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

Abstract: This 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 $\Sigma$, we define $\Sigma^*$ as the set of finite sequences over $\Sigma$ (including the empty sequence $\lambda$). $\Sigma^+$ is the set of non-empty sequences over $\Sigma$. A subset $S$ of $\Sigma^*$ is called a language over $\Sigma$. For every sequence $s \in S$ we define the set of the continuations of $s$ in $S$: $S' = \{ t \in S : st \in S \}$. The projection of a sequence $s$ on $\Sigma_p \subseteq \Sigma$ is noted $P_{\Sigma_p}(s)$. Finally, the length of a sequence $s$ is denoted $|s|$. 

2.1 Labelled Petri nets

**Definition 1.** A labelled Petri net is a tuple $(P,T,A,\ell,L,\Sigma,M_0)$ where:

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

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

A marking is a multiset 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 $\Theta = (P,T,A,\ell,L,\Sigma,M_0)$ where $(P,T,A,\ell,L,\Sigma)$ is a labelled Petri net and $M_0$ an initial marking.

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

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

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

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

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 $\Phi = (P_\Phi,T_\Phi,A_\Phi,\ell_\Phi,P,T,m_{0\Phi})$ of a marked and labelled Petri net $\Theta = (P,T,A,\ell,L,\Sigma,m_0)$, is a labelled Petri net such that:

1. $\forall p \in P_\Phi : |p| \leq 1$.
2. $\Phi$ is acyclic, i.e. for any element $x \in P_\Phi \cup T_\Phi : \neg(x < x)$.
3. $\Phi$ 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 $\neg(x \neq x)$.
5. $\ell_\Phi(P_\Phi) \subseteq P$, $\ell_\Phi(T_\Phi) \subseteq T$;
6. $\forall t \in T_\Phi : \Phi(t)$ is isomorphic to $\bullet$;
7. $\forall t \in T_\Phi : \ell_\Phi(t)$ is isomorphic to $\emptyset$.

We denote by $\min(\Phi) = \{ x \in P_\Phi \cup T_\Phi : \exists y \in P_\Phi \cup T_\Phi, y < x \}$. A configuration $C$ of an unfolding is a set of causally-closed and conflict-free transitions of $T_\Phi$, formally: $t \in C \Rightarrow \forall t' < t, t' \in C \land \forall t', t'' : \neg(t' < t'' \land t' \in C)$. The cut of a configuration $C$ is the set of places $\text{Cut}(C) = (\min(\Phi) \cup C) \setminus C$ where $C = \{ p \in \bullet, t \in C \}$ and $\Phi(C) = \{ p \in \bullet, t \in C \}$ and $\ell_\Phi(\text{Cut}(C))$ characterizes a marking of the net $\Theta$. 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 $\Phi$ also called the finite complete prefix such that any reachable marking of $\Theta$ 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 $\ell_\Phi(\text{Cut}([t'])) = \ell_\Phi(\text{Cut}([t_c]))$ and the relation $t' < t_c$ holds, where $\prec$ is an adequate order (see Esparza et al. (2002)). The finite complete prefix of $\Phi$ 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 $\Phi$.

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

- $P = P_1 \cup P_2$;
- $T = T_1 \cup T_2 \cup T_s$,
  where $T_s = \bigcup_{i=1,2} \{ t_i \mid t_i \notin \Sigma_\ast \}$, $i=1,2$; and $T_s = \bigcup_{i=1,2} \{ t_i \mid t_i \notin \Sigma_\ast \}$, $i=1,2$.
- $A = A_1 \cup A_2 \cup A_\ast$,
  where $A_\ast = \{ (p,t) \in (P \times T) : p \in P_1, t_1 = t \mid t_2, (p,t_1) \in A_1 \}$, $p \in P_2, t_2 = t \mid t_2, (p,t_2) \in A_2 \}$, $p \in P_2, t_2 = t \mid t_2, (p,t_2) \in A_2 \}$.

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)).
This product can be constructed simply by taking the union of the two nets, removing transitions with labels in $\Sigma_s$, and for each $(t_1, t_2) \in T_1 \times T_2$ where $\ell(t_1) = \ell(t_2) \not= \lambda$, 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 $\Theta = \Theta_1 \parallel \Theta_2$ 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: $L(\Theta) = \{s \in (\Sigma_1 \cup \Sigma_2)^* : P_{\Sigma_1}(s) \in L(\Theta_1), i = 1, 2\}$. $L(\Theta)$ also verifies: $s \in L(\Theta)$ if $\exists s_1 \in L(\Theta_1), \exists s_2 \in L(\Theta_2) : P_{\Sigma_1}(s) = P_{\Sigma_1}(s_1) = P_{\Sigma_1}(s_2)$.

### 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 $(P, T, A, \ell, L, \Sigma, m_0)$ such that:

1. (labelling) $L = \{N, F\}$;
2. (initialization) $\forall p \in P : m_0(p) > 0 \Rightarrow \ell(p) = N$;
3. (exclusivity) $\forall p_1, p_2 \in P : \ell(p_1) = N \land \ell(p_2) = F \Rightarrow m(p_1) \times m(p_2) = 0$;
4. (completeness) $\forall m : (\exists s = t_1t_2\ldots : m_0 \Rightarrow m), \forall e \in \Sigma, \exists t \in T : (\ell(t) = e \land \exists p \in P, m(p) > 0)$;
5. (recognition) $\forall m' : (\exists s = t_1t_2\ldots t_n : m_0 \Rightarrow m') : (m' \Rightarrow m \land \exists p \in P : m'(p) > 0 \land \ell(p) = F \Rightarrow \exists p \in P : m(p) = F)$.

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

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.

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{figure2.png}
\caption{$k$ occurrences of the same event $e$.}
\end{figure}

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{figure3.png}
\caption{Stages of the diagnosability analysis.}
\end{figure}

\textbf{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 $S$ over an alphabet $\Sigma$, where $\Sigma$ — the set of events generated by the system — is partitioned into two subsets: $\Sigma_o$ and $\Sigma_u$, 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.

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

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

\[ \exists n \in \mathbb{N}, \forall s \text{ a faulty sequence of } S, \forall t \in S/s : \|\mathcal{P}_{\Sigma_{\mathcal{R}}}(t)\| \geq n \implies \mathcal{P}_{\Sigma_{\mathcal{R}}}(\mathcal{P}_{\Sigma_{\mathcal{R}}}(st)) \subseteq \Omega. \]

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.

\textbf{Definition 6.} A labeled Petri Net $\Theta$ is $\Omega$-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 $\Theta = (P,T,A,\ell,L,\Sigma,m_0)$ vis-a-vis a supervision pattern modeled by a labeled Petri net $\Omega = (P_{\Omega},T_{\Omega},A_{\Omega},\ell_{\Omega},L_{\Omega},\Sigma_{\Omega},m_{0\Omega})$. 
Algorithm 1: Search for ambiguous sequences

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

In order to illustrate every stage of the analysis, we present here a very simple example: the underlying system is modeled by \(\Theta\) of Figure 4 and the pattern \(\Omega\) is the one of Figure 5 representing the occurrence of a single event. Every event of \(\Theta\) is observable except \(u\) and \(f\). Therefore, \(\Sigma = \{a, b, u, f\}, \Sigma_0 = \{a, b\}\) and \(\Sigma_u = \{u, f\}\).

First stage is the computation of the synchronized product \(\Theta \parallel \Omega\) where \(\Sigma_s = \Sigma \cap \Sigma_0 = \{f\}\) which leads to the net \(\Theta_\Omega\) represented on Figure 6.
labels of the resulting marking

\[ \{(a, b)\} \Theta' \]

\( \Theta \) and \( \Theta' \) are fault events, only \( a \) and \( b \) are observable.

Second stage is the computation of the twin-plant, that is the synchronized product of \( \Theta \) with its primed version \( \Theta'_\Omega \) over the set of observables \( \Sigma_o = \{a, b\} \). The result \( \Gamma \) is depicted on Figure 7.

Last stage is the unfolding of \( \Gamma \) 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 \( \Gamma \) the set of labels associated to the places of Cut(\( C(t) \)) that would be marked if \( t \) were fired.

<table>
<thead>
<tr>
<th>cutoff point</th>
<th>labels of the resulting marking</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t_0 )</td>
<td>{( p_4, p_{10}, p_3, p_9 )}</td>
</tr>
<tr>
<td>( t_1 )</td>
<td>{( p_4, p_{11}, p_3, p_8 )}</td>
</tr>
<tr>
<td>( t_2 )</td>
<td>{( p_5, p_{10}, p_2, p_9 )}</td>
</tr>
<tr>
<td>( t_3 )</td>
<td>{( p_5, p_{11}, p_2, p_8 )}</td>
</tr>
</tbody>
</table>

Any element of \( H \) that does not lead to any ambiguity is pruned, i.e. \( H[t_1] \) and \( H[t_2] \) 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 \( \Omega \) is not diagnosable in the system \( \Theta \).

Intuitively, this result is easy to find out on the system \( \Theta \) directly. Both sequences \( s_1 = a f b b b \ldots \) and \( s_2 = a u b b b \ldots \) are infinite, produces the same observable behaviour \( s_0 = a b b b \ldots \) but one is faulty whereas the other is not.
6. CONCLUSION AND PERSPECTIVES

We proposed to analyse the $\Omega$-diagnosability of a supervision pattern by adapting the notion of pattern and twin-plant to Petri net and by using an unfolding technique. The $\Omega$-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 $\Omega$-diagnosability with a particular set of supervision patterns.

REFERENCES


