\) that is not ambiguous which means that the restriction \(M'_2\) of \(M'\) is a matching marking for Π2 (as it is already for Π1):

\[
\square \text{AMBIGUOUS}(\Gamma) \Rightarrow \square(\text{ADM}(\Gamma) \Rightarrow \text{MATCH}(\Pi_2));
\]  

(14)

or it will eventually reach a dead-end

\[
\square \text{AMBIGUOUS}(\Gamma) \Rightarrow \square(\text{dead}).
\]  

(15)

Theorem 3 can then be translated based on expressions (14) and (15) into the following SE-LTL property on Γ.

\textbf{Corollary 1.} The system Θ is \(\Omega\)-diagnosable iff the universally quantified formula \(\text{Diagnosable}(\Theta, \Omega)\) is true:

\[
\text{Diagnosable}(\Theta, \Omega) \equiv \forall \square \text{AMBIGUOUS}(\Gamma) \Rightarrow \square(\text{ADM}(\Gamma) \Rightarrow (\text{MATCH}(\Pi_2) \lor \text{dead})).
\]  

(16)

\section*{B. Algorithm and complexity analysis}

Algorithm 1 summarizes the method. Line 1 implements the different steps to compute Γ from Θ and Ω as detailed in Section V. Line 2 computes the query based on Γ that is presented in Corollary 1. Line 3 calls the model-checker on Γ with the query. The model-checker returns as a response either true or false and in the latter case a counter-example that is an infinite admissible run of Γ that is ambiguous (see Definitions 13 and 14). Line 5 retrieves from this run the involved couple of infinite runs \((r_1, r_2)\) from Θ.

\begin{algorithm}
  \caption{Ω-diagnosability analysis}
  \begin{algorithmic}
    \State input : A system Θ and a pattern Ω
    \State 1 \hspace{1em} Γ ← TwinPlantSynthesis(Θ, Ω);
    \State 2 \hspace{1em} Diagnosable(Θ, Ω) ← QuerySynthesis(Γ);
    \State 3 \hspace{1em} (resp, run) ← CallTina(Γ, Diagnosable(Θ, Ω));
    \State 4 \hspace{1em} if resp = true then return true;
    \State 5 \hspace{1em} else return \((r_1, r_2) ← \text{RunExtractions}()\);
    \State output: \text{true} if Θ is Ω-diagnosable
    \State output: \text{false} if not Ω-diagnosable,
  \end{algorithmic}
\end{algorithm}

The overall complexity of Algorithm 1 is the complexity of the model-checking part of the algorithm that is performed at line 3. Line 3 actually checks the property Diagnosable(Θ, Ω) on the Strong State Class Graph of Γ (SSCG). The number of states of the SSCG is in \(O(2^{P_1})\).

\section*{C. Experimental results}

We report on the experimental results of the 8 diagnosability analyses described in Example 2. Table I recalls the 8 scenarios. Two scenarios are based on the system Θ where transition \(a[0, 3]\) has been replaced by \(a[2, 3]\). As the complexity of the algorithm depends on the number of places in the nets and on the size of the query (16) of Corollary 1, we first apply a few optimizations in the proposed algorithm. Firstly, we optimize the number of non-admissible places \(Na\Theta\) in the twin. In theory, there are as many places \(Na\Theta\) as there are time decompositions in the system, however as \(Na\Theta\) is used to assert whether the current marking is admissible or not, it is actually possible to merge all the places \(Na\Theta\) into one and only one so in the resulting twin, there are only two places namely \(Na\Theta_1\) and \(Na\Theta_2\) to represent the non-admissible runs of the duplicated systems Π1 and Π2 (see Section V-B). Secondly, to reduce the size of the query (16), we do not encode \(\square \text{AMBIGUOUS}(\Gamma)\) (eq (13)) but only \(\text{MATCH}(\Pi_1) \land \text{ADM}(\Pi_2)\) as it is sufficient: we let the model checker handle the case where \(\text{NoMatch}(\Pi_2)\) is false (i.e. no ambiguity so no diagnosability issue) in the premise which implies the global question is true (implication). We also do not encode \(\text{ADM}(\Pi_2)\) required in \(\text{MATCH}(\Pi_2)\) (eq (11)) in the conclusion of the implication as it is implicit (already in the premise of the implication in \(\text{ADM}(\Gamma)\)). Table II presents the experimental results for these scenarios with the optimized algorithm. For instance, consider Case 5 where the pattern is \(\Omega_2(1)\) with \(Q_{\Omega_2(1)} = \{M|M(\text{pp}_1^r) = 1\}\) (see Figure 2b, \(k = 1\)), the actual optimized query used to get the results is

\begin{center}
\begin{tabular}{|c|c|c|}
\hline
Case & Pattern & System \\
\hline
1 & \(\Omega_1^*(1)\): one event \(b\) & Θ \\
2 & \(\Omega_2^*(1)\): two events \(b\) & Θ \\
3 & \(\Omega_1^*(10)\): ten events \(b\) & Θ \\
4 & \(\Omega_1^*(1)\): one event \(f\) & Θ \\
5 & \(\Omega_2^*(1)\): one event \(f\) or one event \(b\) & Θ \\
6 & \(\Omega_2^*(1)\): one event \(f\) or one event \(b\) & Θ: \(a[0, 3] \rightarrow a[2, 3]\) \\
7 & \(\Omega_4(4)\): 4 consecutive events \(b\) without an \(f\) & Θ: \(a[0, 3] \rightarrow a[2, 3]\) \\
8 & \(\Omega_4(4)\): 4 consecutive events \(b\) without an \(f\) & Θ \\
\hline
\end{tabular}
\end{center}

\textbf{Table I:} Diagnosability analyses described in Example 2.
Table II is:

\[
\text{Diagnosable}(\Theta, \Omega_2(1)) \equiv \\
\left( \left( Na\Theta_1 = 0 \land Na\Theta_2 = 0 \land pp1^2 = 1 \right) \Rightarrow \\
\left( \left( Na\Theta_1 = 0 \land Na\Theta_2 = 0 \right) \Rightarrow \left( pp2^2 = 1 \lor \text{dead} \right) \right) \right)
\] (17)

Place name \(pp_1^2\) denotes the place \(pp_2^2\) in the corresponding product \(\Pi_i\) (they are pattern places inside \(\Pi_i\)). MATCH(\(\Pi_i\)) \(\land\) ADM(\(\Pi_2\)) is encoded with \((Na\Theta_1 = 0 \land pp1^2 = 1 \land Na\Theta_2 = 0)\). \(pp2^2 = 1\) is the optimized way to encode MATCH(\(\Pi_2\)).

Table II presents for each scenario the size of the twin \(\Gamma\), the size of the strong state class graph computed by TINA during the verification and the overall time for the computation of the results (time that measures the complete computational time of Algorithm 1). A scenario labeled with OK is a diagnosable scenario, otherwise it is labeled with KO. All the results are consistent with the intuition described in Example 2, the most complex one being obtained in less than 25s in one core of a Ryzen 7 1770 2.2Ghz. We finally recall that, without any quantitative information about time, none of these scenarios would be diagnosable. This explains the size of the SSGs which implicitly represent all the time interleavings of events.

<table>
<thead>
<tr>
<th>(\Gamma) Size (pl/tr/arc)</th>
<th>SSG st.</th>
<th>SSG tr.</th>
<th>Time (s)</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 44/67/224</td>
<td>65101</td>
<td>188706</td>
<td>1.931</td>
<td>OK</td>
</tr>
<tr>
<td>2 46/78/232</td>
<td>101671</td>
<td>290471</td>
<td>2.918</td>
<td>OK</td>
</tr>
<tr>
<td>3 62/94/296</td>
<td>394231</td>
<td>1104591</td>
<td>12.074</td>
<td>OK</td>
</tr>
<tr>
<td>4 44/76/224</td>
<td>85107</td>
<td>231247</td>
<td>1.957</td>
<td>KO</td>
</tr>
<tr>
<td>5 46/84/248</td>
<td>83143</td>
<td>246688</td>
<td>2.560</td>
<td>KO</td>
</tr>
<tr>
<td>6 46/84/248</td>
<td>51734</td>
<td>146106</td>
<td>1.518</td>
<td>OK</td>
</tr>
<tr>
<td>7 52/94/288</td>
<td>145627</td>
<td>390583</td>
<td>4.163</td>
<td>OK</td>
</tr>
<tr>
<td>8 52/94/288</td>
<td>705376</td>
<td>2012231</td>
<td>21.506</td>
<td>KO</td>
</tr>
</tbody>
</table>

**Table II:** Experimental results for Example 2.

**VII. CONCLUSION**

This paper introduces pattern diagnosability for timed discrete event systems as an extension of the single fault diagnosability problem. A fully automated approach to check the problem over safe LTPN is then formally detailed and fully implemented as a model-checking problem. One of the main difficulties is to formally characterize the occurrence of a pattern which is achieved by the definition of an original pattern matching product. In our perspectives, we plan to extend this approach to also deal with time patterns and characterize specific subclasses of decidable problems in LTPN. We also plan to update the model-checker to specifically deal with twin-plant construction as initiated in our recent work [41].

**APPENDIX**

**Lemma 1.** For any timed sequences \(\rho_1, \rho_2 \in L(\Theta)\) such that \(P^\Theta_o(\rho_1) = P^\Theta_o(\rho_2) = \sigma\) there exists a timed sequence \(\rho \in L(\Gamma, Q^\adm_\Gamma)\) such that \(P^\Gamma_o(\rho) = \rho_i, i \in \{1, 2\}\).

**Proof.** Theorem 1 states that one can find a run \(r_i\) in \(\Pi_i\) such that \(\rho_i^t \in \Theta \_ \Gamma_{Pi_i}(r_i)\) and \(\rho_i\) is just as \(\rho_i^t\) by renaming the unobservable event \((e, i)\) from \(\rho_i^t\) to \(e\). As \(P^\Theta_o(\rho_1) = P^\Theta_o(\rho_2) = \sigma\), it follows that both runs \(r_1\) fire observable transitions not only at the same date but also with the same label. By definition of \(\Gamma\) (Definition 11), the firing conditions of any unobservable transitions from both \(\Pi_i\) remain the same in \(\Gamma\). Consider \(t_1\) from \(r_1\) and \(t_2\) from \(r_2\) be the first couple of observable transitions that is fired. The transition fusion in \(\Gamma\) ensures that the transition \(e_{12}\) of \(t_1\|t_2\) (see Figure 4) can be fired at the same date and its postset marking remains admissible. Therefore, this postset marking gathers the postset markings of \(t_1\) and \(t_2\) which means that the firing conditions of next transitions in \(r_1\) and \(r_2\) remain the same in \(\Gamma\). Applying inductively the same reasoning to the sequence of couples \((t_1, t_2)\) of observable transitions fired at the same time in \(r_1\) and \(r_2\), it follows that we can design an admissible run \(r\) of \(\Gamma\) such that \(P^\Gamma_o(\rho) = \sigma\). \(P^\Gamma_o(\rho) = \rho_i, i \in \{1, 2\}\). □

**Lemma 2.** For any timed sequence \(\rho \in L(\Gamma, Q^\adm_\Gamma)\) there exists a couple of timed sequences \(\rho_1, \rho_2 \in L(\Theta)\) such that \(P^\Theta_o(\rho_1) = \rho_i, i \in \{1, 2\}\) and \(P^\Theta_o(\rho_2) = P^\Theta_o(\rho_1) = P^\Theta_o(\rho_2)\).

**Proof.** As \(\rho \in L(\Gamma, Q^\adm_\Gamma)\), there exists an admissible run \(S_{\rho_1} \rightarrow S\) in \(\Gamma\) such that \(\rho \in \Theta \_ \Gamma_{Pi_i}(r_i)\). By Definition 11, \(r\) is a sequence composed of transitions either from the products \(\Pi_{e_i(1, 2)}\) or being transition fusions like \(t_1\|t_2\) where \(t_i\) is in \(\Pi_i\). Consider now \(t_1\|t_2\) as the first transition fusion present in \(r\). As \(r\) is admissible, there must be only one fire of \(cs_{e_i(1, 2)}\) followed by one fire of \(e_{12}\) (transitions from \(t_1\|t_2\), (see Figure 4)) associated with this first transition fusion \(t_1\|t_2\). By definition of \(\Gamma\) (Definition 11), the firing condition of any unobservable transition from \(\Pi_{e_i(1, 2)}\) remain the same in \(\Gamma\), so the sequence \(r_i\) of any unobservable transition fired in \(r\) from \(\Pi_i\) before the fire of transitions \(cs\Theta_i\) can be fired in the same conditions in \(\Pi_i\). As the preset of \(cs\Theta_i\) is the preset of \(t_i\), it means that \(t_i\) is enabled in \(\Pi_i\) at the same time than \(cs\Theta_i\) in \(\Gamma\). As \(e_{12}\) is fired in \(r\) it means that \(t_i\) is fireable at the same absolute date than \(e_{12}\) after the run \(r_i\) in \(\Pi_i\). The postset marking of \(e_{12}\) being admissible, it also gathers the postset markings of \(t_i\) from \(\Pi_i\); it follows that we can design a run \(r_i\) followed by the fire of \(t_i\) in \(\Pi_i\) that leads to the marking \(M_i\) included in the marking of \(\Gamma\) after the fire of \(e_{12}\) in \(r\). Applying inductively this reasoning on the sequence of transition fusions present in \(r\), it follows that we can design a couple of runs \(r_1\) and \(r_2\) from \(r\) such that there exists \(\rho_i \in \Theta \_ \Gamma_{Pi_i}(r_i)\) and \(P^\Pi_o(\rho_i) = P^\Gamma_o(\rho)\). The result then follows from Theorem 1. □

**REFERENCES**
