Fault detection and diagnosis in distributed systems: an approach by partially stochastic Petri nets *

Eric Fabre, Armen Aghasaryan, Albert Benveniste,
Renée Boubour, Claude Jard †

December 22, 1998

Abstract

We address the problem of alarm correlation in large distributed systems. The key idea is to make use of the concurrence of events in order to separate and simplify the state estimation in a faulty network. Petri nets and their causality semantics are used to model concurrency. Special partially stochastic Petri nets are developed, that establish some kind of equivalence between concurrence and independence. The diagnosis problem is defined as the computation of the most likely history of the net given a sequence of observed alarms. Solutions are provided in four contexts, with a gradual complexity on the structure of observations.

*To be referenced as: *Journal of Discrete Event Dynamic Systems, vol. 8, pp. 203-231, 1998.*

†IRISA, Campus de Beaulieu, F-35042 Rennes cedex, FRANCE, fax: +33 2 99 84 71 71, e-mail: name@irisa.fr

Contents

1 Introduction

The complexity of large distributed systems, such as telecommunication or electrical networks, and the huge amount of information carried by them have caused an increase in demand for network management systems. In particular, the area of network fault management requires a lot of expertise and is becoming critical: breakdowns of telecommunication networks cause huge financial losses. Most of the current proposals are built on an *ad hoc* basis, and are usually more involved in structuring the management system than in designing dedicated algorithms. There is a real pressing need for establishing a theoretical foundation of network fault management.

This paper proposes a contribution to this foundation in focusing on the treatment of causal dependencies between alarms and faults. The main idea is to take into account the essentially distributed nature of the problem. This is done by the use of Petri nets and their causality semantics, that are well known as a powerful model for concurrent systems. We base our approach on an explicit description of fault propagations, using capacity-one Petri nets. This allows to deal with multiple faults and to model causal dependencies as well as fault interleaving. Faults play the part of hidden variables; they are not observed directly but manifest their presence by the emission of alarms through the network. Alarms are collected by a network supervisor, the task of which is to "correlate" observations, i.e. to recover coherent fault propagations that explain the observed alarms. This is referred to as the diagnosis operation in the sequel.

The problem is embedded in a stochastic framework that accounts for various random events in the network: reliability of devices, relative frequencies of spontaneous faults, losses of alarms, etc. The randomization also provides a convenient way of introducing robustness against modeling errors on fault propagations. The stochastic model must be designed with care in order to preserve the true concurrence semantics of Petri nets. Traditional stochastic Petri nets fail on this point because they build Markov dynamics on the marking graph of the Petri net, which typically blows up with the amount of concurrency in the system. To avoid an exploding number of possible trajectories of the net, we show that true Markov dynamics must be abandoned. We propose instead "partially stochastic" Petri nets (PSPNs) that provide some kind of equivalence between concurrency and statistical independence. These PSPNs are partially stochastic in the sense they are based both on random and non random variables, related by constraints. A strange but crucial consequence is that they result in dynamics where time is only partially ordered... The natural representation of trajectories for PSPNs relies on the unfolding of the Petri net rather than its marking graph. This construction has a double advantage: it erases obstacles to distributed diagnosis algorithms, and reduces the number of possible trajectories, that are now regarded as causality graphs of faults rather than sequences.

The paper is organized as follows. Section 2 specifies the structure of observations (alarms) and the Petri net model of fault propagations, together with its causality semantics. It analyzes relations between faults and alarms and defines the diagnosis problem. Section 3 is devoted to a review of usual stochastic Petri nets and to the motivation of

PSPNs, that are studied in details. Their trajectories are shown to rely on the unfolding of the Petri net rather than its marking graph. Section 4 provides tools for constructing these trajectories recursively, like a puzzle, relying on a notion of tile. Finally section 5 addresses the diagnosis problem in a progressive way : four levels of difficulty are defined, and solutions are provided, based on the puzzle paradigm.

2 Models for alarms and faults

The notions of *fault* and *alarm* can take very different meanings in the field of network monitoring. In this paper, we adopt the following definition: a fault represents a malfunction event in a system (say the network). Faults are not observed directly, but rather induce the production of alarms, that are collected in some way by a supervisor. This section is devoted to the construction of models for faults and alarms, and to the statement of the diagnosis problem.

2.1 Nature of observations

Due to the size of a network, alarms do not reach the supervisor directly, but are collected through a hierarchy of sensors : local sensors gather alarms stemming from a given region, transmit them to an intermediate supervisor in charge of a bigger region, and so on up to the global supervisor. Protocols defining the nature of alarms incorporate various mechanisms that allow to keep track of *causal or temporal dependencies* between them[1]. We will not enter into the details of each protocol in this paper (an example concerning the SDH protocol can be found in [2] and [1]). However, a reasonable and rather general model states that an alarm bears 1/ information on the fault that generated it, and 2/ information of the kind "has been caused by previous alarms [*list of alarms*]", or 2'/ information like "appeared necessarily after alarms [*list of alarms*]". Of course, we assume relations induced by 2 or 2' don't violate transitivity, so that an observation can be modelled as a directed acyclic graph (DAG) on the finite set of alarms $\{a_1, a_2, \ldots, a_N\}$, as illustrated by fig. 1.

Assumptions 2 or 2' give different meanings to the observed DAG of alarms: it must be regarded respectively as a *causality graph* (CG) or as a *partial order*. The difference is that transitive arcs (such as $a_1 \rightarrow a_3$ on fig. 1) are superfluous in partial orders, while they are meaningful in causality graphs. For simplicity, we assume in this paper the *causality semantics*[2]. Turning back to fig. 1, we shall thus say that alarm b is a *consequence* of alarm a iff the link $a \rightarrow b$ exists. Equivalently, a will be said to be a *cause* of b. More generally, b will be said to be *causally related* to a if there exists an oriented path from a to b. Finally,

[1] We refer for example to *previous alarm* stamping or *local time* stamping techniques for the alarms that are stored in the Management Information Bases of sensors.

[2] *Partial order semantics* can also be captured by the framework presented here, which however requires some technical extensions that we wish to avoid for clarity. This point is postponed to forthcoming publications.

if no causal relation exists between a and b, they will be said to be *concurrent* (denoted by $a \| b$), which means that they could have appeared in any order or even simultaneously.

The "perfect" observation described above is altered by various phenomena, in particular due to the improper behavior of the faulty network :

- Part of the information regarding the links may be unavailable, or simply lost. For example, causal dependencies are derived locally by the sensors, which means that links between alarms collected on different sensors cannot be observed. In other words, the default causality relation between two alarms is "uncertain," which is represented by a dashed arrow on fig. 1, unless it is set to a solid arrow (causality) or an empty link by the sensor that collected these two alarms. We thus observe an "incomplete" causality graph.

- Some alarms may happen to be lost (or masked). This can be caused by buffer overflows, losses of connections, etc. As a consequence, the causal relations regarding such alarms are also lost. Therefore, while causal dependence is a solid information, concurrency has a weaker status since it can result of maskings.

We make two other assumptions on the structure of observations.

(H1) *Causal observation.* Alarms reach the supervisor in a sequence $(a_1, \ldots, a_N)$ such that we never have a_j causally related to a_i and $j < i$.

(H2) Causal dependence relations observed on the alarms coincide with those of the faults that produced these alarms.

Hypothesis (H2) defines an important part of the information used for the diagnosis, and is detailed in the next two sections. The sequence assumed by (H1) will be useful for describing the recursive nature of the diagnosis algorithm. We shall prove however that the result is independent of which sequence is chosen, provided it satisfies the observed CG (included dashed arrows). The proof will rely on an obvious result that we recall right now :

Lemma 1 *Let σ be a causality graph on $\{a_i : 1 \leq i \leq N\}$, and let ϕ be a permutation of the indexes $1 \ldots N$. The sequence $(a_{\phi(1)}, \ldots, a_{\phi(N)})$ is said to* satisfy *or to be* compatible with *σ iff the total order defined by this sequence extends the partial order defined by σ. The set of compatible sequences is denoted by $Lin(\sigma)$.*

If $(a_{\phi(1)}, \ldots, a_{\phi(N)}) \in Lin(\sigma)$, all other compatible sequences are obtained by permutations of successive elements $a_{\phi(k)}$, $a_{\phi(k+1)}$ that are not related by an arrow.

Despite this insensitivity to the choice of the sequence of alarms, (H1) remains technically important as will be discussed in section 5.3. By the way, it already brings an interesting simplification : if the nature of the link between a_k and a_l has been altered for some reason, one would have to test three hypothetic relations. (H1) eliminates the possibility $a_k \leftarrow a_l$ when $k < l$, whence the presence of dashed *arrows* instead of dashed *lines* in the observations. This is the case for the link between a_3 and a_4 on fig. 1. So (H1) reduces the combinatorial explosion on the number of possible (incomplete) CGs when some links are unobserved[3].

Figure 1: (left) A causality graph. On this example, a_2 is a consequence of a_1, a_5 is causally related to a_2 (and thus to a_1), a_3 and a_4 are concurrent. (right) Structure of observations : a damaged or "incomplete" causality graph. The loss of a_6 cancels some causal links. Other relations can't be observed (dashed arrows) : these arrows can either be present or absent.

2.2 Model for fault propagations

We assume the existence of a model describing fault propagations in the network. It is very convenient to express it in the framework of capacity-one Petri nets, a natural tool to represent causality and concurrence relations.

2.2.1 Capacity-one Petri nets

Detailed definitions of Petri nets (PNs) can be found in many books or papers (see [5] for example). Briefly, a net $\mathcal{N} = (P, T, L)$ is composed of finite sets representing *places*, *transitions* and oriented *links* between them : $L \subset (P \times T) \cup (T \times P)$. The preset of a transition $t \in T$, denoted by ${}^\bullet t$ is the set of places that point towards t, while its postset $t^\bullet$ is the set of those pointed by t. Petri nets work through a token game : a state or *marking* is a function that assigns a number of tokens to each place. This number is limited to 1 in capacity-one PNs, that we consider here. A transition such that all places in its preset contain a token, and all places in its postset are empty[4], is said to be *enabled*. An enabled transition can *fire*, in which case tokens are removed in all places of its preset, and one token is put in each place of its postset : this determines a new marking (tokens in all other places don't move). Notice that enabling conditions concern both the preset and the postset : this preserves the firing from violating the capacity-one property. Since tokens (in the preset) but also "holes" (in the postset) are required to enable t, tokens and holes will be considered as equivalent *resources* for t. Observe that although a transition is enabled, it may not fire and its resources may be consumed by another enabled transition instead. Such transitions are said to be in *conflict*. Transitions can also fire simultaneously, provided they require different resources. We then talk about *concurrent* transitions.

[3]In the sequel, a CG of alarms will always refer to an "incomplete" CG, made of dashed and solid arrows, or equivalently to a family of possible "complete" CGs, made of solid arrows only.

[4]To be precise, places of the postset that are not also in the preset must be empty, i.e. places in $t^\bullet \setminus {}^\bullet t$

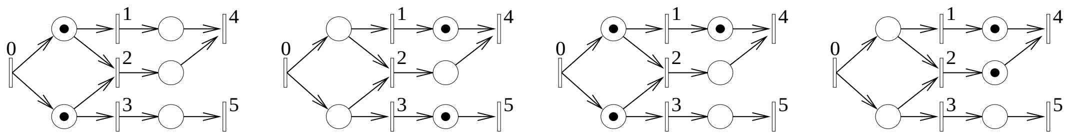

Figure 2: Successive markings in a Petri net. Places are represented as circles, and transitions as flat rectangles, while the arrows stand for the links. Tokens are represented as black patches in the places.

Fig. 2 illustrates the behavior of a Petri net on a toy example. The initial marking is represented on the left : transitions t_1, t_2 and t_3 are enabled, but they can't all fire simultaneously. Nevertheless, t_1 and t_3 don't require the same resources. Firing them gives the second marking. There, only the absorption t_5 and the spontaneous transition t_0 are enabled. Assuming both fire, we get the third marking where only t_2 and t_3 are enabled. Only one of them can fire since they are competing for a token. The last marking is obtained by firing t_2, which enables t_0 and t_4, and so on.

Definition 1 *The initial marking M_0 plus the sequence of fired transitions - here chosen to be $(\{t_1, t_3\}, \{t_0, t_5\}, t_2)$ - will be called a* trajectory, *or* history, *or* path *of the Petri net. We shall denote trajectories by $(M_0, s_1, \ldots, s_N)$, or (M_0, s) for short, where each s_n is a* salvo, *i.e. a (possibly empty) set of simultaneously fired transitions. N will be called the* length *of the trajectory.*

We shall denote by M_n the marking at time n, i.e. the state of the Petri net after applying $s_1, \ldots, s_n$ to M_0. Observe that the n-th salvo s_n produces M_n, and thus applies to M_{n-1}.

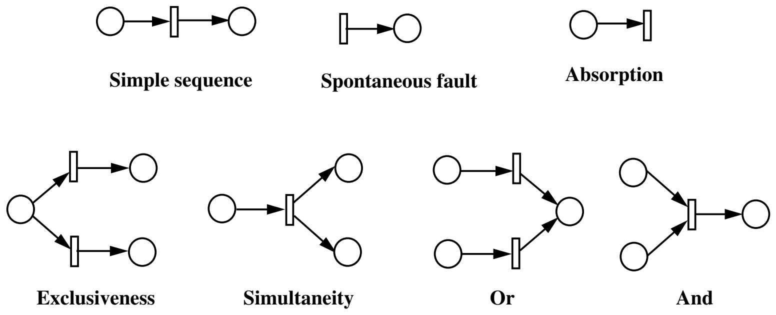

Figure 3: Petri net representation of various dependence relations on faults.

2.2.2 Causality semantics

In this paper, a place represents a given fault. A fault can either be present or absent, whence the restriction to capacity-one places : tokens mark active fault states. Transitions will encode conditions for moving from a fault state to another (fig. 3). We consider PNs with their natural causality semantics on fired transitions: in a sequence s of firings, every transition t_j using a resource that was previously set by transition t_i appears as a direct consequence of t_i. This defines a causality graph $\sigma = CG(s)$. On the toy example above, this yields the CG of fig. 4. Observe that, as a direct consequence of the definition, every sequence s' that would be compatible with this CG - i.e. $s' \in Lin(\sigma)$ - would yield the same final marking, although intermediate ones would be different. A CG derived from a possible sequence of firings is said to be *executable*. CGs of transitions will always be executable in the sequel. Section 4 will define tools to handle these objects.

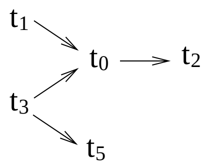

Figure 4: Fired transitions of the toy Petri net of fig. 2 arrange in a causality graph. Of course, repeated occurrences of the same transition would be taken as distinct elements of the CG.

2.3 Nature of the problem

The link between alarms (observations) and faults (hidden variables) is established in the following way : each time a transition fires, it emits an alarm towards the supervisor. This alarm is chosen by each transition t in its alphabet of alarms A_t. The same alarm can be present in several alphabets however. We also add the possibility for a transition to fire silently, which is denoted by the emission of the invisible alarm λ. This mechanism allows to account for losses of alarms. Notice that some transitions may always be silent if their alphabet is reduced to $\{\lambda\}$.

The detailed mechanism that produces links between alarms is not considered by this model, although it is a crucial part of the observation. This would require the specialization to a particular protocol, and to study the inner structure of alarms. We rely on (H2) instead, that allows to keep within a more general framework by stating

$$\begin{array}{lcl} a_i \to a_j \text{ exists} & \Leftrightarrow & a_i \text{ and } a_j \text{ were fired by some } t_i \text{ and } t_j \\ & & \text{such that } t_i \to t_j \text{ exists} \end{array} \qquad (1)$$

In other words, the hidden propagation of faults must exactly match causality relations observed on alarms. However, when a dashed arrow exists between a_i and a_j, both $t_i \to t_j$ and $t_i \quad t_j$ (no arrow) are possible[5]. But of course $t_j \to t_i$ is forbidden. (Notice that, strictly speaking, equivalence (1) is valid provided transitions that fire silently are erased from the causality graph of transitions, as will be explained section 5.5.1.)

[5]Observe that the absence of an arrow between t_i and t_j doesn't imply concurrence, i.e. $t_i \| t_j$, since an indirect causal relation is not excluded.

Now a first diagnosis problem can be stated as follows: given a sequence of alarms, equipped with an "incomplete" set of causal dependence relations (fig. 1), find all sequences of transitions that are compatible with this observation. The following section will embed the problem in a stochastic framework and ask for the most likely sequence.

3 Randomization of the model

Adding probabilities to the algebraic setting defined above has two main advantages. First it allows to incorporate some statistical knowledge on the loss of alarms (maskings) or on the production of faults: some of them may be more likely than others, as stated by reliability tests on devices, or by previous experience on monitoring the network. Secondly, it incorporates some smoothness in the fault net, and allows to account for incomplete knowledge on the consequences of faults, or on the alarms they generate.

Randomness can be introduced at several levels. In the fault net, the spontaneous production or absorption of faults can be assigned different likelihoods. Some faults may also have exclusive possible causes or exclusive consequences: such situations are called *backward* and *forward conflicts* respectively (see the *or* and *exclusiveness* on fig. 3). Here again, all possibilities may not be equally likely. For what concerns observations, a given change in the fault net, i.e. the firing of a transition, can generate non equally likely signatures (alarms), including the invisible alarm λ. Thus the alarm sets A_t should be randomized, at least to account for a random masking of alarms. Losses of links will not be randomized however, since we didn't model their production.

While it is quite easy to randomize the emission/loss of alarms, through conditional probabilities $\mathbb{P}(a|t)$, $t \in T, a \in A_t$, the construction of a relevant stochastic fault net requires some attention. We briefly review below existing formalisms in order to highlight suitable properties that we wish to keep, as well as bad side effects that need to be cancelled.

3.1 Existing stochastic Petri nets

Stochastic Petri nets (SPNs). The usual way transitions are made stochastic is by considering timed Petri nets where waiting times are random [5], [7]. Each enabled transition t in marking M_n selects at random a waiting time, according (usually) to an exponential distribution with parameter $\alpha_t = 1/(\text{average waiting time})$. The transition having the shortest waiting time wins the race and fires, which determines the new marking M_{n+1}. It is quite straightforward to check that if $t_1, \ldots, t_k$ are enabled by M_n, with respective parameters $\alpha_1, \ldots, \alpha_k$, t_i fires with probability $\alpha_i / \sum_{j=1}^{k} \alpha_j$. This protocol makes the sequence $(M_n)_{n \geq 0}$ of successive markings a (discrete time) Markov chain, which is a convenient property in view of a maximum likelihood diagnosis: it brings us into the Hidden Markov Model framework [6].

Discussion. Notice however a crucial drawback of Markov dynamics: the probability of firing t_i depends on the whole current marking M_n, that determines which transitions take part to the race. This feature is rather bothering for the kind of application we have in mind. First of all because of state explosion: we cannot afford to work on the state space of the model. It can be very large, especially because of concurrency (as expected in a telecommunication network for example). Secondly, it asserts that transitions having no common resource, and possibly living far-away one of the other, may however have a *statistical* interaction. Consider the extreme case of two disconnected PNs (fig. 5-a). The standard SPN model claims that gathering formally these two nets in a single one automatically generates interaction, while one would rather expect some kind of statistical independence. Finally, still on the example of fig. 5-a, the probability of firing t_1 and then t_2 is $\alpha_1/(\alpha_1+\alpha_2+\alpha_3+\alpha_4)\cdot\alpha_2/(\alpha_2+\alpha_4)$. This differs from the probability of firing the alternate sequence t_2 then t_1. In other words, the order in which concurrent transitions fire is probabilized. In a fault net, knowing which system failed first is useless if they are not in direct interaction; the relevant information is whether they failed, and what was the most likely cause if they did. One cannot hope for a distributed diagnosis algorithm if the complete order in which failures occur has to be determine, and thus Markov dynamics on the marking graph have to be rejected.

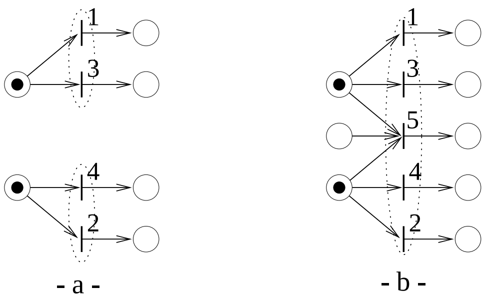

Figure 5: Extended conflict sets defined for probabilistic Petri nets are highlighted by dotted lines. This notion is static: it doesn't take the current marking into account.

Other models. Changing the protocol (pre-selection, priorities, other probability distributions, etc.) do little against the above drawbacks since the probability of firing t may not only depend on the current global marking M_n, but also on extra variables (and possibly previous markings). The difficulty comes from the fact that the selection policy solves conflicts between transitions that are not in direct conflict since they do not share any resource.

Generalized stochastic Petri nets (GSPNs) [7] seem to provide an interesting framework. If we drop their priority rules (useless for our purpose, as mentioned above) and assume all transitions are immediate, we get a sub-family of GSPNs, sometimes referred to as probabilistic Petri nets (PPNs), that explicitly rely on the notion of conflict set. An extended conflict set (ECS) is a set of transitions where conflicts may appear, for some marking. ECSs are defined as equivalence classes for the relation "can be in conflict with," when the latter is completed by reflexivity and transitivity (see fig. 5). Notice that ECSs are static : they are defined a priori and don't depend on the current marking. PPNs work according to the following protocol : they first choose an ECS among those containing at least one enabled transition, then they select one transition in this ECS by comparing relative weights, exactly as in standard SPNs. As a result, only one transition is fired at a time, which brings us back to Markov dynamics and their bad consequences.

Key features. An interesting feature appears however in some particular PPNs[6] provided we "drop the order", as suggested[7] in [8]. Consider the examples of fig. 5, and compute the probability of firing t_1 and t_2, regardless of the order (i.e. sum the probabilities of the two sequences). One easily gets :

$$\mathbb{P}(t_1 \text{ and } t_2 \text{ have fired}) = \frac{\alpha_1}{\alpha_1 + \alpha_3} \cdot \frac{\alpha_2}{\alpha_2 + \alpha_4} \tag{2}$$

which amounts to *independent* selections in *effective* conflict sets. By "effective conflict sets", we mean conflict sets of transitions that are actually competing for a token or a hole in the current marking : notice that (2) holds for both cases of fig. 5.

We exactly aim at such properties in the sequel : *concurrence and statistical independence should coincide.* Indeed, concurrence is the key to distributed algorithms, since it allows to perform local computations and merge them for a global purpose. Its statistical counterpart is independence, therefore one should aim at an exact matching of these notions. Specifically, this induces the following requirements :

1. the probability of firing a given transition should depend only on its own resources, rather than on the complete current marking,

2. the probability of a transition should not depend on what concurrent transitions do, and the order in which concurrent transitions fire should not be randomized,

3. firings should not necessarily be reduced to one transition at a time,

4. no restriction should be put on the PN structure.

At least points 3 and 4 are not satisfied by PPNs, even if we "drop the order". The next section proposes a new randomization of Petri nets that achieves all the above objectives.

3.2 Ideas for new stochastic Petri nets

The discussion above suggests to use a random routing policy, that is to let resources choose which transition they will fire. On fig. 5-a for example, the token in α could select t_1 or t_3 with probabilities $(\frac{\alpha_1}{\alpha_1+\alpha_3}, \frac{\alpha_3}{\alpha_1+\alpha_3})$. The token in β could proceed in the same way, independently, which would yield (2). This formalism is quite appealing, but fails

[6]Namely, in confusion free PPNs, which means that firing a transition in an ECS cannot disable a transition in another ECS.

[7]The phenomenon we describe here is developed on the "PPN part" of GSPNs in [8], i.e. on the set of immediate transitions.

with several respects. First, on the example, it imposes the simultaneous firing of two transitions, while we could wish to fire only one, and wait on the other side. Secondly, it leads to *lockings*, as depicted on fig. 5-*b*: assuming α and β have selected t_5, nothing happens since a token is missing in γ. Thus the null event has a non null probability[8]!

The problem actually comes from the fact that we impose resources to make a choice, which amounts in case *a* to impose the number of transitions that fire. In order to relax this constraint, we allow an extra possibility to resources: that of making no choice! But we *must not* randomize this "wait" since this would immediately assign different probabilities to sequences of concurrent transitions. We thus need an original "hybrid" framework where random and non-random variables cooperate, and need also to extend the notion of likelihood to partially random events.

3.3 Partially stochastic Petri nets (PSPNs)

A general framework has been designed to handle systems involving both random and non-random variables, related by constraints [4]. The interested reader will find there the definition of a generalized likelihood, the counterpart of the HMM (Hidden Markov Model) paradigm, and algorithms to solve MAP (Maximum a Posteriori) problems. PSPNs have been originally described with this formalism [1], [2], [3]. For lack of space, we prefer to present only its specialization to Petri nets and refer the reader to the above mentioned references for a broader view on the subject.

3.3.1 Attributes of a place

As usual SPNs, a PSPN is obtained as a Petri net that handles not only tokens but also extra variables. Namely, three variables are associated to each place $p \in P$:

- Let M_n be the marking at time n, $M_n(p)$ taking values 0 or 1 represents the absence or presence of a token in place p.
- $\rho_n(p)$ is a random variable that encodes the routing choice of the resource lying in place p. $\rho_n(p)$ points either towards an input or an output transition, regardless of the value of $M_n(p)$, i.e. it may choose an output transition even if p is an empty place, or symmetrically.
- $\mu_n(p)$ taking values 1 or 0 encodes whether the place wishes to change state or not between times n and $n+1$. This variable *is not random*, but just unknown.

3.3.2 Evolution rule

A PSPN can be considered as a dynamic system with M_n as state vector and driven by the partially random input (ρ_n, μ_n). Specifically, at each instant n, all places select at

[8] This drawback remains even if tokens are asked to choose among enabled transitions.

random a routing. These selections are *independent* on both indexes n and p, i.e. in time and in space. Places also define their wishes μ_n. The marking at time $n+1$ is determined by the following rule

$$\begin{array}{ll} \forall t \in T, & t \text{ fires at time } n \;\Leftrightarrow \\ & t \text{ is enabled by } M_n \;\wedge\; [\forall p \in {}^{\bullet}t \cup t^{\bullet}, \mu_n(p) = 1, \rho_n(p) = t] \end{array} \tag{3}$$

In other words, an enabled transition fires iff it is elected by all its resources, and the latter wish to change state.

3.3.3 Trajectories

Lemma 2 *A standard Petri net and its partially stochastic version describe the same set of trajectories.*

Proof. Rule (3) is coherent with a legal Petri net behavior : only enabled transitions can fire, and two transitions fire simultaneously only if they are concurrent (would they have a common resource, they couldn't both be chosen by this resource). So every trajectory of a PSPN is legal for its underlying Petri net.

Conversely, every trajectory of a PN can be produced by its PSPN version. To prove it, one has to find a setting of μ and ρ variables that satisfy (3) for this trajectory. Several solutions exist :

1. If a place p is involved in the firing of t at time n, (3) imposes $\mu_n(p) = 1$ and $\rho_n(p) = t$.
2. Otherwise p is involved in no firing at time n. Then
   (a) either $\mu_n(p) = 0$ and the routing choice $\rho_n(p)$ can take any value,
   (b) or $\mu_n(p) = 1$ and $\rho_n(p)$ points to a non fired transition; this corresponds to a locking, since other resources of this transition didn't agree with place p.

Definition 2 *Let (M_0, s) be a length N Petri net trajectory. We call a* realization *of this trajectory any triple (M_0, ρ, μ), where ρ and μ are sequences $(\rho_n), (\mu_n), n = 1..N$ of routing and move variables that generate (M_0, s) in the PSPN framework. We denote it by $(M_0, \rho, \mu) \rightsquigarrow (M_0, s)$.*

3.3.4 Unfolding of time

In order to represent length N trajectories of the PSPN and compute their likelihood, it is very convenient to "unfold time", as illustrated by fig. 6. Specifically, this means that we duplicate each place $N+1$ times to encode its consecutive states in $M_0, \ldots, M_N$. Transitions are also duplicated and establish links between places at time n and places at time $n+1$: each transition reads variables in places of its pre- and post-sets on the left

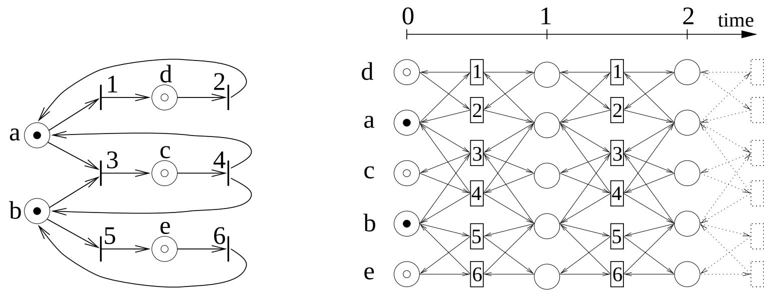

Figure 6: Unfolding of time to represent trajectories of a simple PSPN.

side, and fires or not according to rule (3), which determines the presence or absence of tokens on the right side. A trajectory of the Petri net can be represented on this unfolded system by keeping track of fired transitions only, as illustrated by figure 7.

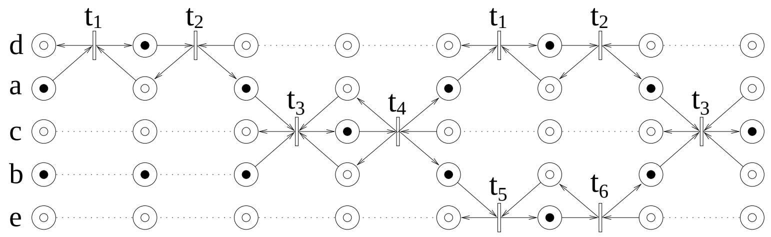

Figure 7: A trajectory, characterized by fired transitions only. Resources not participating to a firing remain unchanged (dotted lines).

In such a representation, PSPN realizations can be considered from a different standpoint [4]. We have a field of variables indexed by the pair $(p, n) \in P \times I\!N$. Some of them, the ρ's, are random and independent. And others are non random: the μ's and the markings M's. A length N trajectory amounts to placing N fired transitions on this field (as on fig. 7), which imposes *constraints* on variables, according to rule (3). If no constraint is violated, we have a valid PN trajectory, and the M's are uniquely determined (assuming places not involved in a firing keep their marking, which is guaranteed by an "extra" constraint represented by dotted lines). However, the (ρ, μ) part of the field can still take several values, whence the existence of several realizations of the same trajectory.

3.4 Likelihood of a path

Introducing the μ variables, we have only added some flexibility in the random routing framework: simultaneous firings as on fig. 5-a remain possible but are not obligatory. However, locking situations have not been erased and are actually more numerous since firing conditions are more restrictive. We show below that they are always less likely than a no locking situation with the same effect.

3.4.1 Definition

One can easily compute the likelihood of a realization (M_0, ρ, μ) : the random variables are the ρ's, more precisely $\rho_n(p)$, $p \in P$, $0 \leq n \leq N-1$. All these routing variables are independent whence :

$$\mathcal{L}(M_0, \rho, \mu) = \prod_{n=0}^{N-1} \prod_{p \in P} I\!P[\rho_n(p)]$$

We are interested in the sequel by the most likely realization (M_0, ρ, μ) given a CG of alarms. However, realizations are not directly observable : only the trajectories they generate can be distinguished by observations. We thus project the notion of likelihood[9] on trajectories by

$$\mathcal{L}(M_0, s) = \max_{(M_0, \rho, \mu) \rightsquigarrow (M_0, s)} \mathcal{L}(M_0, \rho, \mu) \tag{4}$$

3.4.2 Properties

Let (M_0, ρ, μ) be a realization of (M_0, s), such that place p is involved in a locking at time n, i.e. participates to no firing although $\mu_n(p) = 1$. Then (M_0, ρ, μ') obtained by switching $\mu_n(p)$ to 0 is another realization of (M_0, s), and has the same likelihood. In other words, one can restrict the max in (4) to realizations where μ variables vanish for all "static" places. Since $\mu = 1$ for other places - that participate to a firing -, μ is now uniquely determined and the max reduces to ρ variables. In summary, for a given trajectory, a realization leading to lockings is always as likely as another realization without lockings.

For this reduced set of realizations, if $\mu_n(p) = 1$, place p participates to the firing of a transition, say t, between times n and $n+1$, so $\rho_n(p) = t$. Conversely, if $\mu_n(p) = 0$, the place is steady whatever the value of $\rho_n(p)$. Hence

$$\mathcal{L}(M_0, s) = \prod_{n=0}^{N-1} \prod_{p\,:\,\mu_n(p)=1} I\!P[\rho_n(p)] \prod_{p\,:\,\mu_n(p)=0} I\!P^*(p) \tag{5}$$

$$\text{where} \qquad I\!P^*(p) \triangleq \max_t I\!P[\rho_n(p) = t]$$

Observe that the quantity $I\!P^*(p)$ doesn't depend on n : the selection probability of places is supposed to be constant in time.

The exact value of a trajectory likelihood is not important since we shall only consider maximum likelihood problems in the sequel. Therefore, we can re-normalize (5) by the constant $[\prod_{p \in P} I\!P^*(p)]^N$, which yields a notion of *generalized* likelihood (also denoted by

[9] One could wish to take a sum in (4), instead of a max. But this would raise a conceptual difficulty, because the space of the (ρ, μ)'s is not a random field since μ is non-random. Some authors have defined this object, however, and named it a "belief" instead of a "likelihood."

$\mathcal{L}$)

$$\mathcal{L}(M_0, s) = \prod_{n=0}^{N-1} \prod_{p\,:\,\mu_n(p)=1} I\!P[\rho_n(p)]/I\!P^*(p) \tag{6}$$

(6) reveals that only places involved in firings contribute to the (generalized) likelihood of a trajectory. Since places that wish to move are exactly places that participate to a firing, i.e. $\{p : \mu_n(p) = 1\} = \{p : \exists t \in s_{n+1}, p \in^\bullet t \cup t^\bullet\}$, by gathering places that trigger the same transition one gets

$$\mathcal{L}(M_0, s_1, \ldots, s_N) = \prod_{n=0}^{N-1} \prod_{t\in s_{n+1}} \underbrace{\left[\prod_{p\in^\bullet t\cup t^\bullet} I\!P[\rho_n(p) = t]/I\!P^*(p)\right]}_{\mathcal{L}(t)} \tag{7}$$

The bracketed term $\mathcal{L}(t)$ depends only on transition t, but not on the time index n; we shall call it a "transition likelihood." Observe that $\mathcal{L}(t)$ depends only on choices made by the resources of t, and can be computed locally. Turning back to fig. 7, expression (7) means that the likelihood of a trajectory is obtained as the product of transition likelihoods, taken over all fired transitions. All properties of PSPNs derive from these remarks, as we show below.

3.4.3 Interpretation

Lemma 3 *All trajectories that are compatible with a given causality graph have the same likelihood.*

This is a direct consequence of the product form (7). Therefore one can talk of the likelihood of a CG, which leads to several practical interpretations:

- The order in which two concurrent transitions fire is not randomized.
- Waiting has no influence on the likelihood. In particular, empty salvos are erased by the likelihood computation. Together with the previous point, this induces that all firing sequences represented on fig. 8 have the same likelihood.
- Since only causality graphs of firings are distinguished by the PSPN framework, time behaves as if it was partially ordered, as opposed to standard SPNs that assign different likelihoods to different sequences.
- Again, in contrast with standard SPNs, the "probability" that t fires from M_n is not any more a function of the whole marking, but a fixed quantity that only depends on the resources of t.

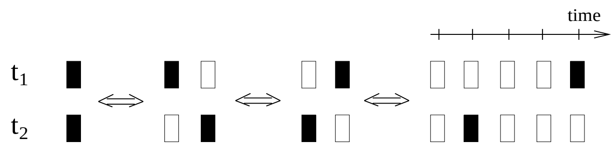

Figure 8: Firing sequences of two concurrent transitions. Black rectangle = firing ; white rectangle = waiting. All sequences have the same likelihood.

At this point, we have proved that only fired transitions should be kept both to describe trajectories and to compute their likelihood. Lemma 3 also reveals that the PSPN framework cannot distinguish sequences of transitions obeying the same causality graph. Therefore, the right notion of trajectory is rather a CG than a sequence.

4 Trajectories as causality graphs

This section provides a framework for handling PSPN trajectories. We first define the building element : *tiles*, associated to transitions. Then we show how to connect tiles in a causality graph to form a PSPN trajectory, that we shall call a *puzzle*. The formalism defined here is used in the next section to implement the diagnosis algorithms.

4.1 Tiles

4.1.1 Sub-markings

Let $\mathcal{M}$ be the set of markings, and $Q \subset P$ a set of places. We define the equivalence relation $\sim_Q$ on $\mathcal{M}$ by

$$\forall M, M' \in \mathcal{M}, \quad M \sim_Q M' \Leftrightarrow \forall p \in Q, M(p) = M'(p)$$

Equivalence classes of $\sim_Q$ are composed of markings that coincide on the places of Q. They will be denoted by m and represented by their values on Q :

$$m(p) = \begin{cases} M(p), \ p \in Q, \ \forall M \in m \\ \varepsilon \text{ otherwise} \end{cases}$$

The symbol ε for places out of Q stands for "free", since markings in m can either have 0 or 1 token in these places. For this reason, m will be called a *sub-marking*. The natural inclusion relation on sub-markings takes the following meaning here : let m_1 and m_2 be two equivalence classes for $\sim_{Q_1}$ and $\sim_{Q_2}$, then

$$m_1 \subset m_2 \quad \Leftrightarrow \quad Q_1 \supset Q_2, \text{ and } \forall p \in Q_2, \ m_1(p) = m_2(p)$$

which means that m_1 specifies more places that m_2. We shall also make use of a symmetric difference operator Δ on sub-markings, that specifies places where sub-markings impose different values:

$$m_1 \Delta m_2 = \{p \in Q_1 \cap Q_2 \;:\; m_1(p) \neq m_2(p)\}$$

Sub-markings m_1 and m_2 such that $m_1 \Delta m_2 = \emptyset$ will be said to be *compatible*, which means that $m_1 \cap m_2 \neq \emptyset$.

Example. With $P = \{p_1, \ldots, p_5\}$ let $m_1 = [\varepsilon, 1, 0, 1, \varepsilon]$ and $m_2 = [1, 1, 1, 0, \varepsilon]$, assuming a vector notation. Then $m_1 \Delta m_2 = \{p_3, p_4\}$.

4.1.2 Elementary tiles

Firing transition t only requires a partial knowledge of the current marking, and the effect of this firing is also limited. We thus define sub-markings m_t^- and m_t^+ as the minimal "past" and "future" of transition t:

$$m_t^-(p) = \begin{cases} 1 \text{ if } p \in {}^\bullet t & \text{(full preset)} \\ 0 \text{ if } p \in t^\bullet \setminus {}^\bullet t & \text{(empty postset)} \\ \varepsilon \text{ otherwise} \end{cases} \tag{8}$$

$$m_t^+(p) = \begin{cases} 0 \text{ if } p \in {}^\bullet t \setminus t^\bullet & \text{(empty preset)} \\ 1 \text{ if } p \in t^\bullet & \text{(full postset)} \\ \varepsilon \text{ otherwise} \end{cases} \tag{9}$$

Definition 3 *For any $t \in T$, let m_t^- and m_t^+ be defined by (8) and (9), the triple (m_t^-, t, m_t^+) is the* elementary tile *associated to t. It satisfies*

$$M \in m_t^- \Leftrightarrow M[t\rangle \qquad \text{and} \qquad M' \in m_t^+ \Leftrightarrow [t\rangle M'$$

where $M[t\rangle$ means that t is enabled by M, and $[t\rangle M'$ that M' is a possible consequence of t.

In the sequel, a tile will be equivalently described by its *applicability condition* m_t^- and its *action* $m_t^- \Delta m_t^+$, i.e. the set of places modified by t. Fig. 9 lists the tiles corresponding to the Petri net of fig. 6.

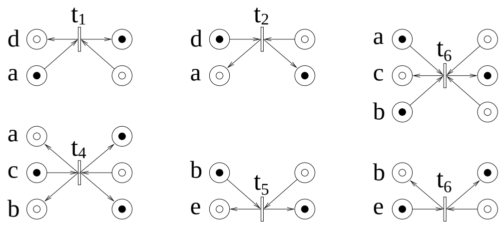

Figure 9: Tiles of the Petri net used on figs. 6 and 7. Places marked by ϵ are not represented for simplicity (they are on the next figure).

4.2 Puzzle

Our objective is now to view histories of the Petri net as causality graphs of transitions rather than sequences. This section provides a tool for constructing CGs by connection of tiles.

4.2.1 Connection of tiles

We consider a sequence of transitions $s = (t_1, \ldots, t_n)$ and wish to extend the notion of tile to s. This object, denoted by $(m_s^-, t_1 t_2 \ldots t_n, m_s^+)$ or (m_s^-, s, m_s^+) for short, is obtained recursively by *connecting* the tiles $t_1, \ldots, t_n$, in this order. Thus (m_s^-, s, m_s^+) will be called a *puzzle*.

In the tile t, m_t^- and m_t^+ are equivalence classes of $\sim_{Q_t}$ where $Q_t = {}^\bullet t \cup t^\bullet$ is the set of places possibly altered by t. By extension, we also define $Q_s = \cup_{i=1}^n Q_{t_i}$, the set of places involved in the sequence s. So m_s^- and m_s^+ are equivalence classes of $\sim_{Q_s}$, as stated below :

Definition 4 *(recursive definition) Let s be a sequence of transitions, the* puzzle (m_s^-, s, m_s^+) *is the connection of its tiles. The tile (m_t^-, t, m_t^+) is* connectable *to the puzzle (m_s^-, s, m_s^+) iff m_t^- is compatible with m_s^+. The connection yields the puzzle (m_{st}^-, st, m_{st}^+) defined by*

$$m_{st}^-(p) = \begin{cases} m_s^-(p) & \textit{if } p \in Q_s \\ m_t^-(p) & \textit{if } p \in Q_t, p \notin Q_s \\ \varepsilon & \textit{otherwise} \end{cases} \tag{10}$$

$$m_{st}^+(p) = \begin{cases} m_t^+(p) & \textit{if } p \in Q_t \\ m_s^+(p) & \textit{if } p \in Q_s, p \notin Q_t \\ \varepsilon & \textit{otherwise} \end{cases} \tag{11}$$

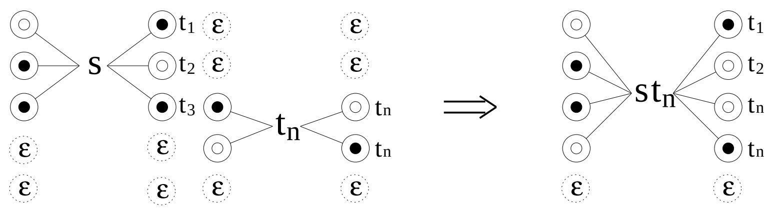

Figure 10: Connection of an extra tile to a puzzle. The name of the transition that produced each resource (the "father" transition) is stored to recover the causality graph associated to the sequence. On the example, t_n becomes a direct consequence of t_3.

Obviously, t is connectable[10] to s if there exist markings that allow to fire first s and then

[10] With a slight abuse of terms, t will both refer to a transition or to its tile, and s to a sequence or its puzzle.

t. Such markings all belong to the same equivalence class, as proved by the following lemma.

Lemma 4 *Let s be a sequence of transitions, $\forall M, M' \in \mathcal{M}$ we have*

$$\begin{array}{rcl} M[s\rangle & \Leftrightarrow & M \in m_s^- \\ [s\rangle M' & \Leftrightarrow & M' \in m_s^+ \end{array}$$

Proof. By recursion. 1/ The result is true for a single tile. 2/ Assume the result is true for sequence s. Let $M[s\rangle M'[t\rangle M''$. Then $M \in m_s^-$ and $M' \in m_t^-$. Places outside Q_s are not modified by s since $M[s\rangle M' \Rightarrow M \Delta M' \subset Q_s$, so $M'(p) = M(p)$ for any place $p \in Q_t, p \notin Q_s$. This proves $M[st\rangle \Rightarrow M \in m_{st}^-$. The sufficient condition, and the symmetric statement for m_{st}^+ follow in the same way.

Remark (i). The compatibility condition and (10), (11) hold for the connection of a tile to a sub-marking m instead of a puzzle: one just has to take $m = m_s^+$. This allows to use the notation $m[t\rangle m'$ for sub-markings (with $m' = m_{st}^+$).

Remark (ii). One easily checks, by recursion, that the compatibility condition and the connection formulae given in definition 4 extend to the connection of a sequence s' to s.

4.2.2 Causality graphs

The causality semantics of Petri nets appears clearly in the connection procedure:

- t_1 and t_2 are *concurrent* transitions iff $Q_{t_1} \cap Q_{t_2} = \emptyset$, which means they use different resources. This has several consequences. First t_2 is connectable to t_1, and conversely. Secondly, $m_{t_1 t_2}^- = m_{t_2 t_1}^-$ and $m_{t_1 t_2}^+ = m_{t_2 t_1}^+$: whatever the order in which they fire, one gets the same result. Notice however that this commutativity property alone is not sufficient to prove concurrency since it doesn't imply $Q_{t_1} \cap Q_{t_2} = \emptyset$.

- If $Q_{t_1} \cap Q_{t_2} \neq \emptyset$ and t_2 is compatible with t_1, t_2 becomes a *direct consequence* of t_1 in the sequence $t_1 t_2$ since it consumes resources set by t_1 (see section 2.2.2), whence the arrow $t_1 \to t_2$.

To compute the causality graph associated to a given sequence $s = (t_1, \ldots, t_n)$, one must keep track of which transition produced each resource $m_s^+(p)$, for $p \in Q_s$. This can be done by the "father" function, illustrated on fig. 10: $f_s(p) \in \{t_1, \ldots, t_n\}$ for $p \in Q_s$, and $f_s(p) = \emptyset$ otherwise. When t_{n+1} is connected to s, it appears as a direct consequence of each transition $f_s(p)$ for $p \in Q_{t_{n+1}}$. t_{n+1} changes tokens in $Q_{t_{n+1}}$, so the "father" function must be updated according to:

$$f_{st_{n+1}}(p) = \begin{cases} t_{n+1} & \text{if } p \in Q_{t_{n+1}} \\ f_s(p) & \text{otherwise} \end{cases} \tag{12}$$

Remark (iii). Once again, the update formula (12) obviously holds for the connection of a sequence s' to s.

Lemma 5 *Let $s = (t_1, \ldots, t_n)$ be a sequence of transitions and σ its associated causality graph on $\{t_1, \ldots, t_n\}$. Let s' be another sequence in $Lin(\sigma)$. Then $m_s^- = m_{s'}^-$, $m_s^+ = m_{s'}^+$ and $f_s = f_{s'}$.*

Proof. By making use of lemma 1, we only have to show that m_s^-, m_s^+ and f_s are preserved by permutation of consecutive concurrent transitions, say t_k and t_{k+1}. Using remarks (ii) and (iii) above, we know that the puzzle s and its father function f_s can be obtained by the connection of three puzzles corresponding subsequences $(t_1, \ldots, t_{k-1})$, (t_k, t_{k+1}) and $(t_{k+2}, \ldots, t_n)$. But m^-, m_+ and f are identical for both puzzles (t_k, t_{k+1}) and (t_{k+1}, t_k), whence the result.

Lemma 5 extends the use of m^-, m^+ and f to causality graphs σ, and allow to view them as puzzles. So it becomes legal to check whether two CGs σ and σ' are compatible. Using f_σ, it is very easy to connect directly a transition t to the CG σ and get the resulting CG $\sigma' = \sigma t$. This capability will be central in the next section. The connection of two general CGs σ and σ' raises a difficulty however, since we need to define the causality graph $\sigma\sigma'$. This can be done in the following way: Let $s \in Lin(\sigma)$ and $s' \in Lin(\sigma')$, the CG σ'' associated to the sequence $s\,s'$ doesn't depend on the choices of s and s'. So we define $\sigma'' = \sigma\sigma'$. The direct construction of σ'' from σ and σ' would require some extra material that we don't develop here.

Theorem 1 *Gathering all results above, a PSPN trajectory can now be defined as a pair composed of an initial marking M_0 and a causality graph of transitions σ, compatible with M_0, i.e. such that $M_0 \Delta m_\sigma^- = \emptyset$. The likelihood of this trajectory $\mathcal{L}(\sigma)$ is the product $\prod \mathcal{L}(t)$ over tiles used to build σ.*

5 Diagnosis algorithm: the Viterbi puzzle

5.1 Four stages

The diagnosis problem consists in providing the most likely Petri net trajectory given an observed set of alarms. In order to stress the difference between SPNs based on Markov dynamics and PSPNs, we assume that the initial marking M_0 is unknown. In particular, no probability law is given on M_0, which preserves the likelihood expression of a PSPN trajectory. This trajectory must be understood as a causality graph σ of transitions, enabled by some initial marking. However, the causal observation assumption (H1) provides alarms as a sequence $(a_1, \ldots, a_N)$, so we shall actually construct trajectories as sequences, and then view them as causality graphs. To simplify the presentation of the diagnosis algorithm, we define four subproblems:

1. Causal dependence relations on alarms are not observed. This is equivalent to assuming that each alarm a_n is related to every a_k, $k < n$, by a dashed arrow: a causal dependence may exist or not. So we don't have to check whether a candidate sequence of transitions does match causality relations on alarms. This first stage

also assumes that no alarm is lost, i.e. all fired transitions are related to an observed alarm.

2. Causal dependence relations on alarms are observed, which restricts the set of possible trajectories. Now the CG associated to a candidate sequence of transitions must match the observed CG of alarms (up to dashed arrows). But alarms can't be lost: there is no silent transition.

3. We turn back to the first framework, where causality relations on alarms are not observed, but now allow alarm losses, i.e. assume that some transitions may fire silently.

4. General case, with causal relations on alarms and possible silent transitions.

5.2 Problem 1

5.2.1 The basic Viterbi puzzle

This problem is solved by a standard dynamic programming procedure. The recursion index n is the number of alarms that have been taken into account in the sequence $(a_1, \dots, a_N)$, and the right notion of system state is a sub-marking m_n of the Petri net. Let $\mathcal{M}_n$ denote the set of sub-markings at time n that can be reached through the observation of $(a_1, \dots, a_n)$:

$$m_n \in \mathcal{M}_n \quad \Leftrightarrow \quad \exists (t_1, \dots, t_n) \; : \; m^+_{t_1 \cdots t_n} = m_n, \;\; a_k \in A_{t_k}, \; 1 \leq k \leq n$$

where A_{t_k} represents the set of alarms that can be emitted by transition t_k. In terms of global markings, $m^+_{t_1 \cdots t_n} = m_n$, means that for every $M_n \in m_n$, there exists an initial marking M_0 such that $M_0[t_1 \cdots t_n\rangle M_n$.

The objective is to compute the best sequence $(t_1, \dots, t_n)$ leading to every reachable m_n, in the sense that it maximizes the joint likelihood of $(t_1, a_1, \dots, t_n, a_n)$. This best likelihood is defined as

$$\mathcal{L}^*(m_n) = \max_{s=(t_1,\dots,t_n)\,:\, m^+_s = m_n} \prod_{k=1}^{n} \mathcal{L}(t_k) \mathbb{P}(a_k | t_k)$$

where we assume $\mathbb{P}(a_k|t_k) = 0$ whenever $a_k \notin A_{t_k}$. The solution to problem 1 is thus the best sequence leading to the best final sub-marking in $\mathcal{M}_N$. We obviously have

$$\mathcal{L}^*(m_{n+1}) = \max_{\substack{m_n \in \mathcal{M}_n, \; t \in T \; : \\ m_n[t\rangle m_{n+1}}} \mathcal{L}^*(m_n) \mathcal{L}(t) \mathbb{P}(a_{n+1}|t) \tag{13}$$

The transition reaching the max is stored as $t^*_{n+1}(m_{n+1})$, and the best previous sub-marking as $m^*_n(m_{n+1})$. The best final sub-marking

$$m^*_N = \arg \max_{m_N \in \mathcal{M}_n} \mathcal{L}^*(m_N)$$

yields the most likely sequence $(t_1^*, \ldots, t_N^*)$ thanks to a backtrack procedure:

$$t_{n+1}^* = t_{n+1}^*(m_{n+1}^*) \qquad\qquad m_n^* = m_n^*(m_{n+1}^*)$$

5.2.2 Comments

- The algorithm above builds trajectories by connecting tiles that are compatible with the observed alarms. We can extend the definition of a tile to a pair transition + alarm: (m_t^-, t, a, m_t^+), $t \in T, a \in A_t$, with likelihood $\mathcal{L}(t,a) = \mathcal{L}(t)\mathbb{P}(a|t)$. At time n, the game becomes the connection of an a_n-tile to sub-markings in $\mathcal{M}_{n-1}$, and the selection of the best last connection among those that produce the same m_n.

- The dynamic programming procedure may yield several optimal sequences of transitions, because a max can have several arguments. In this case, $t_{n+1}^*(m_{n+1})$ corresponds to a *set* of optimal predecessors, and each of them starts a new optimal trajectory in the backtrack procedure.

- The PSPN framework only distinguishes causality graphs, so optimal sequences have to be expressed under this form. One can use the procedure described section 4.2.2 for this purpose, with a father function pointing to a tile (t, a) rather than a tile t.

- One could be interested in sub-products of these optimal causality graphs. For example, only trajectories in terms of transitions may be of interest. The latter can be obtained by erasing alarms in the previous CGs, or by direct construction. Several optimal sequences can thus yield the same CG of transitions. Consider for example the observation $\alpha \dashrightarrow \beta$, and assume transitions t and t' are concurrent and can both produce α or β, with equal likelihoods. Then trajectories $((t,\alpha),(t',\beta))$ and $((t',\alpha),(t,\beta))$ are both possible and equally likely. But they correspond to the same CG $t\|t'$ of transitions. Conversely, one could be interested in the resulting causality graph of alarms, since these relations are not observed. Once again, the example gives a unique solution $\alpha\|\beta$.

5.3 Problem 2

We now add an extra constraint on possible solutions: they have to satisfy an observed causality graph on alarms. It is quite easy to check *a posteriori* whether a given sequence $(t_1, a_1, \ldots, t_n, a_n)$ of tiles (t, a) is valid or not: one only has to check if the resulting CG on alarms matches the observed one. But this can't be done on the solutions to problem 1, since none of them may be compatible. So we have to construct compatible sequences only.

The trick is to mix two recursions: (13), the dynamic programming procedure, and (12), that provides the CG on alarms. The right notion of system state is now a pair (m_n, f_n) where f_n is a father function that stores which alarm among $\{a_1, \ldots, a_n\}$ was consumed to produce each resource in m_n. Since m_n is a sub-marking, we have $f_n(p) = \emptyset$ for places p such that $m_n(p) = \epsilon$. The connection of a tile (t, a_{n+1}) to (m_n, f_n) is legal iff

1. t is compatible with m_n, and
2. the set of alarms $\{f_n(p), p \in Q_t\}$ that produced the resources of t coincide with the observed causes of a_{n+1}.

The connection yields new pairs (m_{n+1}, f_{n+1}), and the best last tile and best predecessor (m_n, f_n) are kept among connections that reach the same state. The rest of the algorithm remains unchanged.

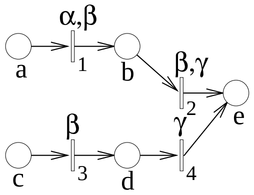

Figure 11: A toy Petri net. Greek letters represent possible alarms for a transition.

Example. (Probabilities are not made explicit for simplicity). We consider the network of fig. 11, and assume that the causality graph of alarms $\alpha \dashrightarrow \beta \longrightarrow \gamma$ has been observed. The three steps of the algorithm are illustrated by fig. 12, where the complete trajectories are depicted, for clarity, although the algorithm only handles their extremity.

Step 1 (left). The first alarm α can only be produced by t_1, whence a unique possibility for m_1. f_1 reflects that places a and b have been changed by the acceptance of α.

Step 2 (center). From m_1, β can be produced by either t_3 or t_2. The first one makes β concurrent with α, while the second assumes β is a consequence of α. Since β is related to α by a dashed arrow, both possibilities are accepted, whence pairs (m_2, f_2) and (m_2', f_2').

Step 3 (right). γ can't be produced from m_2', so this trajectory is discarded. From m_2, one can obtain γ either by t_2 or t_4. In the first case, γ would become a direct consequence of α, which contradicts the observation: the arrow $\alpha \longrightarrow \gamma$ is not present. The second case is correct: γ becomes a direct consequence of β, which is observed. So only this possibility is accepted, which finally reveals that the uncertain relation between α and β is actually a concurrence.

Lemma 6 *The result of the diagnosis algorithm, in terms of causality graph, is independent of which ordering is chosen on alarms, provided it satisfies the observed causality graph (including dashed arrows).*

Proof. We rely on lemma 1. Assume a_k and a_{k+1} are not related by an arrow, neither solid nor dashed, in the sequence a of alarms. Therefore a_k and a_{k+1} are necessarily concurrent (a_{k+1} can't be causally related to a_k). We build the sequence a' by inverting

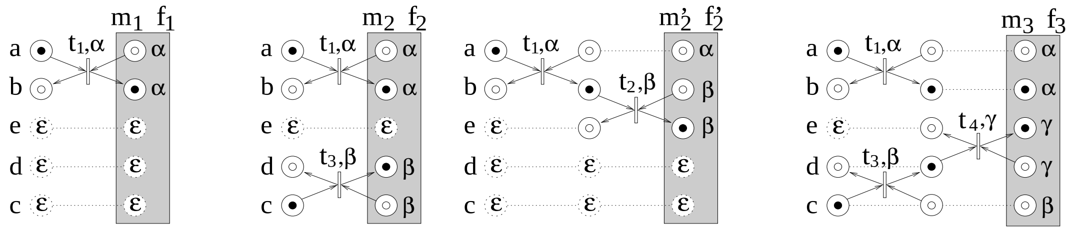

Figure 12: The three steps of the diagnosis algorithm.

these two alarms and keeping all dependence relations. Every trajectory $s = (t_1, \ldots, t_n)$ compatible with sequence a of alarms (and its dependence relations) satisfies $t_k \| t_{k+1}$. So s' obtained by inverting t_k and t_{k+1} is executable, and compatible with the sequence a' (and its dependence relations). Since the permutation doesn't change the likelihood, an optimal sequence for a transforms into an optimal one for a'. Moving to causality graphs, we obtain identical trajectories.

Example (continued). We extend the PN of fig. 11 into that of fig. 13, and take as observation the sequence depicted on the right of the figure. By lemma 6, we get the

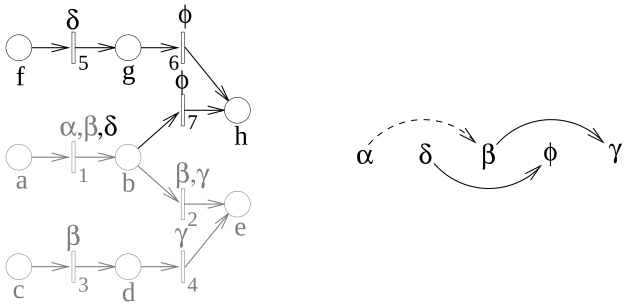

Figure 13: Extension of the PN of fig. 11. The observed CG of alarms is depicted on the right.

same diagnosis with the permuted sequence $\alpha \dashrightarrow \beta \rightarrow \gamma \quad \delta \rightarrow \phi$. The same transitions are concerned by alarms α, β and γ. Since tile likelihoods are computed locally, they are not influenced by the status of extra places, thus the result of the diagnosis performed above can be taken as a starting point here[11]. The same reasoning holds for the sub-sequence $\delta \rightarrow \phi$ and transitions t_1, t_5, t_6, t_7. This part of the diagnosis could also be done independently on a reduced net, which yields two possibilities $\bar{m}_2$ and $\bar{m}_2'$ (fig. 14).

This suggests another way of performing the global diagnosis: by merging two sub-trajectories corresponding to the concurrent sub-sequences. The observed concurrence means that we have to connect puzzles involving different sets of places. The only possibility is the pair m_3 and $\bar{m}_2$ which gives the trajectory $(t_1 \| t_3 \rightarrow t_4) \| (t_5 \rightarrow t_6)$. This

[11] Sub-markings and father functions have to be extended formally, with ϵ and $\emptyset$ for new places.

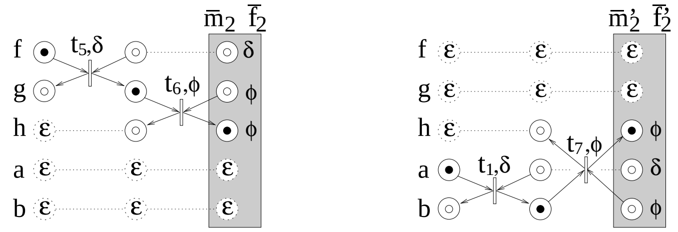

Figure 14: Partial diagnosis for the subsequence $\delta \to \phi$.

reveals part of the ideas that allow the distribution of the algorithm.

We can now realize the importance of the causal observation hypothesis (H1). Lemma 6 shows that any sequence is valid provided the arrows are all oriented from left to right. If it is not the case, i.e. if (H1) is violated, one could start building a trajectory and suddenly have to connect an extra tile "somewhere in the past", instead of at the extremity. This is not permitted by the present formalism. However it seems such a capability could be expected from a distributed diagnosis algorithm.

5.4 Problem 3

Forgetting causality relations for a while, we turn back to problem 1 and introduce another kind of difficulty: we assume that some alarms can get lost in the faulty network, or equivalently that transitions can fire silently.

5.4.1 A recursion based on macro-tiles

The recursion index remains n, the number of alarms taken into account in the sequence. But now an unknown number k of hidden transitions $h_1, \ldots, h_k$ may be fired before a t corresponding to a_{n+1} be hit. Therefore (13) becomes

$$\mathcal{L}^*(m_{n+1}) \;=\; \max_{m_n, t} \; \mathcal{L}^*(m_n) \max_{\substack{k,\; h_1, \ldots, h_k \;: \\ m_n[h_1 \ldots h_k t\rangle m_{n+1}}} \mathcal{L}(t, a_{n+1}) \prod_{i=1}^{k} \mathcal{L}(h_i, \lambda) \tag{14}$$

where one obviously has to select the most likely hidden path between m_n and m_{n+1}. The new recursion (14) does the same job as (13) but relies on "macro-tiles" $h_1 \ldots h_k t$ where only one transition is visible (not silent).

Definition 5 *A* t-macro-tile *is the puzzle of a causality graph π_t, such that π_t is made of silent transitions plus the (visible) tile t, π_t satisfies $m^-_{\pi_t} = m^-, m^+_{\pi_t} = m^+$, and it achieves the best likelihood given the pair (m^-, m^+).*

Naturally, denoting by $(h_1, \ldots, h_k, t)$ an element of $Lin(\pi_t)$, we have $\mathcal{L}(\pi_t) = \mathcal{L}(t) \prod_{i=1}^{k} \mathcal{L}(h_i, \lambda)$. As before, we can also consider macro-tiles (π_t, a), with $\mathcal{L}(\pi_t, a) = \mathcal{L}(\pi_t) I\!P(a|t)$. Macro-tiles take the place of the second max in (14), which now behaves exactly as (13) :

$$\mathcal{L}^*(m_{n+1}) \;=\; \max_{m_n, \pi_t \;:\; m_n[\pi_t\rangle m_{n+1}} \mathcal{L}^*(m_n)\mathcal{L}(\pi_t, a_{n+1}) \tag{15}$$

So we are back to the framework of problem 1. The possibility that several macro-tiles π_t have the same extremities (m^-, m^+) and (by definition) the same likelihood is not excluded; they are said to be *equivalent*. All must be kept since they define different optimal trajectories in the backtrack procedure. However, they can be handled as the same formal object by the diagnosis algorithm.

5.4.2 Selection of useful macro-tiles

The definition of macro-tiles is not constructive, and doesn't specify which pairs (m^-, m^+) should be considered. The following two lemmas clarify this point and eliminate useless macro-tiles.

Lemma 7 *(structural reduction of the set of macro-tiles)*
The macro-tile π_t is useless to the diagnosis algorithm if it doesn't satisfy the following properties :

1. *there is no silent maximum in π_t, or equivalently the (visible) tile t is the unique maximum of π_t,*

2. *t appears only once in π_t,*

3. *π_t has no loop, i.e. there is no $s = (h_1, \ldots, h_k, t) \in Lin(\pi_t)$ such that a subsequence $s' = (h_i, \ldots, h_j)$ has an empty action, $1 \leq i \leq j \leq k$.*

Proof.
3 • Assume π_t has a loop, and take s and s' as above. Since the action of s' is empty, the subsequence $(h_{j+1}, \ldots, h_k, t)$ is compatible with $(h_1, \ldots, h_{i-1})$. Their connection $s'' = (h_1, \ldots, h_{i-1}, h_{j+1}, \ldots, h_k, t)$ has the same action as π_t and satisfies $m^-_{\pi_t} \subset m^-_{s''}$. But the puzzle s'' contains less tiles than π_t, so it has a greater likelihood since a tile likelihood is less or equal to one. So, finally, π_t can always be replaced, in any trajectory, by a better macro-tile corresponding to $(m^-_{s''}, m^+_{s''})$, and thus π_t is useless.

It could happen however that every h_l of s' is a "free" tile, i.e. satisfies $\mathcal{L}(h_l, \lambda) = 1$. Free loops like s' must be erased anyway since they are unobservable and could be repeated infinitely many times.
1 • This involves the recursive structure of the diagnosis algorithm. Let (u, λ) be a silent maximal tile in π_t. There exists $s \in Lin(\pi_t)$ finishing with u, for example $s = (h_1, \ldots, h_{k-1}, t, u)$. Let us consider two consecutive steps in the recursion : $m_n[s\rangle m_{n+1}[s'\rangle m_{n+2}$. We also have $m_n[h_1 \ldots h_{k-1} t\rangle m'_{n+1}[us'\rangle m_{n+2}$.

- The trajectory ss' reaching m_{n+2} from m_n can be obtained through the intermediate sub-marking m'_{n+1} instead of m_{n+1}; in other words, the silent maximum u can be transferred to the next macro-tile. So considering macro-tiles with t as unique maximal tile doesn't reduce the set of trajectories explored by the algorithm.

- If $n+1 = N$, the last index, then m_{n+1} is less likely than m'_{n+1}, whence the same conclusion: π_t is useless.

2 • Assume t appears twice in π_t, once as the unique (visible) maximum, and once as a silent transition that we denote $\bar{t}$. Inverting the roles of t and $\bar{t}$, we get another equivalent macro-tile π'_t. π_t can always be replaced by π'_t and still describe the same trajectory, up to the position of the silent t, which is unobservable anyway. The previous point reveals that π'_t is useless to the diagnosis algorithm, so π_t can also be discarded.

Definition 6 *Let π_t and π'_t be two macro-tiles with t as unique visible transition,*

$$\pi_t \succ \pi'_t \;\Leftrightarrow\; \left\{ \begin{array}{l} \pi'_t \textit{ and } \pi_t \textit{ have the same action} \\ m^-_{\pi'_t} \subset m^-_{\pi_t} \\ \mathcal{L}(\pi_t) > \mathcal{L}(\pi'_t) \end{array} \right.$$

$\pi_t \succsim \pi'_t$ stands for $\succ$ with equality on likelihoods.

Lemma 8 *(likelihood reduction of the set of macro-tiles)*
Macro-tiles that are not maximal for the relation $\succ$ are useless to the diagnosis algorithm.

Proof. This lemma extends the optimality required by the definition of macro-tiles: when $\pi_t \succ \pi'_t$, both macro-tiles change the same tokens (thus allow the same future), and π_t is connectable whenever π'_t is. Since the likelihood is in favor of π_t, π'_t will always be rejected.

5.4.3 Construction of (useful) macro-tiles

Let π_t be a useful t-macro-tile corresponding to the sequence $(h_1, \ldots, h_k, t)$. One easily checks that $(h_2, \ldots, h_k, t)$ induces another useful t-macro-tile. This suggests that t-macro-tiles can be obtained recursively, by left connection of silent transitions, checking at each step that lemmas 7 and 8 are satisfied. Let $\Pi_t(k)$ represent the set of candidate t-macro-tiles with k silent transitions. Construction rules are given below:

1. $\Pi_t(0)$ is reduced to the visible tile (m^-_t, t, m^+_t).

2. Possible elements of $\Pi_t(k)$ are built by connecting a silent transition $h \neq t$ to an element τ_{k-1} in $\Pi_t(k-1)$: $\tau_k = h\tau_{k-1}$ (this assumes compatibility and $\mathcal{L}(h, \lambda) \neq 0$). h must be the cause of one transition in τ_{k-1} in order to satisfy 1 of lemma 7; this is guaranteed by $Q_h \cap Q_{\tau_{k-1}} \neq \emptyset$.

3. Each τ_k has to be compared to every $\tau_i, 0 \leq i \leq k$, for relations $\prec$ and $\succ$. If such a relation is detected, the lower element must be rejected, according to lemma 8.

4. If (h, λ) is a free tile, τ_k must also be compared to every $\tau_i, 0 \leq i \leq k-1$ for relation $\precsim$. If $\tau_k \precsim \tau_i$ is detected, h may have closed a free loop in τ_i, which has to be checked. A τ_i containing a free loop must be rejected.

5. Stop if all elements of $\Pi_t(k)$ have been rejected by rules 2 and 3, in which case the useful t-macro-tiles are the elements of $\cup_{i=0}^{k-1} \Pi_t(i)$. Otherwise proceed to $\Pi_t(k+1)$.

5.5 Problem 4

Gathering problems 2 and 3, we now have to track causality relations despite the presence of silent transitions. As for problem 2, the right notion of system state is a pair (m_n, f_n) rather than m_n alone. This requires to understand the effect of connecting a silent transition, and consequently to adapt the definition of macro-tiles in order to take their action on f_n into account.

5.5.1 Effect of a silent tile

Let $(t_1, a_1, \ldots, t_n, a_n)$ be a sequence of transitions and their associated alarms, some of which are a λ. The resulting causality graph of alarms is obtained in the following way. One first computes the CG of tiles (t_k, a_k), which is identical to the CG of the a_k's alone. Then the latter is reduced by "erasing" every a_k such that $a_k = \lambda$, since we have assumed in section 2.1 that causality relations involving masked alarms are also lost. It is important to notice that one would have obtained the same result by directly erasing a (t_k, λ) tile in the recursive construction of the causality graph. To erase means here to replace pointers to λ-tiles by $\emptyset$ in the father function f, as illustrated by fig. 15.

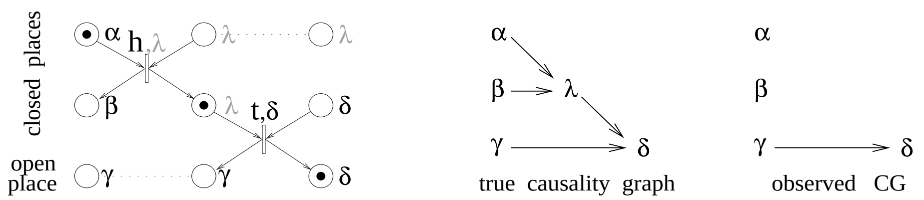

Figure 15: Construction of the CG of alarms (Greek letters). When connecting the silent transition h, pointers to previous alarms are erased for places of Q_h. This "closes some legs" for transition t, i.e. blocks the observation of causality relations.

Fig. 15 reveals the method for updating the father function f_n when the tile t, matching alarm a_{n+1}, is connected by means of the hidden path $s = (h_1, \ldots, h_k, t)$:

$$f_{n+1}(p) = \begin{cases} a_{n+1} & \text{if } p \in Q_t \\ \emptyset & \text{if } p \in Q_s \setminus Q_t \\ f_n(p) & \text{otherwise} \end{cases} \tag{16}$$

Now, using f_n to decide whether the sequence s is compatible with causality relations observed on alarm a_{n+1} is a bit more delicate. We observe on fig. 15 that t induces a visible causal dependence only through places that have not been "captured" by hidden transitions. We thus define the set of *open places* of s as $O_s = Q_t \setminus (\cup_{i=1}^{k} Q_{h_i})$. Therefore, s is connectable to (m_n, f_n) iff

1. s is compatible with m_n, and
2. the set of alarms in the open places of s, $f_n(O_s)$, coincides with the observed causes of a_{n+1}.

To extend the connection procedure to a causality graph π_t, with t as single visible transition, we only need to define the set of open places of π_t. Let the sequence $s = (h_1, \ldots, h_{j-1}, t, h_{j+1}, \ldots, h_k)$ be a linear extension of π_t, i.e. $s \in Lin(\pi_t)$, we set $O_{\pi_t} = Q_t \setminus (\cup_{i=1}^{j-1} Q_{h_i})$. The result is independent of the sequence s (use lemma 1).

5.5.2 New definition of macro-tiles

Macro-tiles were defined as the best CGs for a given applicability condition and a given effect on the state vector m. Applied to a pair (m, f) as notion of state, this yields

Definition 7 *A* t-macro-tile *is the puzzle of a causality graph π_t, such that π_t is made of silent transitions plus the (visible) tile t, π_t satisfies $m^-_{\pi_t} = m^-, m^+_{\pi_t} = m^+, O_{\pi_t} = 0$, and it achieves the best likelihood given (m^-, m^+) and the set of open places O.*

As for problem 2, not all macro-tiles are useful. However, the selection is less drastic. First there is no counterpart of lemma 8: π_t would always replace π'_t if it produced the same change on the state vector. This requires having the same set of open places in order to capture the same causality dependences. This requires also $Q_{\pi_t} = Q_{\pi'_t}$, otherwise f wouldn't be updated on identical sets. Together with $m^-_{\pi'_t} \subset m^-_{\pi_t}$, one gets $m^-_{\pi'_t} = m^-_{\pi_t}$. Finally, having the same action imposes also equality of the m^+ part, so π_t and π'_t are equivalent macro-tiles, that can be considered as a single one by the algorithm.

Nevertheless, the structural reduction of lemma 7 remains valid. But for the same reasons as above, we need to adapt it.

Lemma 9 *The macro-tile π_t is useless to the diagnosis algorithm if it doesn't satisfy the following properties:*

1. *there is no silent maximum in π_t, or equivalently the (visible) tile t is the unique maximum of π_t,*

2. *(no counterpart: π_t can contain silent t's.)*

3. *π_t has no loop, i.e. there is no $s = (h_1, \ldots, h_k, t) \in Lin(\pi_t)$ such that a subsequence $s' = (h_i, \ldots, h_j)$ has an empty action, and the shortened sequence $s'' = (h_1, \ldots, h_{i-1}, h_{j+1}, \ldots, h_k, t)$ has the same Q and O sets as π_t.*

Proof.
3 • Same proof as lemma 7, but the definition of a loop is more restrictive, since the shortened sequence s'' must have the same effect as s. In particular, this limitation cancels the possibility $m^-_{\pi_t} \subsetneq m^-_{s''}$. Free loops satisfying this new definition must still be erased.
1 • Same proof as lemma 7.
2 • This point of lemma 7 has no extension here. The proof relied on the possibility to exchange the final visible t with a silent t, denoted by $\bar{t}$, hidden in the body of the tile. Doing so here would disable the possibility to observe a causality relation $t \to t'$ with a next transition t', so the permutation does have an effect on the algorithm.

The construction of useful t-macro-tiles follows follows the lines of section 5.4.3. Rule 3 becomes useless, rule 2 has to allow the connection of a silent t, and rule 4 must look for free loops in equivalent tiles only.

6 Conclusion

The partially stochastic Petri nets developed in this paper provide independent behaviors to regions of the net that are not directly interacting. They thus reach some kind of equivalence between concurrency of events and independence, and so are well adapted to large distributed systems. Their trajectories are causality graphs of transitions, or equivalently parts of the Petri net unfolding, that can be obtained recursively like a puzzle, by connection of tiles.

The diagnosis problem addressed in this paper assumes a sequence of alarms. It builds optimal trajectories in the spirit of the Viterbi algorithm, by connecting tiles that match the observed sequence of alarms and their causal dependence relations. Whence the name "Viterbi puzzle." We have proved that the resulting optimal trajectory, as a causality graph, didn't depend on which sequence of alarms was observed, provided the same causality graph was satisfied. This suggests to directly consider observations as an incomplete causality graph of alarms.

Efforts are now oriented towards the distribution of the diagnosis algorithm, based on the puzzle paradigm. The observed CG of alarms can be split into pieces for which local diagnoses can be computed, as illustrated by our example. The latter can then be considered as new "tiles" for a global diagnosis. This suggests very interesting potential capabilities of PSPNs. One can imagine for example distributing the diagnosis algorithm on a hierarchy of sensors, each one computing locally the possible components of a trajectory, and relying on the upper level for the connection. This would require a weaker version of (H1), since the causal observation property would only be needed locally. Another target application of distribution properties is the design of a supervising structure that would mirror the physical structure of the network. This is a natural way of keeping monitoring algorithms up to date, which remains a key difficulty for many other models.

References

[1] A. Aghasaryan, R. Boubour, E. Fabre, C. Jard, A. Benveniste, “A Petri net approach to fault detection and diagnosis in distributed systems.” IRISA Research Report no. 1117, August 1997.

[2] R. Boubour, C. Jard, A. Aghasaryan, E. Fabre, A. Benveniste, “A Petri net approach to fault detection and diagnosis in distributed systems. Part I : application to telecommunication networks, motivations and modeling.” CDC’97 Proceedings, San Diego, December 1997.

[3] A. Aghasaryan, E. Fabre, A. Benveniste, R. Boubour, C. Jard, “A Petri net approach to fault detection and diagnosis in distributed systems. Part II : extending Viterbi algorithm and HMM techniques to Petri nets.” CDC’97 Proceedings, San Diego, December 1997.

[4] A. Benveniste, B.C. Levy, E. Fabre, P. Le Guernic, “A Calculus of Stochastic Systems : Specification, Simulation, and Hidden State Estimation,” Theoretical Computer Science, no. 152, pp. 171-217, 1995.

[5] R. David, H. Alla, “Petri Nets for Modeling of Dynamic Systems - A Survey,” Automatica, vol. 30, no. 2, pp. 175-202, 1994.

[6] L.R. Rabiner, “A tutorial on Hidden Markov Models and Selected Applications in Speech Recognition,” Proceedings of IEEE, vol. 77, no.2, February 1989.

[7] M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, G. Franceschinis, “Modeling with Generalized Stochastic Petri Nets,” Wiley Series in Parallel Computing, 1995.

[8] M. Ajmone Marsan, G. Balbo, G. Chiola, G. Conte, “Generalized Stochastic Petri Nets Revisited : Random Switches and Priorities,” in Proc. of PNPM ’87, IEEE-CS Press, pp. 44-53.

[9] F. Baccelli, G. Cohen, G.J. Olsder, J.-P. Quadrat, “Synchronization and Linearity, An Algebra for Discrete Event Systems,” Wiley Series in Probability and Mathematical Statistics, 1992.

[10] W. Vogler, “Modular Construction and Partial Order Semantics of Petri Nets,” LNCS no. 625, 1992.