# Diagnosability analysis of patterns on bounded labeled prioritized Petri nets

Houssam Eddine Gougam, Yannick Pencolé, Audine Subias

# Diagnosability analysis of patterns on bounded labeled prioritized Petri nets

Houssam-Eddine Gougam, Yannick Pencolé, Audine Subias
LAAS-CNRS, Université de Toulouse, CNRS,INSA, Toulouse, France.
e-mail: (hegougam@laas.fr, ypencole@laas.fr, subias@laas.fr)

March 2017

**Abstract**

Checking the diagnosability of a discrete event system aims at determining whether a fault can always be identified with certainty after the observation of a bounded number of events. This paper investigates the problem of pattern diagnosability of systems modeled as bounded labeled prioritized Petri nets that extends the diagnosability problem on single fault events to more complex behaviors. An effective method to automatically analyze the diagnosability of a pattern is proposed. It relies on a specific Petri net product that turns the pattern diagnosability problem into a model-checking problem.
Keywords: Fault diagnosis, Diagnosability, Pattern, Petri nets.

## 1 Introduction

Fault diagnosis of Discrete Event Systems (DES) is a generic problem that is driven by the needs of many different application domains such as manufacturing, transportation, communication networks to name a few of them (Zaytoon and Lafortune (2013)). Fault diagnosis aims at detecting, isolating and identifying the underlying physical phenomena (sensor faults, actuator faults, execution faults,...) that cause the system's failure, relying on a model of the system (faulty or not) and a set of observations of the system. Every fault diagnosis problem is associated with a fault diagnosability analysis that consists in determining whether the available sources of observations of the system are sufficient to always be able to assert with certainty which faults have effectively occurred with a bounded number of observations.

Sampath et al (1995) formally define the Fault Diagnosis problem and the diagnosability analysis of DES based on the classical automaton formalism. From this early approach several contributions were proposed exploring different modeling formalisms among them, communicating automata (Rozé and Cordier (2002), Pencolé and Cordier (2005), Lamperti and Zanella (2006)) or Petri nets (Benveniste et al (2003), Genc and Lafortune (2007), Lefebvre and Delherm (2007), Basile et al (2012), Cabasino et al (2014)) leading to various diagnosis and diagnosability methods. The intrinsically distributed nature of Petri Nets and their analytic capabilities make this tool powerful to

address diagnosis and diagnosability problems in different contexts (distributed, time contexts, probabilistic, ...) (Lai et al (2008)).

Diagnosability analysis was originally described based on a system modeled with an explicit set of fault events (see Sampath et al (1995)). In this paper, we address a more generic problem, called *pattern diagnosability analysis*. Patterns are a way to extend the notion of fault events by introducing more complex behaviors. Indeed, a system might be failing due to a specific sequence, i.e. a specific ordering of events occurring in the system. Any event of the sequence, independently from any other event, might not be a fault event, but the occurrence ordering of them is the cause of the failure. Patterns can model faulty behaviors but also any normal behaviors of interest (Jéron et al (2006)) or system's specifications (Jiang and Kumar (2004)). Patterns thus allow to tackle a larger spectrum of diagnosis problems: simple faulty event, multiple faults, fault repetitions....and more generally any behavior of interest (faulty or not). Another advantage of using patterns is to emphasize on the separation between the behavioral model of the system and the objectives of the diagnosis tasks defined as patterns (Zanella and Lamperti (2004)), also reinforcing the pattern reusability to address other diagnosis problems on other systems.

This paper proposes a formal and effective method to check pattern diagnosability in a system. We first propose a modeling framework where both patterns and systems are represented in a similar manner: the *L-type Labeled Prioritized Petri nets* (LLPPN). This formalism extends labeled Petri nets with transition priorities that can manage transition conflicts in the system models. Thanks to this formalism, the patterns are modeled in a concise way that is independent from the system's behavior. We then formally define the pattern diagnosability problem within this framework and propose an automated method to solve it. To do so, we define a specific operator on Petri nets, called the *system-pattern product*, that characterizes how a given pattern *is matched* by the system with the help of Petri net priorities. Finally, pattern diagnosability of LLPPN is translated into a model-checking problem that is efficiently solved by the Petri analyzer TINA (Berthomieu et al (2007)).

The paper is organized as follows. Section 2 presents a discussion on related work. Then Section 3 introduces the LLPPN formalism and presents how patterns and systems are modeled. Section 4 addresses the definition of the pattern diagnosability problem and Section 5 describes the formal method to solve it. Implementation aspects based on model checking techniques are given in Section 6. Experimental results on two case studies are presented in Section 7. Finally, we conclude in Section 8 and give some perspectives.

## 2 Related work

The problem of fault diagnosis in DESs was introduced a few decades ago in Lin (1994), Sampath et al (1995). A recent survey about this problem can be found in Zaytoon and Lafortune (2013). Diagnosability is the property that asserts whether a diagnoser can conclude with certainty or not depending on the observability of the underlying system. This property was initially introduced for single faults. The initial works on fault diagnosis and diagnosability rely on the automaton formalism. In Sam-

path et al (1995), diagnosability is checked by detecting fault-indeterminate cycles in a global diagnoser, the main problem of this approach is that the algorithm is exponential in the number of states of the global behavioral model. In Jiang et al (2001) and independently in Yoo and Lafortune (2002), the authors analyze the diagnosability without the use of the diagnoser, directly based on the behavioral model. The idea consists in determining whether there exist at least two infinite behaviors in the system, one faulty and one non-faulty, that have the same observable projection. The existence of such a couple proves that the system is not diagnosable, in Jiang et al (2001), the detection of such behaviors is based on the twin plant construction and in Yoo and Lafortune (2002) it is based on the construction of verifiers. Both algorithms are polynomial in the number of states in the behavioral model. Such a couple of behaviors is also called a critical pair in Cimatti et al (2003): in this work a more restrictive diagnosability property is defined as a temporal logic property of the twin plant that can be solved by a symbolic model checker. In Schumann and Pencolé (2007), a diagnosability checker is also proposed but the main difference here is that the authors use the distributed nature of the component-based DES to avoid the construction of a global twin plant by a successive set of diagnosability tests on a subset of components as initially introduced in Pencolé (2004). In Grastien (2009), the same diagnosability problem is proposed to be solved as a model-checking problem.

More recently, the diagnosis problem was also defined on Petri nets (Benveniste et al (2003), Genc and Lafortune (2007), Lefebvre and Delherm (2007)). The diagnosability problem on Petri nets was introduced in Haar et al (2003). In Giua and Seatzu (2005), Cabasino et al (2009), a novel approach to check diagnosability is introduced. It follows the principle of the diagnoser approach of Sampath et al (1995) but builds a finite state machine (called the basis reachability graph) that only represents the subpart of the states space that is sufficient to conclude about the diagnosability question, which is achieved by using the structure of the Petri net (in contrast to using its behavior). In Cabasino et al (2014), the authors extend the work of Cabasino et al (2009) to labeled Petri nets where different transitions can share the same label. In Basile et al (2012), the authors check the $k$-diagnosability (a fault is $k$-diagnosable if it is diagnosable within a window of $k$ observations after the occurrence of the fault) using integer linear programming techniques. Finally, in Liu et al (2014b), the way to solve the diagnosability problem is to perform the analysis on-the-fly while the machine is being built.

Presented works, detailed here above, develop some methods for the same problem that is the diagnosability of single faults in the system. With the emergence of complex engineered systems and the need of automated analyzes, some authors extend the initial problem of fault diagnosis in DESs to the pattern diagnosis in DESs. In Jiang and Kumar (2004), a pattern is defined as a temporal logic specification over a finite-state automaton. A fault is then defined as a behavior that *does not match* the pattern (that is called a specification in Jiang and Kumar (2004)). To check diagnosability, they propose an *ad hoc* algorithm that builds the part of interest of the Kripke structure of the twin plant by synchronizing faulty and non-faulty behaviors. No experimental results are provided.

In Jéron et al (2006), the patterns are defined as automata. The diagnosability question is similar to the one investigated in this paper: is it possible to always assert

that the system *matched* a pattern after a finite sequence of observations. As opposed to our proposal, the representation of the patterns in Jéron et al (2006) is not succinct: a pattern must explicitly define the set of behaviors that the system should match based on the whole event set of the system that makes the pattern totally system-dependent. As in Jiang and Kumar (2004), the authors propose an *ad hoc* algorithm that builds the Kripke structure of the twin plant by successive synchronizations. Here also, no experimental results are provided. In Ye and Dague (2012), the same problem is solved in a decentralized manner by aggregating pattern recognizers.

Our contribution is to bring the notion of patterns introduced in Jéron et al (2006) into the Labeled Prioritized Petri net formalism and then extend previous works on fault class diagnosability on Petri nets to more sophisticated faulty behaviors. With the help of this formalism, the patterns that we define are succinct which is more intuitive to model, moreover a pattern can then be analyzed on several systems as soon as the systems generate at least the set of events defined in the pattern. The complex interactions between the pattern and the system models are fully handled by the system-pattern product that we propose. We then update the twin plant method of Jiang et al (2001) by building a new twin plant LLPPN and apply a model-checking method on it by updating the methods of Cimatti et al (2003) and Grastien (2009).

# 3 Modeling

This paper addresses the diagnosability analysis of behavioral patterns. A pattern is said to be diagnosable in a system if its occurrence can always be diagnosed with certainty based on a finite set of observations. Patterns aim at modeling complex faulty behaviors or any normal behavior of interest. This section presents the modeling framework of the pattern diagnosability problem.

To have a complete characterization of the diagnosability problem, we use Labeled Petri nets with transition *priorities* (Berthomieu et al (2006)). This formalism extends the classical Labeled Petri net. The use of priorities is a convenient way to manage transition conflicts in a model and then to enhance the expressiveness of the model. In particular, transition *priorities* will be used here to overcome the problem of modeling the interactions between the system and a pattern. This section first presents the formalism, called *L-type Labeled Prioritized Petri Nets*, that is used throughout this paper to solve the pattern diagnosability problem. Then the modeling of the system and the pattern are presented.

## 3.1 L-type Labeled Prioritized Petri Nets

**Definition 1 (LLPPN)** *An L-type Labeled Prioritized Petri Net (LLPPN for short) is a 7-uple* $N = \langle P, T, A, \succ, \ell, \Sigma, Q \rangle$ *where:*

- $P$ *is a finite set of nodes called places;*
- $T$ *is a finite set of nodes called transitions and* $P \cap T = \varnothing$*;*
- $A \subseteq (P \times T) \cup (T \times P)$ *is a binary relation that models the arcs between the places and the transitions;*

- *the priority relation* $\succ \subseteq T^2$ *is a binary relation that is:*
  - *transitive* ($t_1 \succ t_2 \wedge t_2 \succ t_3 \Longrightarrow t_1 \succ t_3$)*;*
  - *not reflexive* ($\forall t \in T, t \not\succ t$)*;*
  - *and not symmetrical* ($\forall t_1, t_2 \in T, t_1 \succ t_2 \Longrightarrow t_2 \not\succ t_1$)*;*
- $\Sigma$ *is a finite set of transition labels (events);*
- $\ell : T \to \Sigma$ *is the transition labeling function;*
- $Q$ *is the set of final markings.*

The state of an LLPPN is defined as a *marking*. A marking $M$ is a function $M : P \to \mathbb{N}$ which maps any place of the net to a number of tokens contained in this place. A marking $M$ can also be characterized as a multiset $\mathscr{M}_S(M)$ of places. For any place $p$ of $P$ such that $M(p) = k$, there is exactly $k$ copies of $p$ in $\mathscr{M}_S(M)$. For the sake of simplicity, as both representations are equivalent, the notation $M$ denotes either the function or the corresponding multiset depending on the context. A *marked LLPPN* is a couple $\langle N, M_0 \rangle$ also denoted as the 8-tuple $\langle P, T, A, \succ, \ell, \Sigma, Q, M_0 \rangle$. The *preset* $\text{pre}(n)$ of a node $n$ is the set of nodes $\text{pre}(n) = \{n' \in P \cup T : (n', n) \in A\}$ and the *postset* $\text{post}(n)$ of a node $n$ is the set of nodes $\text{post}(n) = \{n' \in P \cup T : (n, n') \in A\}$. By construction the preset and postset of a transition (respectively a place) are two sets of places (respectively two sets of transitions). An LLPPN is a classical Petri Net formalism with some additional constraints or pieces of information.

Static priorities between transitions are introduced (relation $\succ$) to manage transition conflicts. Priorities add expressiveness to the Petri net and modify the model's semantics. Indeed a transition cannot be fired if a transition with higher priority is firable at the same time: a net transition $t$ is *firable* from a given marking $M$ if and only if $(\forall p \in \text{pre}(t), M(p) > 0) \wedge (\forall t' \in T, t' \succ t \Longrightarrow \exists p' \in \text{pre}(t'), M(p') = 0)$. This firing rule ensures that any firable transition $t$ is firable in the usual sense (when $\succ = \varnothing$) but it avoids the choice of firing transitions with a lower priority that are also firable in the usual sense.

Firing a transition $t$ consumes a token from any place in the preset and adds a token in any place of the postset. From a marking $M$, the fire of a transition $t$ leads to the new marking $M'$ such that $M' = M \setminus \text{pre}(t) \cup \text{post}(t)$, which is denoted $M \xrightarrow{t} M'$. A marking $M$ is *reachable* in a marked LLPPN if it exists a sequence of firable transitions $M_0 \xrightarrow{t_1} M_1 \xrightarrow{t_2} \ldots \xrightarrow{t_k} M$, also denoted $M_0 \xrightarrow{r} M$ where $r$ is the sequence $t_1 t_2 \ldots t_k$, the sequence $r$ is called a run of the net. For a given LLPPN $N$ and a given marking $M$, the set of reachable markings from $M$ is denoted: $R(N, M)$.

An LLPPN also contains a transition labeling function $\ell$ that maps any transition to a label of a set $\Sigma$, this is the way to associate physical events with transitions. Last but not least, an LLPPN has a set of final markings $Q$. The set $Q$ is the way to define the language that is associated with an LLPPN (L-type language of Peterson (1977)).

Let $r = t_1 \ldots t_k \in T^*$ and let $\ell(r)$ denote the sequence $\ell(t_1) \ldots \ell(t_k) \in \Sigma^*$, the language generated by a given LLPPN is defined as follows:

**Definition 2 (Language of an LLPPN)** *The language generated by an LLPPN* $N = \langle P, T, A, \succ, \ell, \Sigma, Q, M_0 \rangle$ *is:*

$$\mathscr{L}(N) = \{\ell(r) : r \in T^* \wedge M_0 \xrightarrow{r} M \wedge M \in Q\}.$$

## 3.2 System model

We propose in this paper to model a system over the set of events $\Sigma$ as an LLPPN $\Theta = \langle P, T, A, \succ, \ell, \Sigma, Q, M_0 \rangle$. As introduced in Sampath et al (1995), systems generate prefix-closed regular languages: for any run $r$ of the system, any of its prefix $r' : r = r'r''$ is indeed a run. It follows that such a system can be represented by a bounded LLPPN (regular language) and with a set of final markings corresponding to the set of reachable markings $Q = R(\Theta, M_0)$ (prefix-closed language). The set of events $\Sigma$ generated by the system is partitioned into two subsets: $\Sigma_o$ is the set of observable events and $\Sigma_u$ the set of unobservable ones.

**Definition 3 (System model)** *The model of a system is an LLPPN* $\Theta = \langle P, T, A, \succ, \ell, \Sigma_o \cup \Sigma_u, Q, M_0 \rangle$ *such that:*

- *$Q$ is the set of reachable markings $R(\Theta, M_0)$;*
- *the Petri net is bounded ($\exists n \in \mathbb{N}^+ : \forall M \in R(\Theta, M_0), \forall p \in P, M(p) \leq n$).*

In this paper, we also set two classical assumptions about the system model.

1. *The system has no deadlock.* From any reachable marking of the system there is always at least one firable transition.
2. *The reachability graph of the system has no cycle of unobservable events*. In other words, from any reachable marking of the system, an observable event will always occur after a finite sequence of unobservable events, the observability of the system is live.

Figure 1 presents an example of such a model $\Theta_1$. Bold events **c** and **d** are the observable events of the system whereas the events $\{a, b, e, f\} = \Sigma_u$ are the unobservable events. The dashed line represents the priority between the transitions of event $b$ and $a$. It ensures that the transition labeled with $b$ is always fired before the one labeled with $a$ in a marking $M$ such that $M(p_0) \geq 1 \wedge M(p_1) \geq 1$ (as the initial marking of the system).

## 3.3 Pattern model

The problem of detecting and diagnosing faults in a DES was originally described based on a system modeled with an explicit set of fault events (see Sampath et al (1995)). Patterns are a way to extend the notion of fault events by introducing more complex faulty behaviors or any normal behaviors of interest (Jéron et al (2006)). This notion of pattern requires the definition of a general and flexible framework allowing to design a large spectrum of diagnosis problems including simple faulty event, multiple

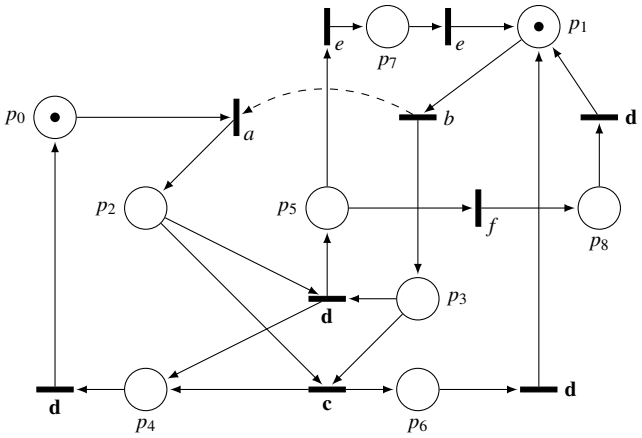


Figure 1: System example $\Theta_1$ with $\Sigma_o = \{c, d\}$.

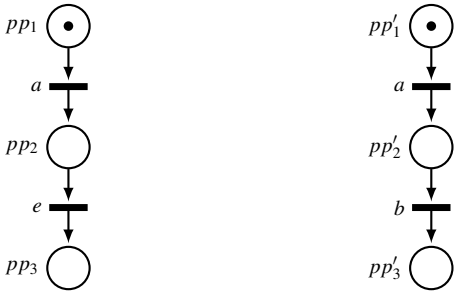


Figure 2: Two sequence patterns $\Omega_{a\to e}$ and $\Omega_{a\to b}$ with $Q_{\Omega_{a\to e}} = \{M : M(pp_3) = 1\}$ and $Q_{\Omega_{a\to b}} = \{M : M(pp'_3) = 1\}$.

faults, fault repetitions....and more generally any behavior of interest (faulty or not). The use of fault pattern helps to clearly separate the system and the diagnosis objectives (see Zanella and Lamperti (2004)). The reusability is then reinforced to tackle new problems. For complex patterns this separation principle improves the applicability. Indeed, a modification of the system model in order to merge with the pattern is a tricky task that might even be impracticable in real cases. This would suppose to search in the system model for all the traces where the pattern has occurred and to modify them accordingly to the pattern model. Moreover, the system model would have to be modified each time a new pattern has to be analyzed.

**Definition 4 (Pattern)** *A pattern $\Omega$ is an LLPPN*

$$\Omega = \langle P, T, A, \succ, \ell, \Sigma, Q, M_0 \rangle$$

*such that*

1. *(initialization) $M_0 \notin Q$ i.e. the language of the pattern does not contain $\varepsilon$;*
2. *(well-formed) from any reachable marking $M$ of $R(\Omega, M_0)$ there exists a sequence of transitions $r \in T^*$ such that $M \xrightarrow{r} M'$ and $M' \in Q$;*
3. *(event determinism) from any reachable marking $M$ of $R(\Omega, M_0)$ there is no event $e \in \Sigma$ that labels more than one firable transition;*
4. *(stability) $\forall M \in Q, \forall M' : (\exists t \in T, M \xrightarrow{t} M') \Rightarrow M' \in Q$;*
5. *(no priority) $\succ = \varnothing$.*

Firstly, Figure 2 presents two similar patterns that will be used as running examples. The set of final markings of $\Omega_{a \to e}$ only contains one reachable marking, the one with $pp_3 = 1$. The language associated to $\Omega_{a \to e}$ is $\mathscr{L}(\Omega_{a \to e}) = \{ae\}$ that is the sequence of event $a$ followed by $e$.

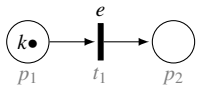


Figure 3: Pattern ($\Omega_1$) about $k$ occurrences of an event $e$ with the final markings $Q_{\Omega_1} = \{M : M(p_2) = k\}$.

Figure 3 shows a pattern $\Omega_1$ representing $k$ occurrences of the event $e$ ($k \in \mathbb{N}^+$). Figure 4 presents the LLPPN model of a pattern $\Omega_2$ representing the occurrence of $n$ events $\{e_1, e_2, \ldots, e_n\}$ in no fixed order. The language $\mathscr{L}(\Omega_1)$ generated by pattern $\Omega_1$ is $\mathscr{L}(\Omega_1) = \{e^k\}$ that is the sequence of exactly $k$ events $e$. The language $\mathscr{L}(\Omega_2)$ gathers any possible interleavings of the events $\{e_1, \ldots, e_n\}$, the cardinality of the language is $|\mathscr{L}(\Omega_2)| = n!$.

In this way, a pattern can easily be modeled directly disregarding any notion of system language contrarily to Jéron et al (2006) and to our previous work (Gougam

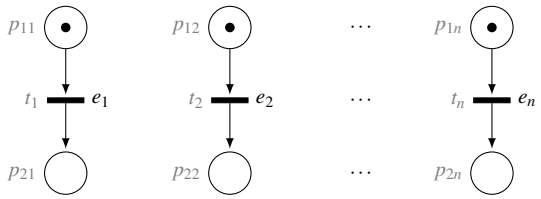


Figure 4: Pattern ($\Omega_2$) about the occurrence of $n$ events in no fixed order with the final markings $Q_{\Omega_2} = \{\{p_{21}, p_{22}, \ldots, p_{2n}\}\}$.

et al (2013)). In Jéron et al (2006) a pattern is modeled as an automaton that must describe the full event set of the system and must be complete (from any automaton's state, any event of the system must be associated with exactly one output transition of the state). In Gougam et al (2013), the presented patterns require completeness so they are also more dependent from the system. Definition 4 proposes a more concise way to model pattern that also reinforces the separation between the system and the diagnosis objectives. The way to handle the complex interactions between a system and a pattern is then not part of the modeling phase. It is managed by a specific product operator that is defined in Section 5.2.

# 4 Pattern diagnosability: problem statement

Intuitively speaking, pattern diagnosis is the problem of determining the set of patterns that have effectively occurred and that could explain a sequence of observations. Associated with this problem is the diagnosability problem. Checking whether a pattern is diagnosable over a system consists in checking whether it is always possible for a diagnostic function to assert with certainty that a run of a pattern effectively occurred during one run of the system based on the produced observations. The formal definition of the diagnosis and the diagnosability problem then depend on the notion of pattern occurrence that is defined below.

A pattern $\Omega$ has occurred in a run $r$ of the system if it is possible to extract from the event sequence $\rho = \ell(r) \in \mathscr{L}(\Theta)$ an ordered set of events that is a word of the language $\mathscr{L}(\Omega)$ generated by the pattern, in this case we say that the sequence $\rho$ *matches* the pattern $\Omega$ and is denoted $\rho \Subset \Omega$ (we also say for convenience that the run $r$ matches the pattern $\Omega$ if $\ell(r) \Subset \Omega$).

**Definition 5 (Matching of a pattern)** *A sequence $\rho \in \Sigma^*$ matches a pattern $\Omega$ ($\rho \Subset \Omega$) if there exists at least a sequence $\sigma$ of $\mathscr{L}(\Omega)$ that is matched by $\rho$ ($\rho \Subset \sigma$).*

Definition 5 relies on the definition of sequence matching.

**Definition 6 (Sequence matching)** *A sequence $\rho \in \Sigma^*$ matches another sequence $\sigma \in \Sigma^*$, denoted $\rho \Subset \sigma$, if:*

- *$\sigma$ is empty ($\sigma = \varepsilon$); or*

- $\sigma = s.\sigma_1, s \in \Sigma, \sigma_1 \in \Sigma^*$ *and there exist two sequences* $\rho_0, \rho_1 \in \Sigma^*$ *such that:*
  1. $\rho = \rho_0 s \rho_1$*;*
  2. $\rho_1 \ni \sigma_1$.

A sequence $\rho$ matches a sequence $\sigma$ if $\rho$ contains any event of the sequence $\sigma$ and these events occur in the same order as in $\sigma$. Obviously $\rho$ can contain several collections of events that gather the events of $\sigma$ in the same order; in this case $\sigma$ occurs several times in $\rho$. The pattern $\rho$ matches $\sigma$ if $\sigma$ occurs at least once. Figure 5 illustrates how the sequence $\rho = bcaabcac$ matches the sequence $\sigma = acc$. Definition 6 is recursively applied three times till $\sigma_3 = \varepsilon$. In this example, $\sigma$ occurs twice in $\rho$. Back to the example of Figure 1, the system $\Theta_1$ matches both the patterns $\Omega_{a\to e}$ and

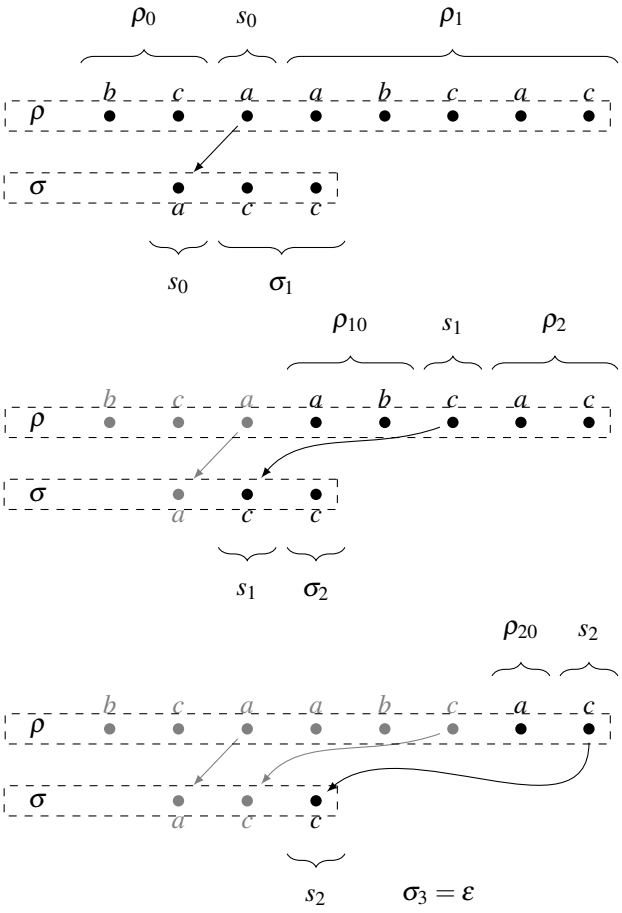


Figure 5: How $\rho = bcaabcac$ matches $\sigma = acc$: $\rho \ni \sigma$.

$\Omega_{a\to b}$ of Figure 2. The sequence $ba\mathbf{d}e$ is a possible run of $\Theta_1$ and $ba\mathbf{d}e \ni ae$. $\Theta_1$ also matches $\Omega_{a\to b}$ as the sequence $ba\mathbf{d}eeb\mathbf{d}$ is a possible run of $\Theta_1$ and $ba\mathbf{d}eeb\mathbf{d} \ni ab$.

Based on the notion of pattern, the classical fault diagnosis problem on DESs can be generalized as proposed in Jéron et al (2006). Let $\mathscr{P}_{\Sigma_1\to\Sigma_2}\colon \Sigma_1^* \to \Sigma_2^*$ denote the projection function of any sequence of $\Sigma_1^*$ to a sequence of $\Sigma_2^*$ defined by:

$$\begin{cases} \mathscr{P}_{\Sigma_1\to\Sigma_2}(\varepsilon) = \varepsilon, \\ \mathscr{P}_{\Sigma_1\to\Sigma_2}(\rho.s) = \mathscr{P}_{\Sigma_1\to\Sigma_2}(\rho) & \text{if } \rho \in \Sigma_1^*, s \notin \Sigma_2, \\ \mathscr{P}_{\Sigma_1\to\Sigma_2}(\rho.s) = \mathscr{P}_{\Sigma_1\to\Sigma_2}(\rho).s & \text{if } \rho \in \Sigma_1^*, s \in \Sigma_2. \end{cases}$$

For a given pattern $\Omega$, the diagnosis problem consists in defining an $\Omega$-diagnoser function that takes as input a sequence of observable events and that produces one of the three symbols $\{\Omega{-}\mathit{faulty}, \Omega{-}\mathit{safe}, \Omega{-}\mathit{ambiguous}\}$ (Pencolé et al (2006)).

**Definition 7 ($\Omega$-diagnoser)** *Let $\Theta$ be the model of a system based on the set of events $\Sigma = \Sigma_u \cup \Sigma_o$, an $\Omega$-diagnoser is a function*

$$\Delta_\Omega : \Sigma_o^* \to \{\Omega{-}\mathit{faulty}, \Omega{-}\mathit{safe}, \Omega{-}\mathit{ambiguous}\}$$

*such that:*

- $\Delta_\Omega(\sigma) = \Omega{-}\mathit{faulty}$ *if for any run* $\rho \in \mathscr{L}(\Theta)$ *that is consistent with* $\sigma$ *(i.e.* $\mathscr{P}_{\Sigma\to\Sigma_o}(\rho) = \sigma$*),* $\rho \ni \Omega$*;*
- $\Delta_\Omega(\sigma) = \Omega{-}\mathit{safe}$ *if for any run* $\rho \in \mathscr{L}(\Theta)$ *that is consistent with* $\sigma$*,* $\rho \notin \Omega$*;*
- $\Delta_\Omega(\sigma) = \Omega{-}\mathit{ambiguous}$ *otherwise.*

Considering now a set of patterns $\Omega_1, \ldots, \Omega_n$, the diagnoser function of a system can be defined as $\Delta : \Sigma_o^* \to \prod_{i=1}^n \{\Omega_i{-}\mathit{faulty}, \Omega_i{-}\mathit{safe}, \Omega_i{-}\mathit{ambiguous}\}$ such that $\Delta(\sigma) = (\Delta_{\Omega_1}(\sigma), \ldots, \Delta_{\Omega_n}(\sigma))$.

As an example, consider first the following observable sequence $\sigma_1 = \mathbf{dd}$ from the system $\Theta_1$ (Figure 1) then $\Delta_{\Omega_{a\to e}}(\sigma_1) = \Omega_{a\to e}{-}\mathit{ambiguous}$ and $\Delta_{\Omega_{a\to b}}(\sigma_1) = \Omega_{a\to b}{-}\mathit{ambiguous}$. In both cases, it is possible to find in $\Theta_1$ two runs which are consistent with $\sigma_1$ (their observable projection on $\Sigma_o$ is exactly $\sigma_1$), one run matches the pattern but not the other one. For instance consider the run $ba\mathbf{d}eeb\mathbf{d}$ of $\Theta_1$ that matches both the pattern $\Omega_{a\to e}$ and the pattern $\Omega_{a\to b}$ and the run $ba\mathbf{d}f\mathbf{d}$ that does not match any of the two patterns.

Consider now the observable sequence $\sigma_2 = \mathbf{cddc}$, then $\Delta_{\Omega_{a\to e}}(\sigma_2) = \Omega_{a\to e}{-}\mathit{safe}$ and $\Delta_{\Omega_{a\to b}}(\sigma_1) = \Omega_{a\to b}{-}\mathit{faulty}$. There is no run in $\Theta_1$ consistent with $\sigma_2$ that contains an event $e$ so none of them can match $\Omega_{a\to e}$. On the other hand, any run consistent with $\sigma_2$ starts with $ba$ and must contain one more $b$ so any of them matches $\Omega_{a\to b}$.

The definition of the pattern diagnosability also relies on the notion of pattern matching by a system sequence. Let $\mathscr{L}$ be a language, let $\rho \in \mathscr{L}$, we denote by $\mathscr{L}/\rho$ the set of continuations of $\rho$ in $\mathscr{L}$:

$$\mathscr{L}/\rho = \{\rho' : \rho.\rho' \in \mathscr{L}\}.$$

**Definition 8 (Pattern-diagnosability)** *A pattern $\Omega$ is diagnosable over a system $\Theta$ if:*

$$\exists n \in \mathbb{N}, \forall \rho \in \mathscr{L}(\Theta), \rho \ni \Omega \Longrightarrow$$
$$(\forall \zeta \in \mathscr{L}(\Theta)/\rho, \|\mathscr{P}_{\Sigma\to\Sigma_o}(\zeta)\| \geq n \Longrightarrow$$
$$(\forall \rho' \in \{\rho'' : \rho'' \in \mathscr{L}(\Theta) \wedge \mathscr{P}_{\Sigma\to\Sigma_o}(\rho'') = \mathscr{P}_{\Sigma\to\Sigma_o}(\rho.\zeta)\}, \rho' \ni \Omega)).$$

As soon as the current run $r$ of the system matches a pattern ($\rho = \ell(r) \ni \Omega$), it is sufficient to wait for a finite observable sequence $\sigma$ to ensure that any possible run $r'$ of the system that generates the same observable sequence $\mathscr{P}_{\Sigma\to\Sigma_o}(\ell(r')) = \mathscr{P}_{\Sigma\to\Sigma_o}(\rho)\sigma$ also matches $\Omega$. In other words, as soon as the current run $r$ of the system matches a pattern $\Omega$, the $\Omega$-diagnoser returns either $\Omega-$*faulty* or $\Omega-$*ambiguous*. If the result is $\Omega-$*ambiguous*, pattern diagnosability then guarantees that waiting for a supplementary finite set of observable events will lead the $\Omega$-diagnoser to definitely return $\Omega-$*faulty*.

**Definition 9 (System diagnosability)** *Given a set of patterns, a system is diagnosable if each pattern is diagnosable over the system.*

Back to the patterns of Figure 2, the pattern $\Omega_{a\to e}$ is not diagnosable in the system $\Theta_1$. Consider for instance the run $\rho = ba\mathbf{d}e$ that matches $\Omega_{a\to e}$ ($\rho \ni \Omega_{a\to e}$). From this run $\rho$, there exists a continuation $\zeta$ that is arbitrarily long (for instance $\zeta = (e\mathbf{d}ba\mathbf{d}e)^k$ for any given $k \in \mathbb{N}$), so that $\rho\zeta \in \mathscr{L}(\Theta_1)$ is an arbitrarily long run of $\Theta_1$. The observable projection of $\zeta$ is arbitrarily long as well: $\mathscr{P}_{\Sigma\to\Sigma_o}(\zeta) = \mathbf{d}^{2k}$. Now consider the sequence $\rho' = ba\mathbf{d}.(f\mathbf{d}\mathbf{d}ba\mathbf{d})^{k'}$ for any given $k' \in \mathbb{N}$, $\rho'$ is also a run of $\Theta_1$. Consider now any couple $(k,k')$ such that $2k = 3k'$ (there is an infinite number of such couples), then $\mathscr{P}_{\Sigma\to\Sigma_o}(\rho') = \mathbf{d}.(\mathbf{d})^{3k'} = \mathbf{d}.(\mathbf{d})^{2k} = \mathscr{P}_{\Sigma\to\Sigma_o}(\rho.\zeta)$. Therefore, there is at least a continuation $\zeta$ of $\rho$ that is arbitrarily long for which there always exists in $\Theta_1$ a run $\rho' \not\ni \Omega_{a\to e}$ that has the same observable projection as $\rho.\zeta$. As opposed to $\Omega_{a\to e}$, the second pattern $\Omega_{a\to b}$ is diagnosable. In $\Theta_1$, it can be noticed that any run $\rho$ such that $|\rho| \geq 8$ matches $\Omega_{a\to b}$. Therefore, for any run $\rho$ that matches $\Omega_{a\to b}$, for any continuation $\zeta$ with $|\mathscr{P}_{\Sigma\to\Sigma_o}(\zeta)| \geq 8$, any $\rho'$ such that $\mathscr{P}_{\Sigma\to\Sigma_o}(\rho') = \mathscr{P}_{\Sigma\to\Sigma_o}(\rho\zeta)$ is such that $|\rho'| \geq 8$ so $\rho' \ni \Omega_{a\to b}$, hence the diagnosability of $\Omega_{a\to b}$ in $\Theta_1$.

Analyzing the diagnosability of a system relatively to a set of patterns consists in analyzing the diagnosability of each pattern. Next section presents how to analyze the diagnosability of a pattern over a system.

# 5 Pattern diagnosability analysis

## 5.1 Method overview

Let us start with a brief overview of the proposed method. The pattern diagnosability method relies on four main steps.

1. *Searching for the system's sequences matching the pattern.* These sequences can be obtained by combining the system and pattern models into a new model where:

   - any evolution of the pattern is always synchronized with an evolution of the system;
   - the system might evolve while the pattern does not.

   This combined model relies on a specific operator called *system-pattern product*. This product is asymmetric.

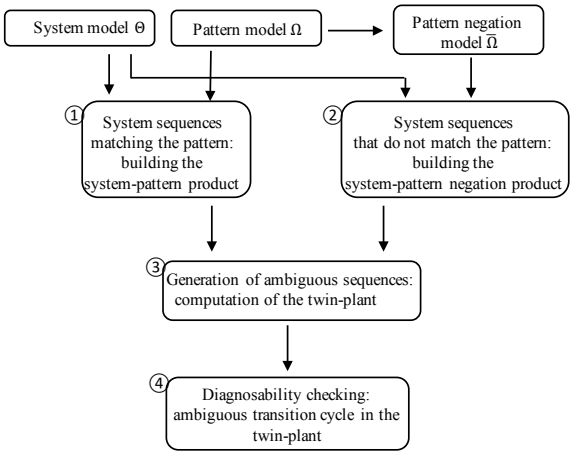


Figure 6: General principle of the diagnosability analysis of a pattern $\Omega$ over a system $\Theta$.

2. *Searching for the system's sequences that do not match the pattern.* This step is similar to the previous one but based on the negation of the pattern. The negation of the pattern is simply given by the pattern with a different set of final markings.

3. *Comparing the sets of sequences obtained in steps* 1 *and* 2 *to obtain the set of ambiguous sequences*. Two sequences are *ambiguous* if they have the same observable projection and only one of them matches the pattern. This step is an adaptation of the classical method to analyze the diagnosability of single faults in a system, independently introduced in Yoo and Lafortune (2002) and Jiang et al (2001). It is based on the computation of a *twin plant* that compares system's sequences matching the pattern and its negation that have the same observable projection. Here, the construction of the twin plant relies on the classical synchronized product of Petri nets where synchronizations are applied on observable events only.

4. *Searching for ambiguous sequences with an infinite number of observable events.* If such sequences exist then the pattern is not diagnosable over the system. In our framework, looking for such ambiguous sequences means to look for a transition cycle in the twin plant. Looking for transition cycles in the twin plant is performed by a model-checking technique (Section 6).

Figure 6 summarizes the overall principle to analyze the diagnosability of a pattern $\Omega$ over a system $\Theta$. Next section presents formally each step of the method in details.

## 5.2 Searching for the system's sequences matching the pattern

This stage aims to determine system's runs where a given pattern has occurred i.e. system's sequences matching the pattern. For this, we introduce a specific product operator to combine the system $\Theta$ and the pattern $\Omega$ in order to retain in the system any run that effectively matches the pattern: this product is called the system-pattern product and is denoted: $\Theta \ltimes \Omega$. Informally speaking, $\Theta \ltimes \Omega$ consists of the union of the places of $\Theta$ and $\Omega$. The set of transitions is composed of the set of transitions of $\Theta$ and a set of synchronized and prioritized transitions replacing the transitions from $\Omega$.

**Definition 10 (system-pattern product)** *Let* $\Theta = \langle P_\Theta, T_\Theta, A_\Theta, \succ_\Theta, \ell_\Theta, \Sigma_\Theta, Q_\Theta, M_{\Theta 0}\rangle$, $\Omega = \langle P_\Omega, T_\Omega, A_\Omega, \varnothing, \ell_\Omega, \Sigma_\Omega, Q_\Omega, M_{\Omega 0}\rangle$ *be respectively a system and a pattern such that* $P_\Theta \cap P_\Omega = \varnothing$, $T_\Theta \cap T_\Omega = \varnothing$ *and* $\Sigma_\Omega \subseteq \Sigma_\Theta$, *the system-pattern product* $\Theta \ltimes \Omega$ *is the LLPPN* $\langle P, T, A, \succ, \ell, \Sigma, Q, M_0\rangle$ *defined as follows.*

- $P = P_\Theta \cup P_\Omega$.
- $T = T_\Theta \cup T_s$ *such that*
  - $T_s = \bigcup_{l \in \Sigma_\Omega} \{t_\Theta \| t_\Omega : \exists t_\Theta \in T_\Theta \wedge \exists t_\Omega \in T_\Omega \wedge \ell_\Theta(t_\Theta) = \ell_\Omega(t_\Omega) = l\}$ *is the set of synchronized transitions, the transition denoted* $t_\Theta \| t_\Omega$ *resulting from the synchronization of* $t_\Theta \in T_\Theta$ *and* $t_\Omega \in T_\Omega$.
- $A = A_\Theta \cup A_s$ *such that*

$$\begin{aligned} A_s = &\{(p,t) : p \in P_\Theta, t = t_\Theta \| t_\Omega, (p, t_\Theta) \in A_\Theta\} \\ &\cup \{(p,t) : p \in P_\Omega, t = t_\Theta \| t_\Omega, (p, t_\Omega) \in A_\Omega\} \\ &\cup \{(t,p) : p \in P_\Theta, t = t_\Theta \| t_\Omega, (t_\Theta, p) \in A_\Theta\} \\ &\cup \{(t,p) : p \in P_\Omega, t = t_\Theta \| t_\Omega, (t_\Omega, p) \in A_\Omega\}. \end{aligned}$$

- $\succ = \succ_\Theta \cup \succ_s$, *with:*

$$\begin{aligned} \succ_s = &\{(t_s, t_\Theta) : t_s = t_\Theta \| t_\Omega \in T_s\} \cup \\ &\bigcup_{t_s = t_\Theta \| t_\Omega \in T_s} (\{(t_s, t) : (t_\Theta, t) \in \succ_\Theta\} \cup \{(t, t_s) : (t, t_\Theta) \in \succ_\Theta\}) \\ &\cup \{(t_\Theta^1 \,\|\, t_\Omega^1, t_\Theta^2 \,\|\, t_\Omega^2) : t_\Theta^1 \,\|\, t_\Omega^1 \in T_s, t_\Theta^2 \,\|\, t_\Omega^2 \in T_s, (t_\Theta^1, t_\Theta^2) \in \succ_\Theta\}. \end{aligned}$$

- $\forall t = t_\Theta || t_\Omega \in T_s, \ell(t) = \ell_\Theta(t_\Theta), \forall t \in T \cap T_\Theta, \ell(t) = \ell_\Theta(t)$.
- $\Sigma = \Sigma_\Theta$.
- $Q = \{M_\Theta \cup M_\Omega : M_\Theta \in Q_\Theta \wedge M_\Omega \in Q_\Omega\}$.
- $M_0 = M_{\Theta 0} \cup M_{\Omega 0}$.

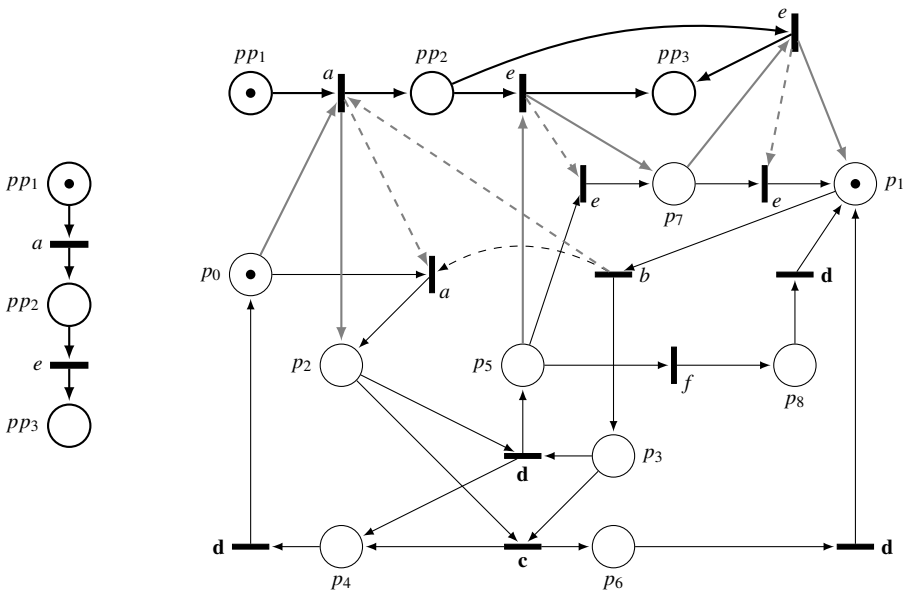


Figure 7: On the left, the sequence pattern $\Omega_{a\to e}$. On the right the system-pattern product between the system of Figure 1 and $\Omega_{a\to e}$.

Intuitively, the pattern $\Omega$ is applied as a filter on the system $\Theta$ by using the product $\Theta \ltimes \Omega$. Any transition of label $e$ in the system is synchronized with a transition of the pattern $\Omega$ labeled with the same event $e$. The product is asymmetric in the sense that $\Theta \ltimes \Omega$ contains all the transitions of $\Theta$ whereas the transitions of $\Omega$ with such an event $e$ have been replaced by a set of synchronized transitions with $\Theta$. Given the current marking of $\Theta \ltimes \Omega$ and the occurrence of a new event $e$, two cases can happen. If the current marking of $\Theta \ltimes \Omega$ enables a synchronized transition labeled with $e$, as it is prioritized, it will be triggered (the event $e$ is therefore part of the pattern), otherwise only a transition from the system $\Theta$ can be triggered in $\Theta \ltimes \Omega$ (the system keeps running by producing the event $e$ but $e$ is not part of the pattern).

Figure 7 illustrates the system-pattern product $\Theta_1 \ltimes \Omega_{a\to e}$ modeled in Figure 1. Events $a$ and $e$ are parts of the system and the pattern. For each of the three transitions of the system labeled with an event $a$ or $e$, a synchronized and prioritized transition is created in $\Theta_1 \ltimes \Omega_{a\to e}$. The sequence $ba\mathbf{d}e$ is generated by a possible run of the system $\Theta_1$ of Figure 1 that obviously matches the pattern $\Omega_{a\to e}$. In $\Theta_1 \ltimes \Omega_{a\to e}$, the sequence $ba\mathbf{d}e$ is associated with the run $\{p_0, p_1, pp_1\} \xrightarrow{b} \{p_0, p_3, pp_1\} \xrightarrow{a} \{p_2, p_3, pp_2\} \xrightarrow{\mathbf{d}} \{p_4, p_5, pp_2\} \xrightarrow{e} \{p_4, p_7, pp_3\}$ that leads to one of its final markings (the reachable markings $M$ such that $M(pp_3) = 1$). By construction of the product, the first occurrence of $a$ and the first occurrence of $e$ are part of the pattern. The use of priorities here forbid the transition $\{p_0, p_3, pp_1\} \xrightarrow{a} \{p_2, p_3, pp_1\}$ to be a run of the product $\Theta_1 \ltimes \Omega_{a\to e}$.

With the help of this product, it is possible to identify in the system $\Theta$ the set of

runs where the pattern $\Omega$ has occurred. The following results assert this intuition.

Let $\Theta_\Omega = \Theta \ltimes \Omega$ and $M_{\Theta_\Omega 0}$ be the initial marking of $\Theta_\Omega$. Let $M_{\Theta 0}$ (resp. $M_{\Omega 0}$) be the initial marking of $\Theta$ (resp. $\Omega$).

**Lemma 1** *For any reachable marking $M \in R(\Theta_\Omega, M_{\Theta_\Omega 0})$ of $\Theta_\Omega$, there exists a reachable marking $M_1 \in R(\Theta, M_{\Theta 0})$ and a reachable marking $M_2 \in R(\Omega, M_{\Omega 0})$ such that $M = M_1 \cup M_2$.*

Proof: see Appendix.

Lemma 1 simply states that any run of the LLPPN $\Theta_\Omega$ represents a run of $\Theta$ and a run of the pattern $\Omega$. Based on this first result, we can now present the fundamental result about the system-pattern product $\ltimes$.

**Theorem 1** *Let $\Theta$ be the LLPPN of a system over the alphabet $\Sigma$ and $\Omega$ be the LLPPN of a pattern:*

$$\mathscr{L}(\Theta \ltimes \Omega) = \{\rho \in \mathscr{L}(\Theta) : \rho \ni \Omega\}.$$

Proof: see Appendix.

Theorem 1 is the fundamental property of the operator $\ltimes$. The set of runs that are characterized by the language $\mathscr{L}(\Theta \ltimes \Omega)$ are exactly the set of runs that match the pattern $\Omega$. Theorem 1 asserts, for example, that the language of the product between the system $\Theta_1$ and the pattern $\Omega_{a\to e}$ is such that:

$$\mathscr{L}(\Theta_1 \ltimes \Omega_{a\to e}) = \mathscr{L}(\Theta_1) \cap ((\Sigma \setminus \{a\})^*.a.(\Sigma \setminus \{e\})^*.e.\Sigma^*),$$

where $\Sigma = \{a,b,c,d,e,f\}$ is the set of events of $\Theta_1$.

## 5.3 Searching for the system's sequences that do not match the pattern

By considering the negation of the pattern, the objective is to determine in the system any run that does not match the pattern. Basically speaking, the negation of $\Omega$ is the LLPPN $\Omega$ associated with the complementary set of the final markings of $\Omega$.

**Definition 11** *The negation $\overline{\Omega}$ of the pattern $\Omega = \langle P, T, A, \succ, \ell, \Sigma, Q, M_0\rangle$ is the LLPPN $\overline{\Omega} = \langle P, T, A, \succ, \ell, \Sigma, \overline{Q}, M_0\rangle$ where:*

$$\overline{Q} = R(\Omega, M_0) \setminus Q.$$

For example, back to the pattern $\Omega_{a\to e}$, the set of reachable markings of $\Omega_{a\to e}$ is $R(\Omega_{a\to e}, \{pp_1\}) = \{\{pp_1\}, \{pp_2\}, \{pp_3\}\}$ so the final markings of $\overline{\Omega}_{a\to e}$ are $\overline{Q_{\Omega_{a\to e}}} = \{\{pp_1\}, \{pp_2\}\}$.

This set of final markings leads to the language

$$\mathscr{L}(\overline{\Omega}_{a\to e}) = \{\varepsilon, a\}.$$

In the pattern of Figure 3, $\overline{Q_{\Omega_1}} = \{M \in R(\Omega_1, \{kp_1\}) : M(p_2) < k\}$ expresses that any run generating a sequence of $\mathscr{L}(\overline{\Omega_1})$ contains less than $k$ occurrences of event $e$. Considering now the pattern of Figure 4, the pattern $\Omega_2$ has only one final marking $Q_{\Omega_2} =$

$\{\{p_{21},\ldots,p_{2n}\}\}$ and the corresponding language $\mathscr{L}(\Omega_2)$ gathers the set of permutations of the sequence $e_1 \ldots e_n$. Now, the negation $\overline{\Omega_2}$ has exactly the same net structure but $\overline{Q_{\Omega_2}}$ is any possible reachable marking of $R(\overline{\Omega_2}, M_0)$ except $\{\{p_{21},\ldots,p_{2n}\}\}$. By construction, any word $\sigma \in \mathscr{L}(\overline{\Omega_2})$ is a prefix of a permutation of the sequence $e_1 \ldots e_n$ whose length is strictly lower than $n$.

The only difference between a pattern $\Omega$ and its negation $\overline{\Omega}$ is the definition of its final markings, so it is possible to apply straightforwardly the system-pattern product on $\overline{\Omega}$. We can then introduce the language $\mathscr{L}(\Theta \ltimes \overline{\Omega})$ that actually corresponds to the sequences of the system that do not match the pattern.

**Corollary 1** *Let $\Theta$ be the LLPPN of a system over an alphabet $\Sigma$ and $\Omega$ be the LLPPN of a pattern:*

$$\mathscr{L}(\Theta \ltimes \overline{\Omega}) = \{\rho \in \mathscr{L}(\Theta) : \forall \sigma \in \mathscr{L}(\Omega) : \rho \not\trianglerighteq \sigma\}.$$

Proof: see Appendix.

Corollary 1 asserts, for example, that the language of the product between the system $\Theta_1$ and the pattern negation $\overline{\Omega}_{a\to e}$ is such that:

$$\begin{aligned}\mathscr{L}(\Theta_1 \ltimes \overline{\Omega}_{a\to e}) &= \mathscr{L}(\Theta_1) \cap (\Sigma^* \setminus ((\Sigma \setminus \{a\})^*.a.(\Sigma \setminus \{e\})^*.e.\Sigma^*)) \\ &= \mathscr{L}(\Theta_1) \setminus \mathscr{L}(\Theta_1 \ltimes \Omega_{a\to e}),\end{aligned}$$

where $\Sigma = \{a,b,c,d,e,f\}$ is the set of events of $\Theta_1$.

## 5.4 Generation of the ambiguous sequences by a twin plant

The comparison of system's sequences matching the pattern and its negation that have the same observable projection is based on the so-called *twin plant* technique. In the original contributions (Jiang et al (2001),Yoo and Lafortune (2002)) the twin machine results from the composition of the system model with itself based on the synchronization of transitions that share the same observable event. Checking whether diagnosability holds or not in the system then consists in finding in the twin machine a cycle of critical pairs $(x,x')$ where $x$ is the target state of a faulty sequence in the system model and $x'$ is the target state of a non-faulty sequence in the system model. Such a cycle represents a pair of sequences of the system that are infinite with the same infinite observable behaviors, but only one is faulty. In the current framework, we apply the same principle and define a *twin plant* as the composition of the system-pattern product and the system-pattern-negation product.

### 5.4.1 Synchronized product of LLPPNs

The composition is the classical synchronized product $||$ of Petri Nets, also called transition fusions or concurrent composition (Hack (1975)), that is updated for its use with LLPPNs.

**Definition 12 (synchronized product)** *Let $N_1 = \langle P_1, T_1, A_1, \succ_1, \ell_1, \Sigma_1, Q_1, M_{10}\rangle$ and $N_2 = \langle P_2, T_2, A_2, \succ_2, \ell_2, \Sigma_2, Q_2, M_{20}\rangle$ be two LLPPNs. The synchronized product $N_1 \,||\, N_2$ is the LLPPN $N = \langle P, T, A, \succ, \ell, \Sigma, Q, M_0\rangle$ such that:*

- $P = P_1 \cup P_2$;
- $T = \hat{T}_1 \cup \hat{T}_2 \cup T_s$, *with:*

$$\hat{T}_i = \{t \in T_i : \ell_i(t) \notin \Sigma_1 \cap \Sigma_2\}, \textit{for } i = 1,2$$
$$T_s = \bigcup_{l \in \Sigma_1 \cap \Sigma_2} \{t_1 \,\|\, t_2 : \exists (t_1,t_2) \in (T_1 \times T_2) \wedge \ell_1(t_1) = \ell_2(t_2) = l\};$$

- $A = \hat{A}_1 \cup \hat{A}_2 \cup A_s$, *with:*

$$\begin{aligned}
\hat{A}_1 &= A_1 \setminus [(P_1 \times (T_1 \setminus \hat{T}_1)) \cup ((T_1 \setminus \hat{T}_1) \times P_1)] \\
\hat{A}_2 &= A_2 \setminus [(P_2 \times (T_2 \setminus \hat{T}_2)) \cup ((T_2 \setminus \hat{T}_2) \times P_2)] \\
A_s &= \{(p,t) : p \in P_1, t = t_1 \,\|\, t_2 \in T_s, (p,t_1) \in A_1\} \\
&\quad \cup \{(p,t) : p \in P_2, t = t_1 \,\|\, t_2 \in T_s, (p,t_2) \in A_2\} \\
&\quad \cup \{(t,p) : p \in P_1, t = t_1 \,\|\, t_2 \in T_s, (t_1,p) \in A_1\} \\
&\quad \cup \{(t,p) : p \in P_2, t = t_1 \,\|\, t_2 \in T_s, (t_2,p) \in A_2\};
\end{aligned}$$

- $\succ = \bigcup_{i=1}^{2} \{(t_1,t_2) \in \succ_i : t_1 \in \hat{T}_i, t_2 \in \hat{T}_i\} \cup \succ_s$ *with:*

$$\begin{aligned}
\succ_s = & \\
& \bigcup_{i=1}^{2} \{(t_1 \,\|\, t_2, t_3) : t_1 \,\|\, t_2 \in T_s, (t_i,t_3) \in \succ_i\} \\
& \cup \bigcup_{i=1}^{2} \{(t_3, t_1 \,\|\, t_2) : t_1 \,\|\, t_2 \in T_s, (t_3,t_i) \in \succ_i\} \\
& \cup \bigcup_{i=1}^{2} \{(t_1^1 \,\|\, t_1^2, t_2^1 \,\|\, t_2^2) : t_1^1 \,\|\, t_1^2 \in T_s, t_2^1 \,\|\, t_2^2 \in T_s, (t_1^i, t_2^i) \in \succ_i\};
\end{aligned}$$

- $\ell(t) = \ell_1(t_1)$ *if* $t = t_1 \,\|\, t_2 \in T_s$, $\ell(t) = \ell_1(t)$ *if* $t \in \hat{T}_1$, $\ell(t) = \ell_2(t)$ *if* $t \in \hat{T}_2$;
- $\Sigma = \Sigma_1 \cup \Sigma_2$;
- $Q = \{M_1 \cup M_2 : M_1 \in Q_1 \wedge M_2 \in Q_2\}$;
- $M_0 = M_{10} \cup M_{20}$.

Any run $r$ of $N_1||N_2$ corresponds to a run $r_1$ of $N_1$ synchronized with a run $r_2$ of $N_2$ such that their projection on the set of events $\Sigma_1 \cap \Sigma_2$ is similar:

$$\mathscr{P}_{\Sigma_1 \to \Sigma_1 \cap \Sigma_2}(\ell_1(r_1)) = \mathscr{P}_{\Sigma_2 \to \Sigma_1 \cap \Sigma_2}(\ell_2(r_2)) = \mathscr{P}_{\Sigma_1 \cup \Sigma_2 \to \Sigma_1 \cap \Sigma_2}(\ell(r)).$$

As any final marking of the LLPPN $N_1||N_2$ is the union of a final marking of $N_1$ and a final marking of $N_2$, the next result follows.

**Theorem 2** $\mathscr{L}(N_1 \| N_2) = \{\rho \in (\Sigma_1 \cup \Sigma_2)^* : \mathscr{P}_{\Sigma_1 \cup \Sigma_2 \to \Sigma_1}(\rho) \in \mathscr{L}(N_1) \wedge \mathscr{P}_{\Sigma_1 \cup \Sigma_2 \to \Sigma_2}(\rho) \in \mathscr{L}(N_2)\}$.

Proof: see Appendix.

### 5.4.2 Computation of the twin plant

Now let us apply the synchronized product to compute the twin plant. Firstly, the system $\Theta$ is duplicated as the new LLPPN $\Theta'$ where:

1. any place $p$ of $\Theta$ is renamed $p'$ in $\Theta'$;
2. any transition $t$ of $\Theta$ is renamed $t'$ in $\Theta'$;
3. any unobservable label $e$ of transitions in $\Theta$ is renamed $e'$ in $\Theta'$, any observable label $e$ of transitions in $\Theta$ remains unchanged in $\Theta'$.

More formally, recalling that $\Sigma$ is the set of events of the system and among $\Sigma$, $\Sigma_o$ is the set of observable events, we introduce the set $\Sigma' = \Sigma_o \cup \{e' : e \in \Sigma_u\}$.

Let $\mathscr{R}_{\Sigma \to \Sigma'}$ (resp. $\mathscr{R}_{\Sigma' \to \Sigma}$) denote the renaming function such that for any $e \in \Sigma \setminus \Sigma_o$ $\mathscr{R}_{\Sigma \to \Sigma'}(e) = e' \in \Sigma'$ (resp. for any $e' \in \Sigma' \setminus \Sigma_o$ $\mathscr{R}_{\Sigma' \to \Sigma}(e') = e \in \Sigma$) and $\mathscr{R}_{\Sigma \to \Sigma'}(e) = e \in \Sigma'$ (resp. $\mathscr{R}_{\Sigma' \to \Sigma}(e) = e \in \Sigma$) if $e \in \Sigma_o$.

The second step is then to consider the pattern negation $\overline{\Omega}$ (see Definition 11) that is also duplicated in the same manner as $\Theta$ to obtain $\overline{\Omega}'$. We then compute the system-pattern products $\Theta_\Omega = \Theta \ltimes \Omega$ and $\Theta'_{\overline{\Omega}'} = \Theta' \ltimes \overline{\Omega}'$. The so called twin plant $\Gamma$ finally results from the synchronized product of $\Theta_\Omega$ and $\Theta'_{\overline{\Omega}'}$,

$$\Gamma = \Theta_\Omega \,\|\, \Theta'_{\overline{\Omega}'}.$$

Let $\Sigma_\Gamma$ denote $\Sigma \cup \Sigma'$, the following result formally asserts the relationship between the language generated by the twin plant $\Gamma$ and how the system $\Theta$ matches a pattern $\Omega$.

**Theorem 3** $\mathscr{L}(\Gamma) = \{\rho \in \Sigma_\Gamma^*, \exists \rho_1, \rho_2 \in \mathscr{L}(\Theta), \rho_1 \ni \Omega \wedge \rho_2 \not\ni \Omega, \mathscr{P}_{\Sigma \to \Sigma_o}(\rho_1) = \mathscr{P}_{\Sigma \to \Sigma_o}(\rho_2) \wedge \rho_1 = \mathscr{P}_{\Sigma_\Gamma \to \Sigma}(\rho) \wedge \rho_2 = \mathscr{R}_{\Sigma' \to \Sigma}(\mathscr{P}_{\Sigma_\Gamma \to \Sigma'}(\rho))\}$.

Proof: see Appendix.

## 5.5 Diagnosability checking

Let $Q_\Gamma$ denote the set of final markings of $\Gamma$. Theorem 3 asserts that any transition sequence of the twin plant $\Gamma$ that leads to a final marking $M \in Q_\Gamma$ of $\Gamma$ represents two runs $r_1$ and $r_2$, $r_1$ from $\Theta_\Omega$ and $r_2$ from $\Theta'_{\overline{\Omega}'}$, that produce the same observable sequence but, on one hand, the run of $\Theta_\Omega$ matches the pattern $\Omega$ and on the other, the run of $\Theta'_{\overline{\Omega}'}$ does not match the pattern $\Omega$. Any final marking of $\Gamma$ therefore represents an ambiguity about the detection of the pattern.

**Definition 13** *A marking $M$ of $\Gamma$ is ambiguous if $M$ is a final marking of $\Gamma$ ($M \in Q_\Gamma$).*

Definition 8 states that a pattern $\Omega$ is not diagnosable over a system iff we can find an infinite set of arbitrarily long sequences $\rho_1, \ldots, \rho_n, \ldots$ in $\mathscr{L}(\Gamma)$ such that $\rho_i$ is a strict prefix of $\rho_{i+1}$ for any $i \in \mathbb{N}$. So looking for such an infinite sequence in $\Gamma$ means to look for a transition cycle involving an ambiguous marking.

**Definition 14** *A sequence of transitions $c \in T_\Gamma^*$ of $\Gamma$ is a firable cycle if there exist two transition sequences $s,t \in T_\Gamma^*$ such that $t$ is not empty and, for any $n \in \mathbb{N}$, $st^n$ is a transition sequence of $\Gamma$ and $c = st^n$. A firable cycle is ambiguous if $t$ leads to an ambiguous marking.*

The proposed analysis method relies then on the following theorem that is a direct consequence of Definition 8, Theorem 3 and Definition 14 as explained here above:

**Theorem 4** *A pattern $\Omega$ is diagnosable in a system $\Theta$ iff the corresponding twin plant $\Gamma$ does not contain any firable cycle that is ambiguous.*

# 6 Implementation of the diagnosability analysis method by model checking

## 6.1 Model checking problem

The previous section shows that solving the pattern diagnosability problem is equivalent to searching ambiguous cycles in the product $\Gamma = \Theta_\Omega || \Theta'_{\overline{\Omega}'}$. If such a cycle exists then the pattern is not diagnosable. Searching for such a cycle can be defined as a model-checking problem. The definition of a model-checking problem relies on a Kripke structure as presented in Clarke et al (1999).

**Definition 15 (Kripke structure)** *A Kripke structure over a set of atomic propositions $P$ is a four tuple $M = (S, S_0, R, L)$ where*

1. *$S$ is a finite set of states;*
2. *$S_0 \subseteq S$ is the set of initial states;*
3. *$R \subseteq S \times S$ is a transition relation that must be total, that is, for every state $s \in S$, there is a state $s' \in S$ such that $R(s,s')$ holds;*
4. *$L : S \to 2^P$ is a function that labels each state with the set of atomic propositions true in that state.*

A Kripke structure is a representation of a finite-state concurrent system in which each state is labeled with the set of atomic propositions that hold in the state. A model-checking problem can then be described as follows. Given a temporal logic formula $\varphi$ (written in Linear Temporal Logic (LTL), Computation Tree Logic (CTL),...), find the set of all states in $S$ that satisfy $\varphi$:

$$\{s \in S : M, s \models \varphi\},$$

where $M, s \models \varphi$ means that $\varphi$ holds at state $s$ in the Kripke structure $M$ (Clarke et al (1999)).

We propose here to implement the pattern diagnosability test of a system $\Theta$ by using the TINA toolkit of Berthomieu et al (2004) and Berthomieu et al (2007) that provides all the necessary tools to directly generate the Kripke structure of the net $\Gamma$ and to check LTL properties over it. The TINA toolkit specifically aims at analyzing (timed) Petri nets by using state-of-the-art model-checking techniques. TINA actually generates *enriched labeled Kripke structures* from the analyzed Petri nets. In an *enriched Labeled Kripke structure*, $L$ is a multiset function $L : S \to 2^{2^P}$ which means that any state of this structure is associated with a set of sets of atomic propositions. For instance, with such an enrichment, it is possible to associate to a state $S$ the multiset $\{\{p = 2\}, \{p = 3\}, \dots\}$ which can be implicitly expressed as $p \geq 2$. With this enrichment it is possible to write formulas like $\varphi = p_1 + p_2 \leq 4$ which is a succinct representation of the explicit formula $\varphi_{exp} = (p_1 = 0 \wedge p_2 = 0) \vee (p_1 = 0 \wedge p_2 = 1) \vee \dots$. Moreover, this structure maps any transition of the net with a label (called event) so that it is possible to also define and check properties about them.

Based on enriched labeled Kripke structures, TINA is able to check properties written in SE-LTL (State/Event Linear Temporal Logic) which extends LTL in the way presented here above. A formula $\psi$ is a SE-LTL formula if it is a universally quantified formula

$$\psi ::= \forall \varphi$$

such that

$$\begin{array}{rcl} \varphi & ::= & cst \mid r \mid \neg\varphi \mid \varphi \vee \varphi \mid\mid \varphi \wedge \varphi \mid \bigcirc \varphi \mid \Box\varphi \mid \Diamond\varphi \mid \varphi \mathbf{U} \varphi \\ r & ::= & e \mid e \triangle e \\ e & ::= & p \mid a \mid c \mid e \bigtriangledown e \\ cst & ::= & \mathbf{dead} \mid \mathbf{div} \mid \mathbf{sub} \end{array}$$

with $p$ a place symbol, $a$ a transition symbol, $c \in \mathbb{N}$, $\triangle \in \{=, <, >, \leq, \geq\}$ and $\bigtriangledown \in \{+, -, *, /\}$. The operators $\bigcirc$ (next), $\Box$ (always), $\Diamond$ (eventually) and $\mathbf{U}$ (until) have their usual LTL semantics. The constants **dead**, **div**, **sub**, respectively stand for:

- deadlock (blocking state);
- temporal divergence;
- partially known state.

## 6.2 Diagnosability as a model-checking problem with TINA

SE-LTL is a temporal logic that aims at expressing properties about the runs (sequence of states (markings)/events (transitions)) of the system which is exactly the type of properties to check on the twin net $\Gamma$ to solve the pattern diagnosability problem. A

system is diagnosable if and only if $\Gamma$ does not contain any cycle of ambiguous markings (Theorem 4). Our objective is to determine a way to write this property as a SE-LTL formula.

As the system is supposed to have no deadlock and as it does not contain any cycle of unobservable events, it follows that from any reachable marking of $\Theta$, any sequence of unobservable transitions has a finite length and will always lead to the enabling of an observable transition of $\Theta$. This property has some consequences on the twin $\Gamma$. For any sequence $r$ of transitions from the initial marking of $\Gamma$ that leads to an ambiguous reachable marking, two cases hold.

1. There is a finite maximal continuation $r'$ of $r$ in $\Gamma$, it means that the twin $\Gamma$ reaches a blocking marking $M_{dead}$ after the trigger of $rr'$. If the blocking marking is not ambiguous, there is obviously no diagnosability problems. If $M_{dead}$ is ambiguous, the sequence $rr'$ then represents two runs $r_1$ and $r_2$ of $\Theta$, one of them is matching $\Omega$ but not the other. There is no unbounded continuation of $r_1$ that has the same observable projection than an unbounded continuation of $r_2$. No diagnosability problems can result from this case.

2. There is an unbounded continuation of $r$ in $\Gamma$, it means that the twin $\Gamma$ reaches a cycle of markings. If the cycle contains ambiguous markings, the system is not diagnosable.

To summarize both cases, checking in $\Gamma$ that there is no cycle of ambiguous markings is equivalent to checking, for any sequence of $\Gamma$ leading to an ambiguous marking, whether any of its continuations either leads to a *deadlock* or *eventually* leads to a marking that is not ambiguous (so that it will finally lead to a cycle of non-ambiguous markings). This is exactly this property that we propose to check on $\Gamma$ with the help of TINA, formally:

$$\varphi_{\text{DIAG}} = \forall\Box((M \in Q_\Gamma) \Rightarrow (\Diamond(M' \notin Q_\Gamma \vee \textbf{dead})))$$

where $Q_\Gamma$ is the set of final markings of $\Gamma$. The constraints $M \in Q_\Gamma$, $M' \notin Q_\Gamma$ of $\varphi_{\text{DIAG}}$ are implicitly described as constraints on the places of $\Gamma$ as described in Section 6.1 (see also examples in Section 7). Intuitively speaking $\varphi_{\text{DIAG}}$ asserts that in any trace of $\Gamma$ ($\forall$), it is always true ($\Box$) that as soon as a marking $M$ of the trace is ambiguous ($(M \in Q_\Gamma) \Rightarrow$), the continuation of the trace after $M$ will eventually ($\Diamond$) lead to a deadlock (**dead**) or to a marking $M'$ that is not ambiguous ($M' \notin Q_\Gamma$).

To end this section, we describe the complete and automated implementation of the method to solve the pattern diagnosability problem on a system $\Theta$:

1. Compute $\Gamma = \Theta_\Omega || \Theta'_{\overline{\Omega}'}$.

2. Compute $\varphi_{\text{DIAG}}$ from $\Gamma$.

3. Call TINA to compute the enriched labeled Kripke structure $K(\Gamma)$ of $\Gamma$ (`sift` tool),

4. Call TINA to check $\varphi_{\text{DIAG}}$ on $K(\Gamma)$ (`selt` tool).

### 6.3 Complexity analysis

This section analyzes the complexity of the method to check the diagnosability of a pattern $\Omega$ in a system $\Theta$. Let $p_\Omega$ (resp. $t_\Omega$) be the number of places (resp. transitions) in the pattern $\Omega$. Let $p_\Theta$ (resp. $t_\Theta$) be the number of places (resp. transitions) in the system $\Theta$. By construction of the system-pattern product, the net $\Theta \ltimes \Omega$ has $p_\Omega + p_\Theta$ places and, in the worst case, contains $t_\Theta + t_\Omega \times t_\Theta$ transitions so the complexity of the construction of $\Theta \ltimes \Omega$ is linear with the number of transitions of $t_\Omega$ and the number of transitions of $t_\Theta$. This is explained by the fact that $\Theta \ltimes \Omega$ implicitly describes how the system behaves with respect to the recognition of the pattern. Any transition of the system with the same label as a transition of the pattern might be actually involved in the recognition of the pattern or not. The twin-plant is based on a duplication of $\Theta \ltimes \Omega$ so it consists of $2 \times (p_\Omega + p_\Theta)$ places and in the worst case, $(t_\Theta + t_\Omega \times t_\Theta)^2$ transitions. Therefore the computation of the twin plant $\Gamma$ is linear in the number of places in the pattern $\Omega$ ($O(p_\Omega)$) and, in the worst case, quadratic in the number of transitions of the pattern $O(t_\Omega^2)$ and the system $O(t_\Theta^2)$. The quadratic complexity with respect to the number of transitions of the system is as in any twin-plant method, required to compare any couple of observable transitions of the system to check whether they share the same label of not. The quadratic complexity with respect to the number of transitions in the pattern comes from the linear complexity of the construction of $\Theta \ltimes \Omega$. The computation of $\varphi_{\text{DIAG}}$ only depends on the pattern $\Omega$ and is, in the worst case, in $O((k+1)^{|p_\Omega|})$ if $\Omega$ is $k$-bounded.

The last two parts of the algorithm solve a model-checking problem based on the net $\Gamma$. For a $k$-bounded net $\Gamma$ the size of the explicit representation of the Kripke structure, equivalent to the reachability graph of $\Gamma$, is in $O((k+1)^{2(p_\Omega+p_\Theta)})$ and the time complexity of its construction is $O((k+1)^{4(p_\Omega+p_\Theta)})$. Finally, the time complexity of the model-checking problem for SE-LTL is in $2^{O(|\varphi|)}.O((k+1)^{2(p_\Omega+p_\Theta)})$ where $|\varphi|$ is the number of symbols in the explicit formula $\varphi$ (Schnoebelen (2003)). The model-checker TINA is in charge of this part of the algorithm. It internally builds an optimized representation of the Kripke structure and the formula $\varphi$ to solve the problem in practice.

## 7 Case studies

This section presents two case studies. The first one comes from the special Benchmark Session at WODES'08, it is a non-prioritized Petri net that was used for testing fault diagnosability in Liu et al (2014a). Based on this benchmark, this section proposes some results about the single fault pattern diagnosability that show that the results of Liu et al (2014a) and the ones of the proposed approach are equivalent. Then, the analyses of other patterns extending the single fault pattern are detailed. The second case study that is described in this section models a system with some priorities. The purpose of this second example is to illustrate the full expressiveness of the models and the patterns that can be handled by the proposed method.

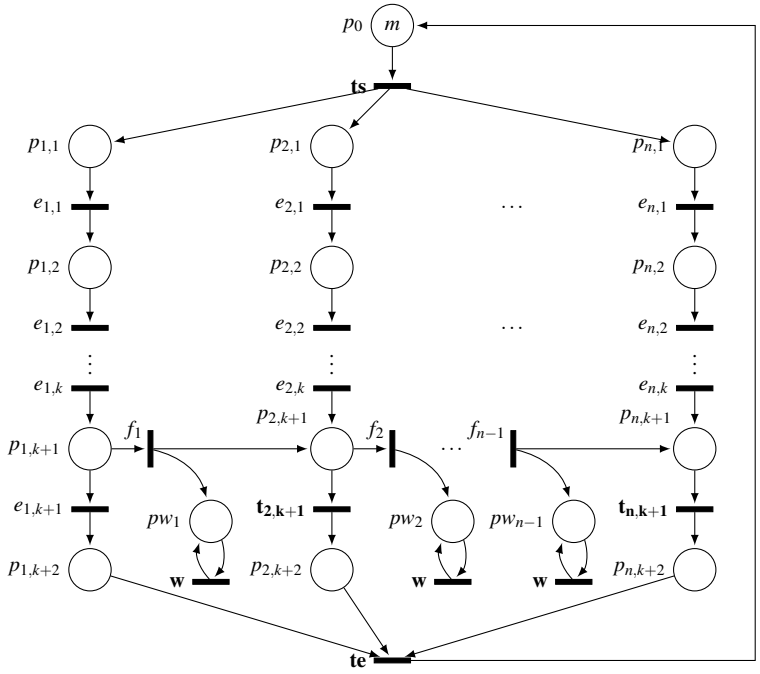


Figure 8: Case study 1: the modified WODES diagnosis benchmark of Giua (2007).

## 7.1 Case study 1

### 7.1.1 Model of the studied system

Figure 8 describes the LLPPN of a modified version of the system introduced in Giua (2007) as a benchmark at the special Benchmark Session at WODES'08. Observable events are in bold. The benchmark describes a family of manufacturing systems characterized by three parameters $n$, $m$ and $k$ where:

- $n$ is the number of production lines;
- $m$ is the number of units of the final product that can be simultaneously produced, each unit of product is composed of $n$ parts;
- $k$ is the number of operations that each part must undergo in each line.

To obtain one unit of final product, $n$ orders are sent, one to each line; this is represented by the observable event **ts** (*transition start*). Each line will produce a part (all parts are identical) and put it in its final buffer. An assembly station will take one part from each buffer to produce the final product, this is modeled by the observable event **te** (*transition end*). The part in line $i \in \{1,\ldots,n\}$ undergoes a series of $k$ operations, represented by unobservable events $e_{i,1}, e_{i,2}, \ldots, e_{i,k}$. After this series of operations two events are possible: either the part is regularly put in the final buffer of the line, or a fault

may occur. Putting the part in the final buffer of line 1 corresponds to unobservable event $e_{1,k+1}$, while putting the part in the final buffer of line $i, i \in \{2,\ldots,n\}$ corresponds to observable event $\mathbf{t_{i,k+1}}$. There are $n-1$ faults, represented by unobservable events $f_i, i \in \{1,\ldots,n-1\}$. Fault $f_i$ moves a part from line $i$ to line $i+1$. Note that on line $i \in \{1,\ldots,n-1\}$ the fault may only occur when the part has finished processing and is ready to be put in its final buffer; the part goes to the same processing stage in line $i+1$. Finally, if a fault $f_i$ occurs, it may cause delays on a line that may generate alarms (**w**), the alarm **w** is observable but does not identify the source of the delays ($n-1$ transitions are labeled with event **w**). This is a slight modification of the WODES'08 benchmark. In the initial version, the system has some deadlocks (for instance, as soon as $m$ faults $f_1$ occur, the system blocks). The proposed modification will replace this blocking behavior by the indefinite generation of **w** watchdog alarms. With this modification, the system's observability is live as requested by the assumptions about the model of the system (see Section 3.2).

### 7.1.2 Experimental results

We present here a set of experimental results about the diagnosability of several patterns that the system of Figure 8 can match. These results were all obtained with a PC linux i5-4670 CPU 3.40GHz with 16GB memory. The results that are retained in these tables are the ones that do not require more than 20GB memory to avoid as much as possible the use of memory swap that requires disk access and slow down the time performance (98% of the presented results actually need less than 16GB).

#### Single class fault pattern

Firstly, in order to show that our diagnosability method extends methods that deal with single event fault, we present the diagnosability results about the single class fault $F = \{f_1,\ldots,f_n\}$. Previous results about this diagnosability problem can be found in Liu et al (2014a). In this problem, any fault event $f_i$ belongs to the same class as any other fault event $f_j$. In other words, the system is considered to be faulty as soon as one of the fault $f_i$ has occurred. With our method, we propose to model this fault class with the pattern $\Omega_3(n)$ presented in Figure 9.

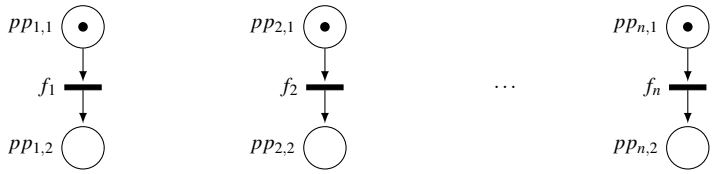


Figure 9: Single class fault pattern $\Omega_3(n)$ with $Q_3 = \{M : \exists i \in \{1,\ldots,n\}, M(pp_{i,2}) = 1\}$. Multiple fault pattern $\Omega_4(n)$, with $Q_4 = \{M : \forall i \in \{1,\ldots,n\}, M(pp_{i,2}) = 1\}$.

The pattern $\Omega_3(n)$ describes the parallel behavior of any of the $n$ faults that could occur in a system of $n+1$ production lines. The pattern $\Omega_3(n)$ is associated with the set of final markings $Q_3 = \{M : \exists i \in \{1,\ldots,n\}, M(pp_{i,2}) = 1\}$. Therefore, as soon as

a fault $f_i$ occurs, the pattern $\Omega_3(n)$ is recognized. This is exactly the expected behavior also represented by the single fault class $F = \{f_1, \ldots, f_n\}$ in Liu et al (2014a). Results are presented in Table 1.

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|
| $m$ | $n$ | $k$ | $nS(\Theta)$ | $nT(\Theta)$ | $nS(K(\Gamma_3))$ | $nT(K(\Gamma_3))$ | t. (s) $\Omega_3$ | Res. $\Omega_3$ | t. (s) $\Omega_4$ | Res. $\Omega_4$ |
| 1 | 2 | 1 | 15 | 27 | 91 | 237 | 0.004 | Yes | 0.004 | Yes |
| 1 | 2 | 2 | 24 | 45 | 276 | 815 | 0.004 | Yes | 0.004 | Yes |
| 1 | 2 | 3 | 35 | 67 | 659 | 2085 | 0.004 | Yes | 0.004 | Yes |
| 1 | 2 | 4 | 48 | 93 | 1348 | 4455 | 0.012 | Yes | 0.008 | Yes |
| 1 | 3 | 1 | 80 | 250 | 1434 | 6640 | 0.020 | Yes | 0.008 | No |
| 1 | 3 | 2 | 159 | 512 | 7071 | 35087 | 0.092 | Yes | 0.048 | No |
| 1 | 3 | 3 | 274 | 892 | 24550 | 126348 | 0.280 | Yes | 0.164 | No |
| 1 | 3 | 4 | 431 | 1408 | 68391 | 360235 | 0.752 | Yes | 0.476 | No |
| 1 | 4 | 1 | 495 | 2286 | 28703 | 222808 | 0.44 | Yes | 0.312 | No |
| 1 | 4 | 2 | 1200 | 5670 | 225792 | 1787416 | 3.864 | Yes | 2.552 | No |
| 1 | 4 | 3 | 2415 | 11486 | 1123695 | 8922268 | 21.276 | Yes | 13.172 | No |
| 1 | 4 | 4 | 4320 | 20550 | 4207640 | 33358840 | 76.076 | Yes | 48.336 | No |
| 1 | 5 | 1 | 3295 | 20382 | 659323 | 8108532 | 18.136 | Yes | 11.956 | No |
| 1 | 5 | 2 | 9691 | 61187 | 8234185 | 99990516 | 218.668 | Yes | 150.468 | No |
| 2 | 2 | 1 | 96 | 278 | 2140 | 8926 | 0.016 | No | 0.016 | No |
| 2 | 2 | 2 | 237 | 746 | 17405 | 85966 | 0.144 | No | 0.148 | No |
| 2 | 3 | 1 | 1484 | 7006 | 232508 | 1756465 | 3.136 | No | 2.964 | No |
| 2 | 3 | 2 | 5949 | 30612 | 5645449 | 48717944 | 82.756 | No | 87.816 | No |
| 2 | 4 | 1 | 28203 | 190144 | 35331057 | 444379980 | 1393.796 | No | 1031.936 | No |
| 3 | 2 | 1 | 377 | 1371 | 21171 | 110757 | 0.196 | No | 0.196 | No |
| 3 | 3 | 1 | 12048 | 69302 | 8118900 | 76016894 | 166.492 | No | 154.576 | No |

Table 1: Diagnosability of the pattern $\Omega_3(n-1)$ (at least one single fault out of the $n-1$ faults on a system with $n$ production lines) and pattern $\Omega_4(n-1)$ (occurrence of the $n-1$ faults on a system with $n$ production lines).

This table represents a set of experiments for different $(m,n,k)$ configurations (Columns 1–3) of the system presented in Figure 8. The set of presented configurations is the one of Liu (2014) (that is a superset of the one of Liu et al (2014a)) and has been selected here for comparison purposes. First, it must be noticed that in Liu (2014), the studied system is the one of Figure 8 without the watchdog alarms $\mathbf{w}$. Adding these alarms do not actually affect the diagnosability of the system (they just add non-discriminant but live observable behaviors to the previous version of the system). This is confirmed by the result of Table 1 where the diagnosability results for $\Omega_3$ in Column 9 are identical to the one of Liu (2014). Columns 4 and 5 present the number of states $nS(\Theta)$ and transitions $nT(\Theta)$ in the reachable marking graph of the system $\Theta$. Here also, it can be noticed that the number of states $nS(\Theta)$ of each configuration is identical to the one of Liu (2014) by construction of the watchdog alarm extensions. The computation

times (Column 8) gather the four computational steps described in Section 6.2. Steps 1 and 2, that are the computations of $\Theta_{\Omega_3}||\Theta'_{\overline{\Omega_3}'}$ and $\varphi_{DIAG3}$, have always a negligible computation time compared to the computation times of Steps 3 and 4. Step 3 builds the corresponding Kripke structure of $\Gamma_3 = \Theta_{\Omega_3}||\Theta'_{\overline{\Omega_3}'}$, Columns 6 and 7 present the number of states $nS(K(\Gamma_3))$ and transitions $nT(K(\Gamma_3))$ of the Kripke structure of $\Gamma_3$ computed by TINA (`sift` tool). Step 4 checks with the help of the TINA tool `selt` whether $\varphi_{DIAG3}$ holds in $K(\Gamma_3)$. For instance, in the configuration $m = 1, n = 3, k = 1$ (gray row in Table 1), we analyze the diagnosability of the pattern $\Omega_3(2)$ (at least one fault among the set $\{f_1, f_2\}$), and the formula $\varphi_{DIAG3}$ is:

$$\forall\Box(((pp_{1,2} \geq 1 \vee pp_{2,2} \geq 1) \wedge (\overline{pp_{1,2}} = 0 \wedge \overline{pp_{2,2}} = 0)) \\ \Rightarrow (\Diamond(\overline{pp_{1,2}} \geq 1 \vee \overline{pp_{2,2}} \geq 1 \vee \mathbf{dead})))$$

where $\overline{pp_{1,2}}$ (resp. $\overline{pp_{2,2}}$) is the place of $\overline{\Omega_3}(2)$ corresponding to the place $pp_{1,2}$ (resp. $pp_{2,2}$) in the pattern $\Omega_3(2)$ (see Figure 9).

**Multiple fault pattern**

The next diagnosability analysis that we present here is the multiple fault diagnosability analysis. In this case, the objective is to determine whether any run of the system where *every fault type* has occurred at least once can be diagnosed with certainty. To perform this analysis, pattern $\Omega_3(n)$ is updated to pattern $\Omega_4(n)$ (see Figure 9). Basically $\Omega_3(n)$ and $\Omega_4(n)$ have the same structure, only the definition of the final markings is different. In $Q_4$, any marking contains at least one token in *any* place $pp_{i,2}, i \in \{1, \ldots, n\}$. This difference obviously has consequences on the formula $\varphi_{DIAG4}$ that is:

$$\forall\Box(((pp_{1,2} \geq 1 \wedge pp_{2,2} \geq 1) \wedge (\overline{pp_{1,2}} = 0 \vee \overline{pp_{2,2}} = 0)) \\ \Rightarrow (\Diamond((\overline{pp_{1,2}} \geq 1 \wedge \overline{pp_{2,2}} \geq 1) \vee \mathbf{dead}))).$$

Results of this analysis are also presented in Table 1 in columns 10 and 11. For the cases where $n = 2$, this analysis is equivalent to the first one (if $n = 2$ there is only one fault $f_1$) which explains why for $m = 1, n = 2$ cases (the first four lines of Table 1), both analyses have the same result. As soon as $n \geq 3$, the multiple fault is not diagnosable.

**Pattern of one normal event over $n$**

The next analysis that we present in this section relies on the pattern $\Omega_5(n)$. The purpose of this analysis is simple. The pattern $\Omega_5(n)$ just represents the set of behaviors on the system $\Theta$ where one of the events $\{e_{1,1}, \ldots, e_{n,1}\}$ occurs at least once. This pattern is obviously diagnosable in any configuration, which is confirmed by the results in Table 2.

This pattern illustrates the fact that we might not necessarily expect to only monitor faulty behaviors. It might be also interesting to monitor normal behaviors of interest.

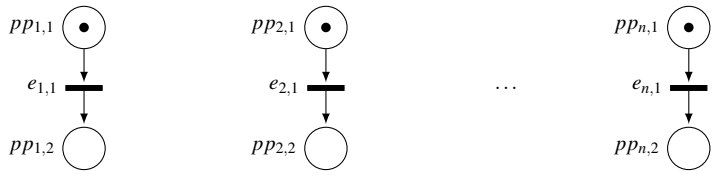


Figure 10: Pattern $\Omega_5(n)$ that is always diagnosable in the benchmark with its final markings $Q_5 = \{M : \exists i \in \{1,\ldots,n\}, M(pp_{i,2}) = 1\}$.

Back to pattern $\Omega_5(n)$ for instance, if it is modified to represent the occurrence of two normal events over $n, n \geq 2$ the related diagnosability question would be: is the occurrence of two distinct events among $\{e_{1,1},\ldots,e_{n,1}\}$ diagnosable or not. The answer is surprisingly negative.

| $m$ | $n$ | $k$ | $nS(K(\Gamma_5))$ | $nT(K(\Gamma_5))$ | Time(s) | Result |
|---|---|---|---|---|---|---|
| 1 | 2 | 1 | 129 | 339 | 0.004 | Yes |
| 1 | 2 | 2 | 371 | 1105 | 0.004 | Yes |
| 1 | 2 | 3 | 853 | 2719 | 0.004 | Yes |
| 1 | 2 | 4 | 1695 | 5637 | 0.008 | Yes |
| 1 | 3 | 1 | 2430 | 11289 | 0.02 | Yes |
| 1 | 3 | 2 | 11238 | 55858 | 0.104 | Yes |
| 1 | 3 | 3 | 37142 | 191409 | 0.376 | Yes |
| 1 | 3 | 4 | 99486 | 524706 | 1.032 | Yes |
| 1 | 4 | 1 | 52901 | 410215 | 0.796 | Yes |
| 1 | 4 | 2 | 393198 | 3103031 | 6.834 | Yes |
| 1 | 4 | 3 | 1865237 | 14754887 | 31.576 | Yes |
| 1 | 4 | 4 | 6708482 | 52988035 | 116.572 | Yes |
| 1 | 5 | 1 | 1265621 | 15542016 | 32.364 | Yes |
| 1 | 5 | 2 | 15124090 | 182988624 | 683.392 | Yes |
| 2 | 2 | 1 | 2470 | 10156 | 0.02 | Yes |
| 2 | 2 | 2 | 18950 | 92732 | 0.2 | Yes |
| 2 | 3 | 1 | 311440 | 2320207 | 5.4 | Yes |
| 2 | 3 | 2 | 6552732 | 55877578 | 141.884 | Yes |
| 3 | 2 | 1 | 22817 | 117993 | 0.26 | Yes |
| 3 | 3 | 1 | 10032272 | 92786651 | 251.288 | Yes |

Table 2: Diagnosability of the diagnosable pattern $\Omega_5(n)$: at least one among the $n$ events $\{e_{k,1}\}_{k\in\{1,\ldots,n\}}$ has occurred.

## 7.2 Case study 2

The aim of the previous example was to compare the results of the proposed method with the ones of Liu et al (2014b) by choosing a single fault pattern and then to extend

the diagnosability analysis to other patterns. The purpose of this second example is to provide experimental results on a system model with priorities and more complex patterns of interest in this system.

### 7.2.1 Description of the system

This second example is a product transportation system. It 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 11 presents the system model. Once a product is detected on the site 1, a *Product1* signal (noted $Pr1$) 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 11, 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.

For this product transportation system, bold events on Figure 11 are the observable events of the system: $\mathbf{Pr_1}, \mathbf{Pr_2}, \mathbf{U}, \mathbf{D}, \mathbf{ERReq_1}, \mathbf{ERReq_1}, \mathbf{ELReq_1}, \mathbf{ELReq_2}$. The events $\{P_1, P_2, EP_1, EP_2, Req_1, Req_2, EPReq_1, EPReq_2\} = \Sigma_u$ are the unobservable events. It must be noticed that in this model, there is no fault event as opposed to the first case study. The faulty behaviours of the system are represented by abnormal combinations of events described as patterns. Next section presents two classes of such patterns

### 7.2.2 Experimental results

For this transportation system to work well, we can imagine that if only one station keeps requesting products and the other one does not make any request, the system is not properly working as only one station is running. It then would be interesting to know whether the set of observations is sufficient to always assert with certainty that the two stations regularly perform requests. The following classes of patterns $\Omega_6(n)$ and $\Omega_7(n,k)$ (see Figure 12) have been designed for this purpose. The results have been obtained under similar conditions to the ones of Section 7.1.2.

#### Multiple consecutive occurrences of an event

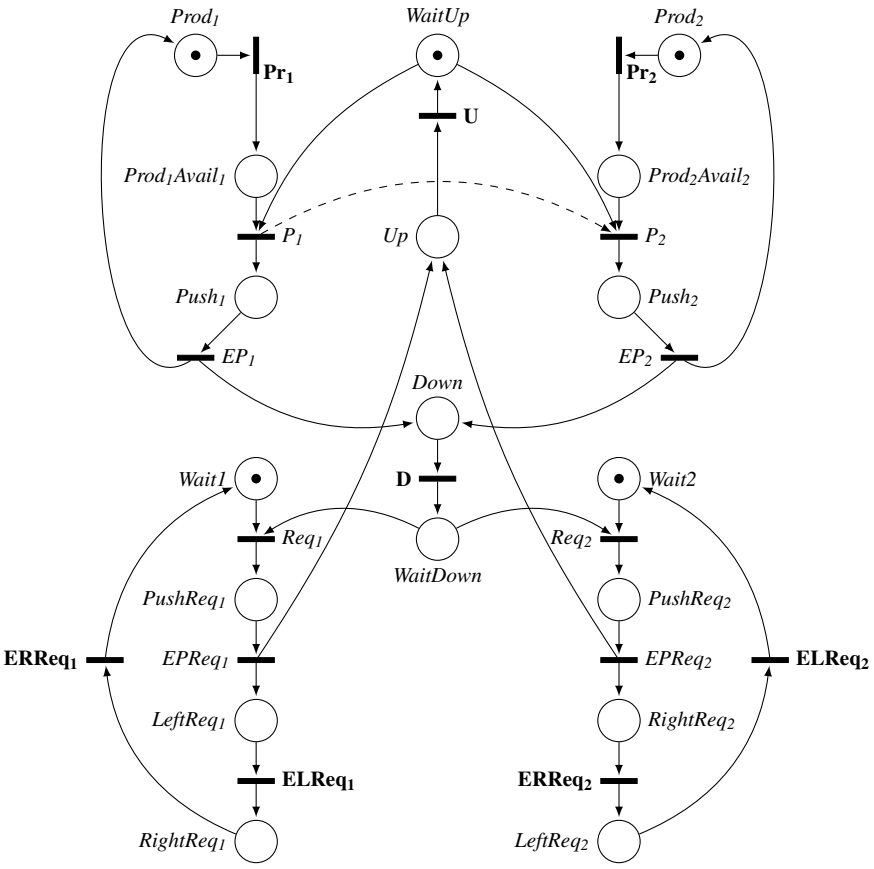


Figure 11: Case study 2 : the product transportation system

In this first diagnosability analysis, the objective is to determine whether any run where $n$ requests of type 1 have occurred consecutively (i.e. with no interleaved request of type 2) can be diagnosed with certainty. The pattern $\Omega_6(n), n = 4$ associated to this behavior is given on the left of Figure 12. Results are presented in Table 3. This table represents a set of experiments for different configurations with respect to the number of consecutive occurrences ($n$) considered in the pattern. The second line shows results for $\Omega_6(4)$ whereas lines 3 to 5 involve higher values for $n$. The diagnosability result is indicated in Column 7. Columns 2 and 3 give the number of places $nP(\Gamma_6)$ and transitions $nT(\Gamma_6)$ of the associated twin Petri net $\Gamma_6$. Columns 4 and 5 present the number of states $nS(K(\Gamma_6))$ and transitions $nT(K(\Gamma_6))$ of the Kripke structure of $\Gamma_6$ computed by TINA. Column 6 gives the computation times (for all the steps described in Section 6.2).

As an example, we present the formula $\varphi_{DIAG6}$ for the diagnosability analysis of the pattern $\Omega_6(4)$:

$$\forall\Box((p_4 \geq 1 \wedge \overline{p_4} \leq 0) \Rightarrow (\Diamond(\overline{p_4} \geq 1 \vee \mathbf{dead})))$$

where $\overline{p_4}$ is the place of $\overline{\Omega_6}(4)$ corresponding to the place $p_4$ in the pattern $\Omega_6(4)$ (see Figure 12 on the left).

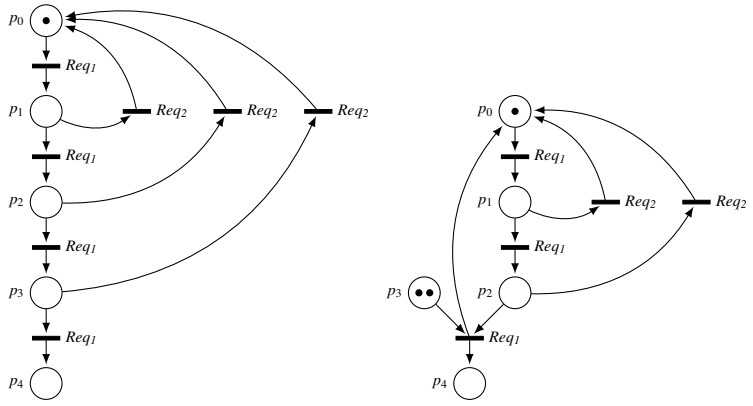


Figure 12: On the left, the multiple consecutive occurrences pattern $\Omega_6(4)$. On the right the pattern $\Omega_7(2,3)$ (2 repetitions of 3 consecutive occurrences of *Req1*).

For any $n$, pattern $\Omega_6(n)$ is not diagnosable which looks counterintuitive. Indeed, as the $Req1$ event is followed by two observable events (**ELReq$_1$** and **ERReq$_1$**) we could imagine to conclude that the pattern is diagnosable. However, after the $Req1$ event, there are possible interleaved behaviors involving $Req2$ that can happen (due to the fact that $EPReq_1$ event is not observable, the station 1 can silently stay in $LeftReq_1$) that lead to the non-diagnosability of the pattern.

It is possible to ensure the diagnosability of the pattern by adding events $EPReq_1$ and $EPReq_2$ to the set of observable events:

| $n$ | $nP(\Gamma_6)$ | $nT(\Gamma_6)$ | $nS(K(\Gamma_6))$ | $nT(K(\Gamma_6))$ | Time(s) | Result |
|---|---|---|---|---|---|---|
| 2 | 42 | 46 | 8248 | 24296 | 0.048 | No |
| 4 | 46 | 54 | 21884 | 65020 | 0.132 | No |
| 6 | 50 | 62 | 41888 | 125088 | 0.216 | No |
| 8 | 54 | 70 | 68260 | 204500 | 0.380 | No |
| 10 | 58 | 78 | 101000 | 303256 | 0.600 | No |

Table 3: Diagnosability of the pattern $\Omega_6(n)$: at least $n$ consecutive requests $Req_1$ have occurred before an occurrence of $Req_2$.

$$\Sigma_o = \{\mathbf{Pr1}, \mathbf{Pr2}, \mathbf{U}, \mathbf{D}, \mathbf{ERReq1}, \mathbf{ERReq1}, \mathbf{ELReq1}, \mathbf{ELReq2}, \mathbf{EPReq1}, \mathbf{EPReq2}\}$$

and $\{P1, P2, EP1, EP2, Req1, Req2\} = \Sigma_u$. Results are shown in Table 4.

| $n$ | $nP(\Gamma_6)$ | $nT(\Gamma_6)$ | $nS(K(\Gamma_6))$ | $nT(K(\Gamma_6))$ | Time(s) | Result |
|---|---|---|---|---|---|---|
| 2 | 42 | 48 | 2052 | 6312 | 0.012 | Yes |
| 4 | 46 | 56 | 3408 | 10492 | 0.024 | Yes |
| 6 | 50 | 64 | 4764 | 14672 | 0.032 | Yes |
| 8 | 54 | 72 | 6120 | 18852 | 0.048 | Yes |
| 10 | 58 | 80 | 7476 | 23032 | 0.060 | Yes |

Table 4: Diagnosability of the pattern $\Omega_6(n)$ on the diagnosable version of the system.

**Repetition of multiple consecutive occurrences of an event**

The next diagnosability analysis extends the previous one to the repetition of multiple consecutive occurrences of an event. In this case, the objective is to determine whether any run of the system where $n$ consecutive occurrences of $Req1$ have occurred $k$ times, can be diagnosed with certainty. To perform this analysis pattern $\Omega_6(n)$ is updated to pattern $\Omega_7(n,k)$ (see Figure 12 on the right for $n = 3$ and $k = 2$). The formula for the pattern $\Omega_7(3,2)$ is $\varphi_{DIAG7}$:

$$\forall\Box((p_4 \geq 2 \wedge \overline{p_4} \leq 1) \Rightarrow (\Diamond(\overline{p_4} \geq 2 \vee \mathbf{dead})))$$

where $\overline{p_4}$ is the place of $\overline{\Omega_7}(3,2)$ corresponding to the place $p_4$ in the pattern $\Omega_7(3,2)$ (see Figure 12 on the right).

Table 5 shows the experimental results of this diagnosability analysis. Several configurations have been tested according to $n$, the number of consecutive occurrences of the event $Req1$, and $k$, the number of repetitions.

In light of the previous results obtained for the $\Omega_6(n)$ pattern, the results of Table 5 about $\Omega_7(n,k)$ are foreseeable. A modification of the system observability similar to the previous one leads to the diagnosability of the pattern $\Omega_7(n,k)$ as presented in Table 6.

| $n$ | $k$ | $nP(\Gamma_7)$ | $nT(\Gamma_7)$ | $nS(K(\Gamma_7))$ | $nT(K(\Gamma_7))$ | Time(s) | Result |
|---|---|---|---|---|---|---|---|
| 2 | 2 | 44 | 46 | 33108 | 97424 | 0.200 | No |
| 3 | 2 | 46 | 50 | 72870 | 214972 | 0.502 | No |
| 4 | 3 | 48 | 54 | 229718 | 677484 | 1.484 | No |
| 10 | 4 | 60 | 78 | 2208716 | 6521248 | 16.416 | No |

Table 5: Diagnosability of the pattern $\Omega_7(n,k)$: at least for $k$ times, $n$ consecutive requests $Req_1$ have occurred before an occurrence of $Req_2$.

| $n$ | $k$ | $nP(\Gamma_7)$ | $nT(\Gamma_7)$ | $nS(K(\Gamma_7))$ | $nT(K(\Gamma_7))$ | Time(s) | Result |
|---|---|---|---|---|---|---|---|
| 2 | 2 | 44 | 48 | 4092 | 12616 | 0.032 | Yes |
| 3 | 2 | 46 | 52 | 6126 | 18886 | 0.052 | Yes |
| 4 | 3 | 48 | 56 | 10890 | 33558 | 0.088 | Yes |
| 10 | 4 | 60 | 80 | 33960 | 104660 | 0.296 | Yes |

Table 6: Diagnosability of the pattern $\Omega_7(n,k)$ on the diagnosable version of the system.

# 8 Conclusions and perspectives

The problem of checking pattern diagnosability on systems modeled as LLPPN is investigated and an effective method to solve the problem is proposed. Genericness is a core strength of the proposed method since it does not impose specific patterns but rather defines a generic framework for pattern modeling. The success of the approach relies firstly on the way a pattern and a system are combined with the *system-pattern product* and secondly on the fact that the pattern diagnosability problem is translated to a model-checking problem on a Petri Net that can be effectively solved by the Petri net analyzer TINA.

Patterns investigated in this paper are regular, in the sense that they generate regular languages. One interesting problem that we are planning to address is to extend the diagnosability analysis to patterns that generate context-free languages. Another research perspective is to extend this work to timed systems by using Time Petri Nets that would lead to monitor more complex and interesting behaviors where time can be observed and that cannot be modeled otherwise. It would especially improve the results by analyzing the impact of the observation of time on the system's diagnosability. To avoid the combinatorial explosion associated with marking graphs, another interesting research topic would be to adapt our proposal to use Petri nets unfoldings. Finally, another challenging issue is the analysis of other properties related to the diagnosis problem, such as discriminability. The discriminability of patterns ensures that one can detect the exclusive occurrence of a particular pattern from another set of patterns. Discriminability is a useful property that is weaker than diagnosability.

# A Proofs

## A.1 Proof of Lemma 1

**Lemma 1** For any reachable marking $M \in R(\Theta_\Omega, M_{\Theta_\Omega 0})$ of $\Theta_\Omega$, there exists a reachable marking $M_1 \in R(\Theta, M_{\Theta 0})$ and a reachable marking $M_2 \in R(\Omega, M_{\Omega 0})$ such that $M = M_1 \cup M_2$.

**Proof.** We respectively denote by $P_\Theta, T_\Theta$, $P_\Omega, T_\Omega$, $P_{\Theta_\Omega}, T_{\Theta_\Omega}$ the set of places and transitions of $\Theta$, $\Omega$, $\Theta_\Omega$. $M_{\Theta 0}$, $M_{\Omega 0}$ and $M_{\Theta_\Omega 0}$ are also their respective initial marking. We proceed by induction on the set of reachable markings of $R(\Theta_\Omega, M_{\Theta_\Omega 0})$ starting from the initial marking $M_{\Theta_\Omega 0}$. The property obviously holds for $M_{\Theta_\Omega 0}$ by definition of the system-pattern product (Definition 10).

Now suppose the existence in $R(\Theta_\Omega, M_{\Theta_\Omega 0})$ of a marking $M'$ such that the property holds: there exist a reachable marking $M'_1 \in R(\Theta, M_{\Theta 0})$ and a reachable marking $M'_2 \in R(\Omega, M_{\Omega 0})$ such that $M' = M'_1 \cup M'_2$. Consider a marking $M$ that is reachable from $M'$: $\exists t \in T_{\Theta_\Omega} : M' \xrightarrow{t} M$, it follows that $M \in R(\Theta_\Omega, M_{\Theta_\Omega 0})$ and $\text{pre}_{\Theta_\Omega}(t) \subseteq M'$. Let $K_\Theta$, $K_\Omega$ respectively denote the transition mask of $\Theta$ over $\Theta_\Omega$ and the transition mask of $\Omega$ over $\Theta_\Omega$ formally defined as follows: let $\varepsilon$ denote here the empty sequence of transitions,

$$K_\Theta(t) = \begin{cases} t & \text{if } t \in T_\Theta \\ \varepsilon & \text{if } t \in T_\Omega \\ t_1 & \text{if } t = t_1 \| t_2 \end{cases} \quad K_\Omega(t) = \begin{cases} \varepsilon & \text{if } t \in T_\Theta \\ t & \text{if } t \in T_\Omega \\ t_2 & \text{if } t = t_1 \| t_2 \end{cases}$$

The transition masks can be easily extended to any sequence of transitions of $\Theta_\Omega$, let $s = t_0 \dots t_n$ be such a sequence, $K_\Theta(s) = K_\Theta(t_0).K_\Theta(t_1) \dots K_\Theta(t_n)$ (similar for $K_\Omega(s)$). From Definition 10, we get:

$$\text{pre}_{\Theta_\Omega}(t) = \text{pre}_\Theta(K_\Theta(t)) \cup \text{pre}_\Omega(K_\Omega(t)),$$

with the following definition extension of pre: $\text{pre}_{\Theta_\Omega}(\varepsilon) = \text{pre}_\Theta(\varepsilon) = \text{pre}_\Omega(\varepsilon) = \varnothing$. However any place of $\text{pre}_\Theta(K_\Theta(t))$ is in $P_\Theta$ and any place of $\text{pre}_\Omega(K_\Omega(t))$ is in $P_\Omega$. As $P_\Theta$ and $P_\Omega$ are disjoint $\text{pre}_\Theta(K_\Theta(t)) \subseteq M'_1$ and $\text{pre}_\Omega(K_\Omega(t)) \subseteq M'_2$. From this, two cases hold.

1. Transition $t = t_1 \| t_2$ is synchronized. This transition results from the synchronization of transition $t_1$ in $\Theta$ and transition $t_2$ in $\Omega$ therefore, as $\text{pre}_\Theta(K_\Theta(t)) = \text{pre}_\Theta(t_1) \subseteq M'_1$, $t_1$ is firable from $M'_1$ in $\Theta$ and, as $\text{pre}_\Omega(K_\Omega(t)) = \text{pre}_\Omega(t_2) \subseteq M'_2$, $t_2$ is firable from $M'_2$ in $\Omega$. By Definition 10, $\text{post}_{\Theta_\Omega}(t) = \text{post}_\Theta(t_1) \cup \text{post}_\Omega(t_2)$.
2. Transition $t$ belongs to $T_\Theta$, so $t$ is firable from $M'_1$ in $\Theta$ and $\text{post}_{\Theta_\Omega}(t) = \text{post}_\Theta(t)$.

Based on the extended definition of post: $\text{post}_{\Theta_\Omega}(\varepsilon) = \text{post}_\Theta(\varepsilon) = \text{post}_\Omega(\varepsilon) = \varnothing$, the three previous cases can be synthesized as follows:

$$\text{post}_{\Theta_\Omega}(t) = \text{post}_\Theta(K_\Theta(t)) \cup \text{post}_\Omega(K_\Omega(t)).$$

We have supposed that $M' \xrightarrow{t} M$ so $M = M' \setminus \text{pre}_{\Theta_\Omega}(t) \cup \text{post}_{\Theta_\Omega}(t)$ which leads to $M = M' \setminus (\text{pre}_\Theta(K_\Theta(t)) \cup \text{pre}_\Omega(K_\Omega(t))) \cup (\text{post}_\Theta(K_\Theta(t)) \cup \text{post}_\Omega(K_\Omega(t)))$ to finally obtain $M = M_1 \cup M_2$ with $M_1 = (M'_1 \setminus \text{pre}_\Theta(K_\Theta(t))) \cup \text{post}_\Theta(K_\Theta(t))$ and $M_2 = (M'_2 \setminus \text{pre}_\Omega(K_\Omega(t))) \cup \text{post}_\Omega(K_\Omega(t))$.

## A.2 Proof of Theorem 1

**Theorem 1** Let $\Theta$ be the LLPPN of a system over the alphabet $\Sigma$ and $\Omega$ be the LLPPN of a pattern:

$$\mathscr{L}(\Theta \ltimes \Omega) = \{\rho \in \mathscr{L}(\Theta) : \rho \ni \Omega\}.$$

**Proof.** By Definition 5, we have to prove that $\mathscr{L}(\Theta \ltimes \Omega) \subseteq \{\rho \in \mathscr{L}(\Theta) : \exists \sigma \in \mathscr{L}(\Omega) : \rho \ni \sigma\}$.

1. Let us prove first that $\mathscr{L}(\Theta \ltimes \Omega) \subseteq \{\rho \in \mathscr{L}(\Theta) : \exists \sigma \in \mathscr{L}(\Omega) : \rho \ni \sigma\}$. The first step is to prove that any word of $\mathscr{L}(\Theta \ltimes \Omega)$ is indeed in $\mathscr{L}(\Theta)$. By Lemma 1, recalling the transition masks $K_\Theta$ and $K_\Omega$ defined in the proof of this lemma, we get: $\forall r \in T^*_{\Theta_\Omega} : M_0 \xrightarrow{r} M, \exists r_1 \in T^*_\Theta = K_\Theta(r) : M_{01} \xrightarrow{r_1} M_1, \exists r_2 \in T^*_\Omega = K_\Omega(r) : M_{02} \xrightarrow{r_2} M_2 \wedge M = M_1 \cup M_2$. Moreover, by Definition 10, $\ell_{\Theta_\Omega}(r) = \ell_\Theta(K_\Theta(r))$. For any firable sequence of $\Theta_\Omega$, there exists a firable sequence in $\Theta$ with the same sequence of labels. As $\mathscr{L}(\Theta)$ is a prefix-closed language ($Q_\Theta = R(\Theta, M_{\Theta 0})$, see Definition 3) any sequence of labels resulting from a firable sequence of $\Theta$ is in $\mathscr{L}(\Theta)$, so $\mathscr{L}(\Theta \ltimes \Omega) \subseteq \mathscr{L}(\Theta)$.

   Consider now such a firable sequence $r : M_0 \xrightarrow{r} M$ of $\Theta \ltimes \Omega$ such that $\rho = \ell_{\Theta \ltimes \Omega}(r) \in \mathscr{L}(\Theta \ltimes \Omega) \subseteq \mathscr{L}(\Theta)$. It means that there exists $M_{02} \xrightarrow{K_\Omega(r)} M_2$ ($M_2 \in Q_\Omega$). Let $\sigma = \ell_\Omega(K_\Omega(r))$ it follows that $\sigma \in \mathscr{L}(\Omega)$. We now prove that $\rho \ni \sigma$. Let $T_s$ denote the set of synchronized transitions of $\Theta \ltimes \Omega$ (see Definition 10). As $\rho \in \mathscr{L}(\Theta \ltimes \Omega), \rho \neq \varepsilon$ and $r$ contains at least one transition $t_s$ of $T_s$. Sequence $r$ can then be decomposed as $r = r' t_s r'' : \forall t \in r', t \notin T_s$. Therefore, $\rho$ can be written as $\ell_\Theta(K_\Theta(r'))\ell_\Theta(K_\Theta(t_s))\ell_\Theta(K_\Theta(r''))$ and $\sigma$ can be written as $\ell_\Omega(K_\Omega(r'))\ell_\Omega(K_\Omega(t_s))\ell_\Omega(K_\Omega(r''))$. As $t_s$ is the first synchronized transition in $r$, $\ell_\Omega(K_\Omega(r')) = \varepsilon$. Moreover, $\ell_\Omega(K_\Omega(t_s)) = \ell_\Theta(K_\Theta(t_s))$, which leads to $\sigma = \ell_\Theta(K_\Theta(t_s))\ell_\Omega(K_\Omega(r''))$. Let $\rho'' = \ell_{\Theta \ltimes \Omega}(r'')$ and $\sigma'' = \ell_\Omega(K_\Omega(r''))$, it follows that $\rho \ni \sigma$ iff $\rho'' \ni \sigma''$. Two cases hold:

   (a) If $r''$ does not contain some transitions of $T_s$, it means that $K_\Omega(r'')$ is an empty sequence: $\ell_\Omega(K_\Omega(r'')) = \varepsilon$, so $\sigma'' = \varepsilon$ and, by Definition 6, $\rho'' \ni \sigma''$.

   (b) If $r''$ contains some transitions of $T_s$, the previous reasoning recursively applies on $r''$ and its first synchronized transition (by replacing $r$ by $r''$, $\rho$ by $\rho''$, $\sigma$ by $\sigma''$) to finally lead to prove that $\rho_{end} \ni \sigma_{end}$ with $\rho_{end} = \ell_{\Theta \ltimes \Omega}(r_{end})$ and $\sigma_{end} = \ell_\Omega(K_\Omega(r_{end}))$ with $r_{end}$ a sequence without any synchronized transition left. This is exactly the previous case a): $\rho_{end} \ni \sigma_{end}$ so finally $\rho'' \ni \sigma''$.

2. Let us prove that $\mathscr{L}(\Theta \ltimes \Omega) \supseteq \{\rho \in \mathscr{L}(\Theta) : \exists \sigma \in \mathscr{L}(\Omega) : \rho \ni \sigma\}$. $\rho$ belongs to $\mathscr{L}(\Theta)$ so there exists a sequence of transitions $r$ from $\Theta$ such that $M_{\Theta 0} \xrightarrow{r} M_r \wedge \ell_\Theta(r) = \rho$. $\sigma$ belongs to $\mathscr{L}(\Omega)$ so $\sigma \neq \varepsilon$ and there exists a sequence of transitions $r'$ from $\Omega$ such that $M_{\Omega 0} \xrightarrow{r'} M_{r'} : \ell_\Omega(r') = \sigma$ and $M_{r'}$ is a final marking of $\Omega$. As $\rho \ni \sigma$, there exists $\sigma_0 \in \Sigma$ such that $\sigma = \sigma_0\sigma'$ and $\rho$ can be written as $\rho = \rho_0\sigma_0\rho_1, \sigma_0 \notin \rho_0$ and $\rho_1 \ni \sigma'$. It implies firstly that $r$ can be written as $M_{\Theta 0} \xrightarrow{r_0} M_{r0} \xrightarrow{t_1} M_{t1} \xrightarrow{r_1} M_r$ with $\ell_\Theta(t_1) = \sigma_0$ and secondly that $r'$ can be written as $M_{\Omega 0} \xrightarrow{t_2} M_{t2} \xrightarrow{r'_1} M_{r'}$ with $\ell_\Omega(t_2) = \sigma_0$. $\sigma_0$ not being a label of a transition in $r_0$, $(M_{\Theta 0} \cup M_{\Omega 0}) \xrightarrow{r_0} (M_{r0} \cup M_{\Omega 0})$ is a sequence of transitions of the reachable marking graph of $\Theta \ltimes \Omega$. By Definition 10, the transition $t_1$ is in $\Theta \ltimes \Omega$ and there exists a transition $t_1 \| t_2$. Both transitions are firable in the marking $(M_{r0} \cup M_{\Omega 0})$ but $t_1 \| t_2 \succ t_1$ so only $t_1 \| t_2$ can be fired: the reachable marking graph of $\Theta \ltimes \Omega$ thus contains a sequence $(M_{\Theta 0} \cup M_{\Omega 0}) \xrightarrow{r_0.t_1\|t_2} (M_{t_1} \cup M_{t_2})$ and $\rho_0\sigma_0 = \ell_{\Theta\ltimes\Omega}(r_0.t_1\|t_2)$. Suppose now that $\sigma = \sigma_0 \dots \sigma_k \in \Sigma^*$, as $\rho_1 \ni \sigma'$, we can reapply $k$ times the previous reasoning starting from $M_{t_1}$ in $\Theta$ and from $M_{t_2}$ in $\Omega$ and then show the existence of reachable markings $M_{\sigma_1}, \dots M_{\sigma_k}$ so that we prove there exists a sequence of transitions $(M_{\Theta 0} \cup M_{\Omega 0}) \xrightarrow{s} M_{\sigma_k} = (M^\Theta_{\sigma_k} \cup M^\Omega_{\sigma_k})$ such that $\ell_{\Theta\ltimes\Omega}(s) = \rho_k\sigma_k$ in $\Theta \ltimes \Omega$, $M^\Theta_{\sigma_k}$ being a marking of $\Theta$ and $M^\Omega_{\sigma_k}$ being a marking of $\Omega$. As $\rho \in \mathscr{L}(\Theta)$, there must finally exist a firable sequence $M^\Theta_{\sigma_k} \xrightarrow{s'} M_r$ in $\Theta$ such that $\ell_\Theta(s.s') = \ell_\Theta(r) = \rho$. On the other side, as $\sigma \in \mathscr{L}(\Omega)$, it follows that $(M^\Theta_{\sigma_k} \cup M^\Omega_{\sigma_k}) \xrightarrow{s'} (M_r \cup M^\Omega_{\sigma_k})$ belongs to the reachable marking graph of $\Theta \ltimes \Omega$. Therefore, there exists a firable sequence of transitions $(M_{\Theta 0} \cup M_{\Omega 0}) \xrightarrow{s''} (M_r \cup M_{r'})$ in $\Theta \ltimes \Omega$ such that $\ell_{\Theta\ltimes\Omega}(s'') = \rho$. Finally as $M_r$ is a final marking of $\Theta$ and $M_{r'}$ is a final marking of $\Omega$, $(M_r \cup M_{r'})$ is a final marking of $\Theta \ltimes \Omega$, hence the result: $\rho \in \mathscr{L}(\Theta \ltimes \Omega)$.

## A.3 Proof of Corollary 1

**Corollary 1** Let $\Theta$ be the LLPPN of a system over an alphabet $\Sigma$ and $\Omega$ be the LLPPN of a pattern:

$$\mathscr{L}(\Theta \ltimes \overline{\Omega}) = \{\rho \in \mathscr{L}(\Theta) : \forall \sigma \in \mathscr{L}(\Omega) : \rho \not\ni \sigma\}.$$

**Proof.** Let $\rho \in \mathscr{L}(\Theta)$, so there exists in $\Theta$ a transition sequence $M_{\Theta 0} \xrightarrow{t_1} \dots \xrightarrow{t_n} M$ such that $\ell_\Theta(t_1 \dots t_n) = \rho$. Then, by Definition 10, there exists in $\Theta \ltimes \Omega$ a transition sequence $M_{\Theta\Omega 0} \xrightarrow{t'_1} \dots \xrightarrow{t'_m} M'$ such that $\ell_{\Theta\ltimes\Omega}(t'_1 \dots t'_m) = \rho$. Let $M'_1 \cup M'_2 = M'$ with $M'_1$ in $R(\Theta, M_{\Theta 0})$ and $M'_2$ in $R(\Omega, M_{\Omega 0})$ (Lemma 1), if $M'_2 \in Q_\Omega$ then $\rho \in \mathscr{L}(\Theta \ltimes \Omega)$, otherwise $M'_2 \in \overline{Q_\Omega}$ and $\rho \in \mathscr{L}(\Theta \ltimes \overline{\Omega})$. From this it follows that $\mathscr{L}(\Theta) = \mathscr{L}(\Theta \ltimes \Omega) \cup \mathscr{L}(\Theta \ltimes \overline{\Omega})$. The result then follows from Theorem 1.

## A.4 Proof of Theorem 2

**Theorem 2** $\mathscr{L}(N_1\|N_2) = \{\rho \in (\Sigma_1 \cup \Sigma_2)^* : \mathscr{P}_{\Sigma_1\cup\Sigma_2\to\Sigma_1}(\rho) \in \mathscr{L}(N_1) \wedge \mathscr{P}_{\Sigma_1\cup\Sigma_2\to\Sigma_2}(\rho) \in \mathscr{L}(N_2)\}$.

**Proof.** First, let us prove that $\mathscr{L}(N_1\|N_2) \subseteq \{\rho \in (\Sigma_1 \cup \Sigma_2)^* : \mathscr{P}_{\Sigma_1\cup\Sigma_2\to\Sigma_1}(\rho) \in \mathscr{L}(N_1) \wedge \mathscr{P}_{\Sigma_1\cup\Sigma_2\to\Sigma_2}(\rho) \in \mathscr{L}(N_2)\}$. Let $\rho \in \mathscr{L}(N_1\|N_2)$, so there exists in $N_1\|N_2$ a transition sequence $M_0 \xrightarrow{r} M$ with $M \in Q$ and $\ell(r) = \rho$. Recalling the definition of masks $K_{N_1}$ and $K_{N_2}$ (see the proof of Lemma 1).

$$K_{N_1}(t) = \begin{cases} t & \text{if } t \in T_{N_1} \\ \varepsilon & \text{if } t \in T_{N_2} \\ t_1 & \text{if } t = t_1\|t_2 \end{cases} \quad K_{N_2}(t) = \begin{cases} t & \text{if } t \in T_{N_2} \\ \varepsilon & \text{if } t \in T_{N_1} \\ t_2 & \text{if } t = t_1\|t_2 \end{cases},$$

by construction of $N_1\|N_2$, the transition sequence $r$ can be associated to a transition sequence $r_1 : M_{01} \xrightarrow{r_1} M_1$ of $N_1$ and a transition sequence $r_2 : M_{02} \xrightarrow{r_2} N_2$ of $N_2$ such that $r_1 = K_{N_1}(r)$ and $r_2 = K_{N_2}(r)$. Moreover, as $M$ belongs to $Q$, $M_1 \in Q_1$ and $M_2 \in Q_2$, therefore $\ell_1(r_1) \in \mathscr{L}(N_1)$ and $\ell_2(r_2) \in \mathscr{L}(N_2)$. Now, remark that $\mathscr{P}_{\Sigma_1\cup\Sigma_2\to\Sigma_1}(\rho) = \ell_1(r_1)$ as any transition of $N_1\|N_2$ labeled with a label of $\Sigma_1$ comes from $N_1$, therefore $\mathscr{P}_{\Sigma_1\cup\Sigma_2\to\Sigma_1}(\rho) \in \mathscr{L}(N_1)$. Similarly $\mathscr{P}_{\Sigma_1\cup\Sigma_2\to\Sigma_2}(\rho) \in \mathscr{L}(N_2)$.

Now we prove $\mathscr{L}(N_1\|N_2) \supseteq \{\rho \in (\Sigma_1 \cup \Sigma_2)^* : \mathscr{P}_{\Sigma_1\cup\Sigma_2\to\Sigma_1}(\rho) \in \mathscr{L}(N_1) \wedge \mathscr{P}_{\Sigma_1\cup\Sigma_2\to\Sigma_2}(\rho) \in \mathscr{L}(N_2)\}$. Let $\rho$ defined as above, suppose that $\rho$ contains a set of $n, n \geq 0$ events from $\Sigma_1 \cap \Sigma_2$ then $\rho$ can be written as $\rho = \rho_0\sigma_1\rho_1 \ldots \sigma_n\rho_n$ with $\forall i \in \{1,\ldots,n\}, \sigma_i \in \Sigma_1 \cap \Sigma_2$ and $\forall i \in \{0,\ldots,n\}, \rho_i \in ((\Sigma_1 \cup \Sigma_2) \setminus (\Sigma_1 \cap \Sigma_2))^*$. As $\mathscr{P}_{\Sigma_1\cup\Sigma_2\to\Sigma_1}(\rho) \in \mathscr{L}(N_1)$, $\mathscr{P}_{\Sigma_1\cup\Sigma_2\to\Sigma_1}(\rho_0)\sigma_1 \ldots \sigma_n\mathscr{P}_{\Sigma_1\cup\Sigma_2\to\Sigma_1}(\rho_n) \in \mathscr{L}(N_1)$. Then there must exist in $N_1$ a sequence of transitions $M_{01} \xrightarrow{r_1} M^1_{n+1} = M_{01} \xrightarrow{r^1_0} M^1_1 \xrightarrow{t^1_1} M^{1'}_1 \ldots \xrightarrow{r^1_{n-1}} M^1_n \xrightarrow{t^1_n} M^{1'}_n \xrightarrow{r^1_n} M^1_{n+1}$ with $r^1_j \in T^*_1, j \in \{0,\ldots,n\}, \ell_1(t^1_i) = \sigma_i, i \in \{1,\ldots,n\}$ and $M^1_{n+1} \in Q_1$ such that $\ell_1(r_1) = \mathscr{P}_{\Sigma_1\cup\Sigma_2\to\Sigma_1}(\rho)$. Similarly, there must exist in $N_2$ a sequence of transitions $M_{02} \xrightarrow{r_2} M^2_{n+1} = M_{02} \xrightarrow{r^2_0} M^2_1 \xrightarrow{t^2_1} M^{2'}_1 \ldots \xrightarrow{r^2_{n-1}} M^2_n \xrightarrow{t^2_n} M^{2'}_n \xrightarrow{r^2_n} M^2_{n+1}$ with $r^2_j \in T^*_2, j \in \{0,\ldots,n\}, \ell_2(t^2_i) = \sigma_i, i \in \{0,\ldots,n\}$ and $M^2_{n+1} \in Q_2$ such that $\ell_2(r_2) = \mathscr{P}_{\Sigma_1\cup\Sigma_2\to\Sigma_2}(\rho)$. Now, let us prove by induction that for any $k$ from 0 to $n$, $M^1_{k+1} \cup M^2_{k+1}$ is a reachable marking of $N_1||N_2$.

- Case $k = 0$. By construction $M_{01} \cup M_{02}$ is a reachable marking of $N_1||N_2$. $M_{01} \xrightarrow{r^1_0} M^1_1$ and $M_{02} \xrightarrow{r^2_0} M^2_1$ do not share any common label and might even be empty (i.e. $r^1_0 = \varepsilon$ and $M^1_1 = M_{01}$ or $r^2_0 = \varepsilon$ and $M^2_1 = M_{02}$). If both are empty then $M_{01} \cup M_{02} = M^1_1 \cup M^2_1$ so $M^1_1 \cup M^2_1$ is reachable. If now at least one of the run is not empty, by construction of $N_1||N_2$ any sequence of transitions $s$ of $N_1||N_2$ such that $K_{N_1}(s) = M_{01} \xrightarrow{r^1_0} M^1_1$ and $K_{N_2}(s) = M_{02} \xrightarrow{r^2_0} M^2_1$ is actually a run of $N_1||N_2$ from $M_{01} \cup M_{02}$, $M^1_1 \cup M^2_1$ is thus reachable.
- General case. Assume $M^1_k \cup M^2_k$ is a reachable marking of $N_1||N_2$ and consider the part of the runs $M^i_k \xrightarrow{t^i_k} M^{i'}_k \xrightarrow{r^i_k} M^i_{k+1}, i \in \{1,2\}$. $t^i_k$ is firable from $M^i_k$ in $N_i$

for $i \in \{1,2\}$. By construction of $N_1||N_2$, there must exist a transition $t_k^1||t_k^2$ such that $\ell(t_k^1||t_k^2) = \sigma_k$ firable from $M_k^1 \cup M_k^2$ and that leads to the marking $M_k^{1'} \cup M_k^{2'}$ that is thus reachable. Considering now the run parts $M_k^{i'} \xrightarrow{r_k^i} M_{k+1}^i, i \in \{1,2\}$, as $M_k^{1'} \cup M_k^{2'}$ is reachable, this is similar reasoning as in the case $k = 0$: it follows that $M_{k+1}^1 \cup M_{k+1}^2$ is also reachable in $N_1||N_2$.

Consequently, $M_{n+1}^1 \cup M_{n+1}^2$ is a reachable marking of $N_1||N_2$. Finally, remark that $M_{n+1}^1 \in Q_1$ and $M_{n+1}^2 \in Q_2$ so $M_{n+1}^1 \cup M_{n+1}^2 \in Q$, therefore $\rho \in \mathscr{L}(N_1||N_2)$.

## A.5 Proof of Theorem 3

**Theorem 3** $\mathscr{L}(\Gamma) = \{\rho \in \Sigma_\Gamma^*, \exists \rho_1, \rho_2 \in \mathscr{L}(\Theta), \rho_1 \ni \Omega \wedge \rho_2 \not\ni \Omega, \mathscr{P}_{\Sigma \to \Sigma_o}(\rho_1) = \mathscr{P}_{\Sigma \to \Sigma_o}(\rho_2) \wedge \rho_1 = \mathscr{P}_{\Sigma_\Gamma \to \Sigma}(\rho) \wedge \rho_2 = \mathscr{R}_{\Sigma' \to \Sigma}(\mathscr{P}_{\Sigma_\Gamma \to \Sigma'}(\rho))\}$.

**Proof.** The synchronized product $\Gamma$ ensures by Theorem 2 that $\mathscr{L}(\Gamma) = \{\rho \in \Sigma_\Gamma^*, \mathscr{P}_{\Sigma_\Gamma \to \Sigma}(\rho) \in \mathscr{L}(\Theta \ltimes \Omega) \wedge \mathscr{P}_{\Sigma_\Gamma \to \Sigma'}(\rho) \in \mathscr{L}(\Theta' \ltimes \overline{\Omega}')\}$. Theorem 1 states that $\mathscr{P}_{\Sigma_\Gamma \to \Sigma}(\rho) \ni \Omega$ and Corollary 1 states that $\mathscr{R}_{\Sigma' \to \Sigma}(\mathscr{P}_{\Sigma_\Gamma \to \Sigma'}(\rho)) \not\ni \Omega$. Finally, as $\Sigma \cap \Sigma' = \Sigma_o$, we get $\mathscr{P}_{\Sigma \to \Sigma_o}(\mathscr{P}_{\Sigma_\Gamma \to \Sigma}(\rho)) = \mathscr{P}_{\Sigma' \to \Sigma_o}(\mathscr{P}_{\Sigma_\Gamma \to \Sigma'}(\rho)) = \mathscr{P}_{\Sigma_\Gamma \to \Sigma_o}(\rho)$.

# References


Basile F, Chiacchio P, De Tommasi G (2012) On K-diagnosability of Petri nets via integer linear programming. Automatica 48(9):2047–2058

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

Berthomieu B, Ribet PO, 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

Berthomieu B, Peres F, Vernadat F (2006) Bridging the gap between timed automata and bounded time Petri nets. In: 4th International Conference Formal Modeling and Analysis of Timed Systems, Paris, France, pp 82–97

Berthomieu B, Peres F, Vernadat F (2007) Model-checking bounded prioriterized time Petri nets. In: Automated Technology for Verification and Analysis, Springer Verlag, LNCS, vol 4762, pp 523–532

Cabasino MP, Giua A, Seatzu C (2009) Diagnosability of bounded Petri nets. In: Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on, IEEE, pp 1254–1260

Cabasino MP, Giua A, Seatzu C (2014) Diagnosability of discrete-event systems using labeled Petri nets. Automation Science and Engineering, IEEE Transactions on 11(1):144–153

Cimatti A, Pecheur C, Cavada R (2003) Formal verification of diagnosability via symbolic model checking. In: 18th International Joint Conference on Artificial Intelligence, Acapulco, Mexico, pp 363–369

Clarke EM, Grumberg O, Peled DA (1999) Model Checking. MIT press

Genc S, Lafortune S (2007) Distributed diagnosis of place-bordered Petri nets. IEEE Transactions on Automation Science and Engineering 4(2):206–219

Giua A (2007) A benchmark for diagnosis. URL http://www.diee.unica.it/giua/WODES/WODES08/media/benchmark_diagnosis.pdf

Giua A, Seatzu C (2005) Fault detection for discrete event systems using Petri nets with unobservable transitions. In: 44th IEEE Conference on Decision and Control, 2005 and 2005 European Control Conference, Seville, Spain, pp 6323–6328

Gougam HE, Subias A, Pencolé Y (2013) Supervision patterns: Formal diagnosability checking by Petri net unfolding. In: 4th IFAC Workshop on Dependable Control of Discrete Systems, York, United Kingdom, pp 73–78

Grastien A (2009) Symbolic testing of diagnosability. In: International Workshop on Principles of Diagnosis (DX-09), pp 131–138

Haar S, Benveniste A, Fabre E, Jard C (2003) Partial order diagnosability of discrete event systems using Petri net unfoldings. In: 42nd IEEE Conference on Decision and Control, Maui, HI, United States, pp 3748–3753

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

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

Jiang S, 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, Huang Z, Chandra V, Kumar R (2001) A polynomial algorithm for testing diagnosability of discrete-event systems. Transactions on Automatic Control 46(8):1318–1321

Lai S, Nessi D, Cabasino MP, Giua A, Seatzu C (2008) A comparison between two diagnostic tools based on automata and Petri nets. In: 9th International Workshop on Discrete Event Systems, Göteborg, Sweden, pp 144–149

Lamperti G, Zanella M (2006) Flexible diagnosis of discrete-event systems by similarity-based reasoning techniques. Artificial Intelligence 170(3):232–297

Lefebvre D, Delherm C (2007) Diagnosis of DES with Petri net models. IEEE Transaction Automation Science and Engineering 4(1):114–118

Lin F (1994) Diagnosability of discrete event systems and its applications. Journal of Discrete Event Dynamic Systems: Theory and Applications 4(2):197–212

Liu B (2014) An efficient approach for diagnosability and diagnosis of des based on labeled Petri nets - untimed and timed contexts. PhD thesis, Univ. Lille Nord

Liu B, Ghazel M, Toguyéni A (2014a) OF-PENDA: A software tool for fault diagnosis of discrete event systems modeled by labeled Petri nets. In: Proceedings of the 1st International Workshop on Petri Nets for Adaptive Discrete-Event Control Systems (ADECS 2014), no. 1161 in CEUR Workshop Proceedings, pp 20–35

Liu B, Ghazel M, Toguyéni A (2014b) Toward an efficient approach for diagnosability analysis of DES modeled by labeled Petri nets. In: 13th European Control Conference, Strasbourg, France, pp 1293–1298

Pencolé Y (2004) Diagnosability analysis of distributed discrete event systems. In: 16th European Conference on Artificial Intelligence, Valencia, Spain, pp 43–47

Pencolé Y, Cordier MO (2005) A formal framework for the decentralised diagnosis of large scale discrete event systems and its application to telecommunication networks. Artificial Intelligence 164(2):121–170

Pencolé Y, Schumann A, Kamenetsky D (2006) Towards low-cost fault diagnosis in large component-based systems. In: 6th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes, Beijing, China, pp 1473–1478

Peterson JL (1977) Petri nets. ACM Computing Surveys 9(3):223–252

Rozé L, Cordier MO (2002) Diagnosing discrete-event systems : extending the diagnoser approach to deal with telecommunication networks. Journal on Discrete-Event Dynamic Systems : Theory and Applications (JDEDS) 12(1):43–81

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

Schnoebelen P (2003) The complexity of temporal logic model checking. Advances in Modal Logic 4:393–436

Schumann A, Pencolé Y (2007) Scalable diagnosability checking of event-driven systems. In: 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, pp 575–580

Ye L, Dague P (2012) A general algorithm for pattern diagnosability of distributed discrete event systems. In: Proceedings of the 24th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2012), pp 130–137

Yoo TS, Lafortune S (2002) Polynomial-time verification of diagnosability of partially observed discrete-event systems. Transactions on Automatic Control 47(9):1491–1495

Zanella M, Lamperti G (2004) Diagnosis of discrete-event systems by separation of concerns, knowledge compilation, and reuse. In: Proceedings of the 16th European Conference on Artificial Intelligence (ECAI04), pp 838–842

Zaytoon J, Lafortune S (2013) Overview of fault diagnosis methods for discrete event systems. Annual Reviews in Control 37:308–320