Hybrid systems Diagnosis using modified particle Petri nets

Quentin Gaudel, Elodie Chanthery, Pauline Ribot, Euriell Le Corronc

▶ **To cite this version:**

Quentin Gaudel, Elodie Chanthery, Pauline Ribot, Euriell Le Corronc. Hybrid systems Diagnosis using modified particle Petri nets. International Workshop on Principles of Diagnosis (DX), Sep 2014, Graz, Austria. hal-01095110

HAL Id: hal-01095110
https://hal.science/hal-01095110

Submitted on 15 Dec 2014

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Hybrid systems Diagnosis using modified particle Petri nets

Quentin Gaudel[1,2], **Elodie Chanthery**[1,2], **Pauline Ribot**[1,3] and **Euriell Le Corronc**[1,3]
[1]CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France
e-mail: {first.last}@laas.fr
[2]Univ de Toulouse, INSA, LAAS, F-31400 Toulouse, France
[3]Univ de Toulouse, UPS, LAAS, F-31400 Toulouse, France

Abstract

This paper presents an approach of model-based diagnosis for the health monitoring of hybrid systems. These systems have both continuous and discrete dynamics. Modified Particle Petri Nets, initially defined in the context of hybrid systems mission monitoring, are extended to estimate the health state of hybrid systems. This formalism takes into account both uncertainties about the system knowledge and about diagnosis results. A generation of a diagnoser is proposed to track online the system health state under uncertainties by using particle filter. An academic example is used all along the article to illustrate the major concepts.

1 Introduction

Systems have become so complex that it is often impossible for humans to capture and explain their behaviors as a whole, especially when they are exposed to failures. It is therefore necessary to develop tools that can support operator tasks but that also reduce the global costs due to unavailability and repair actions. An efficient diagnosis technique has to be adopted to detect and isolate faults leading to failures.

Recent industrial systems exhibit an increasing complexity of dynamics that are both continuous and discrete. It has become difficult to ignore the fact that most systems are hybrid [1]. In previous works [2], we extended the diagnosis approach proposed in [3] in order to integrate diagnosis and prognosis for hybrid systems. The approach uses hybrid automata and stochastic models for the system degradation. The main drawback of this approach is that the discrete event system oriented diagnosis framework explodes in number of states and it does not seem the best suited for the incorporation of the highly probabilistic prognosis task. The goal of this paper is to use a new formalism, based on Modified Particle Petri Nets (MPPN) ([4]), both to specify the system behavior in all its complexity, that is hybrid but also uncertain, and to track system current health state with a diagnoser-like representation.

The paper is organized as follows. Section 2 presents related work on diagnoser specification based on Petri Nets formalism in the special case of hybrid systems under uncertainties. Section 3 recalls the MPPN framework. Section 4 presents how MPPN is applied to health monitoring. Section 5 explains the generation of a behavioral diagnoser using the MPPN formalism. Some conclusions and future work are discussed in the final section.

2 Related work

The diagnoser approach was introduced by [5]. The diagnoser is basically a monitor that is able to process any possible observable event on the system. It consists in recording these observations and providing the set of possible faults whose occurrence is consistent with this observation.
Some approaches, like [6][7][8], extend the diagnoser introduced by [5] to DES modelled by Petri nets. [7] proposes a distributed version of the diagnoser in "Petri net diagnosers". In [8], the authors study the diagnosability of a system, inspired by the diagnosability approach for finite state automata proposed by Sampath [5]. They build the Modified Basis Reachability Graph in order to build the reachability diagnoser that is represented as a graph. However, none of these approaches does take into account continuous aspects, nor uncertainties in the system.
Some works try to take into account uncertainties [9][10][11]. [9] uses partially observed Petri nets. Partially observed Petri net are transformed into an equivalent labelled Petri net and an online monitor has been built to diagnose faults and provide beliefs (degree of confidence) regarding the occurrences of faults. However, this approach is limited, because it does not take into account uncertainties about the model or the event observations, only in diagnosis results. [10] proposes to reduce the explosion of the state space estimation by introducing generalized markings (negative token) to take into account uncertainty about the firing of transitions. [11] uses the Stochastic Petri nets (SPN) to build a formal model of each component of integrated modular avionics architecture. However, for all these approaches, no continuous aspect in the model is taken into account.

To have a more compact representation and to capture all uncertainties related to the system, to the observations and to the diagnosis results, we propose to consider the formalism of Modified Particle Petri Nets (MPPN) defined in [4]. MPPN are an extension of Particle Petri nets [12] that combines a discrete event model (Petri net) and a continuous model (differential equations). The main advantage of MPPN is that uncertainties and hybrid dynamics are taken into account. Particle filter is used to integrate probabilities in the continuous state estimation process. MPPN have

been used for supervision and planning, but never for health monitoring, diagnosis and/or prognosis. In [4], the application is essentially based on the mission monitoring. It does not consider different health states for the system. Moreover, there is no matching with the diagnoser approach and the problem of ambiguous state tracking is not considered. In [13], the authors propose a design approach for the specification and the realization of dynamic system monitoring. They present the integration of system design and monitoring into a unified framework for the reuse of component descriptions and the automatic monitoring component generation. They automatically convert MPPN into an XML description and the monitoring parameters are added to the file. However, this observer is not formally defined, and no equivalence with a diagnoser is given. Moreover, there is no notion of health state for the system. Our paper proposes to use MPPN for health monitoring, taking into account the hybrid aspect and all the possibles uncertainties of real systems. Then it formally specifies a diagnoser-like object for building an on-line diagnosis.

3 Modified Particle Petri Nets for monitoring

In this section, the Modified Particle Petri Nets (MPPN) formalism is described according to the work of [4]. First the model structure is detailed, then its online use is presented.

3.1 Definition

Modified Particle Petri Nets are defined as a tuple $< P, T, Pre, Post, X, C, \gamma, \Omega, M_0 >$ where:

- P is the set of places, partitioned into numerical places P^N and symbolic places P^S.
- T is the set of transitions (numerical T^N, symbolic T^S and mixed T^M).
- Pre and $Post$ are the incidence matrices of the net, of dimension $|P| \times |T|$.
- $X \subset \Re^n$ is the state space of the numerical state vector.
- C is the set of dynamics equations of the system associated with numerical places, representing continuous state evolution.
- $\gamma(p^S)$ is the application that associates tokens with each symbolic places $p^S \in P^S$.
- Ω is the set of conditions associated with the transitions (numerical Ω^N and symbolic Ω^S).
- M_0 is the initial marking of the net.

The marking of the net is composed of tokens, that can be numerical tokens (particles) or symbolic tokens (configurations).

A numerical place $p^N \in P^N$ is associated with a set of dynamics equations representing the continuous behavior of the system. Numerical places thus model continuous dynamics of the system. Numerical places are marked by a set of particles $\pi_k^i = [x_k^i, \mathrm{w}_k^i]$ with $i \in \{1, ..., |M_k^N|\}$ where M_k^N is the set of all the particles in the net at time k. Particles are defined by their corresponding numerical state vector $x_k^i \in X$ and their weight $w_k^i \in [0, 1]$ at time k. The set of particles represents an uncertain distribution over the value of the numerical state vector.

Symbolic places model the behavioral modes of the system. A symbolic place $p^S \in P^S$ is marked by configurations δ_k^j with $j \in \{1, ..., |M_k^S|\}$ where M_k^S is the set of configurations in the net at time k. The set of configurations represents all the possible current modes of the system.

The marking M_k of the MPPN at time k consists of both kinds of tokens:

$$M_k = \{M_k^S, M_k^N\} \tag{1}$$

3.2 Firing rules

A transition models a change in the continuous dynamic and/or a change of the system mode. A symbolic transition is conditioned by an observable discrete event. A numerical transition is conditioned by a set of constraints on continuous observable variables. Finally, a mixed transition is conditioned by an observable discrete event and a set of constraints on continuous observable variables.

Let $Pre(t_j)$ be the set of input places of a transition $t_j \in T$:

$$Pre(t_j) = \{p_i | Pre(i, j) \neq 0, i \in \{1, ..., |P|\}\} \tag{2}$$

As well, $Post(t_j)$ is the set of its output places:

$$Post(t_j) = \{p_i | Post(i, j) \neq 0, i \in \{1, ..., |P|\}\} \tag{3}$$

$\forall p_i \in P$, $M_k(p_i)$ is the set of tokens in p_i at time k and $m_k(p_i) = |M_k(p_i)|$ is the number of tokens in p_i at time k.

Definition 1. *A numerical or a symbolic transition t_j is fire-enabled at time k if:*

$$\forall p_i \in Pre(t_j), m_k(p_i) \geq Pre(i, j) \tag{4}$$

A numerical transition $t_j^N \in T^N$ is associated with conditions $\Omega^N(t_j^N)$, where $\Omega^N(t_j^N)(\pi) = 1$ if the particle satisfies the conditions. For example, if $\pi = [x, \mathrm{w}]$ follows the constraint equation c and b is a trigger value, a numerical condition can be defined as $\Omega^N(t_j^N)(\pi) = (c(x) > b)$. $\Omega^S(t_j^S) = occ(e)$ represents the conditions assigned to a symbolic transition $t_j^S \in T^S$. $occ(e)$ is a boolean indicator of the occurrence of the discrete event e : $occ(e) = 1$ if e has occurred. Then, a configuration δ satisfies the condition $\Omega^S(t_j^S)$ when $\Omega^S(t_j^S)(\delta) = 1$, ie. when the event e has occurred.

We are now going to define transition firing rules. Firing rules of a transition $t_j \in T$ use the following set of variables over their domains of definition. It is suppose that:
$p^N \in (Pre(t_j) \cap P^N), p'^N \in (Post(t_j) \cap P^N)$,
$p^S \in (Pre(t_j) \cap P^S), p'^S \in (Post(t_j) \cap P^S)$.

If $t_j \in (T^N \cup T^M)$, let $\mathcal{S}_k^N(p^N)$ be the set of particles in p^N that satisfy the conditions $\Omega^N(t_j)$ at time k:
$\mathcal{S}_k^N(p^N) \subseteq M_k^N(p^N)$ with $\pi \in \mathcal{S}_k^N(p^N)$ if $\Omega^N(t_j)(\pi) = 1$ at time k. As well, if $t_j \in (T^S \cup T^M)$, $\mathcal{S}_k^S(p^S)$ is the set of configurations in p^S that satisfy the conditions $\Omega^S(t_j)$ at time k:
$\mathcal{S}_k^S(p^S) \subseteq M_k^S(p^S)$ with $\delta \in \mathcal{S}_k^S(p^S)$ if $\Omega^S(t_j)(\delta) = 1$ at time k.

The numerical firing uses the concept of classical firing with the particles satisfying the numerical condition and the concept of pseudo-firing (ie. duplication) for the configurations. The duplication of configurations represents uncertainty about the occurrence of an unobservable discrete event.

Definition 2. *The firing of a fire-enabled numerical transition $t_j^N \in T^N$ at time k is defined by:*

$$\begin{cases} M_{k+1}^N(p^N) = M_k^N(p^N) \backslash \mathcal{S}_k^N(p^N) \\ M_{k+1}^N(p'^N) = M_k^N(p'^N) \cup \mathcal{S}_k^N(p^N) \end{cases} \tag{5}$$

$$\begin{cases} M^S_{k+1}(p^S) = M^S_k(p^S) \\ M^S_{k+1}(p'^S) = M^S_k(p'^S) \cup M^S_k(p^S) \end{cases} \quad (6)$$

An example of a numerical firing from marking at time k to marking at time $k+1$ is illustrated in Figure 1(a).

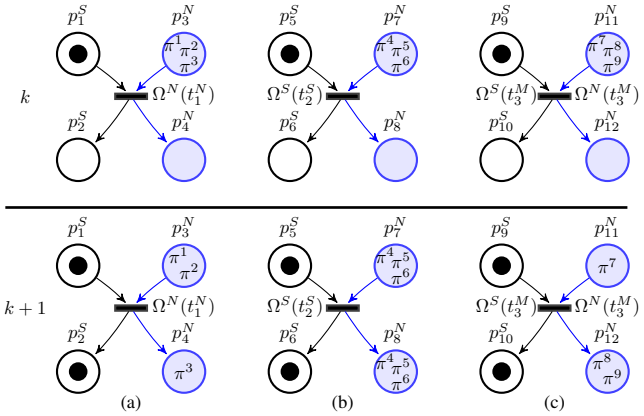

Figure 1: Illustration of firing rules of numerical (a), symbolic (b) and hybrid (c) fire-enabled transitions.

In this example, t^N_1 only has a numerical condition because it is a numerical transition. Particle π^3 satisfies the numerical condition $\Omega^N(t^N_1)$ and thus is moved through the fire-enabled transition t^N_1 to p^N_4. The configuration in place p^S_1 is duplicated in p^S_2.

The symbolic firing uses the concept of pseudo-firing for particles and configurations. The pseudo-firing of all the tokens models uncertainty about the non occurrence of an observable discrete event.

Definition 3. *The firing of a fire-enabled symbolic transition $t^S_j \in T^S$ at time k is defined by:*

$$\begin{cases} M^N_{k+1}(p^N) = M^N_k(p^N) \\ M^N_{k+1}(p'^N) = M^N_k(p'^N) \cup M^N_k(p^N) \end{cases} \quad (7)$$

$$\begin{cases} M^S_{k+1}(p^S) = M^S_k(p^S) \\ M^S_{k+1}(p'^S) = M^S_k(p'^S) \cup M^S_k(p^S) \end{cases} \quad (8)$$

Figure 1(b) illustrates an example of a symbolic firing. The symbolic transition t^S_2 only has a symbolic condition and is fire-enabled. No token satisfies the symbolic condition $\Omega^S(t^S_2)$, however all tokens are duplicated.

Mixed transitions are introduced in [4] to model the interaction between discrete events and system continuous dynamics. In the referred article, they were called "hybrid transitions". A mixed transition merges a symbolic transition with a numerical transition to correlate discrete observations with continuous observations. The firing of the symbolic transition only depends on a discrete event, but the simultaneous firing of the numerical transition models the dependency of the mixed transition on the symbolic part because discrete events are part of the process behavior. A mixed transition $t^M_j \in T^M$ is then associated with both numerical conditions $\Omega^N(t^M_j)$ and with symbolic conditions $\Omega^S(t^M_j)$.

Definition 4. *A mixed transition $t^M_j \in T^M$ is fire-enabled at time k if $\forall p_i \in Pre(t^M_j)$:*

$$\begin{cases} m_k(p^S_i) \geq Pre(i,j), & if \quad p^S_i \in Pre(t^M_j) \cap P^S \\ m_k(p^N_i) \geq 0, & if \quad p^N_i \in Pre(t^M_j) \cap P^N \end{cases} \quad (9)$$

The mixed firing uses the concept of classical firing with the particles satisfying the numerical condition and the concept of pseudo-firing with the configurations satisfying the symbolic condition. The pseudo-firing of configurations models uncertainty about the occurrence of an observable discrete event which is supported by a change of continuous dynamics.

Definition 5. *The firing of a mixed transition $t^M_j \in T^M$ at time k is defined by:*

$$\begin{cases} M^N_{k+1}(p^N) = M^N_k(p^N) \backslash \mathcal{S}^N_k(p^N) \\ M^N_{k+1}(p'^N) = M^N_k(p'^N) \cup \mathcal{S}^N_k(p^N) \end{cases} \quad (10)$$

$$\begin{cases} M^S_{k+1}(p^S) = M^S_k(p^S) \\ M^S_{k+1}(p'^S) = M^S_k(p'^S) \cup \mathcal{S}^S_k(p^S) \end{cases} \quad (11)$$

An example of a mixed firing is illustrated in Figure 1(c). t^M_3 is a mixed transition therefore it has a symbolic condition and a numerical condition. The configuration in place p^S_9 is duplicated because it satisfies the symbolic condition $\Omega^S(t^M_3)$. Regarding the numerical part, particles π^8 and π^9 satisfy $\Omega^N(t^M_3)$ and so they are moved through t^M_3. Furthermore, π^7 stays in place p^N_{11} because it does not satisfy $\Omega^N(t^M_3)$.

Heterogeneous systems are defined as systems that have a discrete, continuous or both discrete and continuous dynamics. MPPN can easily model heterogeneous systems by using only the symbolic or numerical subpart of the model or both in the case of hybrid systems.

3.3 State estimation

The problem of hybrid state estimation in MPPN has been introduced in [4] and consists of a prediction step and a correction step, illustrated in Figure 2.

For the sake of clarity in this paper we assume that a hybrid state is represented by a couple (p^S_i, p^N_j) of a symbolic place and a numerical place. The initial marking of the MPPN is $M_0 = \{M^S_0, M^N_0\}$ and the estimated marking at time k is $\hat{M}_k = \{\hat{M}^S_k, \hat{M}^N_k\}$ where $\hat{M}_k = \hat{M}_{k|k}$. The observations start at time $k = 1$, $O_1 = (O^S_1, O^N_1)$ where O^S and O^N respectively represent the observations corresponding to the symbolic part and the numerical part.

(1) The prediction step is based on the evolution of the MPPN marking and on the estimation of the particle values. It aims at determining all possible next states of the system $\hat{M}_{k+1|k} = \{\hat{M}^S_{k+1|k}, \hat{M}^N_{k+1|k}\}$. A noise is added during the particle values update to take into account uncertainty about the dynamics equations and thus about the continuous system model.

(2) The correction step is based on the update of the prediction according to new observations on the system.

(a) A numerical correction, based on particle filter algorithms, produces a probability distribution Pr_{DN} of the particles $\hat{M}^N_{k+1|k+1}$ over the value

of the numerical state vector. At this step, particle weights are updated using a probability distribution function depending on a random noise that models uncertainty about continuous observations O^N_{k+1}.

(b) A symbolic correction then computes a probability distribution $\Pr_{DS}$ over the symbolic states of the system, depending on discrete observations O^S_{k+1} and on $\Pr_{DN}$ making the process hybrid.

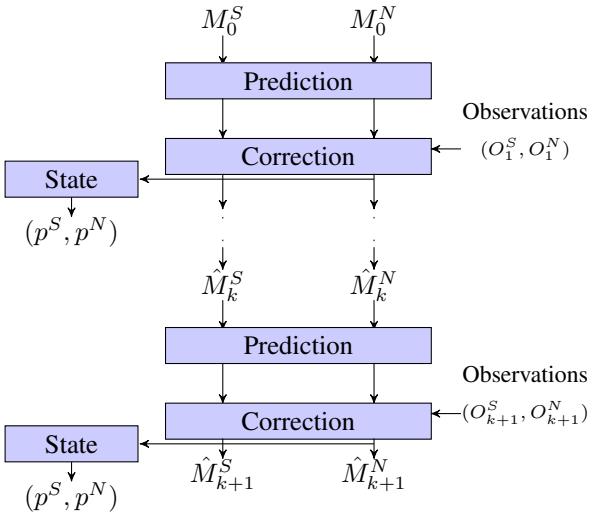

Figure 2: Hybrid sate estimation process of MPPN.

Finally, in order to update the complete predicted marking $\hat{M}_{k+1|k}$, a decision making method is required. The result of the whole state estimation process is the estimated marking at time $k+1$, $\hat{M}_{k+1|k+1} = \{\hat{M}^S_{k+1|k+1}, \hat{M}^N_{k+1|k+1}\}$.

Modified Particle Petri Nets have been originally designed to monitor hybrid system mission in [13]. The main advantage provided by MPPN is the way they manage uncertainties. In this article, we will focus on a way to use them in a context of health monitoring.

4 Application to health monitoring

The main objective of the system health monitoring is to determine the health state of the system at any time [2]. Diagnosis is used to identify the probable causes of the failures by reasoning on system observation. Thus diagnosis reasoning consists in detecting and isolating faults that may cause a system failure. Results of the diagnosis function lead to the current health state of the system. We are interesting in representing changes in system dynamics when one or several anticipated faults happen. Thinking that way, we define a *health mode* by a discrete health state coupled to a continuous behavior. As long as the system does not encounter any fault, it is in a *nominal mode*. We assume that tracked faults are permanent. This means that once a fault happens, the system moves from a nominal mode to a *degraded mode* or faulty mode. Without repair, system evolution is unidirectional and ends with a *failure mode* whereas the system is not operational anymore. This evolution is illustrated in Figure 3.

With the definition of the MPPN abstraction provided in previous sections, it is possible to model a hybrid system behavior. Indeed, MPPN numerical places can be used to represent system dynamics, and symbolic places can be used to represent the different discrete health states of the system. Systems dynamics are then represented by differential equations. Thus, a hybrid state (p^S_i, p^N_j) will represent a health mode of the system. We designate by $Q = \{q_m\}$ the set of health modes of our system:

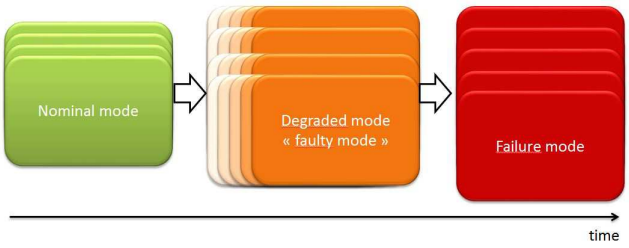

Figure 3: Unidirectional system evolution without maintenance or repair action.

$$q_m = (p^S_i, p^N_j) \in Q \;\; if \;\; \exists t_l \in T, (p^S_i, p^N_j) \in (Post(t_l))^2 \tag{12}$$

Using places that way, it becomes possible to use symbolic conditions to model the occurrence of observable discrete events belonging to Σ_o and unobservable discrete events belonging to Σ_{uo} (faults, mission events, interaction with the environment, etc ...). $\Sigma = \Sigma_o \cup \Sigma_{uo}$ is defined as the set of discrete events of the system.

An example of a system behavioral model is described in Figure 4. In this example, the system has three different dynamics represented by p^N_5, p^N_6, p^N_7 and four different health states p^S_1, p^S_2, p^S_3 and p^S_4. Using Equation 12, five health modes are distinguishable.
Health modes $q_1 = (p^S_1, p^N_5)$ and $q_2 = (p^S_1, p^N_6)$ are two nominal modes changing from the one to the other when condition $\Omega^S(t^S_1) = occ(e_1)$ or condition $\Omega^S(t^S_2) = occ(e_2)$ is satisfied. These conditions represent respectively the occurrence of observable events $e_1 \in \Sigma_o$ and $e_2 \in \Sigma_o$ supporting a change of behavior between p^N_5 and p^N_6. Health modes $q_3 = (p^S_2, p^N_6)$ and $q_4 = (p^S_3, p^N_6)$ are two degraded modes reachable from health mode q_1 by satisfying the conditions $\Omega^S(t^S_3) = occ(f_1)$ and $\Omega^S(t^S_4) = occ(f_2)$ respectively. These two conditions represent respectively the occurrence of two unobservable fault events $f_1 \in \Sigma_{uo}$ and $f_2 \in \Sigma_{uo}$. Finally, $q_5 = (p^S_4, p^N_7)$ is a failure mode in which both f_1 and f_2 occurred and is reachable from the two degraded modes. Therefore $\Omega^N(t^S_5) = occ(f_1)$ is associated to the occurrence of f_1 and $\Omega^S(t^S_6) = occ(f_2)$ is associated with the occurrence of f_2.

Section 5 will now present a methodology to build a state tracker object called a diagnoser from the behavioral system model.

5 Diagnosis

In health monitoring, diagnosis is used to track system current health state. To do so, a common way is to generate a diagnoser of the system from the system model [5]. The diagnoser is basically a monitor that is able to process any possible observable event on the system. It consists in recording these observations and providing the set of possible faults whose occurrence is consistent with these observations.

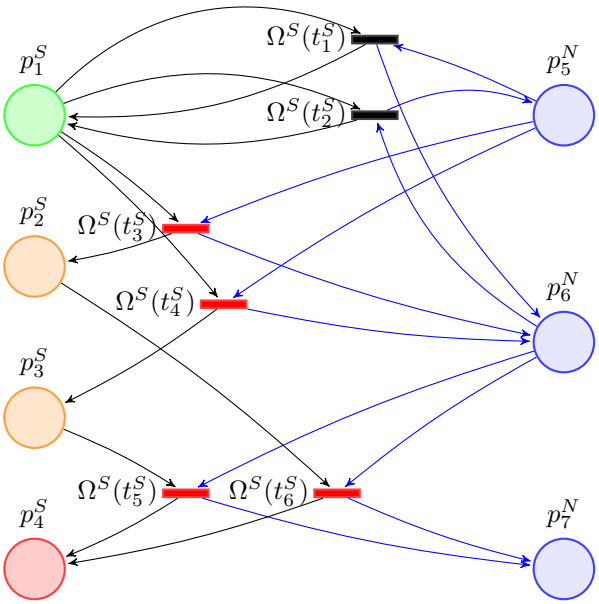

Figure 4: Example of system behavioral model using MPPN.

Concerning hybrid systems, one approach is to build a hybrid diagnoser [3] from a hybrid automaton describing the system. The major idea is to abstract the continuous part of the system to only work with a discrete view of the system. This abstraction is done by using consistency tests, that take the form of a set of analytical redundancy relations (ARR). The diagnoser method is then directly applied on the resulting discrete event system. In previous works [2], we extended this approach in order to integrate diagnosis and prognosis for hybrid systems. The main drawback of this approach is that the DES oriented diagnosis framework seems not the best suited for the incorporation of the highly probabilistic prognosis task. With the MPPN representation, we succeed in capturing all the uncertainties on the state knowledge, but also on the observations. Consequently, we have to develop a new diagnoser build from an MPPN. Moreover, the classical diagnoser is a finite state machine. If this theoretical object is very interesting for studying properties on system, like diagnosability or controllability, it is absolutely not suited for embedded systems, because the number of states of the diagnoser explodes for large models. Consequently, we choose to build a diagnoser based on a MPPN model for the following reasons:

- it is not necessary to transform the specification of the behavior of the system in another formalism,
- MPPN model captures all the uncertainties, so the integration with prognosis becomes more natural,
- this representation is more compact than hybrid automaton description, so the problem of embeddability of the diagnoser is reduced.

The diagnoser takes as input the MPPN specifying the behavior of the system and the set of online observations on the system. The output of the diagnoser is an estimation of the health state of the system that takes the form of a marking of the diagnoser $\Delta_k = \hat{M}_k = \{\hat{M}_k^S, \hat{M}_k^N\}$. Next sections describe how to generate a diagnoser from an MPPN specifying the behavior of a system, then define what is finally called a diagnosis and how this object may be used for health monitoring.

5.1 Diagnoser generation based on MPPN

The goal of this section is to generate a MPPN that is able to monitor the system current health state thanks to the observations. Let suppose that the MPPN specifying the behavior of the system is a tuple $< P, T, Pre, Post, X, F, \gamma, \Omega, M_0 >$ as defined in Section 3.1. The set of places of the diagnoser remains the same as the one of the system. Concerning the transitions, there are two aspects to take into account.

First, it is necessary to follow the continuous behavior of the system with information issued from the observed variables of the system. A set of analytical redundancy relations (ARR) can be generated from the set of differential equations C of the system model. In the linear case, ARRs can be computed by using the parity space approach [14]. The parity space approach has been extended to multi-mode systems in [15]. In our case, a relation ARR_i is associated to each numerical place p_i^N. A numerical condition $\Omega^N(t_l)$ associated with a transition t_l linking two numerical places p_i^N and p_j^N carries ARR_{ij} satisfaction test, with $(i,j) \in \{1,...,|P^N|\}^2$ and $l \in \{1,...,|T|\}$. This means that $\Omega^N(t_l)(\pi)$ is satisfied when ARR_{ij} is satisfied for π. ARRs are satisfied if the observations satisfy the model constraints. Since ARRs are constraints that only contain observable variables, they can be evaluated online with the incoming observations given by the sensors. It is thus possible to check the consistency of the observed system behavior with the predicted one.

Secondly, because the diagnoser only captures the observable behavior of the system, a condition representing the occurence of an unobservable discrete event would never be satisfied. Consequently, all the symbolic conditions representing the occurences of unobservable events are removed from Ω without loss of information. Concerning the observable discrete part of the system, occurrences of observable discrete events will be used as symbolic condition triggers.

Once the system behavioral model is defined and all numerical conditions are computed from the ARRs generation, the corresponding diagnoser can be generated with the following steps:

Step 1: Add corresponding numerical conditions $\Omega^N(t_j^S)$ to every symbolic transition $t_j^S \in T^S$, with $j \in \{1,...,|T|\}$. As a result, the symbolic transition t_j^S will be upgraded into a mixed transition $t_j^M \in T^M$.

Step 2: Remove, from any mixed transition $t_j^M \in T^M$, symbolic conditions $\Omega^S(t_j^M)$ covering the occurrence of an unobservable event, because these conditions would never be satisfied. Consequently, the mixed transition t_j^M is transformed in a numerical transitions $t_j^N \in T^N$.

Ambiguity: Hybrid system diagnosis consists in determining the health state of the system wherein observations are consistent. Diagnosis challenge is the ability to diagnose anticipated but unobservable faults in the system. In

this context, modeling unobservable events can lead to ambiguity in the diagnoser. Indeed, the occurrence of several faults that can not be distinguishable with the observations of the systems will lead to ambiguous health states for the diagnoser. Therefore, a third step is needed during the diagnoser generation to track ambiguity. To do so, it is necessary to define a merger property to merge two numerical transitions. Two numerical transitions are mergeable if they are conditioned by the same dynamics change and if they share the same symbolic places in their sets of inputs places. In a more formal way:

Definition 6. *Two numerical transitions* $(t_i^N, t_j^N) \in (T^N)^2$, *with* $(i,j) \in \{1,...,|T^N|\}^2$ *and* $i \neq j$ *are mergeable if :*

$$(Pre(t_i^N) = Pre(t_j^N)) \wedge (Post(t_i^N) \cap P^N \cap Post(t_j^N) \neq \emptyset) \quad (13)$$

Note that condition (13) implies that the two transitions share the same numerical condition: $\Omega^N(t_i^N) = \Omega^N(t_j^N)$.

Step 3: Merge all mergeable transitions while there is at least two mergeable transitions using the following merging definition:

Definition 7. *The merging of two mergeable numerical transitions* $(t_i^N, t_j^N) \in (T^N)^2$, *with* $(i,j) \in \{1,...,|T^N|\}^2$ *and* $i \neq j$ *is defined by two steps as follows:*

(1) *Creation of a new transition* t_{ij}^N *characterized by:*

$$\begin{cases} Pre(t_{ij}^N) & = & Pre(t_i^N) \\ Post(t_{ij}^N) & = & Post(t_i^N) \cup Post(t_j^N) \\ \Omega^N(t_{ij}^N) & = & \Omega^N(t_i^N) \end{cases} \quad (14)$$

(2) *Introduction of* t_{ij}^N *and deletion of* t_i^N *and* t_j^N *in* T*:*

$$T = (T \backslash \{t_i^N, t_j^N\}) \cup \{t_{ij}^N\} \quad (15)$$

The resulting diagnoser of the model in Figure 4, after computing the third steps above, is presented in Figure 5.

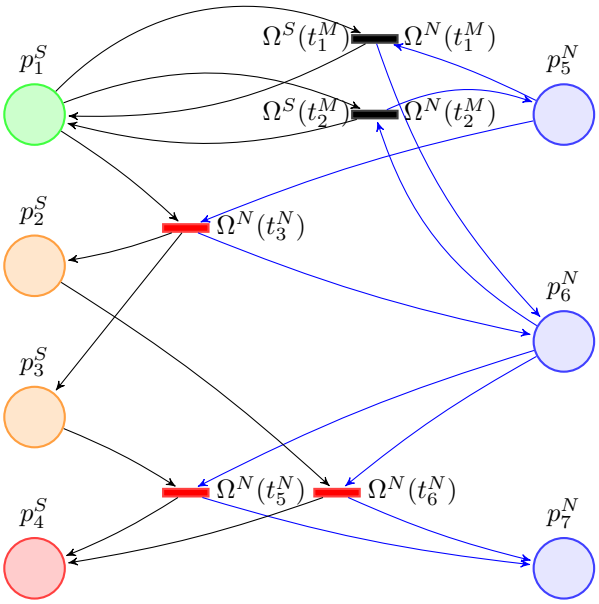

Figure 5: Example of diagnoser of system using MPPN.

In Figure 5, performing Step 1 has generated numerical condition Ω^N to every transition. Indeed, all transitions where supported by a change of dynamics that can be observed with the generation of the ARR. After these steps, all transitions are upgraded into mixed transitions. As there were unobservable events, symbolic conditions associated with the occurrence of f_1 and f_2 have been removed from the diagnoser model during Step 2, transforming t_3, t_4, t_5 and t_6 into numerical transitions. Finally, because transitions t_3 and t_4 were generating a change of dynamics from p_1^N to p_2^N, they were mergeable and thus have been merged into one single numerical transition t_3^N.

5.2 Diagnosis results

The diagnosis is defined at each clock tick as the state of the diagnoser. By using the MPPN, the diagnosis Δ_k at time k is the distribution of health mode believes that depends on particle values and weights and is deduced from the marking of the diagnoser at time k :

$$\Delta_k = \hat{M}_k = \{\hat{M}_k^S, \hat{M}_k^N\} \quad (16)$$

The marking $\hat{M}_k$ indicates the belief on the fault occurrences. It gives the same information than a classical diagnoser mode in terms of faults occurrences, with the same ambiguity. The difference is that in a classical diagnoser, every possible diagnosis has the same belief degree. With MPPN-based diagnoser, the ambiguity is valued by the knowledge about the weights of each particle of the marking.

Consequently, using the diagnosis results for health management becomes easier. Indeed, in the case of classical diagnoser, it is very difficult to "choose" a belief state for the system in case of decision making. It is then very important to obtain the less ambiguous diagnosis as possible. In the case of MPPN-based diagnoser, each possible state of the system is valued, so it is easy to evaluate the more probable state at each clock tick.

6 Conclusion and future work

This paper proposed to use the MPPN formalism to represent hybrid systems. The main advantage of this formalism is to take into account all the possible uncertainties about the knowledge of the system and the online observations. MPPN can be used to model a diagnoser to monitor both discrete and continuous behaviors of the system. Moreover this representation is intuitive and compact and then facilitates the modeling task. The methodology is illustrated with an academic example. The building of such a diagnoser is a first step to perform prognosis and health management of hybrid systems under uncertainty. Moreover, diagnosis results can be used as probability distributions for decision making.

In future works, we will also consider the system degradation depending on the health state of the system and propose a prognosis method on MPPN. Moreover, this work will be implemented and tested on an embedded system. The prognosis methodology will be formally described considering the InterDP framework introduced in [2] that interleaves diagnosis and prognosis methods to let results be more accurate.

References

[1] T. Henzinger. The theory of hybrid automata. In *Proceedings of the* 11^{th} *Annual IEEE Symposium on Logic in Computer Science*, pages 278–292, 1996.

[2] Elodie Chanthery and Pauline Ribot. An integrated framework for diagnosis and prognosis of hybrid systems. In *the 3rd Workshop on Hybrid Autonomous System (HAS)*, Roma, Italy, March 2013.

[3] Mehdi Bayoudh, Louise Travé-Massuyes, Xavier Olive, and Thales Alenia Space. Hybrid systems diagnosis by coupling continuous and discrete event techniques. In *Proceedings of the IFAC World Congress*, pages 7265–7270, Seoul, Korea, 2008.

[4] L Zouaghi, A Alexopoulos, A Wagner, and E Badreddin. Modified particle petri nets for hybrid dynamical systems monitoring under environmental uncertainties. In *IEEE/SICE International Symposium on System Integration (SII)*, pages 497–502. IEEE, 2011.

[5] Meera Sampath, Raja Sengupta, Stéphane Lafortune, Kasim Sinnamohideen, and Demosthenis Teneketzis. Diagnosability of discrete-event systems. *IEEE Transactions on Automatic Control*, 40(9):1555–1575, 1995.

[6] Renée Boubour, Claude Jard, Armen Aghasaryan, Eric Fabre, and Albert Benveniste. A petri net approach to fault detection and diagnosis in distributed systems. part i: application to telecommunication networks, motivations, and modelling. In *the 36th Conference on Decision & Control*, San Diego, Califomia, USA, December 1997.

[7] Sahica Genc and Stéphane Lafortune. Distributed diagnosis of place-bordered petri nets. *IEEE Transactions on Automation Science and Engineering*, 4(2):206–219, April 2007.

[8] Maria Paola Cabasino, Alessandro Giua, and Carla Seatzu. Diagnosability of discrete-event systems using labeled petri nets. *IEEE Transactions on Automation Science and Engineering*, 11(1):144–153, 2014.

[9] Yu Ru and Christoforos N. Hadjicostis. Fault diagnosis in discrete event systems modeled by partially observed petri nets. *Discrete Event Dynamic Systems*, 19(4):551–575, 2009.

[10] F. Basile, P. Chiacchio, and G. De Tommasi. Fault diagnosis and prognosis in petri nets by using a single generalized marking estimation. In *7th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes (SAFEPROCESS)*, Barcelona, Spain, 2009.

[11] Wan Jianxiong, Xiang Xudong, Bai Xiaoying, Lin Chuang, Kong Xiangzhen, and Li Jianxiang. Performability analysis of avionics system with multilayer hm/fm using stochastic petri nets. *Chinese Journal of Aeronautics*, 26(2):363–377, 2013.

[12] Charles Lesire and Catherine Tessier. Particle petri nets for aircraft procedure monitoring under uncertainty. In *Applications and Theory of Petri Nets*, pages 329–348. Springer, 2005.

[13] L Zouaghi, A Alexopoulos, A Wagner, and E Badreddin. Probabilistic online-generated monitoring models for mobile robot navigation using modified petri net. In *15th International Conference on Advanced Robotics (ICAR)*, pages 594–599. IEEE, 2011.

[14] Marcel Staroswiecki and G Comtet-Varga. Analytical redundancy relations for fault detection and isolation in algebraic dynamic systems. *Automatica*, 37(5):687–699, 2001.

[15] Vincent Cocquempot, Touria El Mezyani, and Marcel Staroswiecki. Fault detection and isolation for hybrid systems using structured parity residuals. In *5th Asian Control Conference.*, volume 2, pages 1204–1212. IEEE, 2004.