Supervision Patterns: Diagnosability Checking by Petri Net Unfolding

Houssam-Eddine Gougam [*] **Audine Subias** [*]
Yannick Pencolé [**]

[*] *CNRS; LAAS; 7 avenue du Colonel Roche, F-31400 Toulouse, France*
Univ de Toulouse, INSA, LAAS, F-31400 Toulouse, France
[**] *CNRS; LAAS; 7 avenue du Colonel Roche, F-31400 Toulouse, France*
Univ de Toulouse, LAAS, F-31400 Toulouse, France

AbstractThis paper addresses the problem of checking diagnosability of supervision patterns in discrete-event systems. With a supervision pattern, it is possible to represent a complex behaviour of the system, and especially a faulty behaviour. As opposed to classical diagnosability analysers that check by exploring the marking graph of the underlying net, the proposed method relies on Petri net unfoldings and thus avoids the combinatorial explosion induced by the use of marking graphs. The method is an adaptation of the twin-plant method to net unfolding: a pattern is diagnosable if the unfolding representing the twin-plant does not implicitly contain infinite sequences of events that are ambiguous.

Keywords: TODO Five to ten keywords, preferably chosen from the IFAC keyword list.

1. INTRODUCTION

Monitoring a system or a process in order to ensure sustainable and optimal performance is a critical task that requires the deployment of supervision tools such as sensors, monitors, diagnosers, etc. In the context of discrete-event systems (DES), the supervision task consists in analysing the sequence of observed events and determining whether an abnormal/faulty situation has occurred before deciding what kind of actions to perform in order to recover the optimal performance of the system. By the intrinsic nature of DES, an abnormal situation is characterised by a partial order of observable/non-observable events called an *event pattern*.

In this paper, we address the problem of checking the diagnosability of such patterns that is to provide a formal way to answer the following question: given a DES and a supervision pattern, is the supervisor *always* able to determine *with certainty* that the pattern has occurred or not in the system after observing a *finite* sequence of events? This problem is well-known and many formal analyses have already been proposed as in Jiang and Kumar (2004) or in Jéron et al. (2006) but still the intrinsic nature of the problem induces a combinatorial explosion which is not taken into account in these previous works and makes unrealistic their use in practice. In our proposal, we tackle the problem with two original choices. The first one is the use of Petri net to model patterns as well as the system (Basile et al. (2009),Dotoli et al. (2009)) so that we benefit of the natural way to represent event concurrency. The second one is the use of net unfoldings (McMillan (1995),Esparza et al. (2002),Benveniste et al. (2003)) in order to benefit of a representation as a partial order of events which avoids the explicit enumeration of event sequences that is performed by techniques based on marking graphs like in Cabasino et al. (2012).

Analysing the diagnosability of a system can be intuitively seen as looking for the existence of a diagnosis function that returns "Normal", "Faulty" or "Ambiguous" for each observable execution of the system. We do require some properties on this function. First, the returned value must be *correct* meaning that if it is equal to "Faulty" all possible explanation of the observable trace must be "Faulty", and the same apply for the "Normal" value, otherwise the answer must be "Ambiguous". The function must also be *bounded*, i.e. a faulty sequence yields a "Faulty" return value after a bounded delay.

This intuitive vision rises some questions: how is the system modeled? What is a faulty sequence, and how is it modeled? What is a delay? The analysis method is driven by the answers to these questions.

For instance, in Sampath et al. (1995) the system is modeled by a finite state machine, a faulty sequence is a sequence containing a fault event and the delay is the occurrence of a number of events. Other choices can be imagined, a faulty sequence for Jiang et al. (2003) is the multiple occurrences of a particular event, in Jéron et al. (2006) a fault is a sequence that is recognized by some pattern, Pencolé and Subias (2009) model the faults by chronicles, and the delay is an amount of time.

In this paper, we consider a system modeled by a formal language, a pattern (which also modeled by a formal language) that recognize faulty sequences, and a delay is the occurrence of a number of event.

The paper is organised as follows. Section 2 recalls the necessary background. Section 3 introduces a Petri net representation of patterns. Section 4 presents the problem and the proposed analysis method. Finally, Section 5 illustrate the method by an example.

2. BACKGROUND

This section briefly recalls the necessary theoretical background which this paper relies on.

Given an alphabet Σ, we define Σ^* as the set of finite sequences over Σ (including the empty sequence λ). Σ^+ is the set of non empty sequences over Σ. A subset $\mathcal{S}$ of Σ^* is called a *language* over Σ. For every sequence $s \in \mathcal{S}$ we define the set of the continuations of s in $\mathcal{S}$: $\mathcal{S}/s = \{t \in \mathcal{S} : st \in \mathcal{S}\}$. The projection of a sequence s on $\Sigma_p \subseteq \Sigma$ is noted $\mathcal{P}_{\Sigma_p}(s)$. Finally, the length of a sequence s is denoted $\|s\|$.

2.1 Labelled Petri nets

Definition 1. A *labelled Petri net* is a tuple $\langle P, T, A, \ell, L, \Sigma \rangle$ where:

- P: a set of places;
- T: a set of transitions with $P \cap T = \varnothing$;
- $A \subseteq (P \times T) \cup (T \times P)$: a set of arcs;
- $\ell : P \cup T \to L \cup \Sigma \cup \{\lambda\}$: a labelling function where L is the set of place labels, Σ is the set of transition labels and λ denotes the empty sequence;

Note that this definition is restricted to labelled Petri nets with arcs of weight 1.

A *marking* is a multiset M of places, i.e. a map from P to $\mathbb{N}$ which maps any place to the number of tokens contained in it.

A marked and labelled Petri net is a tuple $\Theta = \langle P, T, A, \ell, L, \Sigma, M_0 \rangle$ where $\langle P, T, A, \ell, L, \Sigma \rangle$ is a labelled Petri net and M_0 an initial marking.

The current state of a Petri net is defined by its marking.

Let $\bullet t = \{p \in P : (p,t) \in A\}$ be the preset of t and $t\bullet = \{p \in P : (t,p) \in A\}$ its postset (we define similarly the preset and postset of a place). The transition t is *firable* from a given marking M iff: $\forall p \in \bullet t : M(p) > 0$. Firing t leads to a new marking M' such that $M' = (M \setminus \bullet t) \cup t\bullet$ and which is denoted by $M \xrightarrow{t} M'$. A marking M is reachable if there exists a firing sequence $s = t_0 t_1 \dots t_n$ such that $M_0 \xrightarrow{t_0} M_1 \xrightarrow{t_1} \dots \xrightarrow{t_n} M$, we can also write $M_0 \xrightarrow{s} M$.

Let $<$ be the transitive closure of A — called the causal relation, and $\leq$ the reflexive closure of $<$. The couple of elements $(x, y) \in (P \cup T)^2$ is a conflict (denoted $x \# y$) if there exists two transitions $t_1, t_2 \in T$, such that $t_1 \neq t_2$, $\bullet t_1 \cup \bullet t_2 \neq \varnothing$ and $(t_1, x), (t_2, y) \in <$.

Given a marked and labelled Petri Net Θ and a set of markings Q, the G-type language (see Peterson (1977)) generated by Θ is given by: $\mathcal{L}(\Theta) = \{\ell(s) : s \in \Sigma^* \wedge M_0 \xrightarrow{s} M \wedge \exists M' \in Q, M' \subseteq M\}$.

To analyse Petri net reachability, one can compute the *marking graph* of a marked net however it usually leads to a combinatorial explosion. Another technique that attempts to avoid this problem is the use of *Petri net unfolding* that exploits partial orders (McMillan (1995)).

2.2 Petri net unfolding

A Petri net unfolding is another labelled Petri net that is generally infinite.

Definition 2. The unfolding $\Phi = \langle P_\Phi, T_\Phi, A_\Phi, \ell_\Phi, P, T, m_{\Phi_0} \rangle$ of a marked and labelled Petri net $\mathcal{N} = \langle P, T, A, \ell, L, \Sigma, m_0 \rangle$, is a labelled Petri net such that:

(1) $\forall p \in P_\Phi : |\bullet p| \leq 1$.
(2) Φ is acyclic, i.e. for any element $x \in P_\Phi \cup T_\Phi : \neg(x < x)$
(3) Φ is finitely preceded, i.e., for every $x \in P_\Phi \cup T_\Phi$, the set of elements $y \in P_\Phi \cup T_\Phi$ such that $y < x$ is finite.
(4) No $x \in P_\Phi \cup T_\Phi$ is in self-conflict $\neg(x \# x)$.
(5) $\ell_\Phi(P_\Phi) \subseteq P$, $\ell_\Phi(T_\Phi) \subseteq T$;
(6) $\forall t \in T_\Phi : \bullet\ell_\Phi(t)$ is isomorphic to $\bullet t$;
(7) $\forall t \in T_\Phi : \ell_\Phi(t)\bullet$ is isomorphic to $t\bullet$.

We denote by $\min(\Phi) = \{x \in P_\Phi \cup T_\Phi : \nexists y \in P_\Phi \cup T_\Phi, y < x\}$. A *configuration* C of an unfolding is a set of causally-closed and conflict-free transitions of T_Φ, formally: $t \in C \Rightarrow \forall t' < t, t' \in C$ and $\forall t, t' \in C : \neg(t \# t')$. The cut of a configuration C is the set of places $\mathrm{Cut}(C) = (\min(\Phi) \cup C\bullet) \setminus \bullet C$ where $C\bullet = \{p \in t\bullet, t \in C\}$ and $\bullet C = \{p \in \bullet t, t \in C\}$ and $\ell_\Phi(\mathrm{Cut}(C))$ characterises a marking of the net $\mathcal{N}$. The *local configuration* of a transition $t \in T_\Phi$ is the configuration $[t]$ such that $\forall t' \in [t], t' \neq t \Rightarrow t' < t$.

McMillan (1995) shows the existence of a fragment of the unfolding Φ also called the *finite complete prefix* such that any reachable marking of $\mathcal{N}$ is represented by the cut of a configuration of that fragment. The construction of such a prefix relies on the search for *cutoff points* (McMillan (1995), Esparza et al. (2002)). A cutoff point is a transition $t_c \in T_\Phi$ such that its local configuration $[t_c]$ contains a transition t' such that $\ell_\Phi(\mathrm{Cut}([t'])) = \ell_\Phi(\mathrm{Cut}([t_c]))$ and the relation $[t'] \prec [t]$ holds, where $\prec$ is an *adequate order* (see Esparza et al. (2002)). The finite complete prefix of Φ is then defined as the subset of elements $x \in P_\Phi \cup T_\Phi$ such that $x < t_c$ where t_c is a cutoff point of Φ.

The method described hereafter always uses the finite complete prefix of the unfolding as a support, so from now on, we use the notion of unfolding as a shortcut for designing its finite complete prefix.

2.3 Synchronised product of labelled Petri nets

Let $\Theta_1 = \langle P_1, T_1, A_1, \ell_1, L_1, \Sigma_1, m_{10} \rangle$ and $\Theta_2 = \langle P_2, T_2, A_2, \ell_2, L_2, \Sigma_2, m_{20} \rangle$ be two labelled Petri nets. The synchronized product $\Theta = \Theta_1 \|_{\Sigma_s} \Theta_2$ over a set of transition labels Σ_s is the labelled Petri net $\Theta = \langle P, T, A, \ell, L, \Sigma, m_0 \rangle$ such that:

- $P = P_1 \cup P_2$;
- $T = \hat{T}_1 \cup \hat{T}_2 \cup T_s$,
  where: $\hat{T}_i = \{t \in T_i : \ell_i(t) \notin \Sigma_s\}, i = 1, 2;$ and $T_s = \bigcup_{l \in \Sigma_s} \{t_1 \| t_2 : \exists (t_1, t_2) \in (T_1 \times T_2) : \ell_1(t_1) = \ell_2(t_2) = l\}$;
- $A = \hat{A}_1 \cup \hat{A}_2 \cup A_s$,
  $\hat{A}_i = A_i \setminus [(P_i \times (T_i \setminus \hat{T}_i)) \cup ((T_i \setminus \hat{T}_i) \times P_i)], i = 1, 2;$ and
  $$\begin{aligned} A_s = &\{(p,t) \in (P \times T) : p \in P_1, t = t_1 \| t_2, (p, t_1) \in A_1\} \\ &\cup \{(p,t) \in (P \times T) : p \in P_2, t = t_1 \| t_2, (p, t_2) \in A_2\} \\ &\cup \{(t,p) \in (T \times P) : p \in P_1, t = t_1 \| t_2, (t_1, p) \in A_1\} \\ &\cup \{(t,p) \in (T \times P) : p \in P_2, t = t_1 \| t_2, (t_2, p) \in A_2\} \end{aligned}$$

- $\ell(p) = \begin{cases} \ell_1(p) & \text{if } p \in P_1 \\ \ell_2(p) & \text{if } p \in P_2 \end{cases}$;
- $\ell(t) = \begin{cases} \ell_1(t_1) & \text{if } t = t_1 \parallel t_2 \\ \ell_1(t) & \text{if } t \in T_1 \\ \ell_2(t) & \text{if } t \in T_2 \end{cases}$;
- $L = L_1 \cup L_2$;
- $\Sigma = \Sigma_1 \cup \Sigma_2$;
- $m_0(p) = \begin{cases} m_{10}(p) & \text{if } p \in P_1 \\ m_{20}(p) & \text{if } p \in P_2 \end{cases}$.

This product can be constructed simply by taking the union of the two nets, removing transitions with labels in Σ_s, and for each $(t_1, t_2) \in T_1 \times T_2$ where $\ell(t_1) = \ell(t_2) \neq \lambda$, add a transition $t_1 \parallel t_2$ to the product inheriting their label and arcs.

Let Q_i be the set of final markings of $\Theta_i, i = 1, 2$. We define the set of final markings of the product $\Theta = \Theta_1 \parallel_{\Sigma_s} \Theta_2$ as $Q = \{q = q_1 \cup q_2 : (q_1, q_2) \in Q_1 \times Q_2\}$.

If $\Sigma_s = \Sigma_1 \cup \Sigma_2$, the language generated by the product, given Q, is defined as follow: $\mathcal{L}(\Theta) = \{s \in (\Sigma_1 \cup \Sigma_2)^* : \mathcal{P}_{\Sigma_i}(s) \in \mathcal{L}(\Theta_i), i = 1, 2\}$. $\mathcal{L}(\Theta)$ also verifies: $s \in \mathcal{L}(\Theta)$ if $\exists s_1 \in \mathcal{L}(\Theta_1), \exists s_2 \in \mathcal{L}(\Theta_2) : \mathcal{P}_{\Sigma_s}(s) = \mathcal{P}_{\Sigma_s}(s_1) = \mathcal{P}_{\Sigma_s}(s_2)$.

3. SUPERVISION PATTERNS

Detecting and diagnosing faults in an SED generally rely on the existence of a fault model where a fault is represented by the occurrence of a particular event (Sampath et al. (1995)). Supervision patterns extend the expressiveness of fault models by introducing more complex faulty behaviours as proposed in Jiang and Kumar (2004) and in Jéron et al. (2006), involving several partially ordered events. We propose to represent a supervision pattern as a labelled Petri net.

Definition 3. A *supervision pattern* is a marked labelled Petri net $\langle P, T, A, \ell, L, \Sigma, m_0 \rangle$ such that:

(1) (labelling) $L = \{N, F\}$;
(2) (initialization) $\forall p \in P : m_0(p) > 0 \Rightarrow \ell(p) = N$;
(3) (exclusivity) $\forall p_1, p_2 \in P : \ell(p_1) = N \wedge \ell(p_2) = F \Rightarrow m(p_1) \times m(p_2) = 0$;
(4) (completeness) $\forall m : (\exists s = t_1 t_2 \ldots : m_0 \xrightarrow{s} m), \forall e \in \Sigma, \exists t \in T : (\ell(t) = e \wedge \forall p \in P, m(p) > 0)$;
(5) (recognition) $\forall m : (\exists s = t_1 t_2 \ldots t_n : m_0 \xrightarrow{s} m) : (m' \xrightarrow{t_n} m \wedge \exists p \in P : m'(p) > 0 \wedge \ell(p) = F \Rightarrow \exists p \in P : m(p) = F)$.

Condition 1 states that a place is either normal (labelled with N), which means that the supervision pattern has not occurred, or faulty (labelled with F). Condition 2 states that the initial marking only involves normal places. Condition 3 is the exclusivity condition: a marking reachable from the initial marking never involves a normal place on one hand and a faulty place on the other hand. Condition 4 is the completeness condition: from any reachable marking, any event of Σ can occur. Condition 5 is the recognition condition: once the pattern reaches a faulty marking any successor marking is faulty.

A set of final markings $Q = \{\{p \in P : \ell(p) = F\}\}$ is associated to the supervision pattern.

3.1 Examples

We illustrate the notion of supervision patterns with two examples. The patterns described below result from the translation of the patterns proposed by Jéron et al. (2006). Note that any pattern of Jéron et al. (2006) can be translated into a pattern defined by Definition 3.

Event concurrency Let $E = \{e_1, e_2, \ldots\}$ be a set of events. The pattern describing the concurrent occurrence of these events is the labelled Petri net $\langle P, T, A, \ell, L, E, m_0 \rangle$ defined as follows:

- $P = \{p_0\} \cup \bigcup_{i=1}^{2} \bigcup_{j=1}^{|E|} \{p_{ij}\}$;
- $T = \{t_0\} \cup \bigcup_{i=1}^{3} \bigcup_{j=1}^{|E|} \{t_{ij}\}$;
- $A = \bigcup_{i=1}^{|E|} \{(p_{1i}, t_{1i}), (p_{2i}, t_{2i}), (p_{2i}, t_0), (p_0, t_{3i})\} \cup \bigcup_{i=1}^{|E|} \{(t_{1i}, p_{2i}), (t_{2i}, p_{2i}), (t_{3i}, p_0)\} \cup \{(t_0, p_0)\}$;
- $\ell(p) = \begin{cases} F & \text{if } p = p_0 \\ N & \text{otherwise} \end{cases}$;
- $\ell(t) = \begin{cases} \lambda & \text{if } t = t_0 \\ e_j & \text{if } {}^{t = t_{ij} \text{ for any } i \in \{1,2,3\} \text{ and }}_{j \in \{1, \ldots, |E|\}} \end{cases}$;
- $m_0(p) = \begin{cases} 1 & \text{if } p \in \bigcup_{i=1}^{|E|} \{p_{1i}\} \\ 0 & \text{otherwise} \end{cases}$.

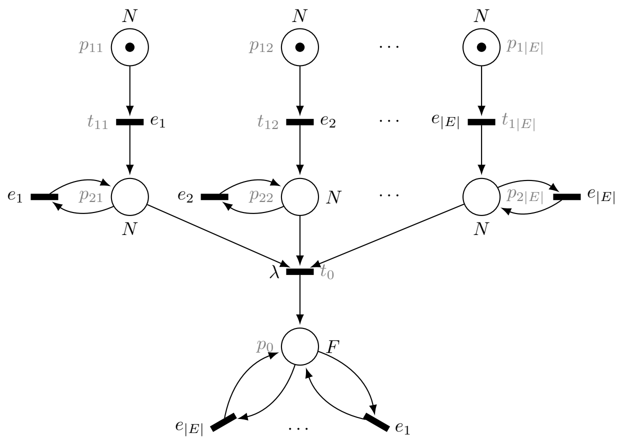

Figure 1. A set of concurrent events.

Figure 1 depicts the graphical representation of the pattern. Transition t_0, as it is labelled with λ, is firable as soon as every event of E has occurred. This example perfectly illustrates the representation compactness of Petri net with regards to automaton when modelling concurrency. In Jéron et al. (2006), the corresponding pattern is represented with an automaton with $2^{|E|}$ states whereas the number of required places to represent the same pattern as a net is $2|E| + 1$.

Multiple occurrences of the same event The pattern representing k occurrences ($k > 0$) of the same event e is characterised by the net $\langle P, T, A, \ell, L, E, m_0 \rangle$ defined as follows:

- $P = \bigcup_{i=0}^{k} \{p_i\}$;
- $T = \bigcup_{i=0}^{k} \{t_i\}$;
- $A = \bigcup_{i=0}^{k} \{(p_i, t_i)\} \cup \bigcup_{i=0}^{k-1} \{(t_i, p_{i+1})\} \cup \{(t_k, p_k)\}$;
- $\ell(p) = \begin{cases} F & \text{if } p = p_k \\ N & \text{otherwise} \end{cases}$;
- $\ell(t) = e, \forall t \in T$;
- $\Sigma = \{e\}$;
- $m_0(p) = \begin{cases} 1 & \text{if } p = p_0 \\ 0 & \text{otherwise} \end{cases}$.

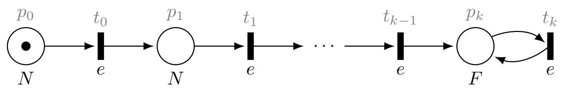

Figure 2. k occurrences of the same event e.

Figure 2 is the graphical representation of the pattern. This pattern graphically looks like the automaton representing the equivalent pattern in Jéron et al. (2006). This similarity is due to the fact that this pattern is purely sequential (no concurrency). However the supervision pattern defined by Definition 3 only represents the evolution of the events in the alphabet $\Sigma = \{e\}$ of the pattern and not the events of the underlying system as in Jéron et al. (2006).

Finally, this pattern is a generalization of the simple fault pattern ($k = 1$) which can be used to represent the classical diagnosability problem as in Sampath et al. (1995) and more recently in Cabasino et al. (2012).

4. DIAGNOSABILITY OF SUPERVISION PATTERNS

The problem of supervision pattern diagnosability in discrete event systems is defined in the following context.

The system is modeled by a language $\mathcal{S}$ over an alphabet Σ, where Σ — the set of events generated by the system — is partitioned into two subsets: Σ_o and Σ_u, respectively representing the set of *observable* and *unobservable* events. The supervision pattern $\mathcal{R}$ is a language over $\Sigma_{\mathcal{R}} \subseteq \Sigma_u$ that recognize $\mathcal{R}$-faulty sequences.

Definition 4. A sequence $s \in \Sigma^*$ is said to be $\mathcal{R}$-faulty if: $\mathcal{P}_{\Sigma_{\mathcal{R}}}(s) \in \mathcal{R}$.

Definition 5. A language $\mathcal{S}$ is $\mathcal{R}$-diagnosable if:

$$\exists n \in \mathbb{N}, \forall s \text{ a faulty sequence of } \mathcal{S}, \forall t \in \mathcal{S}/s :$$
$$\|\mathcal{P}_{\Sigma_o}(t)\| \geq n \implies \mathcal{P}^{-1}_{\Sigma_o}(\mathcal{P}_{\Sigma_o}(st)) \subseteq \Omega.$$

To tackle the problem of $\mathcal{R}$-diagnosability and given that every formal language can be modeled by a G-type labeled Petri net language, we propose to use labeled Petri nets to model the system and the pattern.

Definition 6. A labeled Petri Net Θ is Ω-diagnosable if $\mathcal{L}(\Theta)$ is $\mathcal{L}(\Omega)$-diagnosable.

The problem can be redefined as being the analysis of the diagnosability of a labeled Petri net $\Theta = \langle P, T, A, \ell, L, \Sigma, m_0 \rangle$ vis-a-vis a supervision pattern modeled by a labeled Petri net $\Omega = \langle P_\Omega, T_\Omega, A_\Omega, \ell_\Omega, L_\Omega, \Sigma_\Omega, m_{\Omega 0} \rangle$.

The classical method to analyse the diagnosability of faults in a discrete event system — introduced in Yoo and Lafortune (2002) and Jiang et al. (2000), is the use of a *twin-plant* on the observable events to detect normal and faulty executions that share the same observable behavior — if any — and thus conclude about the non-diagnosability of the system by looking for such sequences that have infinite number of observable events.

In the supervision pattern context, the same principle applies. But, first, the supervision pattern behavior must be taken into account by realising the synchronized product of the pattern and the system.

The analysis method is detailed in the following.

4.1 Analysis stages

Figure 3 depicts the necessary stages to perform the diagnosability analysis. The proposed method is an adaptation of the one in Jéron et al. (2006) to the Petri nets. The main difference is firstly the use of a *synchronized* product instead of a *synchronous* product and secondly the use of a net unfolding technique.

First, let us emphasize on the fact that patterns of Definition 3 are more concise than the ones of Jéron et al. (2006) as the language generated by the proposed patterns is usually not universal. The consequence is that the applied product is not synchronous but synchronized.

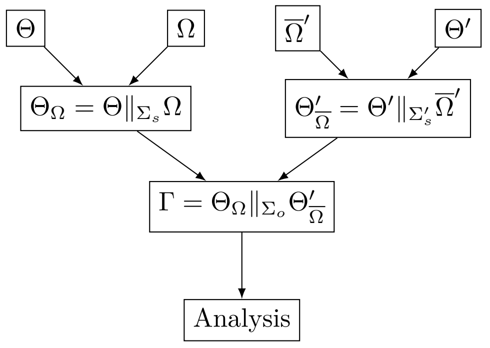
Θ
Ω
$\overline{\Omega}'$
Θ'
$\Theta_\Omega = \Theta \|_{\Sigma_s} \Omega$
$\Theta'_{\overline{\Omega}} = \Theta' \|_{\Sigma'_s} \overline{\Omega}'$
$\Gamma = \Theta_\Omega \|_{\Sigma_o} \Theta'_{\overline{\Omega}}$
Analysis

Figure 3. Stages of the diagnosability analysis.

We associate a set of final markings $Q = \{\varnothing\}$ to Θ and $Q_\Omega = \{\{p \in P_\Omega : \ell(p) = F\}\}$ to Ω. $\overline{\Omega}$ is defined as Ω associated with the set of final markings $\overline{Q}_\Omega = \bigcup_{p \in \ell^{-1}(N)} \{\{p\}\}$.

Proposition 7. $s \in \mathcal{S}$ is $\mathcal{R}$-faulty and $\mathcal{L}(\Omega) = \mathcal{R} \iff s \in \mathcal{L}(\Theta_\Omega)$

Proposition 8. $s \in \mathcal{S}$ is not $\mathcal{R}$-faulty and $\mathcal{L}(\Omega) = \mathcal{R} \iff s \in \mathcal{L}(\Theta'_{\overline{\Omega}})$

Proof. from the language of Petri nets product, the definition of $\mathcal{R}$-faulty and the fact that $\Sigma_{\mathcal{R}} = \Sigma_\Omega \subset \Sigma$.

Finally, we define an "'" operator applicable to sequences, alphabets, and labeled Petri nets, which transforms appropriately its operand by priming unobservable events and places and transitions — if any — and keeping the rest as is.

As a first stage, the net Θ_Ω is computed as the synchronized product of Θ and Ω over the set of synchronized labels

$\Sigma_s = \Sigma \cap \Sigma_\Omega$. $\Theta'_{\overline{\Omega}}$ is calculated in the same manner. In practice, this product is done once, the results being structurally identical, only the associated final markings need to be calculated twice.

The synchronized product $\Theta_\Omega \parallel_{\Sigma_o} \Theta'_{\overline{\Omega}}$ is performed (the result of this product is commonly called the *twin-plant*) and is denoted Γ.

Finally, the analysis consists in finding in Γ the possible sequences $c : m_0 \xrightarrow{c} m$, from m_0 the initial marking of Γ that lead to an ambiguous marking that can indefinitely occur.

Definition 9. A marking m in Γ is ambiguous if: $\exists p, p' \in P_\Gamma, m(p) \times m(p') \neq 0 \wedge \ell(p) = N \wedge \ell(p') = F'$.

Proposition 10. A marking m of Γ is ambiguous $\iff \exists s \in \mathcal{L}(\Gamma) : m_0 \xrightarrow{s} m$.

Proof. By construction of Γ.

Γ being finite, a sequences $c : m_0 \xrightarrow{c} m$ that occur indefinitely is necessary a cycle.

Definition 11. c is a cycle in $\mathcal{L}(\Gamma)$ if $\exists s \in \Sigma_\Gamma^*, \exists t \in \Sigma_\Gamma^+ : \forall n \in \mathbb{N}, st^n \in \mathcal{L}(\Gamma) \wedge c = st$

Theorem 12. Θ is not Ω-diagnosable $\iff$ Γ contains ambiguous cycles.

Proof.

($\Longrightarrow$) : Θ is not Ω-diagnosable if there are two sequences one faulty ($w_1 \in \mathcal{L}(\Theta_\Omega)$) the other non-faulty ($w'_2 \in \mathcal{L}(\Theta'_{\overline{\Omega}})$), which are arbitrary long and have the same observable behavior, which means there exists a sequence w in $\mathcal{L}(\Gamma)$ with the same observable execution as w_1 and w_2.

By hypothesis, the system can not produce cycles of unobservable events, so the projection of w_1 and w_2 on the observables must be arbitrary long, and so is the projection of w.

$\mathcal{P}_{\Sigma_o}(w)$ is arbitrary long, implies that w is arbitrary long too, which implies that Γ has at least one cycle w.

($\Longleftarrow$) : Γ contains ambiguous cycles means that $\mathcal{L}(\Gamma)$ contains cycles. Let w be a cycle in $\mathcal{L}(\Gamma)$, by definition: $\exists s \in \Sigma_\Gamma^*, \exists t \in \Sigma_\Gamma^+ : \forall n \in \mathbb{N}, st^n \in \mathcal{L}(\Gamma)$.

Γ being the product of Θ_Ω and $\Theta'_{\overline{\Omega}}$, ($\forall n \in \mathbb{N}, st^n \in \mathcal{L}(\Gamma)$) implies ($\forall n \in \mathbb{N}, \exists s_1 t_1^n \in \mathcal{L}(\Theta_\Omega), \exists s'_2 t'^n_2 \in \mathcal{L}(\Theta'_{\overline{\Omega}})$: $\mathcal{P}_{\Sigma_o}(s_1 t_1^n) = \mathcal{P}_{\Sigma_o}(s'_2 t'^n_2)$).

There is two sequences, one faulty ($s_1 t_1^n \in \mathcal{L}(\Theta_\Omega)$) the other non-faulty ($s'_2 t'_2 \in \mathcal{L}(\Theta'_{\overline{\Omega}})$), which are arbitrary long and have the same observable behavior, hence Θ is not Ω-diagnosable.

4.2 Algorithm

Algorithm 1 looks for an ambiguous sequence that allows to decide whether a given pattern is diagnosable. As explained in the above section, it is applied to the labelled Petri net Γ.

line 2: the function *unfolding* unfolds a labelled Petri net Θ and returns the unfolding Φ and a data structure H with the cutoff points t and their associated cut $\mathrm{Cut}(C(t))$ (see Section 2.2). A cutoff point is a transition associated to a transition in a cycle in the Petri net Θ.

Algorithm 1 Search for ambiguous sequences

```
1: diagnosability: function ANALYSIS($\Theta$)
2:     $\langle \Phi, H \rangle \leftarrow$ unfolding($\Theta$)
3:     for all $e$ in $H.events$ do
4:         if $\{N', F\} \not\subseteq \ell(\ell_\Phi(H[e])) \wedge \{N, F'\} \not\subseteq \ell(\ell_\Phi(H[e]))$ then
5:             prune($H[e]$)
6:         end if
7:     end for
8:     if $H$ is empty then
9:         return "diagnosable"
10:    else
11:        return "non-diagnosable"
12:    end if
13: end function
```

lines 3-7: every element of H that does not lead to an ambiguous marking is pruned.

lines 8-9: if H is empty then there is no ambiguous marking which can indefinitely occurs, the pattern is diagnosable.

lines 10-12: if H is not empty, i.e. there exists at least, in the unfolding, a sequence containing a cutoff point that leads to an ambiguous marking. It follows that this ambiguous marking can occur indefinitely so the system is not diagnosable.

Proposition 13. The algorithm 1 is correct.

Proof. The correctness of the algorithm is based on the correctness of the following relation:

Θ contains accessible cycles $\iff$ Φ contains cut-off events.

($\Longrightarrow$) : Θ contains cycles — infinite sequences — implies that the total unfolding is infinte.

By hypothesis, Θ is n-safe, so the prefix Φ is necessary finite.

The total unfolding is finite and the prefix is finite means necessary that the prefix Φ contains cutoff events.

($\Longleftarrow$) : Let e be a cut-off event of Φ, so: $\exists e' \in T_\Phi$:

$$\begin{cases} \ell_\Phi(\mathrm{Cut}([e])) = \ell_\Phi(\mathrm{Cut}([e']) = M_c \\ [e'] \subset [e] \end{cases}$$

$$\ell_\Phi(\mathrm{Cut}([e']) = M_c \text{ implies } \exists s \in T^*, M_0 \xrightarrow{s} M_c$$

$$\left.\begin{matrix} \ell_\Phi(\mathrm{Cut}([e])) = M_c \\ [e'] \subset [e] \end{matrix}\right\} \text{ implies } \exists t \in T^+, M_0 \xrightarrow{s} M_c$$

so, $\exists s \in T^*, \exists t \in T^+ : M_0 \xrightarrow{s} M_c \xrightarrow{t} M_c$, thereby, there is accessible cycles in Θ

The following section illustrates the method with the help of an example.

5. EXAMPLE

In order to illustrate every stage of the analysis, we present here a very simple example: the underlying system is modeled by Θ of Figure 4 and the pattern Ω is the one of Figure 5 representing the occurrence of a single event. Every event of Θ is observable except u and f. Therefore, $\Sigma = \{a, b, u, f\}$, $\Sigma_o = \{a, b\}$ and $\Sigma_u = \{u, f\}$.

First stage is the computation of the synchronized product $\Theta \parallel_{\Sigma_s} \Omega$ where $\Sigma_s = \Sigma \cap \Sigma_\Omega = \{f\}$ which leads to the net Θ_Ω represented on Figure 6.

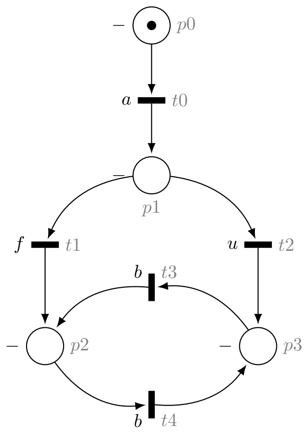

Figure 4. System Θ: f is a fault event, only a and b are observable.

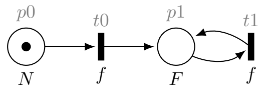

Figure 5. Pattern Ω: occurrence of (at least) a fault event f

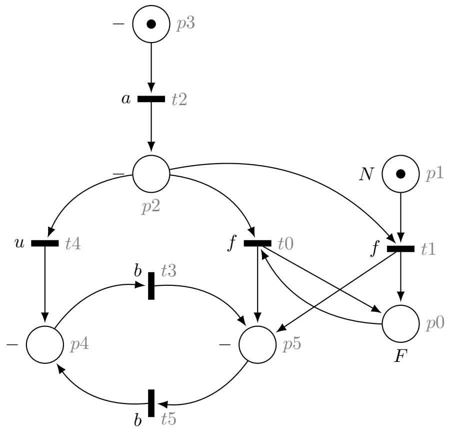

Figure 6. Synchronized product $\Theta_\Omega = \Theta \parallel_{\Sigma_s} \Omega$

Second stage is the computation of the twin-plant, that is the synchronized product of Θ_Ω with its primed version Θ'_Ω over the set of observables $\Sigma_o = \{a, b\}$. The result Γ is depicted on Figure 7.

Last stage is the unfolding of Γ as represented in Figure 8 returned by the function *unfolding* of Algorithm 1. Note that elements in a frame do not belong to the unfolding but aim at better representing the cutoff points.

The function *unfold* also returns the table H (see Table 1) which associates to every cutoff point t of Γ the set of labels associated to the places of $\mathrm{Cut}(C(t))$ that would be marked if t were fired.

Table 1. Cutoff points of H and the labels of their resulting marking in Γ.

cutoff point	labels of the resulting marking
t_0	$\{p_4, p_{10}, p_3, p_9\}$
t_1	$\{p_4, p_{11}, p_3, p_8\}$
t_2	$\{p_5, p_{10}, p_2, p_9\}$
t_3	$\{p_5, p_{11}, p_2, p_8\}$

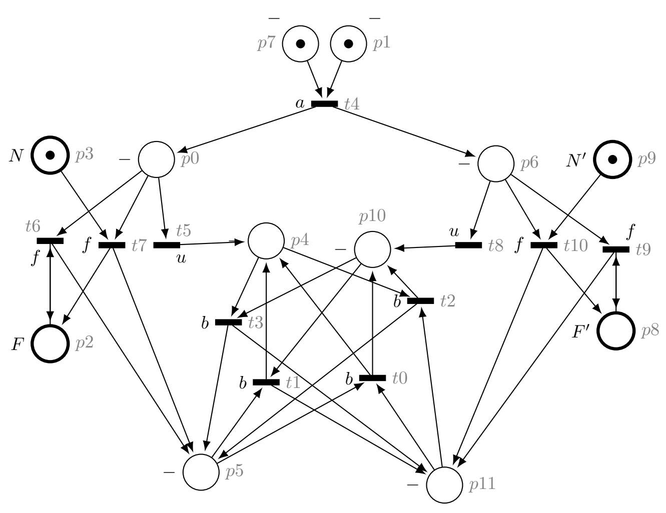

Figure 7. Twin-plant $\Gamma = \Theta_\Omega \parallel_{\{a,b\}} \Theta'_\Omega$

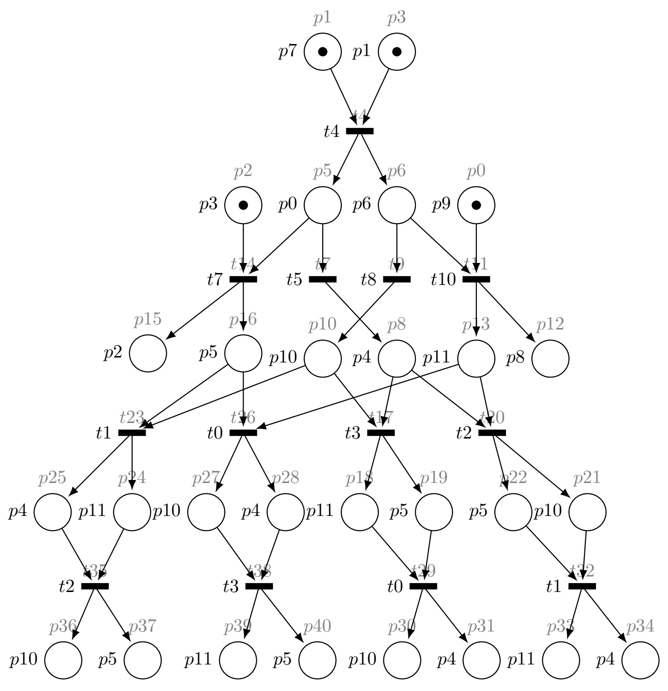

Figure 8. Γ's unfolding

Any element of H that does not lead to any ambiguity is pruned, i.e. $H[t_0]$ and $H[t_3]$ which respectively lead to a normal marking and a faulty marking (see places p_2, p_3, p_8 and p_9 in bold in Figure 7).

Finally, H is not empty as it still contains $H[t_1]$ and $H[t_2]$: there exist two infinite sequences, one is normal and the other is faulty, which have the same observable behaviour, thus the pattern Ω is not diagnosable in the system Θ.

Intuitively, this result is easy to find out on the system Θ directly. Both sequences $s_1 = afbbb\ldots$ and $s_2 = aubbb\ldots$ are infinite, produces the same observable behaviour $s_o = abbb\ldots$ but one is faulty whereas the other is not.

6. CONCLUSION AND PERSPECTIVES

We proposed to analyse the Ω-*diagnosability* of a supervision pattern by adapting the notion of pattern and twin-plant to Petri net and by using an unfolding technique. The Ω-*diagnosability* is a concept introduced in Jéron et al. (2006) and is a generalization of the classical notion defined in Sampath et al. (1995). In Jiang et al. (2003), the notions of K-diagnosability (diagnosability of K occurrences of the same event) and $[1, K]$-diagnosability (diagnosability of J occurrences of the same event, $J \in [1, K]$) are particular cases of Ω-*diagnosability* with a particular set of supervision patterns.

BLABLA A REVOIR DE TOUTE FACON, JE TRADUIS PAS

REFERENCES

Basile, F., Chiacchio, P., and De Tommasi, G. (2009). An efficient approach for online diagnosis of discrete event systems. *Transactions on Automatic Control*, 54(4), 748–759.

Benveniste, A., Fabre, E., Haar, S., and Jard, C. (2003). Diagnosis of asynchronous discrete-event systems: a net unfolding approach. *Transactions on Automatic Control*, 48(5), 714–727.

Cabasino, M.P., Giua, A., Lafortune, S., and Seatzu, C. (2012). A new approach for diagnosability analysis of petri nets using verifier nets. *Transactions on Automatic Control*, 57(12), 3104–3117.

Dotoli, M., Fianti, M.P., Mangini, A.M., and Ukovich, W. (2009). On-line fault detection in discrete event systems by petri nets and integer linear programming. *Automatica*, 45(11), 2665–2672.

Esparza, J., Römer, S., and Vogler, W. (2002). An Improvement of McMillan's Unfolding Algorithm. *Formal Methods in System Design*, 20(3), 285–310.

Jéron, T., Marchand, H., Pinchinat, S., and Cordier, M.O. (2006). Supervision Patterns in Discrete Event Systems Diagnosis. In *Workshop on Discrete Event Systems, WODES'06*, 262–268. IEEE Computer society, Ann-Arbor, USA.

Jiang, S., Huang, Z., Chandra, V., and Kumar, R. (2000). A polynomial algorithm for testing diagnosability of discrete event systems. *IEEE Transactions on Automatic Control*.

Jiang, S. and Kumar, R. (2004). Failure diagnosis of discrete-event systems with linear-time temporal logic specifications. *Transactions on Automatic Control*, 49(6), 934–945.

Jiang, S., Kumar, R., and Garcia, H.E. (2003). Diagnosis of repeated/intermittent failures in discrete event systems. *IEEE Transactions on Robotics*, 19(2), 310–323.

McMillan, K.L. (1995). A technique of state space search based on unfolding. *Formal Methods in System Design*, 6(1), 45–65.

Pencolé, Y. and Subias, A. (2009). A chronicle-based diagnosability approach for discrete timed-event systems: Application to web-services. *J. UCS*, 15(17), 3246–3272.

Peterson, J.L. (1977). Petri nets. *ACM Comput. Surv.*, 9(3), 223–252.

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

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