Diagnosis using Unfoldings of Parametric Time Petri Nets

B. Grabiec[2] , L.M. Traonouez[3], C. Jard[2], D. Lime[1] and O.H. Roux[1⋆]

[1] Ecole centrale & IUT, IRCCyN, Nantes, France

[2] ENS Cachan & INRIA, IRISA, Rennes, France
Université européenne de Bretagne

[3] Software Technologies Laboratory, Università di Firenze, Italy

Abstract. This paper considers the model of Time Petri Nets (TPNs) extended with time parameters and its use to perform on-line diagnosis of distributed systems. We propose to base the method on unfoldings. Given a partial observation, as a possibly structured set of actions, our method determines the causal relation between events in the model that explain the observation. It can also synthesize parametric constraints associated with these explanations. The method is implemented in the tool Romeo. We present its application to the diagnosis of the example of a cowshed with pigs.

Keywords: Unfolding, Time Petri Nets, parameters, diagnosis, supervision

1 Introduction

In this paper, we decided to bring attention on a dynamic verification method, called *model-based supervision*. It is established that diagnosing dynamical systems, represented as discrete-event systems, amounts to finding what happened to the system from existing observations (an event log) derived from sensors. In this context, the diagnostic task consists in determining the trajectories compatible with the observations. The standard situation is that the observed events correspond to the firing of some transitions of the model, while the other transitions are just internal (this situation is called "partial observation" in supervisory control theory [4]). Supervision, based on unfoldings [7,12] in our case, is implemented by the on-the-fly construction of the unfolding, guided by the observations. With this dynamic approach, since we consider only finite sequences of observations, decidability questions become much easier. The only requirement is to be able to decide whether a transition can be fired or not. Petri nets for supervisory control and diagnosis have been proposed in numerous papers (see for instance [16] and [9]). In most cases the construction of diagnosers is based on the state graph (i.e. the interleaving view). The use of unfoldings is

⋆ This work was partially funded by the ANR national research program DOTS (ANR-06-SETI-003).

more recent. Safe ordinary nets are used in [8] with emphasis on distributed diagnosis. This has been extended to safe time Petri nets in [5]. The parametric case has not been considered yet.

The great interest of unfoldings in that task is their ability to infer the possible causal dependencies, which are not in general part of the observations. We think that adding parameters in specifications is a real need. It is often difficult to fix them a priori: indeed, we expect from the analysis some useful information about their possible values. This feature clearly adds some "robustness" to the modeling phase. It is particularly relevant for the supervision activity we consider, in which an arbitrary choice of parameters often avoids to find explanations compatible with the observations. This leads to rejection of the model; moreover, no additional knowledge how to correct it is provided.

We implemented our method inside the Romeo tool developed in the IRCCyN lab in Nantes [11], available as free software. This implementation allowed us to demonstrate the proposed supervision method on small case studies. In this paper we chose to develop a new case study, the "cowshed with pigs". It is freely inspired from [10], in which the idea was to show how Uppaal can handle some hybrid models. Here, we consider a Time Petri Net modelling with time parameters. By observing some particular transitions of the model, we show that it is possible to infer causalities between the corresponding events, allowing us to correlate them in order to find the root causes. Furthermore, the method can compute constraints on parameters that must be satisfied in order to explain the observations.

We consider here for the first time, the possibility of having a structured set of observations. The goal of the supervision is to produce explanations compatible with this set (no contradiction between the respective causal structures).

The contributions of this paper are:

- a general method for on-line diagnosis based on Time Petri Nets with parameters, able of causal, time, and parametric inference from a structured set of observations.
- an illustration using an original model of cowshed, which could be of general interest for the community.

The paper is organised as follows. In Section 2, we first present the Time Petri net model with parameters and the way it can be unfolded. Then the model of observations is presented in Section 3 and how it is used to guide the construction of the unfolding. Section 4 describes our case study and illustrates the method, before a few words of conclusion.

2 Time Petri Nets with parameters and their unfolding

2.1 General notations

We denote by $\mathbb{N}$ the set of non-negative integers, by $\mathbb{Q}$ the set of rational numbers and $\mathbb{R}$ the set of real numbers. For A denoting the sets $\mathbb{Q}$ or $\mathbb{R}$, $A_{\geq 0}$ (resp. $A_{>0}$)

denotes the subset of non-negative (resp. strictly positive) elements of A. Given $a, b \in \mathbb{N}$ such that $a \leq b$, we denote by $[a..b]$ the set of integers greater or equal to a and less or equal to b. For any set X, we denote by $|X|$ its cardinality. In the symbolic expressions, $\wedge$ denotes the logical conjunction, $\vee$ the logical disjunction and $\neg$ the logical negation operators. We will also use $\Rightarrow$ as the logical implication.

For a function f on a domain D and a subset C of D, we denote by $f_{|C}$ the restriction of f to C.

Let X be a finite set. A (rational) *linear expression* on X is an expression of the form $a_1x_1 + \cdots + a_nx_n$, with $n \in \mathbb{N}$, $\forall i, a_i \in \mathbb{Q}$ and $x_i \in X$. The set of linear expressions on X is denoted $Expr(X)$. A *linear constraint* on X is an expression of the form $L_X \sim b$, where L_X is a linear expression on X, $b \in \mathbb{Q}$ and $\sim \in \{<, \leq, \geq, >\}$. We will also use abbreviations like $=$ and $\neq$.

For the sake of readability, when non-ambiguous, we will "flatten" nested tuples, *e.g.* $\langle\langle\langle B, E, F\rangle, l\rangle, v, \theta\rangle$ will be written $\langle B, E, F, l, v, \theta\rangle$.

2.2 Petri nets

Definition 1 (Place/transition net). *A place/transition net (P/T net) is a tuple $\langle P, T, W\rangle$ where: P is a finite set of* places*, T is a finite set of* transitions*, with $P \cap T = \emptyset$ and $W \subseteq (P \times T) \cup (T \times P)$ is the* flow relation*.*

This structure defines a directed bipartite graph.

We further define, for all $x \in P \cup T$, the following sets: ${}^\bullet x = \{y \in P \cup T \mid (y, x) \in W\}$ and $x^\bullet = \{y \in P \cup T \mid (x, y) \in W\}$. These set definitions naturally extend by union to subsets of $P \cup T$.

A *marking* $m : P \to \mathbb{N}$ is a function such that (P, m) is a multiset. For all $p \in P$, $m(p)$ is the number of *tokens* in the place p. In this paper we restrict our study to *1-safe* nets, *i.e.* nets such that $\forall p \in P,\ m(p) \leq 1$. Therefore, in the rest of the paper, we usually identify the marking m with the set of places p such that $m(p) = 1$. In the sequel we will call *Petri net* a marked P/T net, *i.e.* a pair $\langle \mathcal{N}, m\rangle$ where $\mathcal{N}$ is a P/T net and m a marking of $\mathcal{N}$, called *initial marking*.

A transition $t \in T$ is said to be enabled by the marking m if ${}^\bullet t \subseteq m$. We denote by $\mathsf{en}(m)$, the set of transitions enabled by m.

There is a path $x_1, x_2, \ldots, x_n$ in a P/T net iff $\forall i \in [1..n]$, $x_i \in P \cup T$ and $\forall i \in [1..n-1], (x_i, x_{i+1}) \in W$.

In an acyclic P/T net, consider $(x, y) \in P \cup T$. x and y are *causally related*, which we denote by $x < y$, iff there exist a path in the net from x to y. x and y are in *conflict*, which we denote by $x \# y$, iff there exists two paths $p, t, \ldots, x$ and $p, t', \ldots, y$, starting from the same place $p \in P$ but such that $t \neq t'$. It is also convenient to consider the relation of direct conflict between transitions, denoted $x\ \mathsf{conf}\ y$, indicating that they share in their presets the place that originated the conflict (${}^\bullet x \cap {}^\bullet y \neq \emptyset$). x and y are in *concurrency*, which we denote by $x\ \mathsf{co}\ y$, iff none of the two previous relations holds, that is to say $\neg(x < y) \wedge \neg(y < x) \wedge \neg(x \# y)$.

An *occurrence net* is an acyclic P/T net, finite by precedence, and such that no element is in conflict with itself and each place has at most one input transition. We use the classical terminology of *conditions* and *events* to refer to the places and transitions in an occurrence net.

Definition 2 (Branching process). *A* branching process *of a Petri net* $\mathcal{N} = \langle P, T, W, m_0 \rangle$ *is a labeled occurrence net* $\beta = \langle \mathcal{O}, l \rangle$ *where* $\mathcal{O} = \langle B, E, F \rangle$ *is an occurrence net and* $l : B \cup E \to P \cup T$ *is the labeling function such that:*

- $l(B) \subseteq P$ *and* $l(E) \subseteq T$,
- *for all* $e \in E$, *the restriction* $l_{|^\bullet e}$ *of* l *to* $^\bullet e$ *is a bijection between* $^\bullet e$ *and* $^\bullet l(e)$,
- *for all* $e \in E$, *the restriction* $l_{|e^\bullet}$ *of* l *to* $e^\bullet$ *is a bijection between* $e^\bullet$ *and* $l(e)^\bullet$,
- *for all* $e_1, e_2 \in E$, *if* $^\bullet e_1 = {}^\bullet e_2$ *and* $l(e_1) = l(e_2)$ *then* $e_1 = e_2$.

E should also contain the special event $\bot$, such that: $^\bullet \bot = \emptyset$, $l(\bot) = \emptyset$, *and* $l_{|\bot^\bullet}$ *is a bijection between* $\bot^\bullet$ *and* m_0.

Example 1. Fig. 1b shows one branching process of the net presented in Fig. 1a (ignoring any timing or parameter information). The labels are put inside the nodes. We can see that the branching process in Fig. 1b unfolds the loop t_1, t_2, t_0 once. This loop could be unfolded infinitely many times, leading to an infinite branching process.

Branching processes can be partially ordered by a *prefix relation*. There exists the greatest branching process according to this relation for any Petri net $\mathcal{N}$, which is called the *unfolding* of $\mathcal{N}$, denoted $\mathcal{U}(\mathcal{N})$.

Let $\beta = \langle B, E, F, l \rangle$ be a branching process.

A co-*set* in β is a subset B' of B such that $\forall b, b' \in B',\ b \ \mathsf{co}\ b'$.

A *configuration* of β is a set of events $E' \subseteq E$ which is causally closed and conflict-free, that is to say $\forall e' \in E', \forall e \in E,\ e < e' \Rightarrow e \in E'$ and $\forall e, e' \in E', \neg(e \# e')$.

For any co-set B', $l(B')$ defines a subset of the marking of the net. A *cut* is a maximal co-set (inclusion-wise). For any configuration E', we can define the set $\mathsf{Cut}(E') = E'^\bullet \setminus {}^\bullet E'$, which is the marking of the Petri net obtained after executing the sequence of events in E'.

An *extension* of β is a pair $\langle t, e \rangle$ such that e is an event not in E , s.t. $^\bullet e \subseteq B$ is a co-set, the restriction of l to $^\bullet e$ is bijection between $^\bullet e$ and $^\bullet t$ and there is no $e' \in E$ s.t. $l(e') = t$ and $^\bullet e' = {}^\bullet e$. Adding e to E and labeling e with t gives a new branching process. Starting from the event $\bot$, and adding successively possible extensions forms the "unfolding algorithm".

2.3 Parametric Time Petri nets

A mainstream way of adding time to Petri nets is by equipping transitions with a time interval [13,3]. We consider here an extension allowing the designer to leave open the knowledge of time bounds by putting symbolic expressions on parameters in time intervals instead of rational constants.

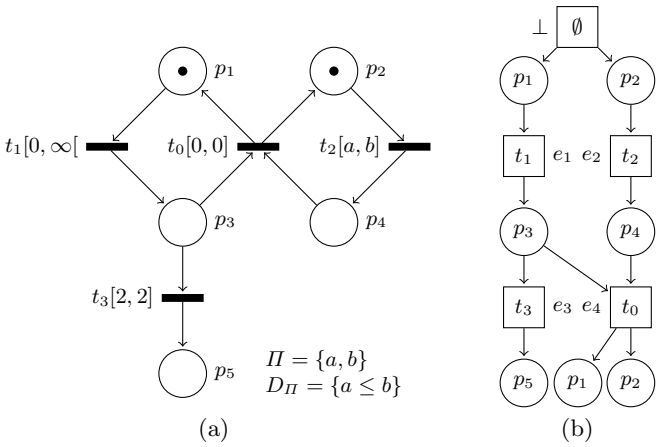

Fig. 1. A parametric time Petri net (a) and a prefix of the unfolding of its underlying (untimed) Petri net (b).

Definition 3 (Parametric Time Petri net). *A parametric Time Petri Net (PTPN) is a tuple* $\langle P, T, W, m_0, \mathsf{eft}, \mathsf{lft}, \Pi, D_\Pi \rangle$ *where:*

$\langle P, T, W, m_0 \rangle$ *is a Petri net,* Π *is a finite set of* parameters ($\Pi \cap (P \cup T) = \emptyset$)*,* D_Π *is a conjunction of linear constraints describing the set of* initial constraints *on the parameters, and* $\mathsf{eft} : T \to \mathbb{Q}_{\geq 0} \cup Expr(\Pi)$ *and* $\mathsf{lft} : T \to \mathbb{Q}_{\geq 0} \cup \{\infty\} \cup Expr(\Pi)$ *are functions respectively called* earliest (eft) *and* latest (lft) *transition firing times. For each transition* t*, if* $\mathsf{eft}(t)$ *and* $lft(t)$ *are constants, it is assumed that* $\mathsf{eft}(t) \leq \mathsf{lft}(t)$*, otherwise, it is assumed that* $D_\Pi \Rightarrow \mathsf{eft}(t) \leq \mathsf{lft}(t)$*.*

Example 2. Fig. 1a gives an example of a PTPN. Notice that the time interval of transition t_2 refers to two parameters a and b. The only initial constraint is $D_\Pi = \{a \leq b\}$.

Given a PTPN $\mathcal{N} = \langle P, T, W, m_0, \mathsf{eft}, \mathsf{lft}, \Pi, D_\Pi \rangle$, we denote by $\mathsf{Untimed}(\mathcal{N})$ the Petri net $\langle P, T, W, m_0 \rangle$. The definition of unfolding for PTPN is developed in [14,15]. It relies on an extension and improvement of [6] to have a compact representation of the unfolding and to deal with parameters. The idea is to decorate the unfolding of the underlying net in associating to each event e a symbolic expression $\theta(e)$ representing the constraints that must be satisfied to justify the occurrence of e. For each event e, we consider its firing date represented by the variable θ_e. The expressions on events are boolean expressions on linear constraints on the set of variables and parameters. Fig.2 gives an example of such "decorated" unfolding.

Let $\mathcal{N} = \langle P, T, W, m_0, \mathsf{eft}, \mathsf{lft}, \Pi, D_\Pi \rangle$ be a PTPN and $\beta = \langle B, E, F, l \rangle$ be the associated unfolding of $\mathsf{Untimed}(\mathcal{N})$. We define the *enabling date* of an event $e \in E$ as the expression $\mathsf{TOE}(e)$ standing for $\max_{f \in {}^\bullet e} \theta_f$. It gives the date at which the corresponding transition has been enabled.

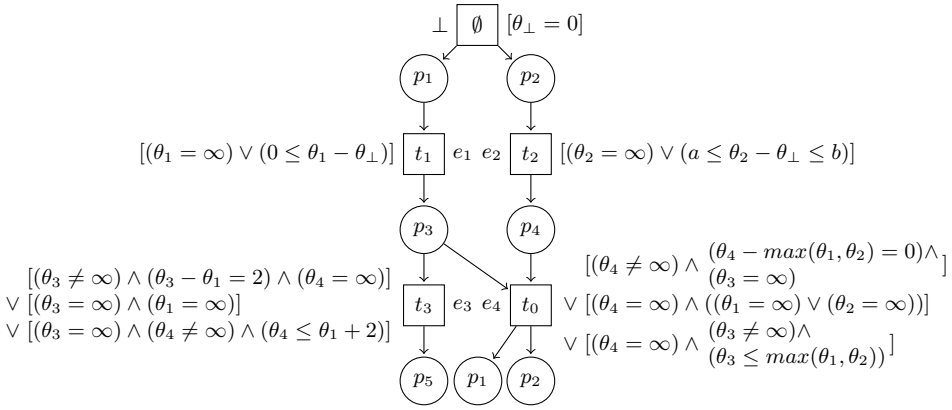

Fig. 2. A prefix of the symbolic unfolding of the PTPN of Fig. 1a.

Definition 4 (Valid timing function for an unfolding). *Given a PTPN* $\mathcal{N} = \langle P, T, W, m_0, \mathsf{eft}, \mathsf{lft}, \Pi, D_\Pi \rangle$. *Let* $\beta = \langle B, E, F, l \rangle$ *be the unfolding of* $\mathsf{Untimed}(\mathcal{N})$. *The* timing function θ *is defined by* $\theta(\perp) = 0$ *and* $\forall e \in E$ $(e \neq \perp)$,

$$\begin{cases} \big[(\theta_e \neq \infty) \ \wedge \ (\mathsf{eft}(l(e)) \leq \theta_e - \mathsf{TOE}(e) \leq \mathsf{lft}(l(e))) \wedge \bigwedge_{e' \in E, e' \ \mathsf{conf} \ e} (\theta_{e'} = \infty)\big] \\ \vee \ \big[(\theta_e = \infty) \ \wedge \ \bigvee_{b \in {}^\bullet e} (\theta_{{}^\bullet b} = \infty)\big] \\ \vee \ \big[(\theta_e = \infty) \ \wedge \ \bigvee_{e' \in E, e' \ \mathsf{conf} \ e} [(\theta_{e'} \neq \infty) \wedge \ (\theta_{e'} \leq \mathsf{TOE}(e) + \mathsf{lft}(l(e)))]\big] \end{cases}$$

Note that in this definition, the parameters appear through the functions eft and lft.

The first line of the expression means that the event e has been fired, and consequently that no conflicting events has been fired and that its firing date must conform to its time interval according to the TPN semantics. The remaining two lines consider the case in which the event e has not been fired (coded by the expression $\theta_e = \infty$). There are two possibilities: either a preceding event has not yet fired, or a conflicting event has been fired and has prevented e to occur. This latter case means that such conflicting event has fired while e was enabled.

Example 3. Fig. 2 shows a symbolic prefix of the unfolding of the PTPN in Fig. 1a. We can see that each event is attributed with a symbolic expression. The expressions are formed with variables denoting the firing dates of the considered event and of its neighbourhood (the events that directly precede and those in conflict) and parameters. In practice, the expressions are implemented in polyhedrons.

We also define the set of events temporally preceding an event $e \in E$ as: $\mathsf{Earlier}(e) = \{e' \in E \mid \theta(e') < \theta(e) \text{ is satisfiable}\}$.

Following the standard semantics of TPNs, [2] has defined the notion of valid time configuration, which can be used here:

Definition 5 (valid time configuration). *A configuration E' of $\mathcal{U}(\mathsf{Untimed}(\mathcal{N}))$ is a* valid time configuration *of $\mathcal{U}(\mathcal{N})$ iff $(\theta_\perp = 0)$ and*

$$\bigwedge_{e \in E' \setminus \{\perp\}} [\theta_e \geq \mathsf{TOE}(e) + \mathsf{eft}(l(e)) \wedge \bigwedge_{e' \in \mathsf{en}(l(\mathsf{Cut}(\mathsf{Earlier}(e))))} \theta_e \leq \mathsf{TOE}(e') + \mathsf{lft}(l(e'))]$$

Let us consider a maximal (in term of set inclusion) configuration E' of $\mathcal{U}(\mathsf{Untimed}(\mathcal{N}))$, extended with the events that are in direct conflict E'' and equipped with the corresponding symbolic expressions of $\mathcal{U}(\mathcal{N})$. Assuming that events in E' have fired and that events in E'' not, E' is a valid time configuration if the conjunction of all expressions of $E' \cup E''$ is satisfiable. This leads to the following theorem [14].

Theorem 1 (Correctness). *Let $\langle B, E, F, l, v, \theta \rangle$ be the unfolding of a parametric time Petri net $\mathcal{N} = \langle P, T, W, m_0, \mathsf{eft}, \mathsf{lft}, \Pi, D_\Pi \rangle$. Consider a maximal configuration $E' \subseteq E$, and $E'' = \{e \in E \mid \exists e' \in E', e \mathsf{\ conf\ } e'\}$*
E' is a valid time configuration iff

$$[\bigwedge_{e \in E'} (\theta_e \neq \infty) \wedge \bigwedge_{e \in E''} (\theta_e = \infty)] \Rightarrow \bigwedge_{e \in E' \cup E''} \theta(e) \text{ is satisfiable.}$$

3 Application to supervision

We first define the notion of structured observation and then show how to guide the construction of a finite unfolding containing the configurations that are compatible with the observations. We consider that the real distributed system under supervision has been instrumented in such a way that it will produce events (like prints used for debugging) during its execution. These events have a name, picked up in some finite alphabet Σ and can be possibly related to each other. In practice, we consider three cases: two events can be causally related, they can be concurrent, or their relation is not known. As usual, the causal relation must be an order. The two others are just symmetric.

Definition 6 (Observation). *An* observation *is a finite set of events O, equipped with a causal order $\preceq$ and a symmetric relation co. If two events are not related, their relation is said to be "unknown". An event also has a name, addressed by the labelling function $\lambda : O \to \Sigma$.*

In order to relate the observation and the model, we also consider that transitions of the PTPN are labelled by a similar function $\lambda : T \to \Sigma \cup \{\epsilon\}$. The ϵ symbol not belonging to Σ is used to indicate that the occurrence of the transition cannot be linked to an observable event. The labelling does not need to be injective, and in general the same observation can be explained by several trajectories of the model.

We construct the unfolding compatible with the observation. To define this notion of compatibility, we consider the maximal configurations and ask they do not contain events and relations that contradicts the observation. Given an observation O, we consider the Parikh vector $\varpi(O) = (|\lambda^{-1}(a)|)_{a\in\Sigma}$, which counts the number of occurrences of each action in O. The same function can also be applied to configurations, considering that for each event e, $\lambda(e)$ is in fact $\lambda(l(e))$.

Definition 7 (Compatibility). *The unfolding of a PTPN $\mathcal{N}$ is* compatible *with an observation O if all its maximal (in the sense of set inclusion) configurations are. A configuration E is* compatible *with an observation iff:*

- $\forall e \in E, \varpi(E) = \varpi(O)$ *and*
- $\forall o_1, o_2 \in O, o_1 \preceq o_2 \Rightarrow$
  $\exists e_1, e_2 \in E$ *s.t.* $(\lambda(o_1) = \lambda(e_1)) \wedge (\lambda(o_2) = \lambda(e_2)) \wedge (e_1 \leq e_2)$
- $\forall o_1, o_2 \in O, o_1$ *co* $o_2 \Rightarrow$
  $\exists e_1, e_2 \in E$ *s.t.* $(\lambda(o_1) = \lambda(e_1)) \wedge (\lambda(o_2) = \lambda(e_2)) \wedge (e_1$ *co* $e_2)$

Theorem 2 (Finiteness). *Given a finite observation, if the PTPN does not contain loops of ϵ transitions, the set of compatible configurations is finite and thus the unfolding.*

Proof. Because of the finiteness of the original Petri net, the only possibility to obtain an infinite object is to have an infinite configuration. Such a configuration contains some observable events (events e such that $\lambda(l(e)) \neq \epsilon$). They are in finite number, due to the finiteness of the observation and by application of the Parikh constraint. Thus, the only possibility is to have an infinite number of ϵ events. Because of the safeness of the net, this infinite set of events must form a chain of causality, which is prevented by the absence of ϵ-loop in the net.

At the end of the observation, we obtain a finite unfolding in which each event is equipped with a symbolic expression. From Theorem 1, it is possible to extract the valid timed configurations. This is done by considering the maximal configurations of the underlying untimed net, extended by the events that are in direct conflict with some event of the configuration. The associated symbolic constraint is given by Theorem 1. After Boolean simplification, keeping only the configurations in which the expression is satisfiable, we obtain a set of timed configurations which constitutes the set of "explanations". An explanation adds in general a lot of information to the observation:

- It has inferred some added causal and concurrent relations between the observable events;
- it has inserted also some patterns of non observable events;
- it gives the constraint that must be satisfied between all the firing dates of the events;
- it gives some constraints about the possible values of the parameters.

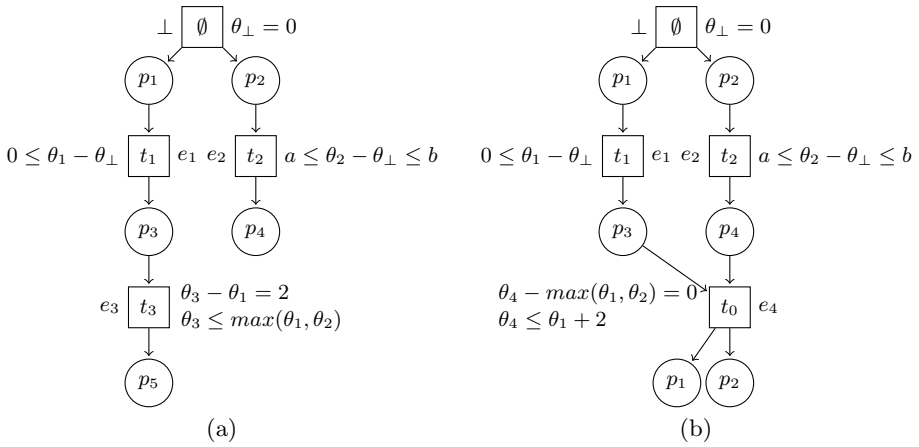

Fig. 3. Two possible explanations.

This is illustrated in Fig. 3. We have considered the PTPN of Fig. 1a in which only transitions t_1 and t_2 are observable and labelled with the same letter. Let us now consider a simple observation formed with only two occurrences of the letter. The finite symbolic unfolding we obtain is the one depicted in Fig. 2. There are only two maximal valid timed configurations as shown in Fig. 3.

4 Case study

4.1 The continuous model

In this section, we present a realistic case-study based on the industrial case study for climate control in a cowshed proposed in [10]. The problem is to keep the temperature, humidity, CO2 and ammonia concentrations at specified levels so that the well-being of pigs is ensured. Though it would be relevant to model temperature, humidity, CO2 and ammonia concentration we limit ourselves to modeling only temperature. It would though be easy to include the disregarded climate parameters since the mixing dynamics are, roughly, identical.

The cowshed is divided into distinct climatic zones which interact by exchanging air flow. Besides internal air flow a zone interact with the ambient environment by activating a ventilator in an exhaust pipe and also by opening a screen to let fresh air into the building. Air flowing from outside into the i^{th} zone is denoted $Q_i^{in}[m^3/s]$. Air flowing from the i^{th} zone to outside is denoted $Q_i^{out}[m^3/s]$. Air flowing from zone i to $i+1$ is denoted $Q_{i,i+1}[m^3/s]$(air flow is defined positive from a lower index to a higher index). A stationary flow balance for each zone i is found: $Q_{i-1,i} + Q_i^{in} = Q_{i,i+1} + Q_i^{out}$ where by definition $Q_{0,1} = Q_{N,N+1} = 0$. The flow balance for zone i is illustrated in Fig. 4.

The temperature in a given zone is impacted in several ways:

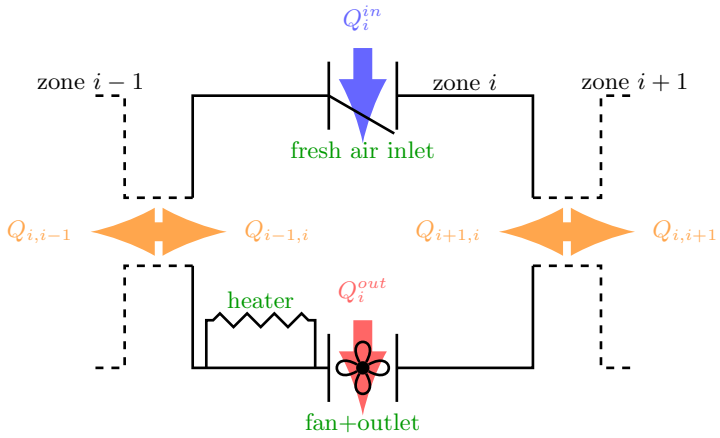

Fig. 4. The zone number i and the air flows through it.

- Each zone is equipped with a heater which can be either on ($u_i = 1$) or off ($u_i = 0$). We denote by $U_i[J/s]$ the resulting heating;
- The pigs in the zone produce heat, denoted by $W_i[J/s]$;
- Air flows from/to adjacent zones;
- Fresh air flows in from outside through the inlet. T_{amb} is the outside temperature. $Q_i^{in,max}$ is the maximum flow of air drawn from outside;
- Air flows outside by means of the fan. $Q_i^{out,max}$ is the maximum flow of air fanned outside.

The evolution of the temperature in zone i is therefore given by the following differential equation, where V_i is the volume of zone i, ρ_{air} the air density [kg/m^3], and c_{air} the specific heat capacity of air [J/kg.C]:

$$\begin{aligned} \tfrac{dT_i}{dt} &= f(T_{i-1}, T_i, T_{i+1}), \text{with} \\ f(T_{i-1}, T_i, T_{i+1}) &= \tfrac{1}{V_i}[Q_i^{in} T_{amb} - Q_i^{out} T_i + Q_{i-1,i} T_{i-1} - Q_{i,i-1} T_i \\ &\quad - Q_{i,i+1} T_i - Q_{i+1,i} T_{i+1} + \tfrac{u_i U_i + W_i}{\rho_{air} c_{air}}] \end{aligned}$$

Among all the factors impacting the temperature in the zone, only three are directly controllable:

- The heater, which is on or off;
- The aperture of the inlet, between 0 and some maximal value inducing $Q_i^{in,max}$;
- The speed of the fan, between 0 and some maximal value inducing Q^{out,max_i}.

In particular, the internal air flows between zones are induced by these last two parameters. We also decided to extend the system with an extra feature which is a possibility of failures of fans (depicted by the state OOO_i in Fig. 5).

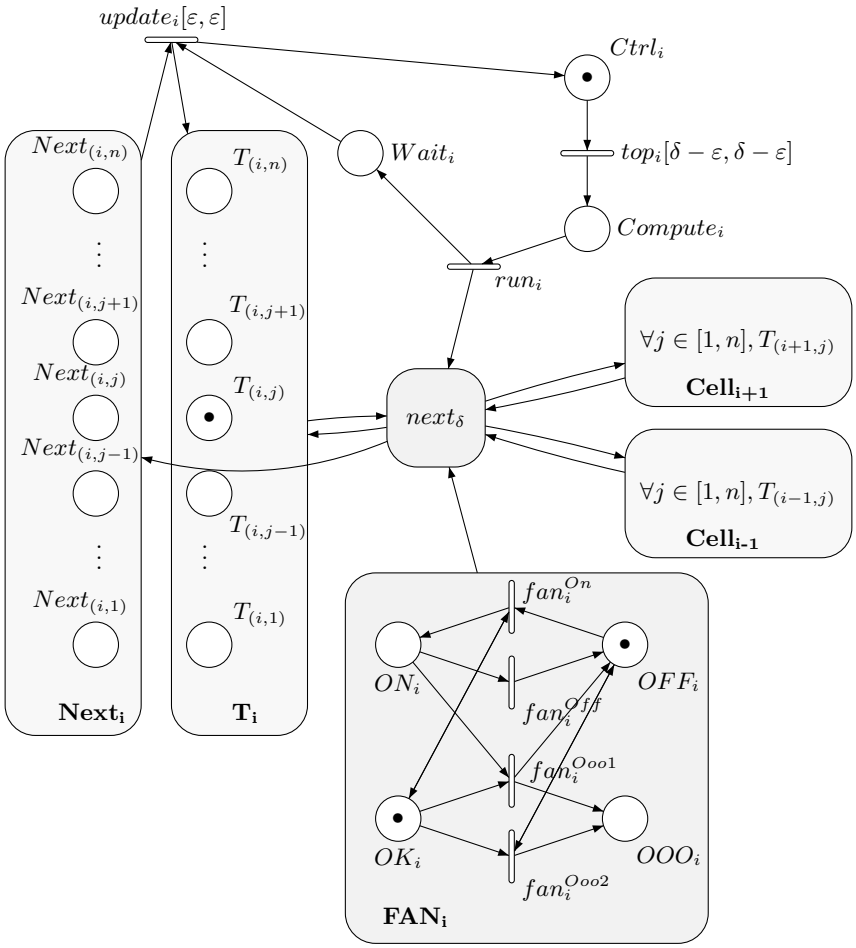

Fig. 5. The cell i.

4.2 The PTPN generation

We consider a discrete evolution of temperature of each cell on a scale of n degrees. Each possible temperature in a cell is represented as a place. The marked place gives the current temperature of the cell. We sample and compute the successor state considering that T_{i-1}, T_i and T_{i+1} are constants. Let us denote $next_\delta(T_i)$ the temperature of cell i, obtained in these conditions after δ units of time (t.u.).

We define $C_\delta \in [0, 1]$ as the coefficient of heat exchange on the duration δ.

1. Without fan (no communication with outside) and without pig:

$$next_\delta(T_i) = T_i + C_\delta * \frac{T_{i-1} - T_i}{3} + C_\delta * \frac{T_{i+1} - T_i}{3}$$

   On an infinite delay ($C_\delta = 1$), we would obtain the heat equilibrium: $next_\delta(T_i) = \frac{T_{i-1}+T_i+T_{i+1}+T_{amb}}{3}$
2. Without fan, but with pigs:
   Let T_δ^W be the heat brought by the pigs during δ time units.

$$next_\delta(T_i) = T_i + C_\delta * \frac{T_{i-1} - T_i}{3} + C_\delta * \frac{T_{i+1} - T_i}{3} + T_\delta^W$$

3. With fan and pigs:
   Let $C_\delta^{amb} \in [0,1[$ the coefficient of heat exchange with outside (depends on the power of the fan: $C_\delta^{amb} = 1$ means a fan with an infinite power).
   $next_\delta(T_i) = C_\delta^{amb} * T_{amb} + (1 - C_\delta^{amb}) * (T_i + C_\delta * \frac{T_{i-1} - T_i}{3} + C_\delta * \frac{T_{i+1} - T_i}{3} + T_\delta^W)$

   The model of a cell i, given in Fig. 5, consists of 4 blocks:

- the block $\mathbf{T_i}$ with one place per temperature,
- the block $\mathbf{Next_i}$ is used to store the intermediate state,
- the block $\mathbf{FAN_i}$ is a model of the behaviour of the fan including possibility of failures,
- the block $next_\delta$ compute the next temperature of the cell and performs the exchange of tokens between bloks $\mathbf{T_i}$ and $\mathbf{Next_i}$ using the block $\mathbf{FAN_i}$ and the temperature of adjacent cells. This exchange is given by the quantization (on the n temperature levels) of the function $next_\delta(T_i)$.

The sampling is controlled by the places $Ctrl_i$ $Compute_i$ and $Wait_i$ and transitions top_i, run_i and $update_i$. The new temperature of the cell i is computed in two steps. First, the new temperature $next_\delta(T_i)$ is computed at $(\delta - \epsilon)$ t.u. and the result is stored in the intermediate places of block $\mathbf{Next_i}$. This intermediate result is obtained in zero t.u. and does not depend of the interleaving since marking of block $\mathbf{T_i}$ is not modified by this computation. Then, the intermediate result is moved from $\mathbf{Next_i}$ to $\mathbf{T_i}$ after ε t.u.

All the possibilities are defined, which leads to a complex graph. With n the number of temperature levels, the model have $2 \times n^3$ transitions.This one is tedious to build by hand. Thus we decided to automatically generate the model by programming a tcl-tk generator, parameterized by the number of considered cells and the temperature scale. This generator of 1000 lines builds an PTPN model as a XML file directly read by Romeo. For example, Fig. 6 gives an insight of the model with 2 cells and 3 levels of temperature.

4.3 The diagnosis experiment

The goal of the experiment is to show a case in which a certain maximal temperature is exceeded in one of the cells. This leads in turn to the death of some pigs. In the model, we assume that we can observe the changes of temperature in each of the cells, and we can get to know if some pigs died. As a result we would like to know the possible explanations of cases in which a pig died. For example, we can imagine a situation where a fan is broken in a cell. Moreover, we would like to obtain some information about the possible dates of death.

The experiment was performed on the system presented in the previous section in Fig. 6. It consists of 2 cells and 3 levels of temperature. In the example, we assume that the temperature of cells is monitored at some given rate, which amounts to 10 units of time.

To be able to execute our scenario, we added some additional transitions to the model (see Fig. 6). They do not change the main functionality of the model. They are used for two reasons. The first reason is to verify whether the critical

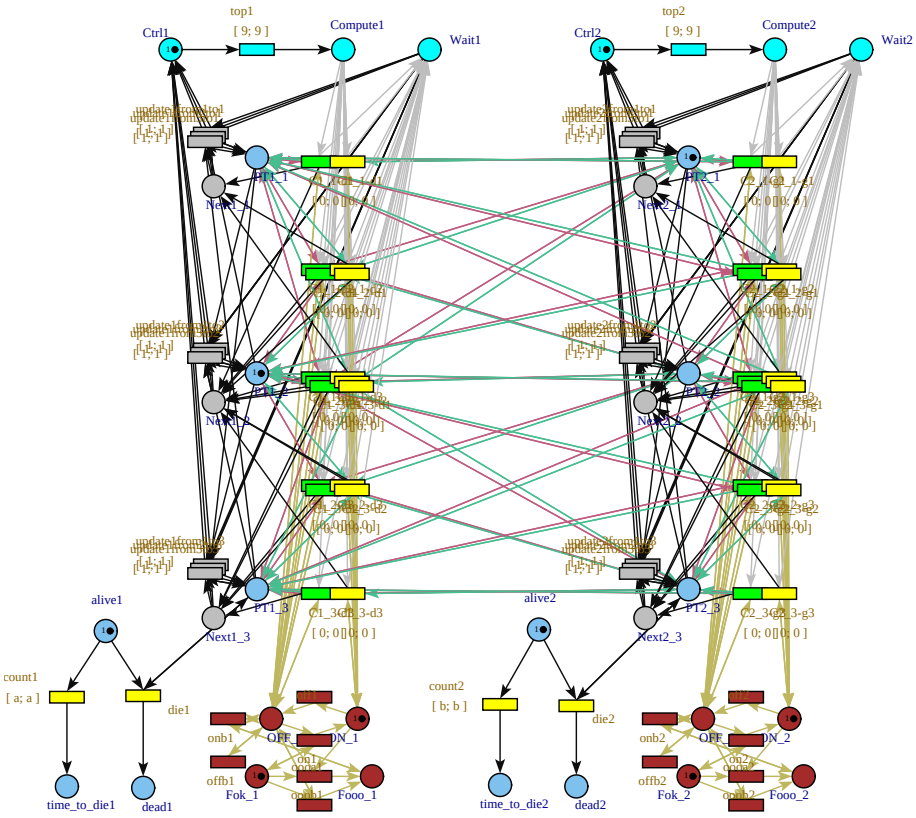

Fig. 6. The model with 2 cells and 3 levels of temperature.

temperature was reached or not. Consequently, each time the extra transitions are fired, one can observe the death of pigs as a consequence of the excessive temperature. The second reason is to compute a minimal delay between two events: the initial event of the model (at time 0), and the potential death of pigs in a cell. The structure which implements this task is based on two transitions. One of the transitions has a parametrized constraint $[a, a]$. The transitions share an input place with a token. The token is active from the very beginning of the model activity. However, the non-parametrized transition t depends also on some other token at place p (in the model it denotes the maximal temperature). Our goal is to get to know when the place p can activate the non-parametrized transition (in the model it denotes the death of pig). We can note that using the structure we get the minimal possible date of firing t by reading the parameter a.

For the purpose of the experiment, we set up an initial temperature for each cell and we entered a set of observed events into our tool for analysis, i.e.: log with unordered temperature measurements, and an event which signals death of an animal in one of the cells. In total, there were 8 observable events and a limit of 6 unobservable events in the observation. As a result, we obtain a prefix consisting of 108 events with 4 possible explanations. From the prefix we can observe that in any of the four scenarios, the fans in the both cells have to be broken before the temperature become critical. Moreover, as a result of the experiment, we get possible valuations of the parameters given in the model. Thus, we get to know that the minimal time amount necessary in order to reach the state dangerous for the pigs amounts to 20 units of time.

To perform the experiment we used a prototype implemented in Romeo, which is a software for analysis of time Petri nets. The experiment was executed on a small machine with 1GB of RAM and 2GHz Intel Pentium processor. The computation time of the example needed about 15 seconds. During our experiments we tested also some different variants of the problem: with more cells, with more levels of temperature, and with different observations. In general, size of the model and observations can strongly influence the time complexity of the diagnosis. It is not difficult to observe that one of the issues which plays a great role in the time consumption of the analysis is the number of unobservable, or indistinguishable events in the system. During the tests we observed many difficult cases in the context of time complexity. We intend to address that issue in our future work and improve the software tool we used for our experiments.

5 Conclusion

The current version of the Romeo tool 2.9.0 is available on the webpage [1]. It offers the possibility of computing symbolic unfoldings for safe time Petri Nets with parameters. When guided by a sequence of actions, this feature allows the user to perform some diagnosis. The diagnosis consists in a finite prefix of the unfolding, presenting all the possible explanations of the input sequence. The explanations show the inferred causal relationships between the events of the model and also give the possible values for the parameters. We think that

such an integrated method is a real added-value for the analysis of concurrent systems, and opens the door to deal with even more complex models like TPNs with stopwatches, or TPNs with more robust time semantics (e.g. with imperfect clocks).

References

1. Roméo - a tool for time Petri nets analysis, *http://romeo.rts-software.org*.
2. T. Aura and J. Lilius. A causal semantics for time Petri nets. *Theoretical Computer Science*, 243(2):409–447, 2000.
3. B. Berthomieu and M. Diaz. Modeling and verification of time dependent systems using time Petri nets. *IEEE trans. on Soft. Eng.*, 17(3):259–273, 1991.
4. C. G. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems, Second Edition.* Springer, 2008.
5. T. Chatain and C. Jard. Time supervision of concurrent systems using symbolic unfoldings of time Petri nets. In *Proceedings of FORMATS*, volume 3829 of *LNCS*, pages 196–210. Springer, 2005.
6. T. Chatain and C. Jard. Complete finite prefixes of symbolic unfoldings of safe time Petri nets. In *Proceedings of ICATPN*, volume 4024 of *LNCS*, pages 125–145. Springer, 2006.
7. J. Esparza and K. Heljanko. *Unfoldings, A Partial-Order Approach to Model Checking.* Monographs in Theoretical Computer Science. Springer, 2008.
8. E. Fabre, A. Benveniste, S. Haar, and C. Jard. Distributed monitoring of concurrent and asynchronous systems. *Journal of Discrete Event Systems, special issue*, pages 33–84, 5 2005.
9. A. Giua. Petri net state estimators based on event observation. In *Proceedings of the IEEE ICDC*, pages 4086–4091, 1997.
10. J.K. Jessen, J.I. Rasmussen, K.G. Larsen, and A. David. Guided controller synthesis for climate controller using uppaal-tiga. In *Proceedings of the 19th International Conference on Formal Modeling and Analysis of Timed Systems*, volume 4763 of *LNCS*, pages 227–240. Springer, 2007.
11. D. Lime, O. H. Roux, C. Seidner, and L.-M. Traonouez. Romeo: A parametric model-checker for Petri nets with stopwatches. In *Proceedings of TACAS*, volume 5505 of *LNCS*, pages 54–57. Springer, 2009.
12. K. L. McMillan. Using unfolding to avoid the state space explosion problem in the verification of asynchronous circuits. In *Proceedings of CAV*, volume 663 of *LNCS*, pages 164–177. Springer, 1992.
13. P. M. Merlin. *A study of the recoverability of computing systems.* PhD thesis, Dep. of Information and Computer Science, University of California, Irvine, CA, 1974.
14. L. M. Traonouez. *Vérification et dépliages de réseaux de Petri temporels paramétrés.* PhD thesis, Ecole centrale de Nantes, available at http://www.irccyn.ec-nantes.fr/˜ traonoue/publications/these.pdf, 2009.
15. L. M. Traonouez, B. Grabiec, C. Jard, D. Lime, and O. H. Roux. Symbolic unfolding of parametric stopwatch petri nets. In *8th International Symposium on Automated Technology for Verification and Analysis (ATVA 2010)*, LNCS, Singapore, September 2010. Springer.
16. T. Ushio, I. Onishi, and K. Okuda. Fault detection based on Petri net models with faulty behaviors. In *Proceedings of the IEEE Int. Conf. on Systems, Man, and Cybernetics*, pages 113–118, 1998.