Modeling complex systems with Heterogeneous Petri Nets (HtPN)

Amaury Vignolles, Pauline Ribot, Elodie Chanthery

Modeling complex systems with Heterogeneous Petri Nets (HtPN)

Amaury Vignolles[1] and **Pauline Ribot**[2] and **Elodie Chanthery**[3]
[1,2,3] LAAS-CNRS, Université de Toulouse, CNRS, INSA, UPS, Toulouse, France
e-mail: {avignolles, pribot, echanthe}@laas.fr

Abstract

This article presents a new formalism based on Petri nets to model aging hybrid systems and take uncertainties into account: the Heterogeneous Petri Nets (HtPN). A new definition of hybrid systems is introduced: a class of systems that can present continuous, discrete or hybrid dynamics by subparts. After presenting the formalism and explaining how it can be used to specify the behavior of complex dynamic systems, the paper presents its application on two examples, a production system from Motorola and a photovoltaic panel system.

1 Introduction

The correct modeling of a system is essential, because it allows, among other things, to specify, control and monitor accurately the system considered. However, with the global complexification of systems, it is necessary for modeling formalisms to evolve as well. For this purpose, a new formalism based on the well-known Petri Nets formalism [1] is introduced in this paper. This formalism, named Heterogeneous Petri Nets (HtPN), was created to fulfill specific needs, although we claim that it can be used to represent everything the usual Petri Net can do.

The specific needs we focus on are modeling and monitoring the health of a hybrid system under all kinds of uncertainty. In short, a hybrid system, according to our definition, is a system in which purely discrete, purely continuous or hybrid parts (i.e. parts mixing discrete and continuous aspects) are all linked and communicate with each other. For example, cyber-physical systems [2] main characteristic is the different nature of their elements. They integrate various devices which have heterogeneous dynamics. Hence, we need a formalism able to represent such hybrid systems. We also need to be able to represent different types of uncertainty (on modeling or observations, for example) through parallelism or noise functions. Finally, monitoring the health of hybrid systems has become such an important challenge for the industry that we wanted to define a formalism that is easy and natural to understand and appropriate. For health monitoring purpose, the formalism has to represent the aging of the system, through degradation dynamics for example. To sum up, the new formalism should make it possible to:

- model and monitor hybrid systems composed of heterogeneous components. This need includes parallelism representation for multi-component systems;
- represent uncertainty on the system, be it of modeling or because of problems on observations (noise or communication problems);
- monitor the health of the system and follow its degradation process.

This paper is organized as follows. Section 2 proposes a new definition for hybrid systems. A survey of the existing formalisms in Section 3 will exhibit that they do not comply with these needs. We thus propose the Heterogeneous Petri Nets (HtPN) based on the work of [3] in Section 4. This new HtPN formalism based on Petri Nets is able to fulfill our needs for simulating and health monitoring of hybrid systems under uncertainty. We implemented a software to simulate systems modeled with HtPN. This implementation was applied on a production system from Motorola already defined in the literature [4] and on a photovoltaic panel system we designed. This work is presented in Section 5.

2 Hybrid Systems

2.1 Definitions

Although the notion of Hybrid Systems (HS) exists in the literature, their formal definition is often too restrictive. The authors in [5] consider hybrid systems where discrete models and continuous models have to communicate during simulation, but hybrid behaviors are never considered. Other works simply consider hybrid systems as the integration of diverse specific components in various domains such as the electrical, mechanical and optical fields [6; 7]. This section aims at proposing a definition of Hybrid Systems (HS) focused on the dynamics of the systems. To better understand this definition, the definitions of Discrete Event Systems (DES) and Continuous Systems (CS) must be recalled.

A Discrete Event System [8] is a system which will only manage discrete data: the state space is a discrete set and the state transition mechanism is event-driven. The state evolution of a DES depends entirely on the occurrence of asynchronous discrete events over time. Some events are observable, whereas some are unobservable (like some spontaneous fault events for diagnosis purpose for example). If continuous data are encountered by a DES, they will be abstracted to generate discrete events.

A Continuous System [8] is a system with continuous time dynamics. The evolution of such a system can be described by a dynamic equation C of the form:

$$C = \left\{ \begin{array}{rcl} x_{k+1} & = & \mathbf{f}(x_k, u_k) + \mathbf{v}(x_k, u_k) \\ y_k & = & \mathbf{h}(x_k, u_k) + \mathbf{w}(x_k, u_k) \end{array} \right. \quad (1)$$

where $x_k \in \mathbb{R}^{n_x}$ is the continuous state vector of n state variables at time k, $u_k \in \mathbb{R}^{n_u}$ is the vector of n_u continuous input variables at time k, $\mathbf{f}$ is the noise-free continuous evolution function, $\mathbf{v}$ is a noise function, $y_k \in \mathbb{R}^{n_y}$ is the vector of n_y continuous output variables at time k, $\mathbf{h}$ is the noise-free output function and $\mathbf{w}$ is the noise function associated with observation.

In [9], a hybrid system is defined as a system that will encounter both discrete and continuous data at any time. This definition may be too restrictive to correspond to heterogeneous multi-component systems. Then, we propose a new definition for HS.

Definition 1 (Hybrid System)**.** *A Hybrid System can be divided into different sub-systems that communicate together. These sub-systems can be either purely discrete, purely continuous, or hybrid merging discrete and continuous. From a data point of view, some parts of an HS are affected solely by discrete data, some only by continuous data and some others by both continuous and discrete data.*

In the context of health monitoring, the aging of the system plays an important role on the evolution of its health state. We therefore define aging Hybrid Systems (aHS):

Definition 2 (aging Hybrid System)**.** *An aHS is a HS which includes the aging of the system as a continuous time function. This aging process is usually represented through degradation dynamics.*

2.2 Running Example of an aging Hybrid System

Our running example is an aHS that can be found in any control process involving a water tank. A water pump has two operating modes: either it is on and pumps water, or it is off. The pump can get stuck, and the system will enter a faulty state and shut down. When the pump is on, there are three ways for the system to stop: the user manually turns it off, the observed (i.e. measured) water level exceeds a given threshold (50 liters here) or a fault occurs on it. The only way to turn on the pump is for the user to start it by pressing the ON button. The system can enter the faulty state from both the on and off states. When in the faulty state, the pump is considered unavailable and cannot be started. When a repair action is made, the system returns into its off state.

In this example of aging hybrid system, one part of the model is hybrid (when the system is on, we consider both discrete and continuous data or observations) and another part is purely discrete (when the system is off, we only consider discrete data and observations). Both parts communicate with each other. An illustration of how this system works will be given with the new HtPN formalism we propose.

3 Related Work

This section aims at studying existing solutions and formalisms to deal with aging hybrid systems corresponding to our new definition. We will try to identify *a priori* solutions likely to satisfy the needs stated in the introduction: parallelism in various systems, representation of uncertainty both in the model and in the observations, representation and monitoring of system aging.

The theory of hybrid automata has been published in [9]. Each discrete state of the automaton represents a possible state of the system. It is associated with continuous dynamics defining the evolution of the continuous space. In this model, only one state can be active at a time. By definition, this is incompatible with the idea of parallelism. The transitions are defined by 5-tuples of the form $(q, Guard, \sigma, Jump, q')$ with q the state before the transition, q' the state after the transition, $Guard$ the condition to fulfill in order to fire the transition, σ the event received or emitted during the transition firing and $Jump$ the changes on the variables taking place during the firing. Concepts such as $Guard$ and $Jump$ are very interesting. However, even if hybrid automata composition is possible, they cannot share a common state. Therefore, it is impossible to represent uncertainty concerning observations or on the current system state.

Petri nets have the advantages to be very intuitive for modeling and designing systems and are recognized for their compactness and their relevance in decision-making and system monitoring. They are also used for proving some properties on systems. In hybrid Petri nets [4], there are two types of places: continuous and discrete places. Tokens in continuous places are real numbers, whereas tokens in discrete places are integer. Two types of transitions can be distinguished: continuous and discrete transitions. A crossing quantity is defined for continuous transitions and acts like a weight on the arcs. It is possible for transitions to have both types of places as inputs, but the discrete place must be an input and an output of the transition, and the weights of the ingoing and outgoing arcs must be the same. In case of a conflict between continuous and discrete transitions, the discrete transition has the priority. The idea of parallelism is applicable, as different tokens can evolve simultaneously in the model. However, continuous places are not associated with any dynamics. It is therefore impossible to associate to a state variable an evolutionary dynamic according to the state of the system, or to make a degradation variable evolve according to a level of stress associated with a state of the system.

In mixed Petri nets, proposed by [10], a place can be continuous and associated to dynamics. In this case, one, or few, differential equation is associated to the place. A place can also represent a discrete phenomenon. The continuous variables' evolution is modeled through two sets, giving the set of equations activated when a place is marked or when a marking is true. One contains the set of the equations activated when a place is marked, whilst the other contains a set of equations activated for a specific marking. The idea of $Jump$ and $Guard$ from hybrid automaton is also present. The idea of parallelism is applicable as long as the marked places do not change the value of the same variable through equations. The continuous variables are shared with the whole system and evolve following the active equations. This is still kind of restricting according to our needs, with the idea of parallelism and uncertainty, as this formalism does not allow two places in parallel to modify the same variables.

In previous works, we proposed a formalism named Hybrid Particle Petri Nets (HPPN) in [11; 12]. A diagnostic method was developed based on this formalism. It represents uncertainty both in the model and in the observations, and system aging can be represented and monitored. However, the HPPN formalism can be applied only on hybrid systems merging discrete and continuous dynamics at any time. We thus worked on the evolution of HPPN towards design and monitoring of any type of systems (discrete, continuous or hybrid systems).

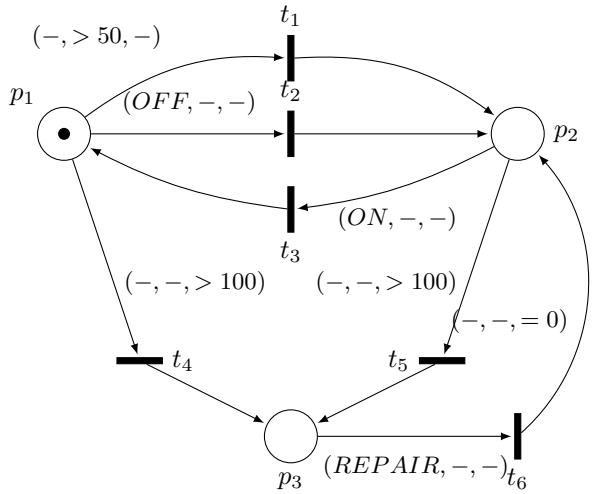
Even if each formalism can bring some interesting ideas for our approach, the definition of a new formalism dedi-

cated to aHS will thus meet a need.

4 HtPN Formalism

As we have seen with the related works, the existing formalisms do not fulfill our particular needs. Hence, we had to define a new formalism, extended from the classical Petri Nets. This section presents this new formalism, the Heterogenous Petri Nets (HtPN) and their semantics.

The hybrid system presented in Section 2.2 was modeled using HtPN and will be used as a running example to illustrate the different notions of the proposed formalism (see Figure 1).



(a)

places	continuous dynamics	degradation dynamics
p_1	$x_{k+1} = x_k + 1$	$\gamma_{k+1} = \gamma_k + 3$
p_2	-	$\gamma_{k+1} = \gamma_k + 1$
p_3	-	-

(b)

Figure 1: Example of an HtPN (a) and dynamics associated to its places (b)

4.1 General Presentation

A HtPN is formally defined as follows.

Definition 3 (HtPN). *A HtPN is a set of 5 elements:* $< P, T, Pre, Post, \mathbb{M}_0 >$ *gathering information to describe discrete, continuous and degradation dynamics through places and the relationships linking these places through transitions:*

- P *is the set of places;*
- T *is the set of transitions;*
- Pre *is the matrix of firing conditions of the system;*
- $Post$ *is the matrix of firing assignments of the system;*
- $\mathbb{M}_0$ *is the initial marking of the network.*

Places

In HtPN, a place is an object which may be associated with a continuous dynamic, and/or a dynamic of degradation. Discrete information is represented by the place itself.

Hence, to a place $p \in P$ can be associated a set of equations $C_p \in C$ modeling continuous dynamic of the system and the associated noise (uncertainties on the evolution and on the measurements) as well as a set of equations $D_p \in D$ modeling degradation dynamic of the system, where C and D respectively are the set of continuous dynamics of the system and the set of degradation dynamics of the system.

$$p = \{C_p, D_p\} \tag{2}$$

The set of equations C_p is defined as in Equation 1 where the functions $\mathbf{f}$, $\mathbf{v}$, $\mathbf{h}$ and $\mathbf{w}$ are dependent on the considered p place.

The set of equations D_p is defined as:

$$D_p = \Big\{ \gamma_{k+1} = d(\gamma_k, b_k, x_k, u_k) + z(\gamma_k, b_k, x_k, u_k) \tag{3}$$

where $\gamma_k \in \mathbb{R}^{n_D}$ is the degradation state vector with n_D degradation state variables. d is the noiseless hybrid degradation function. It depends on discrete events represented by b_k and on the continuous state vector x_k. z is the noise function of the degradation evolution. The functions d and z are dependent on the considered p place. In the example in Figure 1, there are 3 places: p_1, p_2 and p_3 and the associated continuous and degradation dynamics are explained in Figure 1 (b).

In the HtPN formalism, there is no need to associate each place with continuous and/or degradation dynamics. This is illustrated in Figure 1 (b) with the continuous dynamics of p_2 or both dynamics of p_3. By default, if no dynamics are specified, the place p is defined as:

$$p = \{-, -\} \tag{4}$$

Tokens

A place p contains $n_H(p)$ tokens ($n_H(p) \geq 0$). Each token h has three attributes: a discrete attribute, a continuous attribute and a degradation attribute. These three attributes are represented as a set $< \delta_k, \pi_k, \phi_k >$, with δ_k representing discrete information at time k, π_k representing continuous information at time k and ϕ_k representing degradation information at time k. These attributes evolve according to discrete events and dynamics associated to the place p the token belongs to.

The discrete information carried by a token h is called a configuration. The configuration of a token is the set of events that have occurred in the system up to time k. More formally, δ_k is the set b_k of events that occurred up to time k:

$$b_k = \{(v, \kappa) \mid \kappa \leq k\} \tag{5}$$

where (v, κ) represents an event $v \in E$ that occurs at time κ.

The continuous information carried by a token is called the state of the token. The state π_k represents the values of the continuous state vector $x_k \in X$ of the system at time k. The state of a token h evolves according to the continuous dynamics C_p (see Equation **??**) of the place it belongs to. If no continuous dynamic is specified, the state of the token will not evolve and will therefore remain constant.

The degradation information carried by a token is called the status of the token. The status ϕ_k represents the values of the degradation status vector $\gamma_k \in \Gamma$ at time k. The status of a token h evolves according to the degradation dynamics D_p (see Equation 3) of the place it belongs to. If no degradation dynamic is specified, the status of the token will not change and will therefore remain constant.

Definition 4 (Marking)**.** *The marking $\mathbb{M}_k$ of a HtPN at time k is the distribution of tokens in the different places of the network.*

Initial marking $\mathbb{M}_0$ represents the initial conditions of the system. Each token carries its initial configuration (the set of events that have occurred until time 0), its initial continuous state and its initial degradation status.

Arcs

Let denote $A \subset (P \times T \cup T \times P)$ the set of arcs.

Incoming arcs Pre is the matrix containing the firing conditions of arcs connecting the places to the transitions, of dimension $P \times T$. $Pre(t)$ is thus a vector containing the firing conditions of arcs connecting the input places to a given transition t. An arc connecting an input place p to a transition t is noted $a_{p,t}$. Based on the concept of $Guard$ of hybrid automata, the elements of $Pre(p,t)$ are:

$$Pre(p,t) = \begin{cases} ((\Omega^S_{p,t}, \Omega^N_{p,t}, \Omega^D_{p,t}); \rho_{p,t}) \text{ iff } \exists a_{p,t} \in A \\ \emptyset \text{ otherwise} \end{cases} \quad (6)$$

with:

- $(\Omega^S_{p,t}, \Omega^N_{p,t}, \Omega^D_{p,t})$ a triplet of conditions (respectively a symbolic, numerical and a degradation condition)
- $\rho_{p,t} \in \mathbb{N}^+$ a weight.

By default, this set is $Pre(p,t) = \{(\top, \top, \bot); 1\}$, which means that if nothing is specified, the symbolic and numerical conditions are set to TRUE (i.e. they are basically satisfied), the degradation condition to FALSE and the weight to 1. If an element is omitted in the definition of the arc, either the triplet of conditions or the weight, this element will take its default value.

The symbolic condition $\Omega^S_{p,t}$ is a condition related to the configuration of the tokens located in an input place p of a transition t. This condition can be set to TRUE ($\top$), FALSE ($\bot$) or tests a logical equation, i.e. the occurrence of one, or more, events in E. In this case, it takes the form $\Omega^S_{p,t}(\delta_k) = occ(v)$ (which is true if v has occurred).

The numerical condition $\Omega^N_{p,t}$ is a condition related to the state of the tokens in an input place p of the transition t. It can be set to TRUE ($\top$), FALSE ($\bot$) or represents a constraint on the continuous state vector. In this case, $\Omega^N_{pt}(\pi_k) = c(x_k)$ is a test on the continuous state vector x_k.

The degradation condition $\Omega^D_{p,t}$ is a condition related to the status of the tokens in an input place p of the transition t. It can be set to TRUE ($\top$), FALSE ($\bot$), or represents a constraint on the degradation status vector. In this case, $\Omega^D_{p,t}(\phi_k) = c(\gamma_k)$ is a test on the degradation state vector γ_k.

Examples can be seen in the running example (see Figure 1 (a)):

1. $Pre(p_1, t_1)$ contains a numerical condition to test if the continuous state vector is $x_k > 50$ and no symbolic or degradation conditions,
2. $Pre(p_1, t_2)$ contains a symbolic condition to check if the event OFF occurred and no numerical or degradation conditions,
3. $Pre(p_1, t_4)$ contains only a degradation condition to test if the degradation state vector is $\gamma_k > 100$.

Definition 5 (Accepted token)**.** *A token h is said to be accepted by an incoming arc if it satisfies:*

- *either the set of symbolic and numerical conditions of the arc,*
- *or the degradation condition of the arc.*

More formally, let $p \in P$ be a place such that $p \in P \wedge Pre(p,t) \neq 0$:

$$\forall h \in p, Accept(h, a_{p,t}) \equiv \\ \big(< \delta_k, \pi_k, \phi_k > \mid ((\Omega^S_{p,t}(\delta_k) = \top) \wedge (\Omega^N_{p,t}(\pi_k) = \top)) \vee \\ (\Omega^D_{p,t}(\phi_k) = \top) \quad (7)$$

We note $H_a(a_{p,t}, p)$ the set of tokens in the place p which are accepted by the arc $a_{p,t}$:

$$h \in H_a(a_{p,t}, p) \equiv (Accept(h, a_{p,t}) = \top) \quad (8)$$

The weight $\rho_{p,t}$ of an arc connecting a place p to a transition t represents the minimum number of accepted tokens required to validate the arc t.

Definition 6 (Validated arc)**.** *Let consider an arc $a_{p,t}$, $n_{Ha(p,t)}$ the number of tokens accepted by $a_{p,t}$ (i.e. the cardinal of $H_a(a_{p,t}, p)$) and $\rho_{p,t}$ the weight of the arc. The arc $a_{p,t}$ is said to be validated if $n_{Ha(p,t)} \geq \rho_{p,t}$.*

Outgoing arcs $Post$ is the matrix containing the firing assignments of arcs connecting the transitions to the places, of dimension $P \times T$. $Post(t)$ is thus a vector containing the firing assignments of arcs connecting the given t transition to the output places. An arc connecting a transition t to an output place p is noted $a_{t,p}$. Based on the concept of $Jump$ of hybrid automata, the elements of $Post(t,p)$ are:

$$Post(t,p) = \begin{cases} ((\Omega^S_{t,p}, \Omega^N_{t,p}, \Omega^D_{t,p}); \rho_{t,p}) \text{ iff } \exists a_{t,p} \in A \\ \emptyset \text{ otherwise} \end{cases} \quad (9)$$

with:

- $(\Omega^S_{t,p}, \Omega^N_{t,p}, \Omega^D_{t,p})$ a triplet of assignments (respectively a symbolic, numerical and a degradation assignment)
- $\rho_{t,p} \in \mathbb{N}^+$ a weight.

The symbol $-$ for an assignment means that no change is made to the concerned attribute. By default, $Post(t,p) = \{(-,-,-); 1\}$, which means that no assignment is specified. A weight equals to 1 means that only one token will be put in the output place of t.

The symbolic assignment $\Omega^S_{t,p}$ concerns the configurations of the tokens passing through the arc $a_{t,p}$. Let δ_k be the configuration of a token h passing through this arc at time k and wearing the value b_k:

- if $\Omega^S_{t,p} = v$, where $v \in E$, the event v is concatenated with the current configuration of the token passing through the arc: $b_{k+1} \longleftarrow b_k \cup (v, k+1)$,
- if $\Omega^S_{t,p} = b_{new}$, where b_{new} is a set of timed events, the configuration is completely reset and only contains b_{new}: $b_{k+1} \longleftarrow b_{new}$,
- else if $\Omega^S_{t,p} = - : b_{k+1} \longleftarrow b_k$.

The numerical assignment $\Omega^N_{t,p}$ concerns the state of the tokens passing through the arc $a_{t,p}$. Let π_k be the state of a token h crossing the arc at time k. Suppose that π_k carries x_k,

- if $\Omega^N_{t,p} = x_{new}$, where x_{new} represents a new numerical value for the token state π_k then: $x_{k+1} = x_{new}$,

- else if $\Omega^N_{t,p} = -$: $x_{k+1} = x_k$.

The numerical assignment $\Omega^N_{t,p}$ provides the initial condition for the state of the token passing through the arc, then the set of equations $C_p \in C$ defined in Equation **??** determines the evolution of the state of the token in the output place p.

The degradation assignment $\Omega^D_{t,p}$ concerns the status of tokens passing through the arc $a_{t,p}$. Let ϕ_k be the status of a token h crossing the arc at time k and γ_k be the value of ϕ_k,

- if $\Omega^D_{t,p} = \gamma_{new}$, where γ_{new} is a numerical value: $\gamma_{k+1} = \gamma_{new}$,
- else if $\Omega^D_{t,p} = -$: $\gamma_{k+1} = \gamma_k$.

The degradation assignment $\Omega^D_{t,p}$ provides the initial condition for the status of the token passing through the arc, then the set of equations D_p defined in Equation 3 determines the evolution of the status of the token in the output place p.

An example of a status assignment can be seen in Figure 1 (a): $Post(t_6, p_2)$ sets the status of the token to 0.

The weight $\rho_{t,p}$ defines the number of tokens to be put in the output place of the arc $a_{t,p}$, i.e. whether tokens will be duplicated or destroyed, and, if so, in what quantity. A weight $\rho_{t,p}$ less than the number of tokens used to fire the transition will cause the deletion of some of those tokens. A weight greater than the number of tokens used to fire the transition will result in the duplication of some tokens. This deletion or duplication will be performed randomly.

4.2 Firing rules

Enabled Transition

A transition $t \in T$ is said to be enabled at time k if all incoming arcs of t are validated (see Definition 6):

$$enabled(t) \equiv \left(\forall p \text{ s.t } a_{p,t},\ n_{Ha(p,t)} \geq \rho_{p,t}\right) \tag{10}$$

where $n_{Ha(p,t)}$ is the number of tokens accepted by the arc $a_{p,t}$, and $\rho_{p,t}$ is the weight in $Pre(p,t)$.

Set of fired tokens

Let $n_{Ha(p,t)}$ be the number of accepted tokens in the place p by the arc $a_{p,t}$. When $n_{Ha(p,t)} > \rho_{p,t}$, a choice function ${}^\bullet\zeta$ has to be defined to select $\rho_{p,t}$ tokens to be fired among the tokens $H_a(a_{p,t}, p)$:

$${}^\bullet\zeta : \mathbb{N}^+ \times H_a \rightarrow H_a.$$

Let p such that $p \subset P \wedge Pre(p,t) \neq 0$, the set of selected tokens in the place p among accepted tokens is noted $\Psi(p,t)$ and is formally defined as follows:

$$\Psi(p,t) = \begin{cases} {}^\bullet\zeta(\rho_{p,t}, H_a(a_{p,t}, p)) & \text{if } n_{Ha(p,t)} > \rho_{p,t} \\ H_a(a_{p,t}, p) & \text{otherwise.} \end{cases} \tag{11}$$

From $\Psi(p,t)$ can be defined $\Psi(\bullet t)$ which is the set of tokens fired by the transition t.

Let $p_1, p_2, ..., p_i$ be the set of input places of transition t:

$$\Psi(\bullet t) = \Psi(p_1, t) \cup \Psi(p_2, t) \cup ... \cup \Psi(p_i, t) \tag{12}$$

The choice function $\zeta^\bullet$ has also to be defined to select which tokens will be kept, duplicated or deleted among the set of fired tokens $\Psi(\bullet t)$:

$$\zeta^\bullet : \mathbb{N}^+ \times \Psi(\bullet t) \rightarrow \Psi(\bullet t).$$

Transition firing

During a transition firing, the tokens fired by the transition t are moved into the output places of t. The attributes of fired tokens are either kept or updated. As specified in Section 4.1, this update, as well as the possible deletion or duplication of tokens, are defined by the set of assignments $Post(t,p)$ carried by the outgoing arc. As a reminder, if $Post(t,p)$ carries no information, the attributes of the tokens are kept and no duplication will take place. However, deletion of tokens may occur if the number of tokens fired is greater than the weight $\rho_{t,p}$.

The firing of a transition t at time k is formally defined as follows: $\forall p \in P \wedge Pre(p,t) \neq 0$ and $\forall p' \in P \wedge Post(t,p') \neq 0$,

$$\begin{array}{rcl} M_{k+1}(p) & = & M_k(p) - \rho_{p,t} \\ M_{k+1}(p') & = & M_k(p') + \rho_{t,p'} \end{array} \tag{13}$$

where $\rho_{p,t}$ is the weight carried by the arc connecting the place p to the transition t, $\rho_{t,p'}$ is the weight carried by the arc connecting t to the place p', and $M_k(p)$ is the number of tokens in the place p at time k. $\mathbb{M}_k(p)$ represents all the tokens in the place p at time k:

$$\begin{array}{rcl} \mathbb{M}_{k+1}(p) & = & \mathbb{M}_k(p) \setminus \Psi(p,t) \\ \mathbb{M}_{k+1}(p') & = & \mathbb{M}_k(p') \cup \zeta^\bullet(\rho_{t,p'}, \Psi(\bullet t)) \end{array} \tag{14}$$

The HtPN formalism has been formally defined and can be used to model hybrid systems. In the next section, the HtPN formalism is used to model and simulate the behavior of a production system from Motorola and a photovoltaic panel system we designed.

5 Application

A software to simulate systems modeled with HtPN was implemented in python under the alias HeMU. Two systems were implemented as examples, a production system from the literature and a system comprised of two photovoltaic panels. The software is available on gitlab[1], as well as more detailed results for the applications.

5.1 Modeling a hybrid control system

System description

To exemplify the formalism and show that it can represent any type of existing Petri Nets, an example of a Hybrid Petri Nets taken from [4] was represented and simulated. To reduce the computation time, the numbers in the original example were divided by 10. The system with original parameter values is available on gitlab. The chosen system represents a production system from Motorola. It can take care of two types of pieces coming in batch, which are called L-type and R-type. When a L-type batch arrives, it is immediately transformed into 3000 pieces, which will be continuously taken care of, and put in an upstream buffer. When 50 pieces are in this buffer, the processing of the L-type pieces will begin after waiting 30 time units (which corresponds to the set-up of the system for an L-type batch). Once all pieces of the batch are taken care of, the system goes back into an idle state and is available to process another batch. For an R-type batch, the process is almost the same. The main difference is that the beginning of the process is delayed by 100 time units, and that the delay time before processing is set to be 36 time units.

[1] https://gitlab.laas.fr/hymu/hemu

The system is neither purely discrete, as the time is represented by continuous dynamics, nor purely continuous, as the system cannot be solely represented by continuous dynamics and is in need of discrete events: it is a hybrid system. As this model was initially developed for system control, the monitoring aspect with degradation dynamics is not considered and is not represented in this model.

Modeling with HtPN

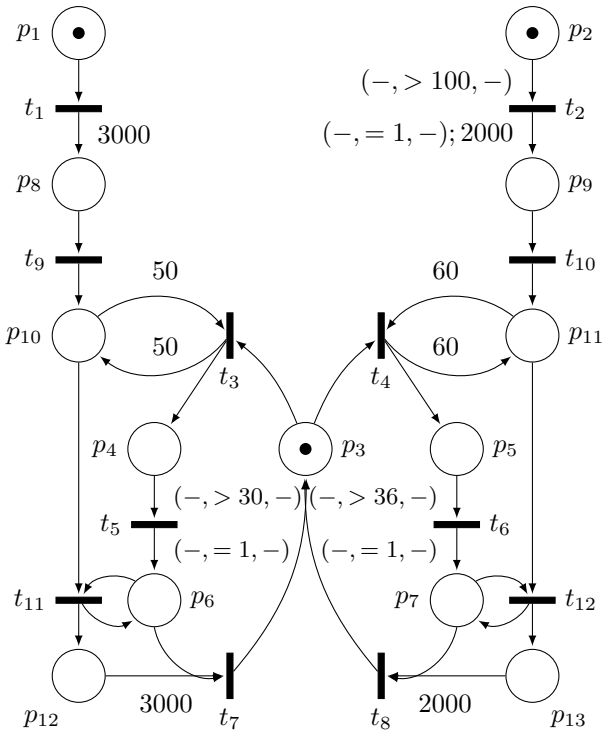


Figure 2: HtPN modeling the behavior of a production system

The system was modeled with HtPN (Figure 2). The left part of the figure represents the L-type pieces, while the right part of the figure represents the R-type pieces. The set P of places is composed of 13 places: $P = \{p_1, \ldots, p_{13}\}$. The model is hybrid, as it is composed of places without any continuous dynamics (purely discrete) communicating with places having continuous dynamics (p_2, p_4 and p_5).

The initial marking is $\mathbb{M}_0 = \{p_1, p_2, p_3\}$. At $\mathbb{M}_0$, the tokens have an empty configuration (which will remain empty as the set of events E is empty), a state $\pi = [1]$ which represent the time elapsed in each place: x_k^h represents the time that the token h has passed in the place p at time k. As there are not any degradation dynamics (as the system is being controlled and its health state is not being monitored), the status ϕ of the token is set to 0.

The set of continuous dynamics C is composed of continuous dynamics incrementing the tokens state by 1 in the places p_2, p_4 and p_5, which are the places concerned by the elapsed time requirement:

$$C_{p_i} = \{x_{k+1} = x_k + 1, \quad i = \{2, 4, 5\} \qquad (15)$$

The set T of transitions is composed of 12 transitions: $T = \{t_1, \ldots, t_{12}\}$.

A numerical condition $\Omega_{p,t}^N$ was used to represent the elapsed time since the arrival of the token in the place requirement. It is for example the case for conditions $Pre(p_2, t_2)$, $Pre(p_4, t_5)$, $Pre(p_5, t_6)$ which will be detailed later in this section.

To simulate the numbers of pieces, the weights on the arcs were set to the needed values. For example, ρ_{t_1,p_8} was set at 3000, to simulate the fact that the batch is transformed into 3000 pieces.

The matrices Pre and $Post$ are given on gitlab. Some are exemplified hereafter:

- $Pre(p_2, t_2) = \{(-, \pi_k > 100, -); 1\}$ to represent the 100 time units waiting in p_2,
- $Pre(p_{10}, t_3) = \{(-, -, -); 50\}$ to represent the fact that 50 pieces have to be in p_{10} before the system begins processing the L-type pieces,
- $Post(t_1, p_8) = \{(-, -, -); 3000\}$ to represent the transformation of an L-type batch into 3000 pieces,
- $Post(t_6, p_7) = \{(-, \pi = 1, -); 1\}$ to reset the state of the token.

The duration of the simulation is 6000 time units, as it is sufficient for the system to process both R-type and L-type batch, and return to its idle state.

When the system is idle, a token is in the place p_3. A token in p_1 represents the fact that a L-type batch is available and waiting to be transformed into pieces. This batch is turned immediately into 3000 pieces, which will be put into the place p_8. This action is represented through the firing of the transition t_1. Transition t_9 occurs and represents the placement in the upstream buffer represented by p_{10}. When the number of pieces in p_{10} is 50, transition t_3 is fired leading to the initialization of the system for the L-type pieces, represented by the place p_4. Then, t_5 will be fired 30 time units later, which corresponds to the duration of the initialization for a L-type batch. A token in p_6 shows that the system's initialization for the L-type pieces is over and that the system is ready to deal with the L-type pieces. Transition t_{11} is then enabled, meaning that the L-type pieces will be processed. Once all 3000 pieces have been processed, t_7 is fired and the systems goes back into an idle state.
For the R-type batch, the process is quite similar putting aside the delay of a 100 time units before the beginning of the process.

Simulation results

A simulation software based on HtPN has been developed in python. The results of the HtPN model simulation for the production system can be found as a video[2].
After the software is launched, the marked places are p_1, p_2 and p_3 which represent that a L-type and a R-type batches are waiting to be processed and that the system is available. Then, the processing begins as it is divided into 3000 pieces, placed in p_8. These pieces are transferred, one by one, to p_{10}. Then, when 50 pieces are in p_{10}, T_3 is fired, leading to a token in p_4 (at 0min04secs on the video). After 30 time units in p_4, T_5 is fired, and a token is placed in p_6, which will lead to the repeated firing of T_{11}. The L-type batch pieces are then being processed. When the 100^{th} time unit has passed, T_2 is fired (at 0min06secs on the video). 2000 pieces from the R-type batch are placed into p_9 and transferred to p_{11} one at a time. The R-type pieces will stop here as the system is not available. At 2min15sec, all the R-type pieces are waiting in p_{11} for the system's availability. It

[2] https://www.youtube.com/watch?v=RZ8yhZum_Pw&feature=youtu.be

still has 900 L-type pieces to process before. At 3min12sec, all the L-type pieces have been manufactured. T_7 is then fired, and the system is available. T_4 is fired immediately and a token is placed into p_5. The system waits 36 time units and T_6 is fired, leading to the processing of the R-type pieces represented by the firing of T_{12}. After all the pieces are processed, T_8 is fired. The system is then back into its idle state and the simulation stops, as 6000 time units have passed.

5.2 Modeling an aHS for health monitoring

System description

A real hybrid system composed of two photovoltaic (PV) panels connected to a lithium-ion battery has been designed. The two PV panels are illuminated by an artificial sun. They are theoretically identical and therefore have the same characteristic parameters. The current and the voltage provided by one of the cells depend on the sun lighting that it captures. In order to charge the battery faster and with a better accuracy, the panels are built in a parallel association. The Lithium-ion battery is connected to the shield so that it can be charged by the panels and to two light bulbs and one motor in order to discharge it or simulate losses. For each panel, data collected from sensors by an Arduino Leonardo card are: the voltage, the provided current and power, the temperature and the illuminance captured by the panel.

The objective is to model this real PV panel system with the HtPN formalism in order to perform health monitoring by specifying a degradation function of panels.

HtPN structure modeling

The first step is to design the system model using a HtPN structure, i.e. $< P, T, Pre, Post >$, by identifying health modes (as defined in [11]) for each panel. Five health modes can be determined for each panel i: two nominal modes, one in which the panel is on (noted $Nom1_i$) and the other in which it is off (noted $Nom2_i$), two degraded modes, in which the panel loses its efficiency, one when the panel is on ($Deg1_i$) and one when it is off ($Deg2_i$), and a failure mode ($Fail_i$) in which the panel must be replaced or repaired.

The resulting HtPN model of the two-panel system, illustrated in Figure 3, contains places corresponding to combinations of these modes and can be found on gitlab. Figure 3 shows the multimode of only one panel, with its two nominal modes, Nom_1 and Nom_2, its two degraded modes, Deg_1 and Deg_2 and its failure mode, $Fail$.

Four observable discrete events $E_o = \{ON1, OFF1, ON2, OFF2\}$ have been identified representing switch on or off a panel. The occurrence of these discrete events is tested in the symbolic conditions Ω^S associated with incoming arcs in the HtPN. A panel can be switched off during a nominal mode, but it can also be switched off during a degraded mode D_i. Some fatal faults, like short circuit or open circuit, are considered that force the panel to switch instantly to a failure mode. These fatal faults are represented by discrete events that are not observable in E_{uo}. These events are included in symbolic conditions Ω^S associated to incoming arcs of the HtPN model.

Continuous and degradation dynamics

Continuous dynamic are then defined in the HtPN model in order to make modes and transitions relevant. Some hypothesis were made about the system physical behavior.

- The received illuminance, that is an input of the system, follows a positive sine during the day with a maximum illuminance located at noon.

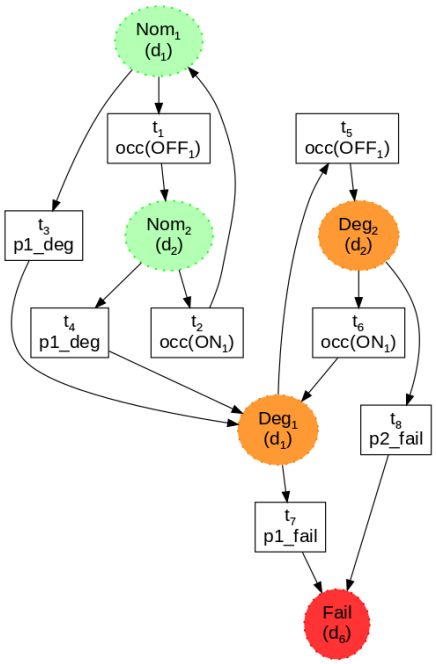


Figure 3: HtPN structure for one panel

- The observable current is computed using a theoretical current, depending on illuminance and degradation.
- According to measures realized on the system, currents are supposed affine depending on illuminance ($I = a.E + b$ with $a = 1.9e-2$ and $b = 7.3$).

Continuous dynamics C_p are the same for all places p of the HtPN and is described by the following equations:

$$\begin{cases} E^i_{th}(k+1) = E_{max}.|sin(\pi \frac{k}{T_{days}})| \\ E^i_{sensor}(k+1) = E^i_{th}(k) + noise(k) \\ I^i_{th}(k+1) = (a.E^i_{th}(k) + b).(1 - \frac{Degradation_i(k)}{I^{max}_{th}}) \\ I^i_{sensor}(k+1) = a.E^i_{sensor}(k) + b \end{cases} \tag{16}$$

where $E_{max} = 120000$ lux is the typical illuminance obtained at the zenith, $T_{days} = 86400$ s is the duration in seconds of a day, and I^{max}_{th} is the theoretical current obtained when the illuminance is maximum. A white noise of 1% on illuminance measured by the luxmeter is considered. These equations make possible to estimate respectively at time $k+1$ the theoretical illuminance, the illuminance recorded by the sensor, the theoretical current that the panels should deliver and finally, the measured current. $Degradation_i$ is a degradation panel indicator that corresponds to an observable current loss that is used in the continuous equations to represent an efficiency loss. This degradation indicator is determined from a degradation function D_p which is identical for all places of the HtPN.

The panel degradation D_p is considered continuous over time and follows a Gaussian distribution which models phenomenon like corrosion, glass break, discoloration, delamination and fissures [13]. This approach makes possible to be quite faithful to reality by inserting a part of randomness in the degradation undergone by the panel. A Gaussian centered on zero is considered to represent the probability of a fault occurrence for the panel. The more time passes, the more we widen the standard deviation of the curve to allow

large degradation to happen because the panel is older and therefore more fragile. The more time increases, the more the curve flattens and the more the probability that the panel fault occurrence increases. The evolution of this degradation will trigger the system evolution from nominal modes to degraded modes or from degraded modes to failure modes according to the following sets of Guard and Jump whose thresholds are defined in [13]:

- $Pre(p_i, t_j) = \{(-, -, 0.2 < \frac{Degradation_i}{I_{th}^{max}}); 1\}$ from p_i where panel i is nominal
- $Post(t_j, p_g) = \{(-, -, -); 1\}$ to p_g where panel 'i' is degraded
- $Pre(p_i, t_j) = \{(-, -, 0.95 < \frac{Degradation_i}{I_{th}^{max}}); 1\}$ from p_i where panel i is degraded
- $Post(t_j, p_g) = \{(-, -, -); 1\}$ to p_g where panel i is failed

The notation $\frac{Degradation_i}{I_{th}^{max}}$ represents the panel loss of efficiency between the best production and what it really produces. With a loss of 95% efficiency, the panel is considered failed. The value of this degradation indicator is therefore between 0 and 1.

5.3 Simulation of the system

For the simulation, the panels are switched on and off periodically every day. The panels are in the "OFF" nominal mode between 10pm and 5am each night.

Figures 4 and 5 illustrate the observed currents and degradation of panels $P1$ and $P2$ that change between these two ON/OFF modes. A fatal fault occurs at t=43200s on the panel $P1$, which is represented by the blue line. The panel $P1$ then enters a failure mode and its real current decrease instantly to zero. When a panel is failed, degradation increase is stopped. The second panel $P2$, represented by the orange curve, continues to operate and its degradation increases, resulting in a loss of current. When its degradation reaches the threshold of 0.95 this panel also enters a failure mode.

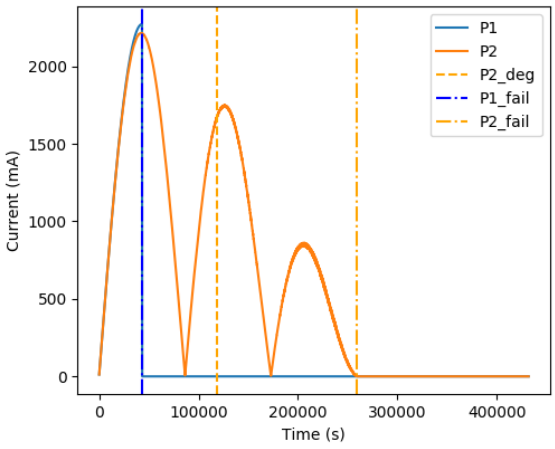


Figure 4: Current measured by sensors for the 2 panel system

Simulation shows that the proposed HtPN model represents the continuous behavior and degradation of a PV system that depends on different operating conditions. This

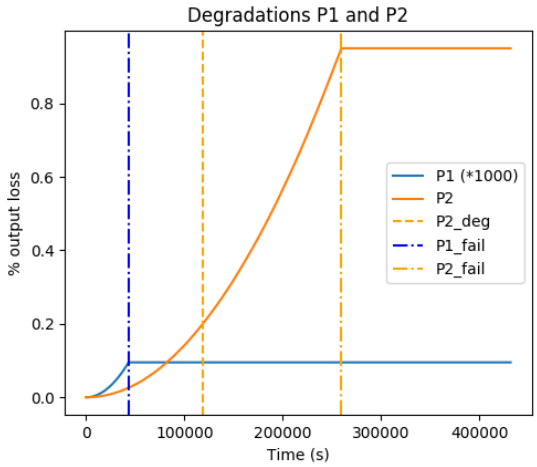


Figure 5: Degradation of the 2 panel system

HtPN model captures all necessary information to implement diagnostic and prognostic functions for system health monitoring.

6 Conclusions and future work

A new definition for a hybrid system has been provided in this paper. For health monitoring purposes, this definition has been extended to aging Hybrid Systems (aHS) in order to take into account the degradation dynamics of the system.

A new formalism based on Petri Nets has been introduced and specified, the Heterogeneous Petri Nets (HtPN). This formalism has been developed to represent everything the usual Petri Net can do and more. It can represent the behavior of a complex hybrid system to simulate control systems for example or monitor the system health modes as well. This representation allows to take into account different types of uncertainty about modeling and observations.

A software implementation has been realized to simulate models of such aging hybrid systems in the proposed formalism and applications to a production system from Motorola and to a photovoltaic panel system have been proposed.

Future work will focus on the development of the health monitoring function of such complex hybrid systems under uncertainty. This function will integrate diagnostic and prognostic capabilities to estimate the current health state of the system and to predict its remaining useful life.

Acknowledgments

We would like to thank J-B. Vidaud, A. Aumaire, G. L'Hostis, H. Laabidi and L. Willem, students in the 2nd year of the Master's program at ENSEEIHT in Toulouse, France, for their work on the panel test bench.

References

[1] J. L Peterson. Petri nets. *ACM Computing Surveys (CSUR)*, 9(3):223–252, 1977.

[2] A. Napoleone, M. Macchi, and A. Pozzetti. A review on the characteristics of cyber-physical systems for the future smart factories. *Journal of Manufacturing Systems*, 54:305 – 335, 2020.

[3] Q. Gaudel, E. Chanthery, and P. Ribot. Health Monitoring of Hybrid Systems Using Hybrid Particle Petri

Nets. In *Annual Conference of the Prognostics and Health Management Society 2014*, page 51, September 2014.

[4] H. Alla and R. David. Continuous and hybrid petri nets. *Journal of Circuits, Systems, and Computers*, 8(01):159–188, 1998.

[5] F. Bouchhima, G. Nicolescu, E. M. Aboulhamid, and M. Abid. Generic discrete–continuous simulation model for accurate validation in heterogeneous systems design. *Microelectronics journal*, 38(6-7):805–815, 2007.

[6] M. B. Ayed, F. Bouchhima, and M. Abid. Codis+: Co-simulation environment for heterogeneous systems. *Journal of Control Engineering and Applied Informatics*, 20(1):98–107, 2018.

[7] P. Ribot and E. Bensana. A generic adaptive prognostic function for heterogeneous multi-component systems: application to helicopters. In *European Safety & Reliability Conference, Troyes, France*, 2011.

[8] C. G. Cassandras and S. Lafortune. *Introduction to discrete event systems*. Springer Science & Business Media, 2009.

[9] T. A. Henzinger. The theory of hybrid automata. In *Verification of digital and hybrid systems*, pages 265–292. Springer, 2000.

[10] C Valentin-Roubinet. Hybrid systems modelling: mixed petri nets. In *IEEE Conference CSCC*, volume 99, 1999.

[11] Q. Gaudel, E. Chanthery, P. Ribot, and M. J. Daigle. Diagnosis of hybrid systems using Hybrid Particle Petri nets: theory and application on a planetary rover. In Moamar Sayed-Mouchaweh, editor, *Fault Diagnosis of Hybrid Dynamic and Complex Systems*, pages 209–241. Springer Verlag, 2018.

[12] Q. Gaudel, E. Chanthery, and P. Ribot. Hybrid Particle Petri Nets for Systems Health Monitoring under Uncertainty. *International Journal of Prognostics and Health Management*, 6, June 2015.

[13] M. Vázquez and I. Rey-Stolle. Photovoltaic module reliability model based on field degradation studies. *Progress in photovoltaics: Research and Applications*, 16(5):419–433, 2008.