Transforming Time Petri Nets into Heterogeneous Petri Nets for Hybrid System Monitoring

Léonie Hatte, Pauline Ribot, Elodie Chanthery

Transforming Time Petri Nets into Heterogeneous Petri Nets for Hybrid System Monitoring

Léonie Hatte * **Pauline Ribot** * **Elodie Chanthery** *

* *LAAS-CNRS, Université de Toulouse, INSA, UPS, Toulouse, France.*
(e-mail: firstname.name@laas.fr).

Abstract: Increasing industrial system complexity challenges maintenance and health monitoring, that involves both diagnosis and prognosis tasks. While formalisms like Time Petri Nets (TPN) simplify the explicit expression of temporal uncertainties, Heterogeneous Petri Nets (HtPN) offer a formalism for hybrid system health monitoring that lacks explicit time representation. This article demonstrates HtPN's ability to explicitly represent temporal information. Key contributions include a lossless transformation from a subclass of the TPN to a subclass of the HtPN and the provision of a Python code for conversion. This transformation enables simulation, diagnosis, and prognosis with explicit time representation.

Keywords: Time Petri Nets, Diagnosis, Prognosis, Hybrid Systems, Health monitoring

1. INTRODUCTION

Industrial systems tend to become increasingly complex. This also means that their maintenance is becoming increasingly complex, which is a major challenge today. System maintenance includes monitoring the state of health of a system, which mainly involves diagnosis and prognosis. The diagnosis is used to estimate the current health state of the system, whereas prognosis computes its future evolution up to failure, in order to predict the system's Remaining Useful Life (RUL). To carry out these monitoring tasks, model-based approaches are based on modeling the system as faithfully as possible.

In order to satisfy the needs, the number of existing formalisms has increased, especially with the introduction of hybrid systems. Hybrid systems (Henzinger (1996)) include a discrete-event evolution and continuous dynamics. Some formalisms allow for the explicit inclusion of time constraints in models, making it easier to understand the temporal aspects of diagnosis and especially prognosis. For that purpose, Time Petri Nets (TPN) (Wang and Wang (1998)) express explicit temporal uncertainties, but do not represent continuous nor hybrid dynamics. On the other side, Heterogeneous Petri Nets (HtPN) (Vignolles et al. (2022)) are a valuable formalism for modeling hybrid systems. They allow representing all the necessary information for diagnosis and prognosis, as well as taking into account the uncertainties within complex systems. As defined, the formalism does not make the time variable or time constraints explicit, as TPN can.

There exists three types of TPN depending on which element of the net the temporal information is associated with (David and Alla (2010)): P-TPN that carry conditions on the places; T-TPN that carry conditions on the transitions; A-TPN that carry conditions on the arcs. The choice of a formalism depends on the application and the implementation targeted in the study. To the best of our knowledge, there exist tools that simulate and analyze the T-TPN behavior, which gives them computational advantages. Nevertheless, we claim that the explicitness of time in HtPN is possible, and we assert that they are more expressive than TPN, ultimately enabling the modeling of systems subject to temporal constraints and encompassing both continuous and discrete dynamics.
This article aims to demonstrate that HtPN can explicitly represent all temporal information inherent in TPN. This capability is crucial for effective health monitoring of systems. The health monitoring process is facilitated by HeMU, a software solution designed not only for simulating hybrid systems, but also for diagnosis and prognosis purposes.

The first contribution of this article is to demonstrate the feasibility of transforming a TPN into a HtPN, without losing any information about the expressiveness. The second contribution is the provision of a Python transformation code that directly converts a T-TPN into a HtPN. It is then possible to start from a T-TPN model, automatically convert it into a model that will be enriched with expert information, for example by defining continuous dynamics or degradation dynamics. At the end of the process, the enriched model can be simulated, diagnosed, and prognosed while preserving the explicit notion of time inherited from the T-TPN initial model.

This article is organized as follows. Section 2 delves into the formalism of HtPN, focusing on the particular class of time-HtPN (t-HtPN). Section 3 focuses on the transformation process, specifically from the A-TPN to the t-HtPN. Section 4 details the use of the Python transformation code that directly converts a T-TPN into a HtPN that can be used in the HeMU software. Finally, Section 5 concludes the paper.

2. BACKGROUND ON HETEROGENEOUS PETRI NETS

A Petri Net is a formalism used to model Discrete Event Systems (DES) whose definition given in Peterson (1977) is recalled hereafter.

Definition 1. (Petri Net). A Petri Net (PN) is a quintuplet $N =< P, T, Pre, Post, M_0 >$:

- P is the set of places;
- T is the set of transitions;
- $Pre : P \times T \rightarrow \mathbb{N}$ is the application of backward incidence. $Pre(p,t)$ is the weight of the arc going from place $p \in P$ to transition $t \in T$. If the weight equals 0, there is no arc between p and t;
- $Post : T \times P \rightarrow \mathbb{N}$ is the application of forward incidence. $Post(t,p)$ is the weight of the arc going from transition $t \in T$ to place $p \in P$. If the weight equals 0, there is no arc between t and p;
- $M_0 \in \mathbb{N}^{|P|}$ is the initial marking representing the number of tokens present in each place of the network.

2.1 HtPN

Heterogeneous Petri nets (HtPN) (Vignolles et al. (2022)) are an extension of Petri nets. They were developed in order to model, simulate and monitor hybrid systems. To represent such systems, HtPN make use of a continuous dynamic $\mathcal{C}_p$ and a degradation dynamic $\mathcal{D}_p$ that may be associated with each place $p \in P$. HtPN formalism uses tokens that are objects with three attributes: a configuration, a continuous state vector and a degradation vector, whose values condition the firing of a transition in the HtPN. This vision comes from colored Petri nets (Gehlot (2019), Liu et al. (2002)) and Petri nets with objects (Lakos (1995)).

The continuous information within the token is termed the token state. It evolves according to the continuous dynamics $\mathcal{C}_p$ associated with the places. Concurrently, the token status represents the degradation information within the token. It evolves according to the degradation dynamics $\mathcal{D}_p$ associated with the places. If no continuous (resp. degradation) dynamics are associated to a place, then the token state (resp. status) remains constant. Introducing the token status, representing the system's degradation in HtPN, is a notable advantage of this formalism. It facilitates the computation of RUL for prognosis and health management purposes. The formal definition of a HtPN is the following.

Definition 2. (HtPN). A HtPN is a tuple $< P, T, Pre, Post, M_0 >$ with:

- P and T defined as for a PN;
- Pre is a structure of size $|P| \times |T|$ that defines the firing conditions for an arc connecting a place p to a transition t : $Pre(p,t) = ((\Omega^S_{p,t}, \Omega^N_{p,t}, \Omega^D_{p,t}), \rho_{p,t})$.
- $Post$ is a structure of size $|P| \times |T|$ that defines the firing assignments for an arc connecting a transition t to a place p: $Post(t,p) = ((\mathbb{A}^S_{t,p}, \mathbb{A}^N_{t,p}, \mathbb{A}^D_{t,p}), \rho_{t,p})$.
- $M_0 \in \mathbb{N}^{|P|}$ is the initial marking.

$Pre(p,t)$ contains a condition triplet and a weight $\rho_{p,t} \in \mathbb{N}$. The condition triplet is composed of a symbolic (i.e. discrete) condition, a numerical (i.e. continuous) condition, and a degradation condition whose outputs are *true* or *false*. They depend on the attributes of the tokens. The symbolic condition $\Omega^S_{p,t}$ can be set to *true*, *false*, or it can test the occurrence of a discrete event. The numerical condition $\Omega^N_{p,t}$ can be set to *true*, *false*, or it can represent a constraint on the continuous state vector of the token to be fired (for example, testing the values of the state vector of the token to be fired). The degradation condition $\Omega^D_{p,t}$ can be set to *true*, *false*, or express a constraint on the token's status vector. By default, when not specified, the symbolic and numerical conditions are set to *true*, and the degradation condition is set to *false*. Similarly, if $\rho_{p,t} = 0$, it means that there is no arc connecting the place to the transition.

$Post(t,p)$ contains an assignment triplet and a weight $\rho_{t,p} \in \mathbb{N}$. The assignment triplet can modify the values of token attributes (configuration, state, status) that cross transitions. It is also composed of a symbolic ($\mathbb{A}^S_{t,p}$), numerical ($\mathbb{A}^N_{t,p}$) and degradation parts ($\mathbb{A}^D_{t,p}$), each of which sets the value of the tokens' attributes. By default, when assignments are not specified, the token values remain unchanged. Similarly, if $\rho_{t,p} = 0$, it means that there is no arc connecting the transition to the place.

The values of the tokens' attributes are tested and condition the firing of transitions. These test conditions are expressed by labels associated with the input arcs.

Definition 3. (Enabled Arc). An arc going from a place p to a transition t is enabled if:

- the number of tokens located in the place downstream of the transition is greater than the weight $\rho_{p,t}$ of the arc;
- the symbolic condition $\Omega^S_{p,t}$ and the numerical condition $\Omega^N_{p,t}$ are *true*, or the degradation condition $\Omega^D_{p,t}$ is *true*:

$$(\Omega^S_{p,t} \wedge \Omega^N_{p,t}) \vee \Omega^D_{p,t}$$

In other words, the arc is enabled if there are at least $\rho_{p,t}$ tokens whose configuration and state satisfy the symbolic and numerical conditions, or if the tokens' status satisfies the arc's degradation condition.

A transition is fireable if all its incoming arcs are enabled. HtPN always operate under strong semantics: the firing of transitions occur as soon as they are fireable.

Compared to classical PN, HtPN therefore extend the definitions of Pre, $Post$, tokens and add dynamics associated with places in P.

2.2 Time HtPN: t-HtPN

To make explicit the temporal information represented in HtPN and facilitate a comparison with the temporal information contained in TPN, we introduce a subclass of HtPN, named t-HtPN (explicit time-HtPN). In a t-HtPN, the numerical condition Ω^N depends only on time and is expressed in the form of a time interval, like those described in TPN.

The explicit notion of time in t-HtPN arises from a restriction on the continuous information carried by tokens. The token state is thus restricted into a single variable, denoted as $\theta \in \mathbb{R}^+$, referred to as the "clock".

Definition 4. (t-HtPN). A t-HtPN is a HtPN restricted as follows:

- P, T, Pre, $Post$, M_0 are defined as for a HtPN;
- the state vector of each token τ is reduced to its clock θ^τ;
- the continuous dynamics $\mathcal{C}_p$ associated with each place $p \in P$ enable the incrementation of the token clock values, such that $\frac{d\theta^\tau}{dt} = 1$ for each token of each place. The clock of a token therefore begins to increment when it is produced in p;
- no degradation dynamics $\mathcal{D}_p$ are defined for the system to lighten the study;
- no symbolic Ω^S or degradation Ω^D conditions are defined;
- the numerical conditions Ω^N are necessarily expressing conditions on time intervals. They are numerical conditions composed by logical conditions "and/or".

3. TRANSFORMATION FUNCTION BETWEEN A-TPN AND T-HTPN

This section delves into the transformation function bridging A-TPN (Arc-Time Petri Net) and t-HtPN (timed Heterogeneous Petri Net). First, we will show that A-TPN, a subset of TPN, incorporate temporal conditions on arcs, and their expressive hierarchy has been established. Subsequently, we introduce a transformation function designed to convert A-TPN into t-HtPN under strong semantics. A proof of equivalence between A-TPN and t-HtPN is thus provided. Finally, some illustrative examples of the transformation process are provided.

3.1 Background on TPN

A TPN with weak semantics means that a transition can remain fireable for an infinite time, meaning its firing is not obligatory as soon as it becomes fireable. Conversely, in a TPN with strong semantics, the firing of transitions are forced whenever they are fireable. Let X represents A, T or P, TPN under weak (resp. strong) semantics are denoted $\underline{X-TPN}$ (resp. $\overline{X-TPN}$).

We define the notation "$\mathcal{N}_1 \subset_\approx \mathcal{N}_2$" that means "the formalism $\mathcal{N}_2$ is strictly more expressive than the formalism $\mathcal{N}_1$", or in other words, "$\mathcal{N}_1$ is generalized by $\mathcal{N}_2$". The following theorems have been demonstrated in Boyer and Roux (2008).

Theorem 5. The strong semantics of all TPN types are strictly more expressive than their weak semantics:

$$\underline{X-TPN} \subset_\approx \overline{X-TPN} \tag{1}$$

This means that a Petri net with weak semantics can be transformed into a Petri net with strong semantics.

Theorem 6. A-TPN are strictly more expressive than T-TPN:

$$T-TPN \subset_\approx A-TPN \tag{2}$$

This means that a T-TPN can be transformed into an A-TPN.

Theorem 7. A-TPN are strictly more expressive than P-TPN:

$$P-TPN \subset_\approx A-TPN \tag{3}$$

This means that a P-TPN can be transformed into an A-TPN.

Thus, by transitivity, A-TPN in strong semantics are strictly more expressive than all other types of Time Petri Nets. This is why, without loss of generality, we will now only focus our study on the transformation of A-TPN into t-HtPN.

3.2 Arc-Time Petri Net

The formal definition of an A-TPN with enabling time intervals associated with network arcs is given below.

Definition 8. (Arc-Time Petri Net). An A-TPN is a couple $< N, I >$ where:

- $N =< P, T, Pre, Post, M_0 >$ is a PN (see Definition 1)
- $I : (P \times T) \cup (T \times P) \to \mathbb{Q}^+ \times (\mathbb{Q}^+ \cup \infty)$ is the time interval function associated with the existing input and output arcs of the system.

Enabling intervals of the arcs of Pre from place $p \in P$ to transition $t \in T$ are associated with time intervals $I(p,t) = [t^{p,t}_{min}, t^{p,t}_{max}]$, with $t^{p,t}_{min}$ the minimum dwell time of the token in p to enable t and $t^{p,t}_{max}$ the maximum dwell time of the token before stopping enabling t.

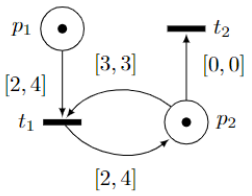
Assignment intervals of the arcs of the set $Post$ from transition $t \in T$ to place $p \in P$ are associated with time intervals $I(t,p) = [t^{t,p}_{min}, t^{t,p}_{max}]$, with $t^{t,p}_{min}$ and $t^{t,p}_{max}$ the minimum and maximum clock values the token can take when produced in p. When tokens are created in the place downstream of the arc, the assignment arc allows expressing uncertainty about the value of the clock associated with the produced tokens. Assignment arcs are not always defined in the literature. When the intervals are not specified on the outgoing arcs, this amounts to saying that the time interval for this assignment is $[0, 0]$, that is to say that the token clock is initialized at 0.

3.3 Transformation function

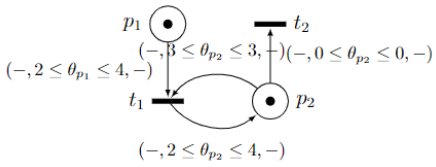
Intuitive model transformation Let us consider the A-TPN represented in Figure 1a where there are trivial time intervals such as $[3, 3]$, as well as non-trivial time intervals like $[2, 4]$ which express uncertainty regarding the token firing times and/or the assignment of token clock values at their production.

As mentioned earlier, HtPN always operate under strong semantics. By extension, t-HtPN also operate under strong semantics. The firing of a fireable transition is forced as soon as the minimum of the upper bounds of the time intervals associated with its incoming arcs is reached. The intuitive transformation of the presented A-TPN into a t-HtPN is given in Figure 1b.

The intuition is to keep the initial structure of the A-TPN. There is here no information about discrete events nor about degradation, so that symbolic conditions and degradation conditions are set to their default values (*true* for Ω^S and *false* for Ω^D). In t-HtPN, time can be expressed in the token state representing continuous information. Time intervals are expressed as time constraints in conditions



(a) A-TPN



(b) HtPN

Fig. 1. Transformation of an A-TPN into an HtPN

associated with incoming arcs in the t-HtPN. These time constraints allow comparing the clock value of the tokens to the values of the enabling intervals of the original A-TPN.

Formal definition of the transformation function The formal definition of the transformation function that maps both A-TPN and t-HtPN formalisms is provided in this subsection.

Let $\mathcal{N}_{A-TPN}$ be the A-TPN set and $\mathcal{N}_{t-HtPN}$ be the t-HtPN set.

Let f be the transformation function from an A-TPN G to a t-HtPN G':

$$f : \left| \begin{array}{c} \mathcal{N}_{A-TPN} \longrightarrow \mathcal{N}_{t-HtPN} \\ G \longmapsto G' \end{array} \right. \tag{4}$$

where the A-TPN G is of the form $< P, T, Pre, Post, I, M_0 >$ and the t-HtPN G' is of the form $< P', T', Pre', Post', M'_0 >$.

Step 1 The function f defines:

$$T' = T \ ; \ P' = P \ ; \ M'_0 = M_0. \tag{5}$$

It associates continuous dynamics C_p with each place $p' \in P'$, that control the evolution of the tokens' clock that are in each place $p' \in P'$.

Step 2 For each existing input arc, numerical conditions are defined by f as:

- $\forall\ I(p,t) = [a,b]$, where $a, b \in \mathbb{R}^+$, defined for G then:

$$Pre'(p', t') = ((-, \theta \in [a,b], -); \rho) \tag{6}$$

Step 3 For each existing output arc, assignments are defined by f as:

- $\forall\ I(t,p) = \emptyset$, then:

$$Post'(t', p') = ((-, \theta = 0, -); \rho) \tag{7}$$

- $\forall\ I(t,p) = [\alpha, \alpha]$ is a trivial time interval, with $\alpha \in \mathbb{R}^+$, then:

$$Post'(t', p') = ((-, \theta = \alpha, -); \rho) \tag{8}$$

- $\forall\ I(t,p) = [\alpha, \beta]$ is a non-trivial time interval indicating uncertainty about the clock value of the token produced in the place p, with

$$Post'(t', p') = ((-, \theta = [\alpha, \beta], -); \rho) \tag{9}$$

3.4 Equivalence proof

This section demonstrates the equivalence between the t-HtPN and A-TPN formalisms by examining three properties of the transformation function f. It's worth noting that if the f function is a bijection, then equivalence is established. We thus demonstrate that f is injective, then surjective, thereby concluding its bijectivity.

Function f is injective

Proof. The function f has a unique solution: every element in the image set of f has at most one inverse image under f. In other words, two distinct elements in its domain cannot have the same image by f. Indeed, when f is used to transform an A-TPN into a t-HtPN, the structure and marking of the final network are identical to the structure and marking of the initial network.

Let us consider two A-TPN, $N =< P, T, Pre, Post, I, M_0 >$ and $M =< P', T', Pre', Post', M'_0 >$, and their images $f(N)$ and $f(M)$. Their images are supposed to be identical, then we have:

$$f(N) = f(M) \iff \begin{cases} P & = P' \\ T & = T' \\ M_0 & = M'_0 \\ Pre & = Pre' \\ Post & = Post' \end{cases} \tag{10}$$

The networks M and N thus have the same structure and the same attributes on the arcs. Therefore, we have:

$$\forall N, M \in \mathcal{N}_{A-TPN}, f(N) = f(M) \iff N = M \tag{11}$$

□

Function f is surjective

Proof. Let M be a t-HtPN of the form $< P', T', Pre', Post', M'_0 >$, we prove that there exists at least one A-TPN N such that $f(N) = M$. Indeed, we can construct N such that $N =< P, T, Pre, Post, I, M_0 >$ with:

$$T = T' \ ; \ P = P' \ ; \ M_0 = M'_0. \tag{12}$$

For every input arc going from p' to t' in the t-HtPN, such that $Pre'(p', t') = (-, \theta \in [a,b], -)$, a time interval $I(p,t) = [a,b]$, and $Pre(p,t)$ are defined for the arc going from p to t.

For every output arc going from t' to p' in the t-HtPN, such that $Post'(t', p') = (-, \theta \in [\alpha, \beta], -)$, a time interval $I(t,p) = [\alpha, \beta]$, and $Post(t,p)$ are defined for the arc going from t to p.

N belongs to $\mathcal{N}_{\mathcal{A}-\mathcal{TPN}}$ the set of A-TPN, therefore f is surjective:

$$\forall M \in \mathcal{N}_{t-HtPN}, \exists N \in \mathcal{N}_{A-TPN} | f(N) = M \tag{13}$$

□

Function f is bijective

The transformation function f is bijective, as it is injective and surjective. Since f is bijective, there exists an application f^{-1} that allows transforming a t-HtPN into an A-TPN. Consequently, we can infer that there exists an equivalence relation between A-TPN and t-HtPN through

the function f. Figure 2 sums up the relation by f between the A-TPN and the t-HtPN, included in the HtPN domain.

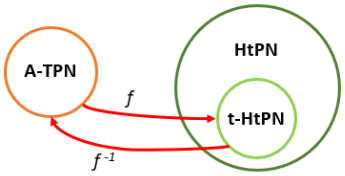


Fig. 2. Graphical representation of the sets

3.5 Illustrations

Some common temporal patterns of A-TPN are compared in this section with their equivalent in t-HtPN, obtained through the transformation function f.

Figure 3 illustrates a pattern with a simple condition associated with an incoming arc. Top left shows the A-TPN and right the corresponding t-HtPN obtained with the function f. The bottom of the figure shows the number of tokens in places p_1 and p'_1 respectively, according to the time. During the period between a and b, the number of tokens in p_1 and p'_1 is not accurately known, as the firing of the transition t_1 is uncertain. It is illustrated by the colored zone in the figure.

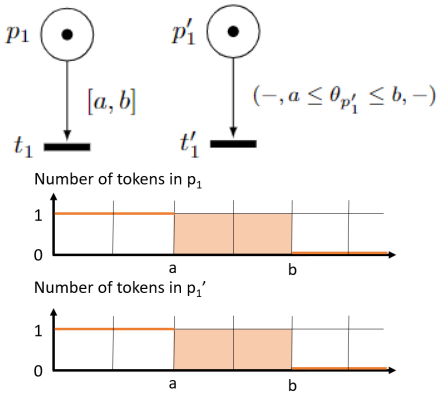


Fig. 3. Simple condition associated to an incoming arc

We can see that both patterns in Figure 3 exhibit the same behavior:

- The token in the A-TPN is used to fire the transition t_1 after the delay a.
- The token in the HtPN is used to fire the transition t'_1 after the delay a.
- If t_1 and t'_1 are fired at the same time $c \in [a, b]$, then the tokens in p_1 and p'_1 are respectively consumed by t_1 and t'_1.
- If the clocks of the tokens in p_1 and p'_1 reach the time b, then according to the strong semantics of A-TPN, t_1 is necessarily fired; and according to the semantics of HtPN, t'_1 is also necessarily fired.

Figure 4 illustrates a pattern with a simple assignation associated to an outgoing arc. Top left shows the A-TPN and right the corresponding t-HtPN obtained with the function f. The bottom of the figure shows the number of tokens in places p_1 and p'_1 respectively, according to the time. During the period between a and b, the number of tokens in p_1 and p'_1 is not accurately known, as the firing of the transition t_1 is uncertain. It is illustrated by the colored zone in the figure.

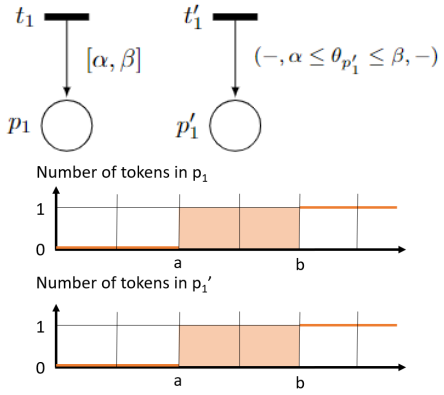


Fig. 4. Simple assignment associated to an outgoing arc

We can see that both patterns in Figure 4 exhibit the same behavior:

- Once t_1 is fired, a token is produced in place p_1.
- Once t'_1 is fired, a token is produced in place p'_1.
- The internal clocks of the tokens produced in p_1 and p'_1 are initialized to a value between a and b. This value is chosen by the system (for example in a manufacturing system, it can correspond to a processing time inherent to the system). So if the HtPN describes the same system as the A-TPN, there is no reason for the clock assignments to be different in the two formalisms.

Figure 5 illustrates a pattern with a basic loop. Top left shows the A-TPN and right the corresponding t-HtPN obtained with the function f. The bottom of the figure shows the number of tokens in places p_1 and p'_1 respectively, according to the time. During the period between a and b, the firing of the transition t_1 and t'_1 is uncertain, so the consumption and the production of tokens are uncertain.

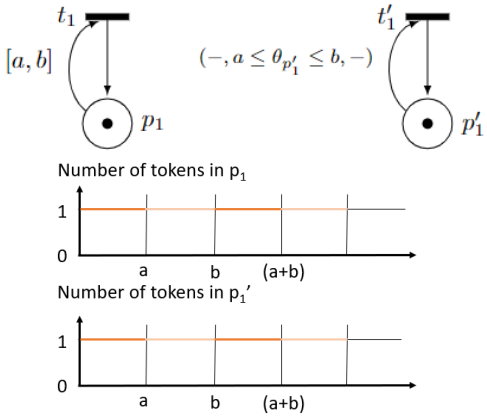


Fig. 5. Basic loop

We can see that both patterns in Figure 5 exhibit the same behavior:

- In the A-TPN, the transition t_1 can be fired only if the token in p_1 has a clock value between a and b.
- In the t-HtPN, the transition t'_1 can be fired only if the token in p'_1 has a clock value between a and b.
- If t_1 and t'_1 are fired simultaneously, then the tokens in p_1 and p'_1 are respectively consumed by t_1 and t'_1.

A transformation function f was proposed to convert an A-TPN into a t-HtPN for later system health monitoring. Indeed, HeMU software enables diagnosis and prognosis from a HtPN model representing hybrid systems and uncertainties. We have shown in this section that HtPN can capture all the temporal aspects expressed in Time Petri Nets.

4. TOOL PROTOTYPE

A Python-based software named HeMU (Vignolles et al. (2021), Chanthery et al. (2019)), which stands for Heterogeneous systems Monitoring under Uncertainty, was developed to perform the simulation, diagnosis, and prognosis of systems represented in the HtPN formalism. This software is available on Git [3] .

A transformation function f has been defined to obtain a t-HtPN from an A-TPN, considering that an A-TPN is more expressive than a T-TPN. Despite this, T-TPN are still the most widely used TPN, as there are a number of software packages available for analyzing their properties and simulating their evolution.

That is why we propose a translator from T-TPN to t-HtPN. Figure 6 illustrates the integration of the T-TPN/t-HtPN translator into the HeMU operating chain. At the beginning of the chain, the TINA software (Berthomieu and Vernadat (2006)) can be used to graphically editing T-TPN. The .ndr file provided by the TINA block can be used as input by the translator Python code (CONVERT block). The theoretical conversion would consist of two steps detailed under the block in the figure. The first step would transform a T-TPN into an A-TPN with the transformation function f_{BR} that is already addressed by Boyer and Roux (2008). The second step would use our f transformation function, discussed in section 3.3. The translator Python code automatically parses and converts the T-TPN in the .ndr file into a Python file representing the t-HtPN that can be given to the HeMU software tool block for system health monitoring. It is then possible to enrich the obtained model with expert information (ENRICH block)to define continuous dynamics or degradation dynamics in order to simulate, diagnose and prognose while keeping the explicit notion of time inherited from the T-TPN initial model.

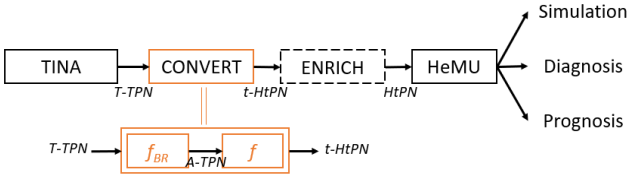


Fig. 6. Integration of the T-TPN/t-HtPN translator into the HeMU operating chain

[3] site: https ://gitlab.laas.fr/hymu/hemu.git

5. CONCLUSION

This article is part of the ambition to model complex systems, and more particularly hybrid systems, in order to carry out model-based diagnosis and prognosis.

It presents a transformation function between two subclasses of TPN and HtPN. The formal equivalence proof between A-TPN and t-HtPN is provided. A Python code is developed, enabling the conversion from a T-TPN model with explicit time constraints to an enriched t-HtPN model incorporating expert information on continuous dynamics and degradation, crucial for health monitoring diagnostic and prognostic functions. This t-HtPN model can be used in the HeMU software. The transformation explicitly interestingly preserves temporal constraints within the framework of HtPN.

Future work will focus on the transformation from T-TPN to A-TPN. This transformation was already discussed in Boyer and Roux (2008), but it could be interesting to implement it in our tool prototype, as T-TPN are the most used formalism.

ACKNOWLEDGEMENTS

We would like to thank Amaury Vignolles for his work on the HtPN formalism and the HeMu software.

REFERENCES

Berthomieu, B. and Vernadat, F. (2006). Time Petri nets analysis with TINA. In *QEST*, volume 6, 123–124.

Boyer, M. and Roux, O.H. (2008). On the compared expressiveness of arc, place and transition time Petri nets. *Fundamenta Informaticae*, 88(3), 225–249.

Chanthery, E., Ribot, P., and Vignolles, A. (2019). HyMU: a software for hybrid systems health monitoring under uncertainty. In *DX'19*.

David, R. and Alla, H. (2010). *Discrete, continuous, and hybrid Petri nets*, volume 1. Springer.

Gehlot, V. (2019). From Petri nets to colored Petri nets: A tutorial introduction to nets based formalism for modeling and simulation. In *2019 Winter Simulation Conf.*, 1519–1533. IEEE.

Henzinger, T.A. (1996). The theory of hybrid automata. In *Proceedings 11th Annual IEEE Symposium on Logic in Computer Science*, 278–292. IEEE.

Lakos, C. (1995). From coloured Petri nets to object Petri nets. In *Application and Theory of Petri Nets*, 278–297. Springer.

Liu, D., Wang, J., Chan, S.C., Sun, J., and Zhang, L. (2002). Modeling workflow processes with colored Petri nets. *computers in industry*, 49(3), 267–281.

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

Vignolles, A., Chanthery, E., and Ribot, P. (2021). A holistic advanced diagnosis approach for systems under uncertainty. In *DX'2021*.

Vignolles, A., Ribot, P., and Chanthery, E. (2022). Modeling complex systems with heterogeneous Petri nets (HtPN). In *DX'2022*.

Wang, J. and Wang, J. (1998). *Time Petri nets*. Springer.