A Model Checking method to solve the event pattern diagnosis problem in safe labeled time Petri nets

Yannick Pencolé[1] and **Audine Subias**[2] and **Camille Coquand**[2]
[1] LAAS-CNRS, Univ. Toulouse, CNRS, Toulouse, France.
e-mail: yannick.pencole@laas.fr
[2] LAAS-CNRS, Univ. Toulouse, INSA, Toulouse, France.
e-mail: audine.subias@laas.fr, camille.coquand@laas.fr

Abstract

This paper addresses the problem of identifying whether a given pattern of unobservable events has occurred in a partially observable timed discrete event system. The systems that are considered here are modelled with safe labeled time Petri nets. To solve the problem, we propose to use a model checking approach. Given the model of the system, the pattern to analyse, and the sequence of observations produced by the system, the method firstly builds automatically a labeled time Petri net and a pair of queries that represent the pattern diagnosis problem to solve. The second step then consists in using a model-checker for time Petri net that provides the answer to the queries. The solution provided by the model-checker is then analysed to determine whether the pattern under investigation has definitely occurred or not.

1 Introduction

Classically, the problem of model-based fault diagnosis in discrete event systems (DES) is the problem of determining whether a set of unobservable and faulty events has occurred in the system based on a complete behavioural model and a sequence of observations produced by the system. This initial problem, originally introduced in [1], has been addressed for decades based on different formalisms such as automata, Petri nets, process algebra and a survey of these techniques can be found in [2] and for specifically Petri nets in [3].

Based on these previous contributions, two independent extensions have been proposed. The first one is about dealing with time. In the initial definition of the problem, time is just characterized by events occurring *sequentially* but their dates of occurrences are not taken into account. The date itself, as a numerical piece of information, is an observation and depending on the date of a given observation the diagnosis might be different. The extension of the problem to deal with time has been formally addressed in [4] where the considered formalism is time automata. The decidability question as well as some complexity results can also be found in [5]. A few method based on time automata have then been proposed in [6] and more recently in [7]. Extensions to deal with time have also been addressed with formalisms like time Petri nets [8; 9; 10; 11].

The second type of extension is about the nature of what is diagnosed. In the previous cited pieces of work, the result of the diagnosis method is about the occurrence of faults represented as *single events*. In [12], the notion of *supervision patterns* is introduced as automata, also called temporal specification in [13] or semantic patterns in [14]. A *supervision pattern* is a specific assembling of events and this assembling is considered as faulty or more generally as critical enough so that knowing whether this pattern of events has definitely occurred in the system is of importance to assist any further decision (repair, reconfiguration, maintenance). A succint representation of such patterns is proposed as Petri nets in [15] and a method for diagnosing such patterns on labeled Petri net is proposed in [16].

The goal of this paper is to extend the method proposed in [16] to Labeled Time Petri nets (LTPN) so that we can benefit of the observation of dates to make the pattern diagnosis more accurate. We propose to turn the pattern diagnosis problem over LTPN as a reachability problem over LTPN that can be solved by a model-checker. Given a pattern, the model of a system and a sequence of observations, the proposed method builds a single LTPN that gathers all the information and designs a set of formal properties over this LTPN that will produce the diagnosis result. The effective construction of this LTPN requires specific products that have been recently introduced in [17].

2 Background

This section recalls the mathematical background that will be used throughout this paper.

2.1 Timed language

A *timed sequence* over a set of labels Σ is a sequence ρ of pairs (s, d) where d is a duration and s is a symbol of $\Sigma \cup \{\lambda\}$. Throughout this paper, symbol λ will represent the occurrence of time and will be always considered as a silent event (λ may be part of Σ or not). The set of timed sequences over Σ is denoted $\mathcal{T}(\Sigma)$ and $\mathcal{T}(\Sigma) \subseteq (\Sigma \cup \{\lambda\} \times \mathbb{R}^+)^+$. Each timed sequence ρ has a canonical representation denoted $[\rho]$ with only one λ at the end. Suppose for instance that $\Sigma = \{a, b, c\}$, then $\rho = (a, 3).(c, 4).(\lambda, 2).(b, 1).(b, 2).(\lambda, 4)$ is a timed sequence of $\mathcal{T}(\Sigma)$ also denoted $3a4c2\lambda1b2b4\lambda$. The canonical representation of ρ is $[\rho] = 3a4c3b2b4\lambda$. Throughout this paper, only canonical representations are considered. Given two alphabets Σ_1, Σ_2, the projection $\mathbf{P}_{\Sigma_1 \to \Sigma_2}$: $\mathcal{T}(\Sigma_1) \to \mathcal{T}(\Sigma_2)$ of a canonical timed sequence is defined as follows.

- $\mathbf{P}_{\Sigma_1 \to \Sigma_2}(\theta\lambda) = \theta\lambda$, for any $\theta \in \mathbb{R}^+$;
- if $e \in \Sigma_2$, $\mathbf{P}_{\Sigma_1 \to \Sigma_2}(\theta e.\rho) = \theta e.\mathbf{P}_{\Sigma_1 \to \Sigma_2}(\rho)$;
- if $e \in \Sigma_1 \setminus \Sigma_2$, two cases hold
  - $\mathbf{P}_{\Sigma_1 \to \Sigma_2}(\theta e\rho) = (\theta + \theta')\lambda$, if $\theta'\lambda = \mathbf{P}_{\Sigma_1 \to \Sigma_2}(\rho)$.
  - $\mathbf{P}_{\Sigma_1 \to \Sigma_2}(\theta e.\rho) = (\theta + \theta')e'.\rho'$, if $\theta' e'.\rho' = \mathbf{P}_{\Sigma_1 \to \Sigma_2}(\rho)$.

As an example, let $\Sigma' = \{c\}$ then $\mathbf{P}_{\Sigma \to \Sigma'}([\rho]) = 7c9\lambda$.

2.2 Labeled time Petri nets

Definition 1 (LPN)**.** *A Labeled Petri Net (LPN for short) is a 5-uple $N = \langle P, T, A, E, \ell \rangle$ such that:*

- *P is a finite set of places;*
- *T is a finite set of transitions ($P \cap T = \varnothing$);*
- *$A \subseteq (P \times T) \cup (T \times P)$ 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 the transition labeling function.*

The *preset* $\mathrm{pre}(t)$ of a transition t is the set of (input) places $\mathrm{pre}(t) = \{p \in P : (p, t) \in A\}$ and the *postset* $\mathrm{post}(t)$ of a transition t is the set of (output) places $\mathrm{post}(t) = \{p \in P : (t, p) \in A\}$. A *state* of a LPN is a *marking* that is a function (M: $P \to \mathbb{N}$) that assigns to each place a number of tokens (or marks). All along this paper, we will consider that any LPN is safe i.e. M: $P \to \{0, 1\}$. A transition t is *enabled* by a given marking M (denoted $t \in \mathit{enabled}(M)$) if and only if ($\forall p \in \mathrm{pre}(t), M(p) = 1$). An LPN is usually associated with an *initial marking* denoted M_0. A Labeled Time Petri net [18] is a specific class of LPN that maps any transition to a static time interval. Throughout this paper, the static interval $I_s(t)$ of a transition t is either closed ($[a, b]$) or semi-closed ($[a, +\infty[$).

Definition 2 (LTPN)**.** *A Labeled Time Petri Net (LTPN for short) is a 6-uple $N = \langle P, T, A, E, \ell, I_s \rangle$ such that:*

- *$\langle P, T, A, E, \ell \rangle$ is a LPN;*
- *$I_s : T \to \mathbb{I}_{\mathbb{Q}^+}$ is the static time interval function that maps any transition to an interval of $\mathbb{I}_{\mathbb{Q}^+}$ (an interval of $\mathbb{I}_{\mathbb{Q}^+}$ is an interval of reals whose bounds are either positive rationals or $+\infty$).*

A *state* of an LTPN is a couple $S = \langle M, I \rangle$ where I is a partial firing interval application I: $T \to \mathbb{I}_{\mathbb{R}^+}$ that associates to any enabled transition t a time interval of $\mathbb{R}^+$ ($\mathbb{I}_{\mathbb{R}^+}$: 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 $\lfloor I(t) \rfloor$, and its upper bound, also called the date of later firing, is denoted $\lceil I(t) \rceil$. The initial state of a marked LTPN is given by $S_0 = \langle M_0, I_0 \rangle$ where I_0 is such that $I_0(t) = I_s(t)$ for any transition t enabled by M_0.

Intuitively, in an LTPN a transition is firable 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 *firable* from a state $S = \langle M, I \rangle$ at a relative date θ if and only if:

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

Firing a firable transition t at the relative date θ from the state $S = \langle M, I \rangle$ leads to a state $S' = \langle M', I' \rangle$ such that:

- M' is such that $\forall p \in \mathrm{pre}(t) \setminus \mathrm{post}(t), M'(p) = 0, \forall p \in \mathrm{post}(t) \setminus \mathrm{pre}(t), M'(p) = 1$, 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 $I'(t') = I_s(t')$.

In the following, the fire of a transition t at the relative date θ is denoted: $\langle M, I \rangle \xrightarrow{\theta t} \langle M', I' \rangle$. A state S is *reachable* in a marked LTPN if it exists a sequence of firable transitions $S_0 \xrightarrow{\theta_1 t_1} S_1 \xrightarrow{\theta_2 t_2} \ldots \xrightarrow{\theta_k t_k} S$, also denoted $S_0 \xrightarrow{r} S$ where r is the timed transition sequence $r = \theta_1 t_1 \theta_2 t_2 \ldots \theta_k t_k$. The sequence r will be called a *run* of the net. Any run $r' = \theta_1 t_1 \ldots \theta_i t_i, i \leq k$ is a *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 $S' = \langle M', I' \rangle$ such that $S' \in R(N, \langle M, I \rangle)$.

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:

$$\theta\_\ell(r) = \{[\theta\_\ell^{\dashv}(r).d\lambda], d \in [0, d_{\max}[\}, \qquad (1)$$

where $\theta\_\ell^{\dashv}(r)$ is the *earliest* timed sequence of r:

- if r is the empty run (no transition) then $\theta\_\ell^{\dashv}(r) = 0\lambda$;
- if $r = \theta_1 t_1 \theta_2 t_2 \ldots \theta_k t_k, k > 0$ then $\theta\_\ell^{\dashv}(r) = [\theta_1 \ell(t_1) \theta_2 \ell(t_2) \ldots \theta_k \ell(t_k) 0\lambda]$.

Date d_{max} is the latest possible duration stay in state S (i.e. the minimal upper bound of the current firing intervals $d_{max} = min_{t \in enabled(M)}(\lceil I(t) \rceil)$). 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 3 (Language of an LTPN over Q)**.** *The language generated by a marked LTPN N over Q is*

$$\mathcal{L}(N, Q) = \bigcup_{r: S_0 \xrightarrow{r} S = \langle M, I \rangle \wedge M \in Q} \theta\_\ell(r)$$

2.3 LTPN extensions

We recall here two classical LTPN extensions that will be also used in this paper: *inhibitor arc* and *transition priorities*. 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. A *priority* defines a binary relation between two transitions of an LTPN, denoted $t_1 \succ t_2$ (t_1 has priority over 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.

3 Problem statement

The fault diagnosis problem has been formally defined with the help of timed automata in [4]. We propose here to extend this definition to deal with more complex behaviors than single faults.

3.1 System model

We consider throughout this paper that the supervised system is modelled as a safe LTPN denoted $\Theta = \langle P_\Theta, T_\Theta, A_\Theta, \Sigma_\Theta, \ell_\Theta, I_{\Theta S}\rangle$ with an initial marking denoted $M_{0\Theta}$. Σ_Θ represents the set of event types that are effectively generated by the system ($\lambda \notin \Sigma_\Theta$). 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 Σ_Θ are then partitioned into two subsets: Σ_Θ^o is the set of observable events and Σ_Θ^u the set of unobservable ones. Finally, we assume that the model does not contain any zeno run (a zeno run is an infinite sequence of transitions that can occur in a finite amount of time which is unrealistic in real systems).

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 $p_i^1, i \in \{1,\dots,5\}$ while the activities of the second process (on the right part of the figure) are modeled by the places $p_i^2, i \in \{1,\dots,6\}$. Both processes are synchronized on transition t_1^{12} whose firing leads to the marking of a shared place p^{12} 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_3^1 and p_6^2 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 $\mathbf{o_1}$ (resp. $\mathbf{o_2}$) events as observations.

3.2 Event pattern model

Event patterns gather a set of specific combinations of untimed and unobservable 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 are the same as in [17] and are defined as LPNs. As for the system model, transitions labels represent events of the system.

Definition 4 (Pattern). *A pattern Ω is an LPN $\Omega = \langle P_\Omega, T_\Omega, A_\Omega, \ell_\Omega, \Sigma_\Omega, M_{0\Omega}\rangle$ associated with a set of final markings $Q_\Omega \subset R(\Omega, M_{0\Omega})$ such that:*

1. *(unobservable) events of Σ_Ω 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 of $R(\Omega, M_{0\Omega})$ 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 $M' \in Q_\Omega$.*

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_k^1 indicates the pattern has occurred then the set of final markings is given by $Q_{\Omega_1^b(k)} = \{M\colon M(pp_k^1) = 1\}$. 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_1^1 and then $Q_{\Omega_2(1)} = \{M\colon M(pp_1^2) = 1\}$. The marking of the place pp_2^2 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 $Q_{\Omega_2(2)} = \{M\colon M(pp_2^2) = 1\}$. The number of layers in the pattern depends on the k value. The k occurrences are obtained when the place pp_k^2 is marked hence $Q_{\Omega_2(k)} = \{M\colon M(pp_k^2) = 1\}$. The pattern $\Omega_2(k)$ 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 $\Omega_3(k)$ 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, $Q_{\Omega_3(k)} = \{M\colon M(pp_k^3) = 1\}$.

3.3 Pattern diagnosis

This subsection introduces the problem of diagnosing patterns in timed discrete event systems. For a given pattern Ω, the diagnosis problem consists in defining an Ω-diagnoser 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 Ω or not. It results in the generation of three possible symbols: $\Omega-$*faulty* (all the possible runs match Ω), $\Omega-$*safe* (none of them match Ω), $\Omega-$*ambiguous* (some of them match Ω, some of them do not) [15]. A run r of the system *matches* a pattern Ω if one can find in a word $\rho = \theta_1 e_1 \dots \theta_n e_n \theta_{n+1}\lambda$ of $\theta\_\ell(r)$ a sub-word $\left(\sum_{i=1}^{j_1}\theta_i\right) e_{j_1} \left(\sum_{i=j_1+1}^{j_2}\theta_i\right) e_{j_2} \dots \left(\sum_{i=j_{k-1}+1}^{j_k}\theta_i\right) e_{j_k}$ such that $\rho_\Omega = e_{j_1}\dots e_{j_k}$ is a word of $\mathcal{L}(\Omega)$: this will be denoted by $\rho \Supset \rho_\Omega$ and more generally by $\rho \Supset \Omega$ and $r \Supset \Omega$.

Checking whether a run of Θ can generate the given observations (i.e. the run is consistent with the observations)

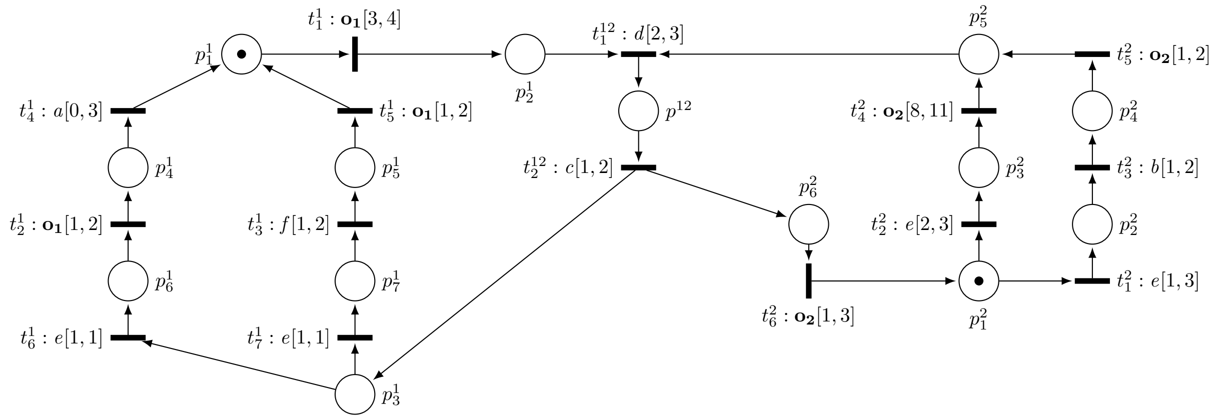


Figure 1: A system Θ composed of two communicating and partially observable timed processes

relies on the timed language projection from the alphabet Σ_Θ to the observable alphabet Σ_Θ^o (see Section 2.1). The Ω-diagnoser can then be formally defined as follows.

Definition 5 (Ω-diagnoser)**.** *An Ω-diagnoser is a function*

$\Delta_\Omega : \mathcal{T}(\Sigma_\Theta^o) \rightarrow \{\Omega{-}\textit{faulty}, \Omega{-}\textit{safe}, \Omega{-}\textit{ambiguous}\}$ *with*

- $\Delta_\Omega(\sigma) = \Omega{-}\textit{faulty}$ *if for any $\rho \in \mathcal{L}(\Theta)$ that is consistent with σ (i.e. $\mathbf{P}_{\Sigma_\Theta \rightarrow \Sigma_\Theta^o}(\rho) = \sigma$), $\rho \Supset \Omega$;*
- $\Delta_\Omega(\sigma) = \Omega{-}\textit{safe}$ *if for any $\rho \in \mathcal{L}(\Theta)$ that is consistent with σ, $\rho \not\Supset \Omega$;*
- $\Delta_\Omega(\sigma) = \Omega{-}\textit{ambiguous}$ *otherwise.*

It must be noticed that, even if Definition 5 uses Petri nets (Θ and Ω notations), this definition is purely language-based. The diagnosis problem is defined over the set of timed sequences ρ in $\mathcal{L}(\Theta)$, the matching relation is defined over event sequences and consistency is achieved by sequence projection. It follows that Definition 5 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 Δ: $\mathcal{T}(\Sigma_\Theta^o) \rightarrow \prod_{i=1}^n \{\Omega_i{-}\textit{faulty}, \Omega_i{-}\textit{safe}, \Omega_i{-}\textit{ambiguous}\}$ 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 σ_1 = $3\mathbf{o_1}2\mathbf{o_2}8\mathbf{o_2}2\mathbf{o_1}1\lambda$ (that is the reception of event $\mathbf{o_1}$ at date 3 then the reception of $\mathbf{o_2}$ at date 5 and another reception of $\mathbf{o_2}$ at date 13 followed by $\mathbf{o_1}$ at date 15 and finally the observed sequence ends at date 16). The pattern $\Omega_1^b(1)$ models the single occurrence of b. Based on σ_1, we know that the first occurrence of $\mathbf{o_2}$ is at the absolute date 5 which means that an occurrence of b has definitely occurred. So $\Delta_{\Omega_1^b(1)}(\sigma_1) = \Omega_1^b(1){-}\textit{faulty}$. Considering now pattern $\Omega_1^b(2)$ (i.e. two occurrences of b). If b has occurred twice, the second occurrence is after the second occurrence of $\mathbf{o_2}$ at date 13. As σ_1 lasts at date 16, it is possible that b has occurred twice (the second occurrence would be at date in* $[15, 16]$*) but not necessary: $\Delta_{\Omega_1^b(2)}(\sigma_1) = \Omega_1^b(2){-}\textit{ambiguous}$.*

The pattern $\Omega_2(2)$ models two occurrences of an event among $\{b, f\}$. Based on σ_1, we know that b has definitely occurred once. As the second occurrence of event $\mathbf{o_1}$ occurs at date 15, it enforces that transition t_3^1 has been fired once so the event f has certainly occurred and $\Delta_{\Omega_2(2)}(\sigma_1) = \Omega_2(2){-}\textit{faulty}$. Finally, $\Delta_{\Omega_3(3)}(\sigma_1) = \Omega_3(3){-}\textit{safe}$ (event f has necessarily occurred once and b at most twice).

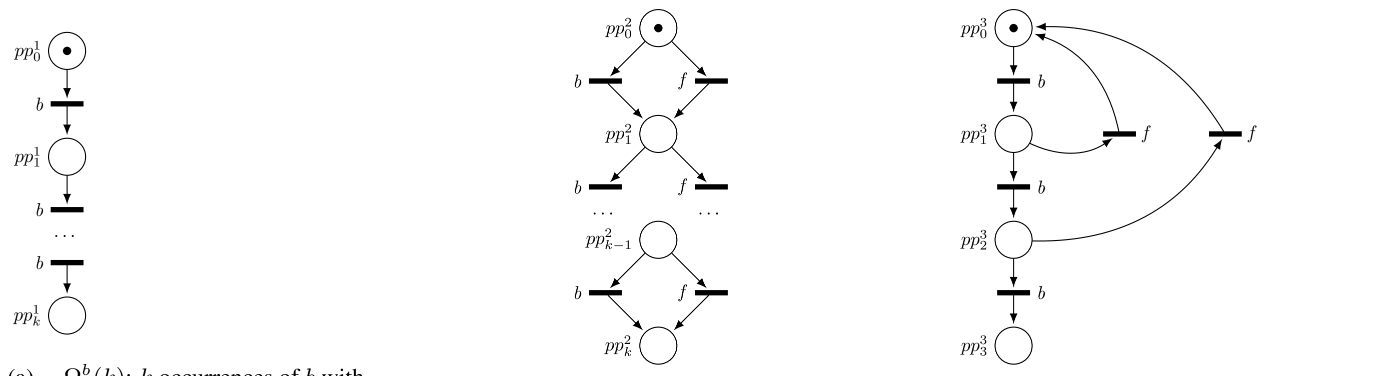
4 Proposed method

The proposed method relies on the formal framework proposed in [17] that is used for analyzing pattern diagnosability on the same type of systems. Figure 3 presents an overview of the proposed method.

The main principle of the method is to synthesize an LTPN, denoted $\Pi_\sigma^{\Theta,\Omega}$, that represents as a whole the set of runs of Θ that are consistent with σ and for each run of these runs, the LTPN should also represent the fact that the pattern Ω is matched or not. Once $\Pi_\sigma^{\Theta,\Omega}$ is obtained, the remaining step is to check into $\Pi_\sigma^{\Theta,\Omega}$ whether the runs match the pattern or not. To do so, we propose first to synthesize a set of formal queries and secondly to use a model-checker on $\Pi_\sigma^{\Theta,\Omega}$ to solve the queries and get the diagnosis (see Section 5 for details about this part). This section now presents how the LTPN $\Pi_\sigma^{\Theta,\Omega}$ is computed based on the LTPN Θ, the LPN Ω and the timed sequence σ. The computation is based on specific compositions of LTPNs that have been formally introduced in [17] and that are described in the next subsections.

4.1 About composing LTPNs

Usually, composition of Petri nets relies on the synchronization of some transitions. However, in time Petri nets, compositionnality is kept only when synchronizations involve transitions with an interval $[0, +\infty[$ [19] 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 to ensure that the synchronization is performed only on transitions with $[0, +\infty[$

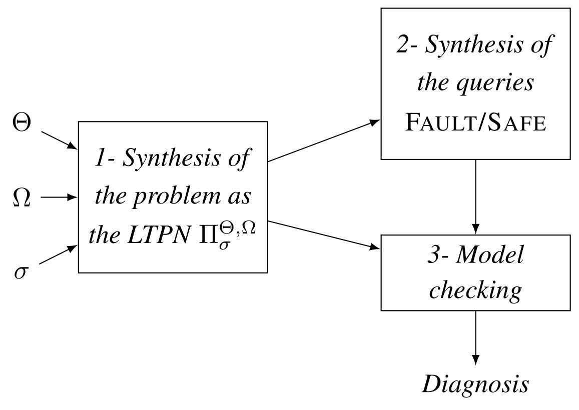


(a) $\Omega_1^b(k)$: k occurrences of b with $Q_{\Omega_1^b(k)} = \{M\colon M(pp_k^1) = 1\}$.

(b) $\Omega_2(k)$: k occurrences of b or f with $Q_{\Omega_2(k)} = \{M\colon M(pp_k^2) = 1\}$.

(c) $\Omega_3(3)$, 3 consecutive occurrences of b without any occurrence of f, $Q_{\Omega_3(3)} = \{M\colon M(pp_3^3) = 1\}$.

Figure 2: A set of patterns.



Θ
Ω
σ
1- Synthesis of the problem as the LTPN $\Pi_\sigma^{\Theta,\Omega}$
2- Synthesis of the queries FAULT/SAFE
3- Model checking
Diagnosis

Figure 3: Overview of the method for diagnosing an occurrence of Ω in the system Θ based on the observations σ.

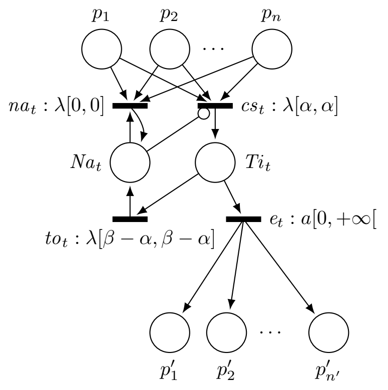


Figure 4: Time decomposition of a transition t of a LTPN N with $\ell(t) = a$ and $I_s = [\alpha,\beta], \alpha > 0$.

time intervals. The time decomposition of transition t with preset $p_1,\dots,p_n$ and postset $p_1',\dots,p_{n'}'$ consists in replacing t in the initial LTPN N by the set of transitions/places defined in Figure 4. This time decomposition ensures that transition e_t is firable exactly in the same conditions as the initial transition t. Indeed, once t is enabled by the marking of $p_1,\dots,p_n$, transition cs_t (i.e. *clock start*) is enabled in the decomposition and is silenty fired (λ event) exactly at time $\alpha > 0$ so transition e_t starts being firable exactly from time α till time β. At β, either e_t or the silent transition to_t (i.e. *time out*) is fired, so e_t is firable in $[\alpha,\beta]$ as t and leads to the same postset. Now, as soon as to_t is fired, it means that the behavior is actually *not admissible* (transition t must have been fired in $[\alpha,\beta]$ as it is conflict-free), the place Na_t then permanently holds a token and inhibits any further fire of cs_t thanks to an inhibitor arc between Na_t and cs_t. The aim of na_t is then to empty the preset of cs_t in case of a non admissible behavior.

4.2 Pattern matching product

The first step of the computation of $\Pi_\sigma^{\Theta,\Omega}$ is to compute an LTPN that represents how a system Θ is actually matching a pattern Ω: this step is based on the *pattern matching product*. This product relies on a specific product between the transitions of Θ and Ω. Consider now t as a transition of Θ (with $\ell_\Theta(t) = a$, $I_{S\Theta}(t) = [\alpha,\beta], \alpha > 0$) [1] with $\mathrm{pre}(t_\Theta) = \{p_1,\dots,p_n\}$ and $\mathrm{post}(t) = \{p_1',\dots,p_{n'}'\}$. Without loss of generality, we assume that t is conflict-free (no other transition has the same preset).[2] Let t_Ω be a transition of Ω labeled with a with $\mathrm{pre}(t_\Omega) = \{q_1,\dots,q_m\}$ and $\mathrm{post}(t_\Omega) = \{q_1',\dots,q_{m'}'\}$. To effectively obtain the pattern matching product between t and t_Ω, first a time decomposition of t is performed in order to replace t with a transition e_t as detailed in the previous subsection. As t_Ω is part of an LPN and not an LTPN, it means that the implicit static time interval of t_Ω is $[0,+\infty[$ so no time decomposition is required for t_Ω. The pattern matching product consists then in replacing e_t and t_Ω by the set of transitions/places defined in Figure 5.

On the left, transition e_t, resulting from the time decomposition of t, is kept as is. On the right, the transition t_Ω is replaced by a synchronized transition $e_t||t_\Omega$ whose preset (resp. postset) is the union of the presets (resp. postsets) of e_t and t_Ω. Finally, a priority constraint is added between

[1]Recall that the system has no zeno runs so we suppose that $\alpha > 0$.

[2]Indeed, if t is in conflict with another transition t', Θ can be transformed by adding two silent transitions $\lambda[0,0]$ enabled by the preset of t and t'. Each silent transition has a postset with one new place that replaces the preset of either t or t'. New versions of t and t' are then conflict-free while the generated langage $\mathcal{L}(\Theta)$ remains the same.

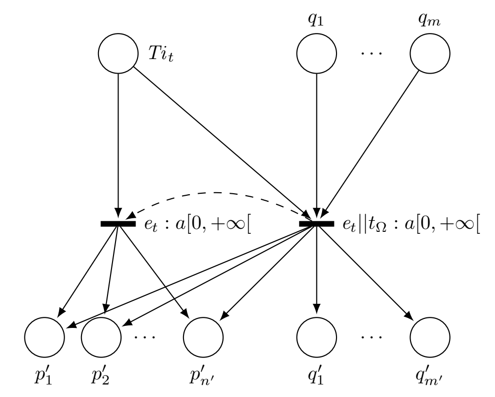


Figure 5: Pattern matching product between a transition t of Θ and a transition of Ω.

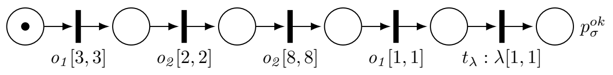


Figure 6: LTPN ξ_σ for observation sequence $3\mathbf{o_1}2\mathbf{o_2}8\mathbf{o_2}1\mathbf{o_1}1\lambda$.

$e_t||t_\Omega$ and e_t.

Applying this product to Θ and Ω then leads to an LTPN denoted $\Theta \ltimes \Omega$. As explained in the previous section, $\Theta \ltimes \Omega$ contains admissible markings (a marking M is admissible if for every place Na_t resulting from the time decomposition of a transition t, $M(Na_t) = 0$). Let $Q^{match}_{\Theta \ltimes \Omega}$ be the set of admissible markings of $\Theta \ltimes \Omega$ such that for every $M \in Q^{match}_{\Theta \ltimes \Omega}$, the restriction of M to the places of Ω belongs to the accepting markings Q_Ω of Ω. Then, as proved in [17], the set of runs in $\Theta \ltimes \Omega$ leading to a marking of $Q^{match}_{\Theta \ltimes \Omega}$ is exactly the set of runs of Θ that match Ω:

Proposition 1.

$$\mathcal{L}(\Theta \ltimes \Omega, Q^{match}_{\Theta \ltimes \Omega}) = \{\rho \in \mathcal{L}(\Theta) | \rho \Supset \Omega\}$$

4.3 Composing with timed observations

To finally get the LTPN $\Pi^{\Theta,\Omega}_\sigma$ as a single LTPN that represents the diagnosis problem, the next step is to find a way to encode whether a run of the system Θ is consistent with the observations σ or not. If a run is consistent with σ it means that it produces the sequence of observations at the correct time. Consider again the observation sequence $3\mathbf{o_1}2\mathbf{o_2}8\mathbf{o_2}1\mathbf{o_1}1\lambda$. The principle is to build out of this sequence, an LTPN (noted ξ_σ) as the one depicted in Figure 6.

It is a single sequence of transitions ending with a specific silent transition t_λ and its postset p_σ^{ok}. By construction of ξ_σ, we have $\mathcal{L}(\xi_\sigma, \{p_\sigma^{ok}\}) = \{\sigma\}$. To fully encode the diagnosis problem in $\Pi^{\Theta,\Omega}_\sigma$, we now need to compose ξ_σ with the pattern product $\Theta \ltimes \Omega$ in such a way that:

1. we keep only the runs of $\Theta \ltimes \Omega$ that produce the events of σ in the same order and at the same date;
2. among these runs we keep the ones which have the same duration than σ.

To implement the first condition, two operations are performed. Firstly, any transition labeled with an observable event that is not shared with at least one transition of ξ_σ must be deactivated. To do so, its postset is simply replaced with a postset containing a single non-admissible place denoted Na_{obs}. Secondly, the idea is to synchronize any transition of ξ_σ with every transition of $\Theta \ltimes \Omega$ that share the same event label. To do so, the first step is to apply the time decomposition on every transition of ξ_σ and on every transition of $\Theta \ltimes \Omega$ that shares an event label $o \in \Sigma^o_\Theta$ with at least one transition of ξ_σ. Once the time decomposition is done, every couple of obtained transitions (e_{t_1}, e_{t_2}), where t_1 belongs to $\Theta \ltimes \Omega$ and t_2 belongs to ξ_σ and such that e_{t_1} and e_{t_2} share the same observable event label $o \in \Sigma^o_\Theta$, is replaced by the synchronized transition $e_{t_1}||e_{t_2}$ labeled with o as detailed in Section 4.2. To implement the second condition, the principle is to finally add a set of constraints as priorities and inhibitor arcs that block any continuation of a run that is fully synchronized with the observations and has the same duration as σ:

1. a priority $t \succ t_\lambda$ is added for any transition t from $\Theta \ltimes \Omega$ or resulting from the time decomposition of a transition in $\Theta \ltimes \Omega$;
2. an inhibitor arc is added between the place p_σ^{ok} and every such transition t.

Priorities ensure that any run that ends with the effective fire of a transition from $\Theta \ltimes \Omega$ at the exact firing date of t_λ has indeed the same duration than σ. Inhibitor arcs then forbid any further fire of any transitions after the fire of t_λ which ensures that any run with greater duration is ignored. From this construction of $\Pi^{\Theta,\Omega}_\sigma$, it follows that:

Proposition 2. *Any run of Θ consistent with σ is an admissible run of $\Pi^{\Theta,\Omega}_\sigma$ that leads to a marking M such that $M(p_\sigma^{ok}) = 1$ and reciprocally.*

5 Pattern diagnosis as a model-checking problem

The previous section details how to build the LTPN $\Pi^{\Theta,\Omega}_\sigma$ and shows that, by solving a reachability problem in $\Pi^{\Theta,\Omega}_\sigma$, it is possible to solve the pattern diagnosis problem. This section now details how to apply a model-checking technique to solve the pattern diagnosis problem based on $\Pi^{\Theta,\Omega}_\sigma$.

5.1 Model-checking problem

Generally speaking, model-checking is a paradigm that gathers the set of techniques that aim at formally checking whether a property φ holds in a finite-state model $\mathcal{M}$:

$$\mathcal{M} \models \varphi. \qquad (2)$$

A model-checker then returns a binary answer:

- either φ holds in $\mathcal{M}$ and the model-checker returns true;
- or φ does not hold in $\mathcal{M}$ and the model-checker returns false and a counter-example which proves the claim (a counter-example e is such that $e \models \mathcal{M} \wedge \neg\varphi$).

We propose to solve the pattern diagnosis problem by using the LTPN model checker TINA [20; 21]. Consider a LTPN N and a property φ written as a SE-LTL formula (State/Event Linear Temporal Logic), TINA then solves the problem $N \models \varphi$.[3] SE-LTL is a temporal logic that is able

[3]Note that the problem $N \models \varphi$ is generally undecidable however TINA is able to determine sufficient conditions for decidable subclasses. In our specific case, the problem we solve are decidable by construction of $\Pi^{\Theta,\Omega}_\sigma$ (safe LTPNs).

to express properties about runs of a LTPN. A formula ψ is a SE-LTL formula if it is a universally quantified formula $\psi ::= \forall\varphi$ where $\varphi ::= cst \mid r \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \bigcirc \varphi \mid \Box\varphi \mid \Diamond\varphi \mid \varphi \mathbf{U} \varphi$. The constant cst can be $\bot$ (false), $\top$ (true), $\mathbf{dead}$ (deadlock), $\mathbf{div}$ (temporal divergence), $\mathbf{sub}$ (partially known state). r defines constraints $e \triangle e$ with $\triangle \in \{=, <, >, \leq, \geq\}$ between arithmetical expressions e involving place and transition symbols from the underlying LTPN. The operators $\bigcirc$ (next), $\Box$ (always), $\Diamond$ (eventually) and $\mathbf{U}$ (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 $p_1, p_2, \dots, p_n$. The set Q of markings $M \in Q$ of N such that $M(p_1) = 1$ and $M(p_2) = 1$ can be characterized by the SE-LTL property denoted $\textsc{Markings}(N, Q)$:

$$\textsc{Markings}(N, Q) \equiv (p_1 = 1 \wedge p_2 = 1)$$

and the problem of checking whether in N, from its initial marking M_0, it is always true ($\Box$) that a run eventually ($\Diamond$) leads to a marking M such that $M(p_1) = 1$ and $M(p_2) = 1$ can be formulated as follows:

$$N \models \Box\Diamond\textsc{Markings}(N, Q)$$

5.2 Design of the diagnosis properties

To solve the diagnosis problem, two SE-LTL properties must be checked. The first one, denoted $\textsc{Faulty}(\Theta, \Omega, \sigma)$, will formally express the property that every run of Θ that is consistent with the observation σ matches the pattern Ω. If the model-checker returns true to the problem $\Pi_\sigma^{\Theta,\Omega} \models \textsc{Faulty}(\Theta, \Omega, \sigma)$ then it means that the Ω-diagnoser must return:

$$\Delta_\Omega(\sigma) = \Omega - faulty. \tag{3}$$

If the model-checker returns false, then it also returns as a counter-example a run consistent with σ that does not match Ω. Then two cases hold:

1. either every run that is consistent with σ, does not match Ω and the diagnosis is Ω-safe
2. or there is another run consistent with σ that matches Ω which means then that the diagnosis is Ω-ambiguous.

To determine whether it is Case 1 or Case 2, a second model-checking problem needs to be solved. The second SE-LTL property to check is denoted $\textsc{Safe}(\Theta, \Omega, \sigma)$ and expresses that every run of Θ that is consistent with the observation σ does not match pattern Ω. If finally the model-checker returns true then it is Case 1, otherwise it is Case 2. Note that, in case of an ambiguity, the method provides two runs, one is matching the pattern Ω and the other is not.

Both formulas $\textsc{Faulty}(\Theta, \Omega, \sigma)$ and $\textsc{Safe}(\Theta, \Omega, \sigma)$ express a property on runs of Θ that are represented as runs leading to admissible markings in $\Pi_\sigma^{\Theta,\Omega}$. A marking M from $\Pi_\theta^{\Theta,\Omega}$ is not admissible as soon as for one place of type $Na\Theta$ (see Section 4.2), we have $M(Na\Theta) \neq 0$ or if $M(Na_{obs}) \neq 0$. Let first define the set of admissible markings of $\Pi_\sigma^{\Theta,\Omega}$ in SE-LTL:

$$\textsc{Adm}(\Pi_\sigma^{\Theta,\Omega}) \equiv \bigwedge_{Na\Theta \in \Pi_\sigma^{\Theta,\Omega}} Na\Theta = 0 \wedge Na_{obs} = 0.$$

The second property is the matching property. By construction of $\Pi_\sigma^{\Theta,\Omega}$ (see Proposition 1), a run matches Ω if it is admissible and leads to a marking which includes an accepting marking of Q_Ω. $\textsc{Match}(\Pi_\sigma^{\Theta,\Omega})$ is the formula representing such markings:

$$\textsc{Match}(\Pi_\sigma^{\Theta,\Omega}) \equiv \textsc{Adm}(\Pi_\sigma^{\Theta,\Omega}) \wedge \textsc{Markings}(\Omega, Q_\Omega).$$

Similarly, the property $\textsc{NoMatch}(\Pi_\sigma^{\Theta,\Omega})$ asserts the admissible marking M of $\Pi_\sigma^{\Theta,\Omega}$ is not a matching property, i.e. its restriction $M_{|\Omega}$ does not belong to the final markings of Ω.

$$\textsc{NoMatch}(\Pi_\sigma^{\Theta,\Omega}) \equiv \textsc{Adm}(\Pi_\sigma^{\Theta,\Omega}) \wedge \neg\textsc{Markings}(\Omega, Q_\Omega).$$

From Proposition 2, a run of Θ that is consistent with σ is characterized by an admissible run of $\Pi_\sigma^{\Theta,\Omega}$ that leads to a marking that includes the marking of p_σ^{ok}.

$$\textsc{Consistent}(\Pi_\sigma^{\Theta,\Omega}) \equiv$$
$$\textsc{Adm}(\Pi_\sigma^{\Theta,\Omega}) \wedge \textsc{Markings}(\Omega, \{p_\sigma^{ok}\}).$$

Finally, $\textsc{Faulty}(\Theta, \Omega, \sigma)$ is defined as follows: it is always ($\Box$) true that if the run is consistent then it matches the pattern.

$$\textsc{Faulty}(\Theta, \Omega, \sigma) \equiv$$
$$\Box(\textsc{Consistent}(\Pi_\sigma^{\Theta,\Omega}) \Rightarrow \textsc{Match}(\Pi_\sigma^{\Theta,\Omega}))$$

$\textsc{Safe}(\Theta, \Omega, \sigma)$ is defined as follows: it is always ($\Box$) true that if the run is consistent then it does not matche the pattern.

$$\textsc{Safe}(\Theta, \Omega, \sigma) \equiv$$
$$\Box(\textsc{Consistent}(\Pi_\sigma^{\Theta,\Omega}) \Rightarrow \textsc{NoMatch}(\Pi_\sigma^{\Theta,\Omega}))$$

5.3 First experimental results

Table 1 presents the implementation results for the scenarios that are presented in Example 1. Each line presents some details about the LTPN $\Pi_\sigma^{\Theta,\Omega}$ that is generated (number of places, transitions, arcs, priorities). To check the properties $\textsc{Faulty}$ and $\textsc{Safe}$, the model-checker TINA first precomputes a Strong State Class Graph (SSG) that is then used as a Kripke structure. Table records the details (states, transitions) for the precomputed SSGs. The result for each query is also presented and shows that there are consistent with the expected result detailed in Example 1.

6 Conclusion

This paper proposes a method to solve the pattern diagnosis problem in timed discrete event systems. This method turns the problem to a couple of model-checking problems over a safe LTPN. The method only takes into account untimed patterns. One perspective would be to extend the method to also deal with timed patterns. Such an extension is not straightforward due to the specificities of the system-pattern product. We will firstly aim at better understanding the pattern matching product for timed patterns by characterizing it as a set of temporal constraints.

References

[1] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. C. Teneketzis. Failure diagnosis using discrete-event models. *IEEE Transactions on Control Systems Technology*, 4(2):105–124, March 1996.

	pl $\Pi_\sigma^{\Theta,\Omega}$	tr $\Pi_\sigma^{\Theta,\Omega}$	arcs $\Pi_\sigma^{\Theta,\Omega}$	prio $\Pi_\sigma^{\Theta,\Omega}$	st SSG	tr SSG	FAULTY	SAFE	time
$\Omega_1^b(1)$	44	56	217	57	273	456	T	F	45ms
$\Omega_1^b(2)$	45	57	222	60	269	451	F	F	49ms
$\Omega_2(2)$	47	62	243	69	273	456	T	F	47ms
$\Omega_3(3)$	48	63	248	72	270	452	F	T	49ms

Table 1: Results for the running example scenarios.

[2] J. Zaytoon and S. Lafortune. Overview of fault diagnosis methods for Discrete Event Systems. *Annual Reviews in Control*, 37(2):308–320, December 2013.

[3] Francesco Basile. Overview of fault diagnosis methods based on Petri net models. In *2014 European Control Conference (ECC)*, pages 2636–2642, June 2014.

[4] Stavros Tripakis. Fault Diagnosis for Timed Automata. In Werner Damm and Ernst Rüdiger Olderog, editors, *Formal Techniques in Real-Time and Fault-Tolerant Systems*, Lecture Notes in Computer Science, pages 205–221. Springer Berlin Heidelberg, 2002.

[5] Patricia Bouyer, Fabrice Chevalier, and Deepak D'Souza. Fault Diagnosis Using Timed Automata. In Vladimiro Sassone, editor, *Foundations of Software Science and Computational Structures*, Lecture Notes in Computer Science, pages 219–233. Springer Berlin Heidelberg, 2005.

[6] Zineb Simeu-Abazi, Maria Di Mascolo, and Michal Knotek. Diagnosis of discrete event systems using timed automata. In *International Conference on cost effective automation in Networked Product Development and Manufacturing*, page 12, Monterrey, Mexico, October 2007.

[7] Patricia Bouyer, Samy Jaziri, and Nicolas Markey. Efficient Timed Diagnosis Using Automata with Timed Domains. In Christian Colombo and Martin Leucker, editors, *Runtime Verification*, Lecture Notes in Computer Science, pages 205–221. Springer International Publishing, 2018.

[8] Thomas Chatain and Claude Jard. Time Supervision of Concurrent Systems Using Symbolic Unfoldings of Time Petri Nets. In Paul Pettersson and Wang Yi, editors, *Formal Modeling and Analysis of Timed Systems*, Lecture Notes in Computer Science, pages 196–210. Springer Berlin Heidelberg, 2005.

[9] René K. Boel and George Jiroveanu. The On-Line Diagnosis of Time Petri Nets. In Carla Seatzu, Manuel Silva, and Jan H. van Schuppen, editors, *Control of Discrete-Event Systems: Automata and Petri Net Perspectives*, Lecture Notes in Control and Information Sciences, pages 343–364. Springer London, London, 2013.

[10] Xu Wang, Cristian Mahulea, and Manuel Silva. Model Checking on Fault Diagnosis Graph. *IFAC Proceedings Volumes*, 47(2):434–439, January 2014.

[11] Francesco Basile, Maria Paola Cabasino, and Carla Seatzu. State Estimation and Fault Diagnosis of Labeled Time Petri Net Systems With Unobservable Transitions. *IEEE Transactions on Automatic Control*, 60(4):997–1009, April 2015.

[12] Thierry Jéron, Hervé Marchand, Sophie Pinchinat, and Marie-Odile Cordier. Supervision patterns in discrete event systems diagnosis. In *2006 8th International Workshop on Discrete Event Systems*, pages 262–268, July 2006.

[13] Shengbing Jiang and Ratnesh Kumar. Failure diagnosis of discrete-event systems with linear-time temporal logic specifications. *IEEE Transactions on Automatic Control*, 49(6):934–945, June 2004.

[14] G. Lamperti and X. Zhao. Diagnosis of Active Systems by Semantic Patterns. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 44(8):1028–1043, August 2014.

[15] Houssam-Eddine Gougam, Yannick Pencolé, and Audine Subias. Diagnosability analysis of patterns on bounded labeled prioritized Petri nets. *Discrete Event Dynamic Systems*, 27(1):143–180, March 2017.

[16] Yannick Pencolé and Audine Subias. Diagnosis of supervision patterns on bounded labeled petri nets by model checking. In Marina Zanella, Ingo Pill, and Alessandro Cimatti, editors, *28th International Workshop on Principles of Diagnosis (DX'17)*, volume 4 of *Kalpa Publications in Computing*, pages 184–199. EasyChair, 2018.

[17] Yannick Pencolé and Audine Subias. Diagnosability of Event Patterns in Safe Labeled Time Petri Nets: A Model-Checking Approach. *IEEE Transactions on Automation Science and Engineering*, pages 1–12, 2021.

[18] Philip M. Merlin and David J. Farber. Recoverability of Communication Protocols - Implications of a Theoretical Study. *IEEE Transactions on Communications*, 24(9):1036–1043, September 1976.

[19] Bernard Berthomieu, Florent Peres, and François Vernadat. Bridging the gap between timed automata and bounded time Petri nets. In *4th International Conference Formal Modeling and Analysis of Timed Systems*, pages 82–97, Paris, France, 9 2006.

[20] Bernard Berthomieu, Pierre-Olivier Ribet, and François Vernadat. The tool TINA - Construction of abstract state spaces for petri nets and time petri nets. *International Journal of Production Research*, 42(14):2741–2756, July 2004.

[21] Bernard Berthomieu, Florent Peres, and François Vernadat. Model Checking Bounded Prioritized Time Petri Nets. In Kedar S. Namjoshi, Tomohiro Yoneda, Teruo Higashino, and Yoshio Okamura, editors, *Automated Technology for Verification and Analysis*, Lecture Notes in Computer Science, pages 523–532. Springer Berlin Heidelberg, 2007.