Observable Simple Temporal Network synthesis for the diagnosis of time patterns in time Petri nets

Camille Coquand, Audine Subias, Yannick Pencolé

Observable Simple Temporal Network synthesis for the diagnosis of time patterns in time Petri nets

Camille Coquand * **Audine Subias** * **Yannick Pencolé** **

* *CNRS, LAAS, Univ de Toulouse, INSA, LAAS, F-31400 Toulouse, France (e-mail: firstname.lastname@laas.fr).*
** *CNRS, LAAS, Univ de Toulouse, LAAS, F-31400 Toulouse, France (e-mail: yannick.pencole@laas.fr)*

Abstract: This paper presents a method for the diagnosis of time patterns in time Petri nets. This method uses a characterization of the pattern called *Observable Simple Temporal Network*, given in the form of a set of observable events with temporal constraints on their occurrence dates. The proposed diagnoser verifies if a part of an input timed sequence of observations is consistent with the characterization. If the pattern has not occurred, this consistency test will lead to the same conclusion. One the other hand, if the pattern has occurred, the consistency test will lead to an ambiguous diagnosis in the general case or to the conclusion that the pattern has definitely occurred if the underlying system is diagnosable.

Keywords: Time pattern, Discrete event system, Time Petri nets, State class graph, Observable Simple Temporal Network.

1. INTRODUCTION

The problem of diagnosis of *Discrete Event Systems* (DES) in which a fault is considered as an unobservable event that may occur during the system life has been well studied since the *90s* (Sampath et al. (1995)). Among the studied formalisms, one can mention Petri nets, for which different diagnosis and diagnosability techniques have been developed (Cabasino et al. (2010) for example). An overview of diagnosis methods on Petri nets can be found in Basile (2014). The introduction of time (Tripakis (2002)) has been a first extension to the problem in order to include a better modeling for temporal phenomena. Some works introduce this extension on Petri net formalism called *Time Petri Nets* (TPN) (Wang et al. (2015)). Another extension proposed in Jéron et al. (2006) aims to model faults as more complex behaviours called *supervision patterns* that are specific assembling of unobservable events. Recent works introduce the problem of diagnosis of *supervision patterns* in TPN (Pencolé and Subias (2021), Pencolé et al. (2021)), and defined *supervision patterns* with time constraints called *timed patterns* (Coquand et al. (2021)).

In this work the diagnosis is performed on a new type of patterns called *time patterns*, that are timed patterns that may occur anytime during the system life. The aim of this work is to provide a diagnosis method for time pattern in time systems modeled as time Petri nets. To do so, the synthesis of a new object called *Observable Simple Temporal Network* that abstracts some observable behaviours of the system for which the pattern has occurred is performed. It is presented as a sequence of observable events linked by time constraints. The diagnosis is performed for a sequence of observations as the recognition of one of the *Observable Simple Temporal Network* of a set abstracting the different pattern occurrences. It returns whether the pattern has not or may have occurred, and certify the diagnosis in case of diagnosable systems.

The paper is articulated as follows. First Section 2 presents the prerequisites necessary to understand the diagnosis problem, its modeling and the problem itself. Section 3 develops the synthesis of the *Observable Simple Temporal Network* for given system and time pattern. Before concluding, Section 4 gives the diagnosis function based on the results of the previous section.

2. DIAGNOSIS OF TIME PATTERN IN TIME PETRI NETS

In this work the system and the pattern are both modeled as Labeled Time Petri Nets (LTPN) (see Section 2.1). While the system is partially observable (some of its transitions are labeled by observable events) the considered time pattern represents some unobservable behaviours. It is then necessary to characterize their occurrences as observable behaviours to perform a diagnosis on the system. The idea is to abstract the different occurrences of a time pattern inside a system as finite sets of constraints relative to the date of occurrence of some observable events in the system. Such an abstraction will be modeled by an *Observable Simple Temporal Network*, denoted *OSTN*.

2.1 Background on labeled time Petri nets

In this section some formal prerequisites on LTPN are recalled.

Definition 1. A Labeled Time Petri Net (LTPN) is a 6-uple $N = \langle P, T, A, \Sigma, \ell, I_s \rangle$ where:

- 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 modeling the arcs between the transitions and the places
- Σ is a finite alphabet of transition labels
- $\ell : T \to \Sigma$ is the transition labeling function
- $I_s : T \to I_{\mathbb{Q}_+}$ is a static interval function $I_s(t)$, for which the lower bound, also called the date of earlier firing is denoted $\downarrow (I_s(t)) \in \mathbb{Q}_+$, and its upper bound, also called the date of later firing, is denoted $\uparrow (I_s(t)) \in \mathbb{Q}_+ \cup \{+\infty\}$

The *preset* of a transition t is the set of input places $pre(t) = \{p \in P |\ (p,t) \in A\}$, and similarly the *postset* of t is the set of output places $post(t) = \{p \in P |\ (t,p) \in A\}$. For a *safe* LTPN, a state is a couple $S = \langle M, I \rangle$ where M is the marking of the net (M: $P \to \{0,1\}$) and I is the partial firing interval application (I: $T \to I_{\mathbb{Q}_+}$) that associates to any transition a time interval of $\mathbb{Q}_+$ in which t can be fired as soon as it is enabled. $S_0 = \langle M_0, I_0 \rangle$ is the initial state of the net where M_0 is the initial marking of the net and I_0 is defined as follows: for any transition t enabled by M_0, $I_0(t) = I_s(t)$, else $I_0(t) = \varnothing$. For a marking M, a transition t is firable at the date θ if and only if:

- t is enabled (*i.e.* $\forall p \in pre(t),\ M(p) > 0$)
- $\theta \in I(t)$ and for all t' enabled by M, $\theta \leq \uparrow (I(t'))$

The fire of a transition t at a date θ is denoted: $\langle M, I \rangle \xrightarrow{\theta t} \langle M', I' \rangle$ and defined such that

- M' is such that $\forall p \in pre(t) \setminus post(t), M'(p) = 0$, $\forall p \in post(t) \setminus pre(t),\ M'(p) = 1$ else $M'(p) = M(p)$
- for any transition $t' \in T$ ($t' \neq t$) enabled by M and still enabled by M', $I(t') = [a,b] \Rightarrow I'(t') = [max(0, a-\theta), b-\theta]$
- for every transition t' enabled by M' and not by M, $I'(t') = I_s(t')$

A state S is reachable in a marked LTPN if there exists a run $r = \theta_1 t_1 \dots \theta_n t_n, n \in \mathbb{N}^*$ *such that* $S_0 \xrightarrow{\theta_1 t_1} S_1 \xrightarrow{\theta_2 t_2} S_2 \dots \xrightarrow{\theta_n t_n} S$. The set of reachable states of a LTPN N is denoted $R(N, S_0)$.

A run $r = \theta_1 t_1 \dots \theta_n t_n$ of a LTPN is said to be admissible if there exist $S_1, \dots S_n$ reachable states of N such that $S_0 \xrightarrow{\theta_1 t_1} S_1 \xrightarrow{\theta_2 t_2} S_2 \dots \xrightarrow{\theta_n t_n} S_n$.

A *timed sequence* over an alphabet Σ is a sequence of pairs $(d, e) \in \mathbb{R}_+ \times \Sigma$ where d corresponds to the date of firing of symbol e. A run produces a unique timed sequence.

Definition 2. The language $\mathcal{L}(N)$ of a LTPN N is the set composed by every timed sequence ρ such that there exists $r = \theta_1 t_1 \dots \theta_n t_n$ an admissible run for N with $\rho = \theta_1 \ell(t_1) \dots \theta_n \ell(t_n)$.

Berthomieu and Menasche (1983) defines a State Class Graph (SCG) which is an abstraction of the LTPN as an automaton. Each state is a class of equivalence between the states of the LTPN that share their marking and their firing domain i.e the time constraints on the firable transitions from the marking. The initial firing domain is defined by $I_0(t)$ for each t enabled by M_0.

Definition 3. A State Class Graph (SCG) of a LTPN $N = \langle P, T, A, \Sigma, \ell, I_s \rangle$ is a triple $(C, \alpha_0, \to)$ *such that*:

- $\alpha_0 = (M_0, F_0)$ where M_0 is the initial marking of N and $F_0 \in (I_{\mathbb{Q}_+})^T$ is the initial firing domain of N
- $C \in \{0,1\}^P \times (I_{\mathbb{Q}_+})^T$ is the set of all classes corresponding to states reachable in N
- $\to \in C \times T \times C$ is the transition function defined as follows : $(M, F) \xrightarrow{t} (M', F')$ iff
  - t is firable from (M, F)
  - $M' = M - pre(t) + post(t)$
  - $F' = next(F, t)$

where $next : (I_{\mathbb{Q}_+})^T \times T \to (I_{\mathbb{Q}_+})^T$ is the procedure to build the firing domain F' associated with a reachable marking M' reached from M by the firing of t that is defined as follows:

(1) for each transition t' enabled in M, compute the firing of t by adding the two constraints $\theta \leq \theta'$ and $\theta' = \theta + \theta'_{upd}$ (θ'_{upd} is a substitution variable)
(2) eliminate variables relative to transitions enabled in M and not in M'
(3) add the constraints relative to the newly enabled transitions (in M')
(4) determine the canonical form of each constraint in F'

2.2 Modeling

In this work the system is modeled as a partially observable safe LTPN $\Theta = \langle P_\Theta, T_\Theta, A_\Theta, \Sigma_\Theta, \ell_\Theta, I_{s,\Theta} \rangle$. Each firing interval is closed with its bounds belonging to $\mathbb{Q}_+$. The labeling function ℓ_Θ is bijective *i.e.* every event is associated with a unique transition of T_Θ. The alphabet is partitioned into two sets: $\Sigma_{o\Theta} = \{o_1, \dots, o_n\}$ the set of observable events on Θ, and $\Sigma_{u\Theta} = \{uo_1, \dots, uo_p\}$ the set of unobservable events. Similary T_Θ is partitioned into $T_{o\Theta}$ the set of transitions labeled by an observable event, and $T_{u\Theta}$ the set of transitions labeled by an unobservable event. Some other assumptions are formulated about Θ:

- **A0** the SCG of the system is finite
- **A1** there is no cycle of unobservable event in the system
- **A2** the system has no zeno run (infinite sequences of transitions that can occur in a finite amount of time)
- **A3** every transition enabled by the initial marking of the system is observable

Condition **A0** ensures that for each transition $t \in T_\Theta$, there is a finite number of arcs in the SCG of Θ labeled by t. Condition **A1** ensures that an observable transition will always be fired in a finite amount of time after another one. Condition **A2** prevents an infinite number of events from occurring in a finite amount of time. Condition **A3** ensures that the start of the system clock is observable.

Time pattern represent faulty behaviours or particular behaviours one wishes to diagnose. Contrary to the patterns investigates in Coquand et al. (2021) in which the pattern always occur in a finite time window after the starting of the system (after a certain amount of time it will not occur), the time patterns considered here are unobservable behaviours that can occur on the considered system.

Definition 4. A time pattern over a system Θ is a safe acyclic LTPN $\Omega = \langle P_\Omega, T_\Omega, A_\Omega, \Sigma_\Omega, \ell_\Omega, I_{s\Omega} \rangle$ where:

(1) $\Sigma_\Omega \subseteq \Sigma_{u\Theta}$

(2) if $(t, t') \in T_\Omega^2$ with $pre(t) \cap pre(t') = \emptyset$, then t and t' cannot be firable simultaneously
(3) there is only one transition enabled in the initial marking $M_{0\Theta}$ and it is such that its firing interval has the form $[a_{init}, +\infty[$. The other transitions have a closed interval.
(4) $M_{0\Omega} \notin Q_\Omega$ with $M_{0\Omega}$ and Q_Ω respectively the initial marking and the set of final markings of Ω
(5) for every marking M reachable in Ω, there exists M' reachable from M such that $M' \in Q_\Omega$
(6) Ω is deterministic
(7) every run starting from a marking of Q_Ω necessarily leads to a marking of Q_Ω

In this work the study is restricted to pattern without parallelism (Condition 2). Condition 3 states that an occurrence of the pattern can start at any time during the system life, but always ends after a finite amount of time. Condition 4 ensures that the language of a pattern does not contain the empty sequence i.e the pattern does not represent empty event sequences. Condition 5 ensures that any run of the pattern is a prefix of a run for which the pattern has occurred. Condition 6 ensures that a pattern is written in a unique way. Finally Condition 7 ensures that the occurrence of the pattern in an execution of Θ is definitive.

Example 1. Figures 1a and 1b show an example of time pattern Ω_1 on a system Θ_1. The observable transitions are t_0, t_1, t_3 and t_5 (colored in blue here) and the unobservable ones are t_2 and t_4 (colored in red). The final marking of Ω_1 is $Pt_3 = 1$.

The occurrence of a pattern in a run of the system is considered as a pattern matching problem, following the definition of Pencolé and Subias (2021):

Definition 5. A timed sequence $\rho \in \mathcal{L}(\Theta)$ matches a pattern Ω (denoted $\rho \Subset \Omega$) if there exists a sub-word ρ' of ρ such that $\rho' \in \mathcal{L}(\Omega)$.

Without ambiguity, it is said that a run r matches a pattern Ω ($r \Subset \Omega$) if the timed sequence ρ produced by r matches Ω.

Example 2. The run $r = 0t_0.2t_4.2t_5.2t_1.1t_2.0t_3.1t_4.2t_5$ is a run of Θ_1 that matches Ω_1, as it produces the timed sequence $\rho = 0o_1.2f.2o_4.2o_2.1init.0o_3.1f.2o_2$ for which $\rho' = 7init.1f$ is a sub-word that belongs to $\mathcal{L}(\Omega_1)$.

The projection of a timed sequence onto the observable alphabet of the system (also called observable timed sequence) is defined as follows:

- $\mathbf{P}_{\Sigma_\Theta \to \Sigma_{o\Theta}}(\theta_1 e_1.\theta_2 e_2 \dots \theta_n e_n) = \theta_1 e_1.\mathbf{P}_{\Sigma_\Theta \to \Sigma_{o\Theta}}(\theta_2 e_2 \dots \theta_n e_n)$ if $e_1 \in \Sigma_{o\Theta}$
- $\mathbf{P}_{\Sigma_\Theta \to \Sigma_{o\Theta}}(\theta_1 e_1.\theta_2 e_2 \dots \theta_n e_n) = \mathbf{P}_{\Sigma_\Theta \to \Sigma_{o\Theta}}((\theta_1 + \theta_2) e_2 \dots \theta_n e_n)$ otherwise

Example 3. The associated observable timed sequence to ρ is $\rho_o = 0o_1.4o_4.2o_2.1o_3.2o_2$.

2.3 Diagnosis problem statement

Intuitively speaking, the diagnosis problem considered is to determine if a pattern has effectively occurred from the knowledge of a sequence of observable events produced by a system.

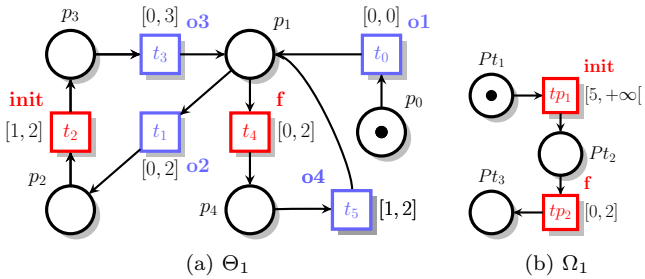

Fig. 1. A system Θ_1 and a pattern on this system Ω_1

The answer is given by a diagnosis function called diagnoser (denoted Ω-diagnoser), returning for an observable timed sequence *Ω-safe* if no run producing this time sequence matches Ω, *Ω-certain* if every run producing the observable timed sequence matches Ω and *Ω-ambiguous* otherwise.

In this paper, we aim at proposing an Ω-diagnoser for an extension of the pattern diagnosis problem introduced in Pencolé et al. (2021), that deals with time pattern as defined in Definition 4. As each run of the system that matches the pattern produces a given set of observable events satisfying specific time constraints, the knowledge of these observable events and time constraints can be exploited for a diagnosis purpose. We then propose to formally define the Ω-diagnoser based on the notion of *Observable Simple Temporal Network* that is defined below:

Definition 6. An *Observable Simple Temporal Network OSTN* is a couple (s, Π) where s is a nonempty sequence of observable events ($s \in \Sigma_{o\Theta}^+$) and Π is a set of constraints relative to the occurrence dates of the events of s.

The aim of this work is to diagnose the occurrences of a time pattern Ω in a system Θ thanks to the recognition of a set of *OSTN*s that capture observable behaviours associated with the system runs that match the pattern *i.e.* to every possible type of occurrence of Ω in Θ. These so called *OSTN*s are built following the procedure of Algorithm 1 presented in the next section.

3. OSTN SYNTHESIS

This section presents the synthesis of the *OSTNs* based on the analysis of the SCGs of the system and the pattern.

3.1 General procedure

Algorithm 1 exposes the calculation procedure of the *OSTNs* that characterize a time pattern Ω in a system Θ. This algorithm is based on both the SCGs of Θ and Ω. Firstly, Function `getObsPredecessors` returns for a given transition t_{init} (the one labeled by the *init* event here) every observable transition that may be the nearest observable predecessor of t_{init} in a run of Θ by exploring the SCG of Θ. In other words, it returns $\{t_{obs} \in T_{o\Theta}$ *s.t.* $\exists (m, n) \in \mathbb{N}^2, m < n, \exists (t_i)_{i \in [1,n]} \in T_\Theta^n, \exists (\alpha_j)_{j \in [0,n]} \in C^{n+1}, \alpha_0 \xrightarrow{t_1} \alpha_1 \dots \alpha_m \xrightarrow{t_m} \alpha_{m+1} \dots \alpha_{n-1} \xrightarrow{t_n} \alpha_n, t_n = t_{init}, t_m = t_{obs}$ (*i.e.* $\ell(t_m) \in \Sigma_{o\Theta}$) $\forall i \in [m+1, n], \ell(t_j) \in \Sigma_{u\Theta}\}$. Secondly, starting from t_{init} and exploring in parallel the SCG of Θ and the one of Ω, the procedure extracts

Algorithm 1 Synthesis of $OSTNs$ for a time pattern Ω and a system Θ

input : A system Θ, a time pattern Ω and their SCG ($SCG(\Theta)$ and $SCG(\Omega)$), the system transition t_{init} (labeled by event $init$)
output: A set of $OSTN$

$obsPredecessors \leftarrow$ `getObsPredecessors`$(t_{init}, SCG(\Theta))$
$seqMatch \leftarrow$ `getMatchingSeq`$(SCG(\Theta), SCG(\Omega), t_{init})$
for $\pi \in seqMatch$, $t_o \in obsPredecessors$ **do**
  $OSTNs \leftarrow OSTNs \cup$ (`obs`$(t_o.\pi)$, `getConstraints`(π, t_o, Ω))
end
return $OSTNs$

from the SCG of Θ the transition sequences from Θ for which there can be an occurrence of Ω i.e the sequences that match the pattern (function `getMatchingSeq`)). `getMatchingSeq` actually returns a set of sequences of pairs of interval/transition, where the interval corresponds to the constraints relative to this transition in the firing domain of the class in which the transition is fired in the SCG of Θ. This sequence of pairs called path is denoted π thereafter. Then, for each couple formed by an observable predecessor transition t_o and a path π, `getConstraints` computes the time constraints on the observable transitions of π relative to the occurrence of Ω and results in the synthesis of the $OSTN$ for the couple (t_o, π). In this step, the knowledge of the different t_0 (i.e the observable predecessors of t_{init}) allows to shift the time origin of the time constraints to make it observable (see Section 3.2). In the end Algorithm 1 returns a set containing the $OSTNs$ abstracting the different matching. It has been shown in Coquand et al. (2021) that this set is finite.

Example 4. Let us consider the system and the pattern of Figure 1. The t_{init} transition is t_2. The observable predecessor of t_2 is t_1. In the system SCG, starting from t_{init} the path matching Ω_1 is $\pi_1 = [0,3]t_3\ .[0,2]t_4\ .[1,2]t_5$. The $OSTN$ building based on these results will be developed thereafter. Note that this example is simple for the sake of readability: there is only one path π_1 matching Ω_1 (so the SCG is not given). However, in the general case, there are many possible paths in a system satisfying the modeling assumptions (see Section 2.2).

3.2 Constraints synthesis

This section describe the function `getConstraints` of Algorithm 1. The paths π previously obtained correspond to the system behaviours that may include occurrences of the pattern. For each π obtained, there exists a run r of π that matches Ω, meaning there exists a path π' in the SCG of Ω corresponding to the occurrence of Ω in π (in terms of transition sequence). From each path π of the system it is then possible to extract time constraints on the firing dates of the observable transitions of the path according the to the time constraints of π' . This step of constraint extraction is formally detailed in Coquand et al. (2021) and summarized in this section.

The constraint extraction for a couple (π, π') is performed in three main steps.

The first step is to synchronise the system and the pattern to simulate the matching. As the *atemporal* matching has been previously verified (see `getMatchingSeq` of Algorithm 1), this step allows to deal with the temporal part of the matching by expressing it in the form of time constraints. To start, the path π returned by `getMatchingSeq` is reduced to its observable transitions and to the transitions responsible for the occurrence of Ω. The unobservable transitions of the path π that do not participate to the pattern occurrence are removed. Their firing date are deported onto the next non-deleted transition (which can be observable, or unobservable but that participates to the pattern occurrence) by resolving an inequation system built from the firing domains of the classes associated to the path taking into account the firing order of the transitions in π (Wang et al. (2015)).

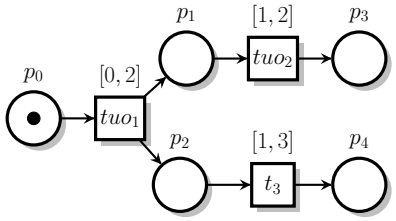

Fig. 2. Example illustrating the reduction step

Example 5. This example illustrates the reduction of a path for a system with one observable transition (t_3) and two unobservable transitions (t_{uo_1} and t_{uo_2}) without considering any pattern. Let us consider the net of Figure 2, and more particularly the path $\pi_{example} = [0,2]t_{uo_1}.[1,2]t_{uo_2}.[0,2]t_3$. The reduction of this path leads to a new path $\pi = [a,b]t_3$ where $(a,b) \in \mathbb{R}^2$. To determine a and b, the inequation system is to establish with y_i the variable associated to the absolute date of firing (relative to the start of the net) of t_i. Following the firing sequences resulting inequations will be: $\{0 \leq y_{uo_1} \leq 2, 1 \leq y_{uo_2} - y_{uo_1} \leq 2, 0 \leq y_3 - y_{uo_2} \leq 2\}$. As t_{uo_2} and t_3 are enabled simultaneously but t_3 fired after, a new inequation is added: $0 \leq y_3 - y_{uo_2}$ (here this last inequation is redundant but in the case of conflicts between transitions this inequation must be considered). The resolution of this inequation system leads to: $a = 1, b = 6$, so $\pi = [1,6]t_3$. Let us consider now the path $[0,2]t_{uo_1}.[1,3]t_3.[0,1]t_{uo_2}$, the reduction will be performed on t_{uo_1} and t_3 only.

Once the reduced path is obtained, it is then cut into blocks to be synchronized with the pattern transitions following π'. A block is part of a path (i.e a sequence of pairs of firing domains/transitions) composed eventually of observable transitions, and of a last transition that is an unobservable transition involved in the pattern occurrence.

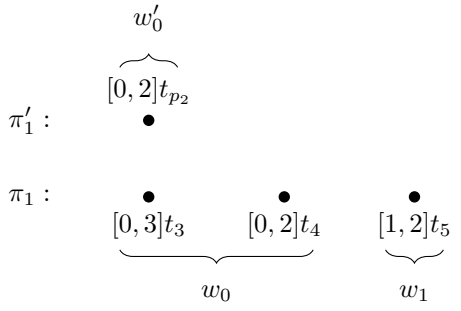

Fig. 3. Block cutting illustration ($\ell_\Theta(t_4) = \ell_\Omega(t_{p_2})$)

Example 6. Back to the example of Figure 1 with $\pi_1 = [0,3]t_3\ .[0,2]t_4\ .[1,2]t_5$ and $\pi'_1 = [0,2]t_{p_2}$, the block cutting is performed as shown in Figure 3. The transitions t_4 and t_{p_2} must be synchronized as they share the same label f. The system path π is then composed of two blocks: $w_0 = [0,3]t_3[0,2]t_4$ and $w_1 = [1,2]t_5$. The second block contains only an observable transition, as it does not need to be synchronized with anything. The pattern is composed of only one block $w'_0 = [0,2]t_{p_2}$.

Proposition 1. The synchronisation of two blocks $w = \mathcal{I}_1 t_1 \dots \mathcal{I}_k t_k$ and $w' = \mathcal{I}_\Omega t_\Omega$ with $\ell(t_k) = \ell(t_\Omega)$ defines the following set of constraints (called synchronisation constraints) to be satisfied:

$$\forall r = d_1 t_1 \dots d_k t_k \subseteq w, r \ni \mathcal{I}_\Omega t_\Omega \Leftrightarrow \begin{cases} d_1 \in \mathcal{I}_1 \\ \dots \\ d_k \in \mathcal{I}_k \\ \sum_{i=1}^{k} d_i \in \mathcal{I}_\Omega \end{cases} \quad (1)$$

During the rest of this work, a synchronisation constraint is considered as its canonical form, which is defined as the restriction for a constraint to the set of values for which there exists a solution for the whole set.

Example 7. Back to the example with $\pi_1 = [0,3]t_3\ .[0,2]t_4\ .[1,2]t_5$ and $\pi'_1 = [0,2]t_{p_2}$, there is no need for reduction in this simple example. The synchronisation of t_4 and t_{p_2}, leads to the following set of constraints: $\{d_3 \in [0,3], d_4 \in [0,2], d_3 + d_4 \in [0,2]\}$. As there is no solution to this set for values of d_3 between 2 and 3, the canonical form of the synchronisation constraints is: $\{d_3 \in [0,2], d_4 \in [0,2], d_3 + d_4 \in [0,2]\}$.

The second step of the constraint extraction, generates another type of constraints called *admissibility constraints*, that provide the necessary conditions for a transition sequence of the SCG to correspond to a part of an admissible run in the system. Indeed the SCG contains for each transition every possible firing date for the transition, but a set of firing dates for a sequence of transitions in the SCG may not be admissible for the system (due to parallelism). Such constraints are calculated from the following rule: the time elapsed between the enabling and the firing of a transition belongs to its firing interval in the system. This time elapsed may be the sum of the firing dates of a group of transitions if a transition is enabled at a time and not fired directly.

Example 8. The admissibility constraints for $\pi_{example} = [0,2]t_{uo_1}.[1,2]t_{uo_2}.[0,2]t_3$ of Figure 2 will be: $1 \leq d_{uo_2} + d_3 \leq 3$.

The third step is to transform the previous constraints into observable ones. For the admissibility constraints a variable change is applied reporting the involved dates of firing onto the next observable transition (if they are relative to an unobservable transition) relatively to the previous observable transition fired in the sequence. The same variable change is applied for the constraints issued from the synchronisation, for which the only unobservable transitions are the transitions synchronized with the transitions of the pattern.

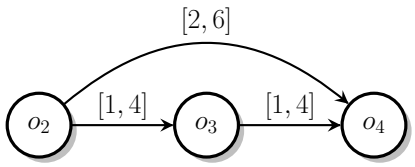

Fig. 4. Graphical representation of $OSTN_{\Omega_1}$

To complete this step it is necessary to shift the time origin of the constraints to make the constraints relative to an observable transition (as $\ell_\Omega(t_{init}) \in \Sigma_{u\Theta}$). A similar variable change to the one performed previously is applied on the first variable of the constraints set. This variable change models the time elapsed between the firing of the observable transition preceeding t_{init} and the first transition of each sequence.

Example 9. As t_3 is observable, the translation into observable constraints is performed with the variable change $d_5^o = d_4 + d_5$. That leads to the following constraints: $\{d_3 \in [0,2], d_5^o \in [1,4], d_3 + d_5^o \in [1,4]\}$. The origin shifting is performed with the variable change $d_3^o = d_2 + d_3$ as the time elapsed since the observable transition preceding the transition labeled by *init* is $d_2 + d_3$ (with $d_2 \in [1,2]$ see Figure 1a). The observable set of constraints for the matching of Ω_1 in Θ_1 is: $\{d_3^o \in [1,4], d_5^o \in [1,4], d_3^o + d_5^o \in [2,6]\}$.

For each tuple $(t_o, \pi, \Pi_{t_o,\pi})$, where $\Pi_{t_o,\pi}$ is the set of constraints associated to (t_o, π), the following $OSTN$ is computed : $OSTN = (obs, \Pi_{t_o,\pi})$ where *obs* is the concatenation of the label of t_o and of the observable labels of π.

Example 10. The $OSTN$ characterizing Ω_1 throughout Θ_1 is: $OSTN_{\Omega_1} = (o_2.o_3.o_4, \{d_3^o \in [1,4], d_5^o \in [1,4], d_3^o + d_5^o \in [2,6]\})$. Figure 4 shows a graphical representation of $OSTN_{\Omega_1}$. An arc between two events labeled with an interval represents the time elapsing since the firing of the starting node until the firing of the ending node. The time elapsing between the occurrence of o_2 and o_4 (which contains the time between o_3 and o_2 and the time between o_4 and o_3) is bounded between 2 and 6 time units.

In the following for a system Θ and a pattern Ω, $N_\Theta(\Omega)$ denotes the set of $OSTN$ abstraction of Ω in Θ. For the example $N_{\Theta_1}(\Omega_1) = \{OSTN_{\Omega_1}\}$.

4. RECOGNITION-BASED DIAGNOSIS

In this section, we consider that the system Θ produces the observed timed sequence σ. Let $(s, \Pi) \in N_\Theta(\Omega)$ be an OSTN. We say that the OSTN (s, Π) is recognized on σ iff:

- there exists $\rho = u.v.w \in \mathcal{L}(\Theta)$ such that the observable events in the timed subsequence v are the ones of s and their dates of occurrences satisfy the constraints Π; and
- let π_ρ be the path of SCG(Θ) associated with the run of Θ producing ρ, the path π based on which the OSTN (s, Π) has been synthetized is a prefix of the path π_ρ.

Let $\mathcal{O}(\sigma)$ be the set of $OSTN$'s that are recognized on σ.

Proposition 2. (1) If a $OSTN$ is in $\mathcal{O}(\sigma)$, then there exists a run r of Θ with σ as observable timed sequence such that $r \ni \Omega$.
(2) Similarly if a run r of the system produces the observable timed sequence $\sigma = \sigma'.\sigma''$ with σ'' not empty and a prefix of r matches Ω while producing σ' then $\mathcal{O}(\sigma) \neq \varnothing$.

Sketch proof: (1) If a $OSTN$ is recognized on σ there exists a path of SCG(Θ) associated with a run whose observable part is σ that has, as a prefix, the path π based on which the $OSTN$ has been built. Then the result is a direct consequence of the constraints synthesis and the bijectivity of ℓ_Θ. (2) The run r is associated with a path π_r of SCG(Θ), as it matches Ω, there exists a strict prefix π of π_r that belongs to $seqMatch$ in Algorithm 1 from which an OSTN exists and will be recognized on σ.

Based on the previous results, we now propose the definition of two diagnosers. The first one is generic and runs on any type of system Θ.

Definition 7. The OSTN-based Ω-diagnoser is the function $\Delta_\Omega : \mathbf{P}_{\Sigma_\Theta \to \Sigma_{o\Theta}}(\mathcal{L}(\Theta)) \to \{\Omega - safe, \Omega - ambiguous\}$ defined as follows.

- $\Delta_\Omega(\sigma) = \Omega - safe$ if $\mathcal{O}(\sigma) = \varnothing$.
- $\Delta_\Omega(\sigma) = \Omega - ambiguous$ otherwise.

This Ω-diagnoser keeps tracking the paths of the SCG that are consistent with σ and checks for the recognition of at least one OSTN. Proposition 2 then ensures that $\Delta_\Omega(\sigma) = \Omega - safe$ iff no pattern has occurred on the runs producing σ. The second version of the diagnoser is more accurate but requires Θ to be Ω-diagnosable (see Pencolé and Subias, 2021).

Definition 8. A system Θ is said to be Ω-diagnosable iff $\exists \tau \in \mathbb{R}_+$ *s.t.* $\forall (\rho_1, \rho_2) \in \mathcal{L}(\Theta)^2$, $\rho_1 = \rho_1'\rho_1''$, $time(\rho_1'') \geq \tau$, $\rho_1' \ni \Omega \wedge \mathbf{P}_{\Sigma \to \Sigma_{o\Theta}}(\rho_2) = \mathbf{P}_{\Sigma \to \Sigma_{o\Theta}}(\rho_1) \Rightarrow \rho_2 \ni \Omega$.

where *time* is the function returning for a timed sequence the time elapsed during this sequence.

Definition 9. Let Θ be a Ω-diagnosable system. The Ω diagnoser is the function $\Delta^d_\Omega : \mathbf{P}_{\Sigma_\Theta \to \Sigma_{o\Theta}}(\mathcal{L}(\Theta)) \to \{\Omega - safe, \Omega - ambiguous, \Omega - certain\}$ such that:

- $\Delta^d_\Omega(\sigma) = \Omega - safe$ if $\mathcal{O}(\sigma) = \varnothing$.
- $\Delta^d_\Omega(\sigma) = \Omega - certain$ if $\mathcal{O}(\sigma) = \{OSTN_{\preceq\pi}, \forall \pi \in \Pi_{SCG(\Theta)}(\sigma)\}$
- $\Delta^d_\Omega(\sigma) = \Omega - ambiguous$ otherwise.

$\Pi_{SCG(\Theta)}(\sigma)$ denotes the set of paths from $SCG(\Theta)$ ending with an observable transition that at least produces a run whose observable part is σ and $OSTN_{\preceq\pi}$ denotes an $OSTN$ that has been synthetized based on a SCG path that is a prefix of π.

Proposition 3. If $\Delta^d_\Omega(\sigma) = \Omega - certain$ then Ω has definitely occurred in the system.

Sketch proof: If $\Delta^d_\Omega(\sigma) = \Omega - certain$, we know that for any path of $\Pi_{SCG(\Theta)}(\sigma)$ there exists at least a possible run of Θ that generates σ and matches Ω (Proposition 2). Now, whatever the observable future of σ is, it will be associated with a path of the SCG whose prefix is in $\Pi_{SCG(\Theta)}(\sigma)$ so there will always be a possible run that matches Ω. But as the system is Ω-diagnosable, then the current real run necessarily matches Ω already.

5. CONCLUSION

This paper adresses the problem of diagnosis in time Petri nets of time patterns that may occur anytime. The diagnosis method is based on the recognition of Observable Simple Temporal Networks ($OSTNs$) that capture the occurrence of observable events constrained by time. These $OSTNs$ are an abstraction of pattern occurrences of in the system. The provided diagnosis helps either to conclude the pattern has not occurred, or to indicate the result is ambiguous. A diagnoser for diagnosable systems is presented using the diagnosability to conclude, that can bring a predictable character to the result. Future work will include diagnosability analysis for such types of time pattern.

REFERENCES

Basile, F. (2014). Overview of fault diagnosis methods based on petri net models. In *2014 European Control Conference (ECC)*, 2636–2642. doi: 10.1109/ECC.2014.6862631.

Berthomieu, B. and Menasche, M. (1983). An enumerative approach for analyzing time Petri nets. In *Proceedings IFIP*, 41–46. Elsevier Science Publishers.

Cabasino, M.P., Giua, A., and Seatzu, C. (2010). Fault detection for discrete event systems using petri nets with unobservable transitions. *Automatica*, 46(9), 1531–1539. doi:https://doi.org/10.1016/j.automatica.2010.06.013.

Coquand, C., Subias, A., and Pencolé, Y. (2021). Signature of timed patterns in time Petri nets: a formal characterization. In *Modélisation des Systèmes Réactifs (MSR'21)*. Paris, France.

Jéron, T., Marchand, H., Pinchinat, S., and Cordier, M.O. (2006). Supervision patterns in discrete event systems diagnosis. In *2006 8th International Workshop on Discrete Event Systems*, 262–268. doi: 10.1109/WODES.2006.1678440.

Pencolé, Y. and Subias, A. (2021). Diagnosability of event patterns in safe labeled time Petri nets: A model-checking approach. *IEEE Transactions on Automation Science and Engineering*, 1–12. doi: 10.1109/TASE.2020.3045565.

Pencolé, Y., Subias, A., and Coquand, C. (2021). A model checking method to solve the event pattern diagnosis problem in safe labeled time petri nets. In *32nd International Workshop on Principles of Diagnosis (DX'21), Hamburg (Germany), September 13-15, 2021.*

Sampath, M., Sengupta, R., Lafortune, S., Sinnamohideen, K., and Teneketzis, D. (1995). Diagnosability of discrete-event systems. *IEEE Transactions on Automatic Control*, 40(9), 1555–1575. doi:10.1109/9.412626.

Tripakis, S. (2002). Fault diagnosis for timed automata. In W. Damm and E.R. Olderog (eds.), *Formal Techniques in Real-Time and Fault-Tolerant Systems*, 205–221. Springer Berlin Heidelberg, Berlin, Heidelberg.

Wang, X., Mahulea, C., and Silva, M. (2015). Diagnosis of time petri nets using fault diagnosis graph. *IEEE Transactions on Automatic Control*, 60(9), 2321–2335. doi:10.1109/TAC.2015.2405293.