How to use Model Checking for Diagnosing Fault Patterns in Petri nets

Johanne Bakalara, Yannick Pencolé, Audine Subias

15th IFAC Workshop on Discrete Event Systems
November 11-13, 2020. Rio de Janeiro, Brazil

How to use Model Checking for Diagnosing Fault Patterns in Petri nets

Johanne Bakalara * **Yannick Pencolé** * **Audine Subias.** *

* *LAAS-CNRS, Université de Toulouse, CNRS,INSA, Toulouse, France (ypencole@laas.fr,subias@laas.fr).*

Abstract: This paper deals with the problem of fault pattern diagnosis in discrete event systems modelled by Petri nets. A general framework for implementing a pattern diagnosis function by model-checking is proposed. Two different approaches with experimental results are presented and compared to a third approach from the literature.

Keywords: Diagnosis, model checking, fault patterns, Petri net, Linear Time Logic

1. INTRODUCTION

The original problem of diagnosis in discrete event systems as introduced in Sampath et al. (1995) is to determine the occurrence of a fault in a system, the fault being modelled by a single event. In this work, our objective is to consider the diagnosis of a fault event but also the diagnosis of more complex faulty behaviours (called *fault patterns*) that cannot be modelled by a simple event (see Jéron et al. (2006), Boussif and Ghazel (2018)). Indeed, for some systems, a change in the order of occurrence of the events, or the joint occurrence of specific events, or the multiple occurrence of a specific type of event may reflect a faulty behaviour. This paper addresses the pattern diagnosis problem that has been introduced in Pencolé and Subias (2017) for Petri nets. The objective is to investigate several manners to solve the problem that use model-checking techniques (Clarke et al. (1999)). The advantages of model checking methods are numerous: speed, no proof building, many logics allowing to express large numbers of properties, generation of counter-examples. Model-checking has been well studied for many years and many techniques are used to counter the main obstacle of the combinatorial explosion (e.g. abstraction, symmetries, reduction). The objective of this article is to consider a range of approaches based on model checking to solve the pattern diagnosis problem and to compare their advantages and drawbacks. The proposed approaches solve the same problem but differ in the way they represent the three elements that define a diagnosis problem: the system, the faulty behaviour and the sequence of events observed. The paper is organized as follows. The problem statement is presented in Section 2 followed by the description of the general method to solve the pattern diagnosis problem with a model-checking approach. Section 3 then presents a first method in which the faulty behaviour and the observations are directly expressed as a complex property to be verified in the system's model. A first series of experimental results are then presented in Section 4 and compared with a method proposed in Pencolé and Subias (2017). Based on these results, a second approach is proposed in Section 5 and compared with the others. Finally, conclusions and some perspectives are given in Section 6.

2. PROBLEM STATEMENT

The system is modeled as a classical bounded labeled Petri net, denoted LPN (possibly with transition priorities and arc inhibitors): some labels represent observable events ($\Sigma_o = \{o_1, ..., o_p\}$) and the others represent silent/unobservable events ($\Sigma_u = \{uo_1, ..., uo_m\}$). A run of the system is a sequence of fired transitions from an initial marking and generates a word $\rho \in (\Sigma_u \cup \Sigma_o)^*$. Now, given a sequence of observed events from the system, denoted $\sigma \in \Sigma_o^*$, the Ω-diagnosis problem consists in determining whether an unobservable pattern of events Ω has occurred or not. A pattern of events Ω denotes a complex assembling of events (a set of words from Σ_u^*) and we say that Ω has occurred in a run if the run generates a word ρ that contains as a subword one of the words of Ω (see Pencolé and Subias (2017) for more details). To solve this problem, the objective is to design a diagnosis function Δ that returns one of the three results: Ω_{Faulty} if the pattern has certainly occurred (any run of the system that generates σ is a run where Ω occurred), Ω_{Absent} if the pattern is certainly absent (no run of the system that generates σ is a run where Ω occurred) and $\Omega_{Ambiguous}$ otherwise. In this paper we investigate several approaches to design Δ by using a model-checking tool called TINA (TIme petri Net Analyzer `http://www.laas.fr/tina/`) that is able to check properties over LPN and compare them with the one of Pencolé and Subias (2017).

2.1 Background on model checking

A model-checking problem consists in checking whether some given states (or runs) of a formal model $\mathcal{M}$ satisfy a given property ϕ encoded in a formula of some temporal logic (Clarke et al. (1999)). A model-checking tool checks if $\mathcal{M}$ satisfies ϕ (denoted $\mathcal{M} \models \phi$) i.e if ϕ is true in all the worlds of $\mathcal{M}$. To do so, one solution is the automata based approach of model-checking. In this case the problem is translated to a language intersection problem by converting $\mathcal{M}$ and $\neg\phi$ as Büchi automata. The Büchi automaton of $\mathcal{M}$ generates the infinitely long words of $\mathcal{L}^\omega(\mathcal{M})$ describing the set of runs of $\mathcal{M}$ while the Büchi automaton of $\neg\phi$ generates the infinitely long

words of $\mathcal{L}^\omega(\neg\phi)$ where $\neg\phi$ holds. The model-checker then checks for $\mathcal{L}(\mathcal{M})\cap\mathcal{L}(\neg\phi) = \varnothing$. If the intersection is empty, then $\mathcal{M} \models \phi$, otherwise $\mathcal{M} \not\models \phi$ and a counterexample is returned as a word of $\mathcal{L}(\mathcal{M})\cap\mathcal{L}(\neg\phi)$. Linear temporal logic (LTL) can be used to express the properties so that their statements can easily be transformed in automata. TINA is a model-checking tool that converts any LPN into a specific enriched Kripke structure $\mathcal{M}$ (Berthomieu et al. (2004), Berthomieu et al. (2007)) and checks properties ϕ that are written in State/Event LTL (SE-LTL) logic. A formula ϕ is a SE-LTL formula if it is a universally quantified formula $\phi ::= \forall\varphi$ where $\varphi ::= cst \mid r \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi\wedge\varphi \mid \bigcirc\varphi \mid \Box\varphi \mid \Diamond\varphi \mid \varphi\sqcup\varphi$. The constant cst can be $\bot$ (false), $\top$ (true), **dead** (deadlock), **div** (temporal divergence), **sub** (partially known state). r denotes constraints $e \triangle e$ with $\triangle \in \{=, <, >, \leq, \geq\}$ between arithmetical expressions e involving place and transition symbols from the underlying LPN. Finally, modal operators $\bigcirc, \Box, \Diamond, \sqcup$ are respectively the operators *next*, *always*, *eventually* and *until*. SE-LTL expresses formulas about places and transitions of Petri nets, however, all along the paper, we will express SE-LTL formulas with events from $\Sigma_u \cup \Sigma_o$. An event e being the label of one or several transitions $t_1, \ldots, t_n$ of the net, by writing e in any of the following formula we actually write the underlying SE-LTL formula $(t_1 \wedge \neg(t_2 \vee \cdots \vee t_n)) \vee (t_2 \wedge \neg(t_1 \vee t_3 \cdots \vee t_n)) \vee \ldots$ that asserts that one (and only one) transition labeled with e is fired.

2.2 General principle for implementing a diagnosis function by model-checking

Algorithm 1 describes the general diagnosis algorithm implementing the diagnosis function Δ by model-checking. $\mathcal{M}$ denotes the formal model that is investigated while $\Psi_{\neg\Omega}$ is a SE-LTL formula corresponding to the question: are we sure that pattern Ω never occurred in a run consistent with the observation sequence σ? If the result is true, Δ can immediately returns Ω_{Absent}. If not, a new question Ψ_Ω is asked: are we sure that Ω always occurred in all the sequences consistent with σ? If the result is true, Δ can return Ω_{Faulty}, otherwise we are sure there is an ambiguity (Δ returns $\Omega_{Ambiguous}$).

Algorithm 1 Diagnosis algorithm Δ by model-checking

```
if M ⊨ Ψ_¬Ω then
    return Δ(σ) = Ω_Absent
else if M ⊨ Ψ_Ω then
    return Δ(σ) = Ω_Faulty
else
    return Δ(σ) = Ω_Ambiguous
end if
```

Algorithm 1 is actually an algorithm template as we have not yet defined what is $\mathcal{M}$, $\Psi_{\neg\Omega}$ and Ψ_Ω. The next sections propose different methods and compare them.

3. A PURE MODEL-CHECKING APPROACH

This section presents a method that is called *pure model-checking approach* (denoted PURE for short). It is called pure in the sense that the method aims at fully encoding the pattern diagnosis problem with the questions Ψ_Ω and $\Psi_{\neg\Omega}$ based on the model of the system that is encoded with $\mathcal{M}$. There are actually two subproblems to solve. The first one is: what are the runs of the system that are consistent with σ? The second one is: among these runs, what are the ones where Ω has occurred and the ones where it has not. The two formulae $\Psi_{\neg\Omega}$ (absence of Ω) and Ψ_Ω (occurrence of Ω) are then as below:

$$\begin{cases} \Psi_{\neg\Omega} \ : \Phi_\sigma \Rightarrow \Phi_{\neg\Omega} \\ \Psi_\Omega \ : \ \Phi_\sigma \Rightarrow \Phi_\Omega \end{cases} \quad (1)$$

Φ_σ is the part of the formula that expresses that any run of interests in the question is consistent with the observation sequence σ while $\Phi_{\neg\Omega}$ (resp. Φ_Ω) expresses that Ω has not occurred in any run (resp. Ω has occurred in any run). It follows that $\Psi_{\neg\Omega}$ is true either if Ω does not occur in any run consistent with σ or if there is no run consistent with σ. For Ψ_Ω, in the remainder it is supposed that the observed sequence is correct and that it always exists at least one run in the model of the system that is consistent with the observation σ. Therefore, Ψ_Ω is true if Ω occurs in any run consistent with σ.

3.1 Definition of Φ_σ

Let us consider now the representation of the observation sequence σ in the $SE-LTL$ formula (Φ_σ). We denote NOE the disjunction of all the unobservable events, $NOE = \bigvee_{1\leq i\leq m} uo_i = uo_1 \ \vee \ uo_2 \ ... \ \vee \ uo_m$ with $NOE = \top$ if the set is empty. An observation sequence $\sigma = o_1o_2$ can be described by the following formula: $NOE\sqcup(o_1\wedge\bigcirc(NOE\sqcup o_2))$. In other words, an observation sequence is a trace including some unobservable events (NOE) until ($\sqcup$) an observable event o_1 and this event o_1 is directly followed ($\bigcirc$) by unobservable events (NOE) until ($\sqcup$) o_2. Note that the Until operator (a $\sqcup$ b) accepts the traces that satisfy a until b but also nothing before b. So, if no unobservable event occurs between the two observed events o_1 and o_2 in a run, this property holds in the run. In the general case an observation sequence σ of $k = |\sigma|$ events is then recursively given by $\Phi_\sigma = \varphi_1$ with φ_n for n from $(k-1)$ to 1 such that:

$$\begin{cases} \varphi_k \ = \ NOE \ \sqcup \ \ o_k \\ \varphi_n \ = \ NOE \ \sqcup \ (\ o_n \ \wedge\bigcirc \ (\varphi_{n+1})\ ) \end{cases} \quad (2)$$

Let us consider $\sigma = o_1o_2o_3$, φ_1 is given by:

$$\begin{array}{lcl} \varphi_3 & = & NOE \sqcup o_3 \\ \varphi_2 & = & NOE \sqcup (o_2 \wedge \bigcirc(\varphi_3))) \\ & = & NOE \sqcup (o_2 \wedge \bigcirc(NOE \sqcup o_3))) \\ \varphi_1 & = & NOE \sqcup (o_1 \wedge \bigcirc(\varphi_2))) \\ & = & NOE \sqcup (o_1 \wedge \bigcirc(NOE \sqcup (o_2 \\ & & \wedge \bigcirc (NOE \sqcup o_3)))))) \end{array}$$

then $\Phi_\sigma = \varphi_1 = NOE \sqcup (o_1 \wedge \bigcirc(NOE \sqcup (o_2 \wedge \bigcirc(NOE \sqcup o_3)))))$.

3.2 Definitions of $\Phi_\Omega, \Phi_{\neg\Omega}$: the single fault case

We consider first the single fault case: Ω is the occurrence of a single fault $f \in \Sigma_u$. To define $\Phi_f, \Phi_{\neg f}$, we need to distinguish two cases: diagnosis problem with or without *silent closure* (introduced in Lamperti and Zanella (2003)). In the case that the diagnoser does not want to address

the silent closure issue (i.e. the diagnosis function does not look for what could have happened *after* the last observed event of σ), only unobservable events occurring before or during executions that are consistent with the observation sequence σ are considered.

Diagnosis without silent closure
To check that the fault has not ocurred in any run (i.e $\Phi_{\neg f}$) it is necessary to express that the fault does not occur neither at the beginning of the observation nor between observable events. Note that at this stage only the relevant runs are considered due to the double implication of Equation (1). As an example let us consider an observation sequence $\sigma = o_1 o_2$, any run in which $\Phi_{\neg f}$ holds must be such that it does not include f until ($\sqcup$) o_1 that is followed ($\bigcirc$) by some events that are not f until o_2. This can be expressed by: $(\neg f \sqcup (o_1 \wedge \bigcirc(\neg f \sqcup o_2)))$.

From this example it appears that the formula can be expressed in a recursive way such that: $\Phi_{\neg f} = \varphi_1$ with φ_n for n from $(k-1)$ to 1.

$$\begin{cases} \varphi_k = \neg f \sqcup o_k \\ \varphi_n = \neg f \sqcup (o_n \wedge \bigcirc(\varphi_{n+1})) \end{cases} \quad (3)$$

Let us consider the observation sequence $\sigma = o_1 o_2 o_3$, φ_1 is then given recursively by:

$$\begin{array}{lcl} \varphi_3 & = & \neg f \sqcup o_3 \\ \varphi_2 & = & \neg f \sqcup (o_2 (\wedge \bigcirc (\varphi_3))) \\ & = & \neg f \sqcup (o_2 (\wedge \bigcirc (\neg f \sqcup o_3))) \\ \varphi_1 & = & \neg f \sqcup (o_1 (\wedge \bigcirc (\varphi_2))) \\ & = & \neg f \sqcup (o_1 (\wedge \bigcirc (\neg f \\ & & \sqcup (o_2 (\wedge \bigcirc (\neg f \sqcup o_3))))) \end{array}$$

Then $\Phi_{\neg f} = \neg f \sqcup (o_1 (\wedge \bigcirc (\neg f \sqcup (o_2 (\wedge \bigcirc (\neg f \sqcup o_3))))))$

Now let us define Φ_f, we need to express that the fault may appear before the observed events or between the observed events. Let us consider a simple sequence of two events $\sigma = o_1 o_2$. Any run where Φ_f holds must include unobservable events different from f ($NOE_{/\{f\}}$) until f or the event o_1 directly followed by unobservable events different from f until f: $NOE_{/\{f\}} \sqcup (f \vee (o_1 \wedge \bigcirc (NOE_{/\{f\}} \sqcup f)))$. This can be formalized in a recursive way by: $\Phi_f = \varphi_1$ with φ_n for n from $(k-1)$ to 1.

$$\begin{cases} \varphi_k = NOE_{/\{f\}} \sqcup f \\ \varphi_n = NOE_{/\{f\}} \sqcup (f \vee (o_n \wedge \bigcirc(\varphi_{n+1}))) \end{cases} \quad (4)$$

To illustrate this formula let us consider $\sigma = o_1 o_2 o_3$.

$$\begin{array}{lcl} \varphi_3 & = & NOE_{/\{f\}} \sqcup f \\ \varphi_2 & = & NOE_{/\{f\}} \sqcup (f \vee (o_2 \wedge \bigcirc(\varphi_3))) \\ & = & NOE_{/\{f\}} \sqcup (f \vee (o_2 \wedge \bigcirc(NOE_{/\{f\}} \sqcup f))) \\ \varphi_1 & = & NOE_{/\{f\}} \sqcup (f \vee (o_1 \wedge \bigcirc(\varphi_2))) \\ & = & NOE_{/\{f\}} \sqcup (f \vee (o_1 \wedge \bigcirc(NOE_{/\{f\}} \sqcup \\ & & (f \vee (o_2 \wedge \bigcirc(NOE_{/\{f\}} \sqcup f)))))) \end{array}$$

then $\Phi_f = \varphi_1 = NOE_{/\{f\}} \sqcup (f \vee (o_1 \wedge \bigcirc(NOE_{/\{f\}} \sqcup (f \vee (o_2 \wedge \bigcirc(NOE_{/\{f\}} \sqcup f)))))))$.

How to deal with silent closure
By considering the silent closure, the diagnoser must now also take into account the fact that the fault can also occur after the last observed event. To check that the fault is absent all along the run, we then need to modify $\Phi_{\neg f}$. Let $OE = \bigvee_{1 \leq i \leq p} o_i = o_1 \vee o_2 \ ... \ \vee o_p$ be the disjunction of all observable events from Σ_o. Let us consider an observation sequence $\sigma = o_1 o_2$. The runs that we want to characterize with $\Phi_{\neg f}$ do not include the fault event f until o_1, that is followed by unobservable events different from f until o_2, that is followed by unobservable events different from f until any observable event. So by generalizing, the formula $\Phi_{\neg f}$ is given in a recursive way as $\Phi_{\neg f} = \varphi_1$ with φ_n for n from k to 1.

$$\begin{cases} \varphi_{k+1} = \neg f \sqcup OE \\ \varphi_n = \neg f \sqcup (o_n \wedge \bigcirc(\varphi_{n+1})) \end{cases} \quad (5)$$

Suppose now that $\mathcal{M} \models \Phi_\sigma \Rightarrow \Phi_{\neg f}$ is true, the diagnosis function stops and returns f_{Absent} (see Algorithm 1). If it is not true, suppose that f occurs before the last observed event in every run such that Φ_σ holds, then with the previous formula Φ_f, $\mathcal{M} \models \Phi_\sigma \Rightarrow \Phi_f$ is true and Δ returns f_{Faulty}. Finally, if there is a run, such that Φ_σ holds, in which f occurs only after the last observed event, then there is also a run, such that Φ_σ holds, in which f does not occur at all (it is a prefix of the previous run) so in this last case, Δ must return $f_{Ambiguous}$. Therefore, the formula Φ_f remains unchanged when dealing with silent closure, only $\Phi_{\neg f}$ is modified.

3.3 Extension to fault class diagnosis

We now propose to extend this pure model-checking approach to deal with a specific type of patterns called *fault class*. A fault class gathers a set of independent fault events denoted $\mathcal{F} = \{f_1, f_2, .., f_q\} \subseteq \Sigma_u$. Following Algorithm 1, the diagnosis principle is to check first that no fault of the class has occurred in a run of the system consistent with the observation sequence. If the result is true (i.e. $\mathcal{M} \models \Psi_{\neg \mathcal{F}}$), the diagnosis result is $\mathcal{F}_{Absent}$ i.e it is sure that the fault class has not occurred. Otherwise, if in any system's run consistent with the observations σ, at least one fault of the class $\mathcal{F}$ has occurred, the diagnosis result is $\mathcal{F}_{Faulty}$. Finally, Δ returns $\mathcal{F}_{Ambiguous}$ if there are at least two runs consistent with σ such that one contains at least a fault of the class $\mathcal{F}$ and the other run does not contain any fault of the class.

Fault class diagnosis without silent closure
The SE-LTL formula that checks the absence of a fault class is similar to the one proposed in the case of simple fault where the simple fault event f is replaced by F that denotes the disjunction of all the faults of $\mathcal{F}$: $F = \bigvee_{f \in \mathcal{F}} f$. Then $\Phi_{\neg \mathcal{F}} = \varphi_1$ with φ_n for n from $(k-1)$ to 1.

$$\begin{cases} \varphi_k = \neg F \sqcup o_k \\ \varphi_n = \neg F \sqcup (o_n \wedge \bigcirc(\varphi_{n+1})) \end{cases} \quad (6)$$

Formula $\Phi_{\mathcal{F}}$ is also derived from the simple case (see Equation (4)). $\Phi_{\mathcal{F}} = \varphi_1$ with φ_n for n from $(k-1)$ to 1.

$$\begin{cases} \varphi_k = NOE_{/\mathcal{F}} \sqcup F \\ \varphi_n = NOE_{/\mathcal{F}} \sqcup (F \vee (o_n \wedge \bigcirc(\varphi_{n+1}))) \end{cases}$$

Case of the silent closure
For the absence of fault class with the silent closure, the

formula is similar to the one given for the simple fault (see Equation (5)) by replacing the event f by F. To check the occurrence of the fault class, the same formula as the one used in the case without the silent closure is considered for the same reason as in the simple fault case.

4. FIRST EXPERIMENTAL RESULTS

This section describes some experimental results about the PURE approach presented here above. These experiments are performed on the example from Pencolé and Subias (2017): a product transportation system.

4.1 Product transportation system

The proposed example is a two-level system composed of two product sites (namely sites 1 and 2) at level 2 where a set of products is stored and two assembly stations at level 1 that request products from level 2. A lift is used between the two levels. Each site has a conveyor belt to move a product from the site to the lift and each station also has a conveyor belt to get the products from the lift. Figure 1 presents the system LPN model. Once a product is detected on the site 1, a *Product1* signal (denoted $Pr1$) is emitted. The product is then put in a box and becomes available. The box is then pushed and sent into the lift (action $Push_1$ that starts with event P_1 and ends with event EP_1). Site 2 behaves in a similar manner. A product from site 1 has a priority access to the lift. In the Petri net model of Figure 1, this priority relation is indicated by a dashed edge between the transitions labeled by P_1 and P_2. After a product has been pushed into the lift, the lift goes down (action *Down*) to reach level 1 (signal D detects the end of the *Down* action). At this level the box is directed to the station that makes the request (either request Req_1 or request Req_2). If it is request Req_1, then the box is pushed on the conveyor belt of station 1 (action $PushReq_1$ that starts with event Req_1 and ends with event $EPReq_1$). Then the conveyor belt performs a move-left action ($LeftReq_1$, $ELReq_1$) to deliver the box with the product and a move-right action to go back to initial position ($RightReq_1$, $ERReq_1$). The behaviour of station 2 is similar except that its conveyor belt moves right first ($RightReq_2$, $ERReq_2$) and then moves left ($LeftReq_2$, $ELReq_2$). Once the box has been pushed on a conveyor belt, the lift goes up (action *Up* ending with the emission of signal U) to level 2. In Figure 1 only underlined events are observable events.

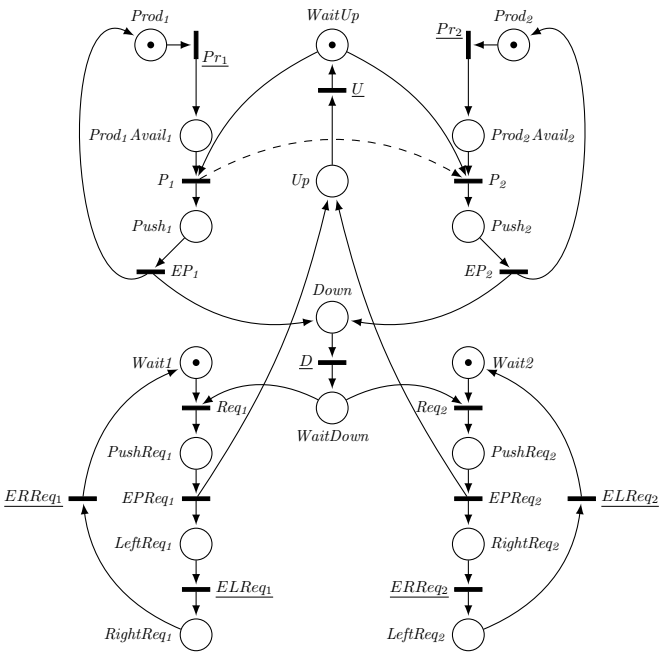

Fig. 1. Case study: the product transportation system

Obs.	$\Psi_{\neg f}$	Ψ_f	Diagnosis	Time(ms)
$Pr1$	T	F	*Absent*	32
$Pr2$	T	F	*Absent*	24
D	F	F	*Ambiguous*	22
U	F	F	*Ambiguous*	50
D	F	F	*Ambiguous*	126
$ERReq2$	F	F	*Ambiguous*	55
$ELReq2$	F	F	*Ambiguous*	3189
$ELReq1$	F	T	*Faulty*	25522
$ERReq1$	F	T	*Faulty*	240524
U	x	x	x	x

Table 1. Fault event diagnosis with silent closure

Obs.	$\Psi_{\neg f}$	Ψ_f	Diagnosis	Time(ms)
$Pr1$	T	F	*Absent*	12
$Pr2$	T	F	*Absent*	14
D	T	F	*Absent*	13
U	F	F	*Ambiguous*	22
D	F	F	*Ambiguous*	153
$ERReq2$	F	F	*Ambiguous*	197
$ELReq2$	F	F	*Ambiguous*	1076
$ELReq1$	F	T	*Faulty*	7121
$ERReq1$	F	T	*Faulty*	66799
U	x	x	x	x

Table 2. Fault event diagnosis without silent closure

4.2 Results

The objective in this experimental setting is to diagnose the occurrence of the event *Req1* based on

σ : *Pr1 Pr2 D U D ERReq2 ELReq2 ELReq1 ERReq1 U*

composed of ten observable events. In the approach that we already described, the Kripke structure $\mathcal{M}$ computed by TINA depends only on the system and has 204 states and 625 transitions. Table 1 presents the results and any line represents one step of the scenario σ (i.e line 1 is *Pr1*, line 2 is *Pr1* followed by *Pr2* etc). Columns 2 and 3 indicate respectively the answer of the question *are we sure that the fault never occurred on a run consistent with the observation sequence ?*($\Psi_{\neg f}$) and *are we sure that f always occurred in any run consistent with the observation sequence ? (* Ψ_f*)*. Column 4 gives the diagnosis conclusion and the last column is the computation times obtained by TINA. Table 2 presents the results in the case without silent closure. First of all, based on Table 2, the PURE approach gets the same results as for the method proposed in Pencolé and Subias (2017) (and denoted PROD here below). Also, if we compare both Tables, we can see that the results with the silent closure are sometimes more ambiguous (line 3) which is explained by the fact that *Req1* may occur between the occurrence of **D** and **U**. Without the silent closure this possible occurrence is simply ignored

(the diagnosis is *Absent*) while the diagnosis is *Ambiguous* when considering the silent closure. Now regarding the computation time, results show the computation time is not satisfactory as the computation time is exponential with the size of the observation sequence and PROD clearly outperforms PURE. A detailed analysis of the set of experiments indicates the source of the complexity problem. As explained in Section 2.1, one step of the model-checking is the computation of the language $\mathcal{L}(\neg\Phi)$, that is the conversion of the formula $\neg\Phi$ into its equivalent Büchi automaton and this conversion is unfortunately exponential to the length of formula Φ (i.e. $O(2^{|\Phi|})$) and so, by construction of the formula $\Psi_f, \Psi_{\neg f}$ (see Section 3), PURE is exponential in the size of σ ($O(2^{|\Phi|}) = O(2^{|\sigma|})$). Looking now at the PROD method, this limitation is not present. Indeed, even if PROD follows the same algorithm (see Algorithm 1), the information stored in $\mathcal{M}$ and in $\Psi_\Omega, \Psi_{\neg\Omega}$ are fundamentally different. PROD relies on a set of specific products of LPNs to gather in one single LPN the information about how a pattern Ω matches any single run of the system and how any single run of the system is consistent with σ. The size of $\mathcal{M}$ is in $O(2^{|\sigma|})$ in PROD while in PURE the size of $\mathcal{M}$ does not depend on $|\sigma|$. Secondly, in PROD, the questions $\Psi_\Omega, \Psi_{\neg\Omega}$ do not depend on the size of σ as opposed to the ones from PURE. As opposed to PROD, our objective is to benefit of the expressivity of SE-LTL to write pattern Ω and not as an LPN like in PROD. To avoid that Ψ_Ω and $\Psi_{\neg\Omega}$ depend on the size of σ, we propose a tradeoff between the PURE and PROD approaches. The idea is to synchronise first the system model with the observation sequence σ by classical transition fusion as in PROD to isolate the runs of the system consistent with the observation sequence. The result of this synchronisation is an LPN that would be converted as a Kripke structure $\mathcal{M}$ (in $O(2^{|\sigma|})$ as in PROD) and $\Psi_\Omega, \Psi_{\neg\Omega}$ would only characterize the pattern matching (independent from σ).

5. AN APPROACH OVER A MODEL SYNCHRONISED WITH THE OBSERVATIONS

This section describes the model-checking approach denoted SYN that is a compromise between PURE and PROD. The observed sequence σ is then modeled by a LPN O as a single sequence of places and transitions. A synchronised product between the LPN of the system and the LPN of the observation sequence is performed based on transition fusions (see Hack (1975)). Figure 2 presents such a synchronisation. The SYN approach is still governed by Algorithm 1 and the formula Ψ_Ω and $\Psi_{\neg\Omega}$ still follow Equation (1). But now, as opposed with the PURE approach, and similarly to the PROD approach, Ψ_σ consists in checking whether a system's run will eventually lead to a marking where p_{obs} is marked in the synchronised LPN (see Figure 2). It follows that in the SYN approach $\Phi_\sigma = \Diamond\, p_{obs}$ whose length does not depend on $|\sigma|$.

5.1 Diagnosis of a fault event

The objective is to check that no run consistent with the observations contains the fault event f at the beginning or between two observed events. The formula $\Phi_{\neg f}$ is then given by: $\Phi_{\neg f} = \Box\neg(f \wedge \bigcirc(\Diamond(o_k \wedge \bigcirc p_{obs})))$. $\Phi_{\neg f}$ asserts

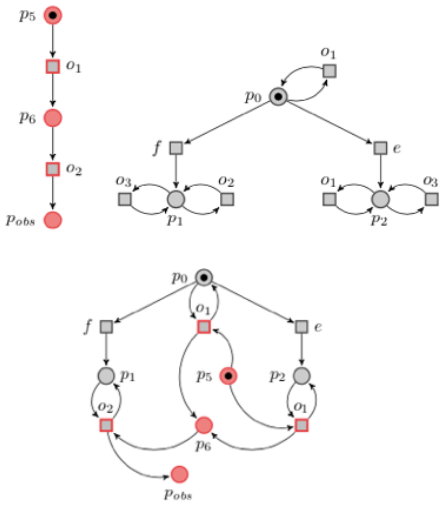

Fig. 2. σ on the left, the system on the right, the synchronisation below

that fault f never occurs ($\Box\neg$) if in the future ($\bigcirc$) [1] the system eventually leads ($\Diamond$) to the firing of the last event of σ ($o_k \wedge \bigcirc p_{obs}$). It is necessary to consider the sequence until the last observed event followed by p_{obs} as the event o_k is not a sufficient condition since the same event can be observed several times in the observation sequence, so its observation does not guarantee the end of the execution of the observation sequence. Moreover p_{obs} is also not sufficient as it remains marked after the execution of the observations whereas some unobservable events can occur after the last observation. The formula Φ_f is then given by $\Phi_f = \Diamond(f \wedge \bigcirc(\Diamond(o_k \wedge \bigcirc p_{obs})))$: it means the system will eventually ($\Diamond$) fire f followed ($\bigcirc$) by a set of transitions to end as a run that is consistent with σ ($o_k \wedge \bigcirc p_{obs}$). Unlike the PURE method, Φ_f has a size independent of $|\sigma|$.

5.2 Diagnosis of a fault class and multiple faults

The problem of fault class diagnosis can also be considered by replacing the event f by F the disjunction of all the faults of the class in the previous formulas $\Phi_{\neg f}$ and Φ_f as in the PURE method. Now we may also be interested in the diagnosis of multiple faults. Suppose for instance a fault class $\mathcal{F} = \{f_1, f_2\}$, the multiple fault associated with $\mathcal{F}$ is the occurrence of fault f_1 and f_2 in the same run. To diagnose such a multiple fault, the formulas $\Phi_{\neg\mathcal{F}}$, $\Phi_\mathcal{F}$ can be adapted: $\Phi_{\neg\mathcal{F}} = \Box\neg(\varphi_2)$ and $\Phi_\mathcal{F} = \varphi_2$ where
$\varphi_2 = \Diamond(f_1 \wedge \bigcirc(\Diamond(o_k \wedge \bigcirc p_{obs}))) \wedge (\Diamond(f_2 \wedge \bigcirc(\Diamond(o_k \wedge \bigcirc p_{obs}))))$.

The problem can be generalised to $\mathcal{F} = \{f_1, f_2, .., f_q\}$, thus $\Phi_{\neg\mathcal{F}} = \Box(\neg(\varphi_q))$ and $\Phi_\mathcal{F} = \varphi_q$ where φ_q is such that

$$\begin{cases} \varphi_n = (\varphi_{n-1}) \wedge (\Diamond(f_n \wedge \bigcirc(\Diamond(o_k \wedge \bigcirc p_{obs})))) \\ \varphi_1 = \quad (\Diamond(f_1 \wedge \bigcirc(\Diamond(o_k \wedge \bigcirc p_{obs})))) \end{cases} \tag{7}$$

5.3 Pattern diagnosis

With the SYN approach, the problem of fault event diagnosis can be extended to the fault pattern diagnosis that aims at diagnosing more complex faulty behaviours or any normal behaviour of interest (Jéron et al. (2006), Pencolé

[1] The next operator is used to traduce that the events f and o_k are not simultaneous, but it is not mandatory.

and Subias (2017)). For instance a pattern can describe a sequence of n unobservable events: let us consider a pattern Ω_1 corresponding to the sequence $e_1e_2e_3$. If Ω_1 occurs in a run then the run first leads to e_1 followed in the future by e_2 that is followed in the future by e_3 followed by the last observed event followed by end of synchronisation: $\Phi_{\Omega_1} = \Diamond(e_1 \wedge \bigcirc(\Diamond(e_2 \wedge \bigcirc\ (\Diamond(e_3 \wedge \bigcirc(\Diamond(o_k \wedge \bigcirc p_{obs})))))))$. And the absence Ω_1 in the run is characterized by the negation: $\Phi_{\neg\Omega_1} = \Box(\neg\Phi_{\Omega_1})$. This approach allows also to

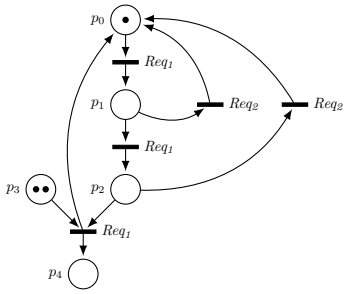

Fig. 3. Pattern $\Omega_2(3,2)$: 2 occurrences of 3 requests Req_1.

consider more complex behaviour that one might want to diagnose. Let us come back to system of Figure 1. Figure 3 describes a pattern $\Omega_2(3,2)$ that models 3 consecutive occurrences of events of type Req_1 that are not interleaved with any occurrence of Req_2 events (the left conveyor keeps having its requests fulfilled while the right conveyor gets stuck) and this pattern should occur 2 times (the pattern has a counter). This pattern is made up of normal events but defines a behaviour that could be unexpected (both conveyors at Level 1 should get items *regularly*). In this case it is necessary first to express that in the future the event $Req1$ is not followed by $Req2$ until $Req1$ that is not followed by $Req2$ until $Req1$. $\Xi = \Diamond(Req1 \wedge \bigcirc((\neg Req2 \sqcup Req1) \wedge \bigcirc((\neg Req2 \sqcup Req1)))$. Then, to check that the behaviour described above occurs twice the formula $\Phi_{\Omega_2(3,2)}$ is given by: $\Phi_{\Omega_2(3,2)} = \Xi \wedge \bigcirc(\Xi \wedge \bigcirc(\Diamond(o_k \wedge \bigcirc p_{obs})))$. The absence of the pattern is checked through $\Phi_{\neg\Omega_2(3,2)} = \Box(\neg(\Phi_{\Omega_2(3,2)}))$.

5.4 Results analysis

The SYN approach handle more complex problems than the PURE approach with satisfactory size of observations but is still not as efficient as PROD (see Table 3). Based on the pattern $\Omega_2(n, counter)$ of Figure 3, several comparisons have been made with the PROD approach by selecting different values for n (the number of *Req1*), *counter* and the size of the observation sequence. As an example Table 3 shows the results of the two approaches for one occurrence of five events $Req1$ without $Req2$, for ten occurrences of ten consecutive events $Req1$, with several sizes for the observation sequence. The SYN approach is efficient relatively to the size of the Kripke structure that only depends on the observation sequence. Nevertheless, for the same reason as for the PURE approach, SYN is limited by the length of the pattern formula, SYN must be only considered for small patterns only.

6. CONCLUSION AND FUTURE WORK

This paper addresses the diagnosis of fault patterns in the context of Petri nets. This problem is formulated as a model checking problem using the TINA tool taking benefit of the expressivity of SE-LTL. According to the knowledge encoded in the question and in the model. several approaches are presented and compared. In our perspectives, we plan to investigate alternative approaches to keep a low complexity both in the formula and in the models given to the model checker. One direction would be to consider a new way for the intersection of Petri nets (Lubat et al. (2019)). The applicability to online diagnosis is also an issue.

size	Req_1	counter	nS	nT	Diagnosis
SYN					
2000	5	1	6601	8000	$\Omega_{\mathcal{F}_{Absent}}$
5000	5	1	16501	20000	$\Omega_{\mathcal{F}_{Absent}}$
2000	10	10	6601	8000	x
PROD					
2000	5	1	1028	1223	$\Omega_{\mathcal{F}_{Absent}}$
5000	5	1	25988	30983	$\Omega_{\mathcal{F}_{Absent}}$
2000	10	10	10388	12383	$\Omega_{\mathcal{F}_{Absent}}$

Table 3. Fault class diagnosis without silent closure

REFERENCES

Berthomieu, B., Peres, F., and Vernadat, F. (2007). Model checking bounded prioritized time Petri nets. In *Automated Technology for Verification and Analysis*. Springer.

Berthomieu, B., Ribet, P.O., and Vernadat, F. (2004). The tool tina – construction of abstract state spaces for Petri nets and time Petri nets. *International Journal of Production Research*, 42(14), 2741–2756.

Boussif, A. and Ghazel, M. (2018). Formal verification of intermittent fault diagnosability of discrete-event systems using model-checking. *International Journal of Critical Computer-Based Systems*, 8(2), 193–213.

Clarke, E., Grumberg, O., and Peled, D. (1999). *Model checking*. MIT press.

Hack, M. (1975). Petri net languages. Technical Report 124, M.I.T. Project MAC, Computation Structures Group, Massachusetts Institute of Technology.

Jéron, T., Marchand, H., Pinchinat, S., and Cordier, M.O. (2006). Supervision patterns in discrete event systems diagnosis. In *8th International Workshop on Discrete Event Systems*, 262–268. Ann Arbor, MI, United States.

Lamperti, G. and Zanella, M. (2003). Continuous diagnosis of discrete-event systems. In *14th International Workshop on Principles of Diagnosis DX03*, 105–111. Washington DC, United States.

Lubat, É., Dal Zilio, S., Le Botlan, D., Pencolé, Y., and Subias, A. (2019). A State Class Construction for Computing the Intersection of Time Petri Nets Languages. In *17th International Conference FORMATS*, volume 11750 of *LNCS*. Springer, Amsterdam, Netherlands.

Pencolé, Y. and Subias, A. (2017). Diagnosis of supervision patterns on bounded labeled Petri nets by model checking. In *28h International Workshop on Principles of Diagnosis DX17*. Brescia, Italy.

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