Diagnosis of asynchronous discrete event systems, a net unfolding approach [*]

Albert Benveniste, Eric Fabre, Claude Jard, and Stefan Haar[†]

13th February 2002

Abstract

In this paper we consider the diagnosis of asynchronous discrete event systems. We follow a so-called true concurrency approach, in which no global state and no global time is available. Instead, we use only local states in combination with a partial order model of time. Our basic mathematical tool is that of *net unfoldings* originating from the Petri net research area. This study was motivated by the problem of event correlation in telecommunications network management.

Keywords: *diagnosis, asynchronous diagnosis, discrete event systems, Petri nets, unfoldings, alarm correlation.*

1 Introduction

In this paper we study the diagnosis of truly asynchronous systems. Typical examples are networked systems, such as shown in Fig. 1. In this figure, the sensor system is distributed.

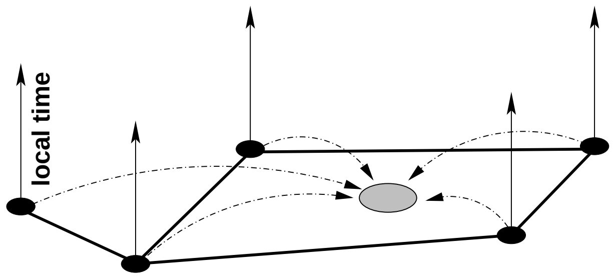


Figure 1: Supervision of a networked system.

It involves several local sensors, attached to some nodes of the network (shown in black). Each sensor has only a partial view of the overall system. The different sensors possess

[*]This work was supported by the RNRT project MAGDA, funded by the Ministère de la Recherche; other partners of the project are France Telecom R&D, Alcatel, Ilog, and Paris-Nord University.

[†]IRISA, Campus de Beaulieu, 35042 Rennes cedex, France. A.B., E.F., S.H. are with Inria, C.J. is with CNRS. Corresponding author Albert.Benveniste@irisa.fr

their own local time, but they are not synchronized. Alarms are reported to the global supervisor (depicted in grey) asynchronously, and this supervisor performs diagnosis. This is the typical architecture in telecommunications network management systems today, our motivating application[1]. Events may be correctly ordered by each indidual sensor, but communicating alarm events via the network causes a loss of synchronization, and results in a *nondeterministic* and *unbounded* interleaving at the supervisor. Hence, the right picture, for what the supervisor collects, is not a sequence of alarms, but rather a *partially ordered* set of alarms.

Fault diagnosis in discrete event systems has attracted a significant attention, see the work of Lafortune and co-authors [34][10] for an overview of the literature and introduction to the subject. Decentralized diagnosis is analyzed in [10], including both algorithms and their diagnosability properties; the solution is formulated in terms of a precomputed decentralized *diagnoser*, consisting of a set of communicating machines that have their states labeled by sets of faults and react to alarm observations and communications; the language oriented framework of Wonham and Ramadge (see [9]) is used, and the systems architecture is that of communicating automata, with a synchronous communication based on a global time, as revealed by the assumption "A6" in [10]. The work [10] has been extended by the same authors in [11] toward considering the effect of (bounded) communication delays in decentralized diagnosis. Difficulties resulting from communications in diagnosis are also investigated by Sengupta in [35]. Finally, the recent work of Tripakis [36] discusses issues of undecidability for a certain type of decentralized observability, this issue has again some relation with asynchrony. Baroni et al. [4] propose a different approach, more in the form of a simulation guided by the observed alarms, for a model of communicating automata. The solution proposed offers a first attempt to handle the problem of state explosion which results from the interleaving of events involving different components.

Diagnosis in the framework of Petri net models has also been investigated by some authors. Hadjicostis and Verghese [22] consider faults in Petri nets in the form of losses or duplications of tokens, this is different than using Petri nets as an asynchronous machine model, for diagnosis. Valette and co-authors [33] use Petri nets to model the normal behavior of systems, and considers as faults the occurrence of events that do not match firing conditions properly. The work closest to ours is that of Giua and co-authors [23][24], it considers the estimation of the current marking from observations.

Event correlation in network management is the subject of a considerable literature, and a number of commercial products are available. We refer the reader to Gardner [21] for a survey. There are two main frameworks for most methods developed in this area. The first framework relates to rule-based or case-based reasoning, an approach very different from the one we study here. The second one uses a causal model, in which the relation between faulty states and alarm events is modelled. The articles by Bouloutas et al. [7][8][27] belong to this family. The case of event correlation in network management

[1]See [28] and the url http://magda.elibel.tm.fr/ for a presentation of the MAGDA project on fault management in telecommunications networks.

also motivated the series of papers by Fabre et al. [6][2][3], on which the present paper relies.

As said before, our present approach was motivated by the problem of fault management in telecommunications networks, so it is worth discussing how this context motivated some of our choices. As seen from our bibliographical discussion, two classes of approaches were available, to handle the diagnosis of asynchronous systems.

A possible approach would consist in constructing a diagnoser in the form of a Petri net, having certain places labeled by faults, and transitions labeled by alarms. Received alarms trigger the net, and visiting a faulty place would indicate that some fault occurred in the original net for monitoring. Another approach would consist in estimating the current marking of the Petri net for monitoring, as in [23][24].

For our application, we needed to support distributed sensor setups, from which wrong interleaving can result. Hence we feel it important, that robustness against a wrong interleaving should be addressed. However, the available approaches typically assume that alarms are received in sequence, and that this sequence is an actual firing sequence of the net.

Also, for our application in fault management in telecommunications networks (where faults are typically transient), providing explanations in the form of *scenarios,* not just snapshots, was essential. Finally, returning all scenarios compatible with the observations, was the requirement from operators in network management. They did not ask for a more elaborated information such as fault *identification,* or *isolation.*

In this paper, we propose an approach to handle unbounded asynchrony in discrete event systems diagnosis by using net *unfoldings,* originally proposed by Nielsen,Plotkin and Winskel [30]. Unfoldings were used by Mc Millan [29] for model checking in verification. They were subsequently developed by Engelfriet [13][32], Esparza, and Römer [14][15][16]. Net unfoldings are branching structures suitable to represent the set of executions of a Petri net using an asynchronous semantics with local states and partially ordered time. In this structure, common prefixes of executions are shared, and executions differing only via the interleaving of their fired transitions are represented only once. Our motivation, for using Petri nets and their unfoldings, is to have an elegant model of asynchronous finite state machines, therefore we restrict ourselves to safe Petri nets throughout this paper. Net unfoldings are not wellknown in the control community, they have been however used for supervisory control in [25][26].

The paper is organized as follows. Section 2 is devoted to the problem setting. Subsection 2.1 collects the needed background material on Petri nets and their unfoldings. Subsection 2.2 introduces our first example. And our problem setting for asynchronous diagnosis is formalized in subsection 2.3, which constitutes per se our first contribution.

In asynchronous diagnosis, some recorded alarm sequences differ only via the interleaving of concurrent alarms, hence it is desirable not to distinguish such alarm sequences. Similarly, it is desirable not to distinguish diagnoses which only differ in the interleaving of concurrent faults. *Diagnosis nets* are introduced to this end in section 3, they express the solution of asynchronous diagnosis by using suitable unfoldings, and constitute the main contribution of this paper. Corresponding (asynchronous) algorithms will be given in the extended version of this paper. Finally we draw some conclusions and perspectives.

2 Asynchronous diagnosis: problem setting

In this section we first introduce the background we need on Petri nets and their unfoldings. Then we introduce informally asynchronous diagnosis on an example. And finally we formally define asynchronous diagnosis.

2.1 Background notions on Petri nets and their unfoldings

Basic references are [31][9][12]. Homomorphisms, conflict, concurrency, and unfoldings, are the essential concepts on which a true concurrency and fully asynchronous view of Petri nets is based. In order to introduce these notions, it will be convenient to consider general "nets" in the sequel.

Nets and homomorphisms. A *net* is a triple $\mathcal{P} = (P, T, \rightarrow)$, where P and T are disjoint sets of *places* and *transitions*, and $\rightarrow \subset (P \times T) \cup (T \times P)$ is the *flow relation.* The reflexive transitive closure of the flow relation $\rightarrow$ is denoted by $\preceq$, and its irreflexive transitive closure is denoted by $\prec$. Places and transitions are called *nodes,* generically denoted by x. For $x \in P \cup T$, we denote by ${}^\bullet x = \{y : y \rightarrow x\}$ the *preset* of node x, and by $x^\bullet = \{y : x \rightarrow y\}$ its *postset.* For $X \subset P \cup T$, we write ${}^\bullet X = \bigcup_{x \in X} {}^\bullet x$ and $X^\bullet = \bigcup_{x \in X} x^\bullet$. An *homomorphism* from a net $\mathcal{P}$ to a net $\mathcal{P}'$ is a map $\varphi : P \cup T \longmapsto P' \cup T'$ such that: 1/ $\varphi(P) \subseteq P'$, $\varphi(T) \subseteq T'$, and 2/ for every node x of $\mathcal{P}$, the restriction of φ to ${}^\bullet x$ is a bijection between ${}^\bullet x$ and ${}^\bullet \varphi(x)$, and the restriction of φ to $x^\bullet$ is a bijection between $x^\bullet$ and $\varphi(x)^\bullet$.

Occurrence nets. Two nodes x, x' of a net $\mathcal{P}$ are *in conflict,* written $x \# x'$, if there exist distinct transitions $t, t' \in T$, such that ${}^\bullet t \cap {}^\bullet t' \neq \emptyset$ and $t \preceq x$, $t' \preceq x'$. A node x is in *self-conflict* if $x \# x$. An *occurrence net* is a net $\mathcal{O} = (B, E, \rightarrow)$, satisfying the following additional properties:

$\forall x \in B \cup E : \neg[x \# x]$	no node is in self-conflict
$\forall x \in B \cup E : \neg[x \prec x]$	$\preceq$ is a partial order
$\forall x \in B \cup E : \lvert\{y : y \prec x\}\rvert < \infty$	$\preceq$ is well founded
$\forall b \in B : \lvert{}^\bullet b\rvert \leq 1$	each place has at most one input transition

We will assume that the set of minimal nodes of $\mathcal{O}$ is contained in B, and we denote by $\min(B)$ or $\min(\mathcal{O})$ this minimal set. Specific terms are used to distinguish occurrence nets from general nets. B is the set of *conditions,* E is the set of *events,* $\prec$ it the *causality* relation.

Nodes x and x' are *concurrent,* written $x \perp\!\!\!\perp x'$, if neither $x \preceq x'$, nor $x \preceq x'$, nor $x \# x'$ hold. A *co-set* is a set X of concurrent conditions. A maximal (for set inclusion) co-set is called a *cut*. A *configuration* κ is a sub-net of $\mathcal{O}$, which is *conflict-free* (no two nodes are in conflict), *causally closed* (if $x' \preceq x$ and $x \in \kappa$, then $x' \in \kappa$), and terminates at a cut.

Occurrence nets are useful to represent executions of Petri nets. They are a subclass of nets, in which essential properties are visible via the topological structure of the bipartite graph—unlike for Petri nets.

Petri nets. For $\mathcal{P}$ a net, a *marking* of $\mathcal{P}$ is a multiset M of places, i.e., a map $M : P \longmapsto \{0, 1, 2, \ldots\}$. A *Petri net* is a pair $\mathcal{P} = (\mathcal{P}, M_0)$, where $\mathcal{P}$ is a net having finite sets of places and transitions, and M_0 is an *initial* marking. A transition $t \in T$ is *enabled* at marking M if $M(p) > 0$ for every $p \in {}^\bullet t$. Such a transition can *fire*, leading to a new marking $M' = M - {}^\bullet t + t^\bullet$, we denote this by $M[t\rangle M'$. The set of *reachable* markings of $\mathcal{P}$ is the smallest (w.r.t. set inclusion) set $M_0[\rangle$ containing M_0 and such that $M \in M_0[\rangle$ and $M[t\rangle M'$ together imply $M' \in M_0[\rangle$. Petri net $\mathcal{P}$ is *safe* if $M(P) \subseteq \{0, 1\}$ for every reachable marking M. Throughout this paper, we consider only safe Petri nets, hence marking M is regarded as a subset of places. A finite occurence net $\mathcal{B}$ can be regarded as a Petri net, where the initial marking is $M_0 = \min(\mathcal{B})$.

Branching processes and unfoldings. A *branching process* of Petri net $\mathcal{P}$ is a pair $\mathcal{B} = (\mathcal{O}, \varphi)$, where $\mathcal{O}$ is an occurrence net, and φ is an homomorphism from $\mathcal{O}$ to $\mathcal{P}$ regarded as nets, such that: 1/ the restriction of φ to $\min(\mathcal{O})$ is a bijection between $\min(\mathcal{O})$ and M_0 (the set of initially marked places), and 2/ for all $e, e' \in E$, ${}^\bullet e = {}^\bullet e'$ and $\varphi(e) = \varphi(e')$ together imply $e = e'$.

The set of all branching processes of Petri net $\mathcal{P}$ is uniquely defined, up to an isomorphism (i.e., a renaming of the conditions and events), and we shall not distinguish isomorphic branching processes. For $\mathcal{B}, \mathcal{B}'$ two branching processes, $\mathcal{B}'$ is a *prefix* of $\mathcal{B}$, written $\mathcal{B}' \sqsubseteq \mathcal{B}$, if there exists an injective homomorphism ψ from $\mathcal{B}'$ into $\mathcal{B}$, such that the composition $\varphi \circ \psi$ coincides with φ'. By theorem 23 of [13], there exists (up to an isomorphism) a unique maximum branching process according to $\sqsubseteq$, we call it the *unfolding* of $\mathcal{P}$, and denote it by $\mathcal{U}_{\mathcal{P}}$.

As announced, configurations are adequate representations of firing sequences of Petri net $\mathcal{P}$. Let $M_0, M_1, M_2, \ldots$ be a maximal firing sequence of $\mathcal{P}$, and let $M_{k-1}[t_k\rangle M_k$ be the associated sequence of fired transitions. Then there exists a unique maximal (for set inclusion) configuration κ of $\mathcal{P}$ having the following properties: κ is the union of a sequence $e_1, e_2, \ldots$ of events and a sequence $\mathbf{c}_0, \mathbf{c}_1, \mathbf{c}_2, \ldots$ of cuts, such that, for each $k > 0$, $\varphi(\mathbf{c}_k) = M_k$, $\varphi(e_k) = t_k$, and $\mathbf{c}_{k-1} \supseteq {}^\bullet e_k, e_k^\bullet \subseteq \mathbf{c}_k$. Conversely, each maximal configuration of $\mathcal{P}$ defines a maximal firing sequence, which is unique up to the interleaving of structurally concurrent transitions—transitions t and t' are structurally concurrent iff ${}^\bullet t' \cap ({}^\bullet t \cup t^\bullet) = \emptyset$ and ${}^\bullet t \cap ({}^\bullet t' \cup t'^\bullet) = \emptyset$.

Example 1. Fig. 2 shows the example we will use throughout this paper. A Petri net $\mathcal{P}$ is shown on the left. Its places are $1, 2, 3, 4, 5, 6, 7$, and its transitions are i, ii, iii, iv, v, vi. Places consituting the initial marking are encircled in thick.

A branching process $\mathcal{B} = (\mathcal{O}, \varphi)$ of $\mathcal{P}$ is shown on the right. Its conditions are depicted

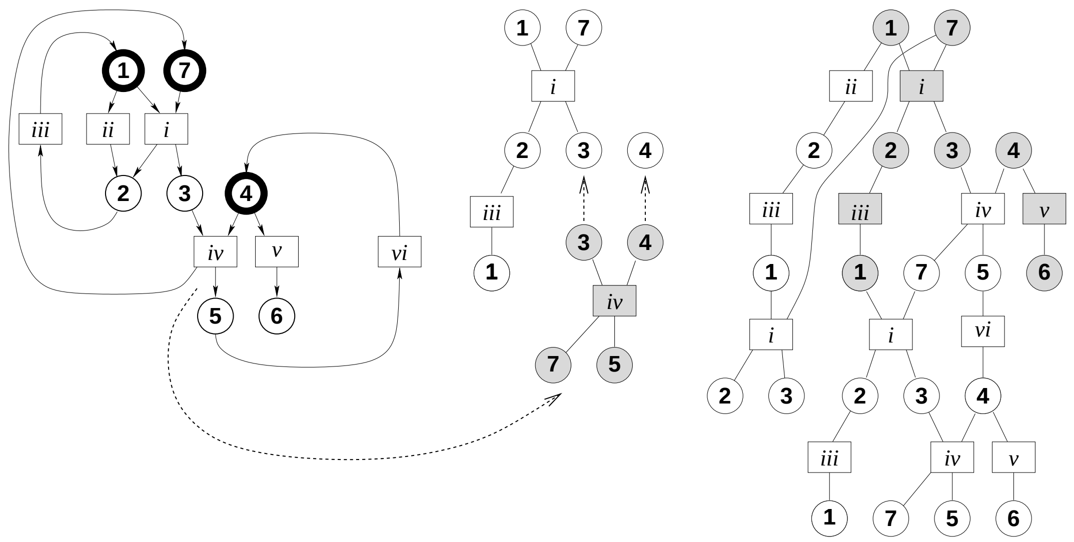


Figure 2: Example 1, its unfolding, and a configuration. For this and subsequent examples, we take the following convention for drawing Petri nets and occurrence nets. In Petri nets, the flow relation is depicted using *directed arrows.* In occurrence nets, since no cycle occurs, the flow relation *progresses downwards,* and therefore there is no need to figure them via directed arrows, standard solid lines are used instead.

by circles, and its events are figured by boxes. Each condition b of $\mathcal{B}$ is labeled by $\varphi(b)$, a place of $\mathcal{P}$. Each event e of $\mathcal{B}$ is labeled by $\varphi(e)$, a transition of $\mathcal{P}$. A configuration of Petri net $\mathcal{P}$ is shown in grey. Note that the minimal condition labeled by 7 is branching in $\mathcal{B}$, although it is not branching in $\mathcal{P}$ itself. The reason is that, in $\mathcal{P}$, the token can freely move along the circuit $1 \to ii \to 2 \to iii \to 1$, and resynchronize afterwards with the token sitting in 7.

The mechanism for constructing the unfolding of Petri net $\mathcal{P}$ is illustrated in the middle, it is informally explained as follows. Put the three conditions labeled by the initial marking of $\mathcal{P}$, this is the minimal branching process of $\mathcal{P}$. Then, for each constructed branching process $\mathcal{B}$, select a co-set X of $\mathcal{B}$, which is labeled by the preset ${}^\bullet t$ of some transition t of $\mathcal{P}$, and has no event labeled by t in its postset within $\mathcal{B}$. Append to X a net isomorphic to ${}^\bullet t \to t \to t^\bullet$ (recall that ${}^\bullet t = X$), and label its additional nodes by t and $t^\bullet$, respectively. Performing this recursively yields all possible finite branching processes of $\mathcal{P}$. Their union is the unfolding $\mathcal{U}_\mathcal{P}$.

Labeled nets and their products. For $\mathcal{P} = (P, T, \to)$ a net, a *labeling* is a map $\lambda : T \longmapsto A$, where A is some finite alphabet. A net $\mathcal{P} = (P, T, \to, \lambda)$ equipped with a labeling λ is called a *labeled net.* For $\mathcal{P}_i = \{P_i, T_i, \to_i, \lambda_i\}$, $i = 1, 2$, two labeled nets, their product $\mathcal{P}_1 \times \mathcal{P}_2$ is the labeled net defined as follows:

$$\mathcal{P}_1 \times \mathcal{P}_2 \quad = \quad (P, T, \to, \lambda). \tag{1}$$

In (1), $P = P_1 \uplus P_2$, where $\uplus$ denotes the disjoint union, and:

$$\begin{array}{rcll} & & \{t_1 \in T_1 : \lambda_1(t_1) \in A_1 \setminus A_2\} & \text{(i)} \\ T & = & \cup \;\; \{(t_1,t_2) \in T_1 \times T_2 : \lambda_1(t_1) = \lambda_2(t_2)\} & \text{(ii)} \\ & & \cup \;\; \{t_2 \in T_2 : \lambda_2(t_2) \in A_2 \setminus A_1\}\,, & \text{(iii)} \end{array}$$

$$p \to t \;\; \text{iff} \;\; \begin{array}{ll} p \in P_1 \text{ and } p \to_1 t_1 & \text{for case (ii) or (i)} \\ p \in P_2 \text{ and } p \to_2 t_2 & \text{for case (ii) or (iii)} \end{array}$$

$$t \to p \;\; \text{iff} \;\; \begin{array}{ll} p \in P_1 \text{ and } t_1 \to_1 p & \text{for case (ii) or (i)} \\ p \in P_2 \text{ and } t_2 \to_2 p & \text{for case (ii) or (iii)} \end{array}$$

In cases (i,iii) only one net fires a transition and this transition has a private label, while the two nets synchronize on transitions with identical labels in case (ii). Petri nets and occurrence nets inherit the above notions of labeling and product.

2.2 Discussing asynchronous diagnosis on example 1

A labeled Petri net model. Our first example of Fig. 2 is redrawn slightly differently in Fig. 3, in the form of a labeled Petri net. The example is now intepreted as two

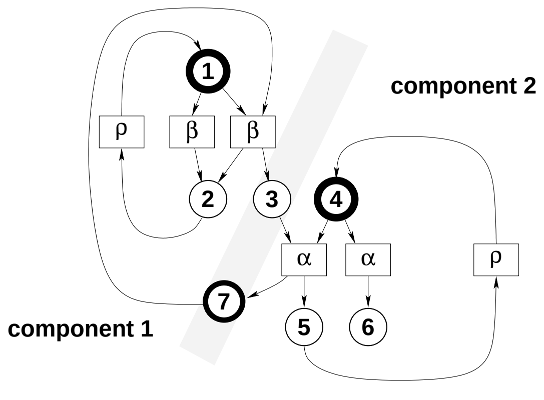


Figure 3: Example 1, two interacting components modelled as a labeled Petri net.

interacting components, numbered 1 and 2. Component 2 uses the services of component 1 for its functioning, and therefore it may fail to deliver its service when component 1 is faulty.

Component 1 has two states: nominal, figured by place 1, and faulty, figured by place 2. When getting into faulty state, the component 1 emits an alarm β, which is associated to transition (i) and (ii) (cf. Fig. 2) as a label. The fault of component 1 is temporary, and therefore self-repair can occur, this is figured by the label ρ associated to transition (iii) (cf. Fig. 2).

Component 2 has three states, figured by places $4, 5, 6$. State 4 is nominal, state 6 indicates that component 2 is faulty, and state 5 indicates that component 2 fails to deliver its service, due to the failure of component 1. Fault 6 is permanent and cannot be repaired.

That component 2 may fail to deliver its service due to a fault of component 1, is modelled by the shared place 3. The monitoring system of component 2 only detects that this component fails to deliver its service, it does not distinguish between the different reasons for this. Hence the *same* alarm α is attached to the two transitions (*iv,v*) as a label (cf. Fig. 2). Since fault 2 of component 1 is temporary, self-repair can also occur for component 2, when in faulty state 5. This self-repair is not synchronized with that of component 1, but is still assumed to be manifested by the same label ρ. Finally, place 7 guarantees that fault propagation, from component 1 to component 2, occurs only when the latter is in nominal state.

The grey area indicates where interaction occurs between the two components. The initial marking consists of the two nominal states $1, 4$, for each component. Labels (alarms α, β or self-repair ρ) attached to the different transitions or events, are generically referred to as *alarms* in the sequel.

The different setups considered, for diagnosis. The first setup assumes that the successive alarms are recorded *in sequence* by a single supervisor, in charge of fault monitoring.

In a second setup, we assume two independent sensors, one per each component. Each sensor records its *local* alarms in sequence, by ignoring the other sensor's records. The two records are collected, *asynchronously,* by a single supervisor. In this setup the interleaving of the two local sequences of alarms is lost.

In the third setup, the fault monitoring is performed in a *distributed* way, by two supervisors cooperating asynchronously. Each supervisor is attached to a component, it records its local alarms in sequence, and can exchange supervision messages with the other supervisor, asynchronously.

In this paper we consider the first and second setups (and generalizations of them), but not the third one, which is an instance of distributed diagnosis. For distributed diagnosis, the reader is referred to [17][18][19].

The different setups are illustrated in Fig. 4, which is a combination of Fig. 2 and Fig. 3. The labeled Petri net of Fig. 3 is redrawn, on the left, with the topology used in Fig. 2. In the middle, we redraw the configuration shown in grey in Fig. 2-right, call it κ, and we relabel its events by their associated alarms. Configuration κ expresses that component 1 went into its faulty state 2, and then was repaired; concurrently, component 2 moved to its faulty state 6, where self-repair cannot occur. Note that the transmission of the fault of component 1 to component 2, via place 3, is preempted, due to the fatal failure of component 2.

How alarms are recorded is modelled by the two occurrence nets shown in the third and fourth diagrams, we call them *alarm patterns.* In the third diagram, we assume the first setup, in which a single sensor is available to collect the alarms. Hence configuration κ produces the alarms β, α, ρ, recorded in sequence. This record is modelled by the linear alarm pattern shown in the third diagram. This alarm pattern has its events labeled by

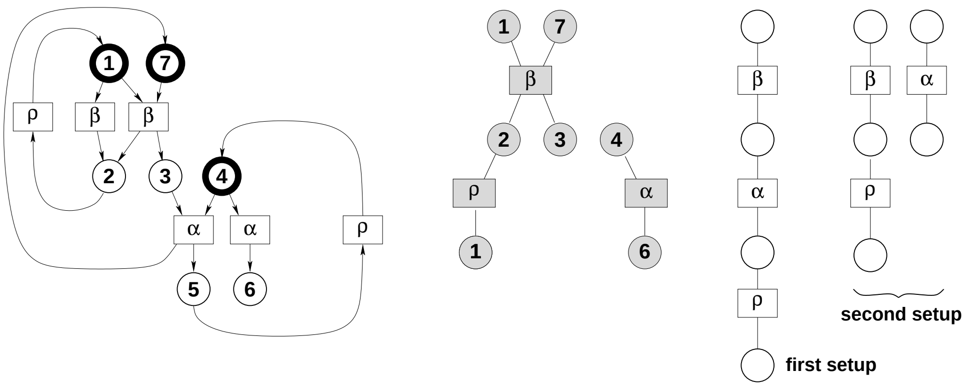


Figure 4: Example 1, a scenario involving a single sensor, and two independent sensors.

alarms, but its conditions are “blind”, i.e., they have no label. This manifests the fact that the different places traversed while producing the alarms β, α, ρ are not observed.

Now, in the last diagram, we show the case of the second setup, in which β, ρ are collected by the first sensor, and α is collected by the second one, independently. The result is an alarm pattern composed of two concurrent parts, corresponding to the records collected by each sensor. When collected by the supervisor, these concurrent parts can interleave arbitrarily—this manifests asynchrony.

Alarm patterns are generically denoted by the symbol $\mathcal{A}$. Note that each sensor delivers, as an alarm pattern, some linear extension of the partial order of events it sees. But the causality relations involving pairs of events seen by different sensors, are lost. In general, *observations may add some causalities, may lose other ones, but they never reverse any of them.* Therefore, the only valid invariant between alarm pattern $\mathcal{A}$ and the configuration κ that produced it, is that $\mathcal{A}$ and κ possess a common linear extension. With this definition, we encompass the different considered setups in a common framework.

Asynchronous diagnosis. From the above discussion on asynchronous diagnosis, we must accept as plausible explanations of an alarm pattern $\mathcal{A}$ any configuration κ such that $\mathcal{A}$ and κ possess a common linear extension. Such κ are said to *explain* $\mathcal{A}$. We are now ready to formalize our problem setting.

2.3 Asynchronous diagnosis: formal problem setting

Now, we formalize what an alarm pattern $\mathcal{A}$ is, and what it means, for $\mathcal{A}$, to be associated with some configuration κ. We are given the following objects, where the different notions have been introduced in subsection 2.1:

- A labeled Petri net $\mathcal{P} = (P, T, \rightarrow, M_0, \lambda)$, where the range of the labeling map λ is the set of possible *alarms,* denoted by A, and
- its unfolding $\mathcal{U}_{\mathcal{P}} = (B, E, \rightarrow, \varphi)$.

Note the following chain of labeling maps:

$$\underbrace{E}_{\text{events}} \xrightarrow{\varphi} \underbrace{T}_{\text{transitions}} \xrightarrow{\lambda} \underbrace{A}_{\text{alarms}} \quad : \quad e \longmapsto \varphi(e) \longmapsto \lambda(\varphi(e)) \stackrel{\Delta}{=} \Lambda(e)\,, \tag{2}$$

which defines the *alarm label* of event e, we denote it by $\Lambda(e)$—we call it also "alarm", for short, when no confusion can occur.

An *extension* of a net $\mathcal{P} = (P, T, \rightarrow)$ is any net obtained by adding places and flow relations but not transitions. Occurrence nets inherit this notion. An occurrence net induces a labeled partial order on the set of its events, extending this occurrence net induces an extension of this labeled partial order [2].

Two labeled occurrence nets $\mathcal{O} = (B, E, \rightarrow, \Lambda)$ and $\mathcal{O}' = (B', E', \rightarrow', \Lambda')$ are called *alarm-isomorphic* if there exists an isomorphism ψ, from $(B, E, \rightarrow)$ onto $(B', E', \rightarrow')$, seen as directed graphs, which preserves the alarm labels, i.e., such that $\forall e \in E : \Lambda'(\psi(e)) = \Lambda(e)$. Two alarm-isomorphic occurrence nets can be regarded as identical if we take into account the alarm labels of their events, but ignore their conditions.

Definition 1 (alarm pattern) *Consider $\mathcal{P}$, $\mathcal{U}_{\mathcal{P}}$, and Λ, as in (2). A labeled occurrence net $\mathcal{A} = (B_{\mathcal{A}}, E_{\mathcal{A}}, \rightarrow_{\mathcal{A}}, \lambda_{\mathcal{A}})$ is an* alarm pattern *of $\mathcal{P}$ iff:*

1. *Its labeling map $\lambda_{\mathcal{A}}$ takes its value in the alphabet A of alarms,*

2. *$\mathcal{A}$ is itself a configuration (it is conflict free), its set of conditions $B_{\mathcal{A}}$ is disjoint from that of $\mathcal{U}_{\mathcal{P}}$, and*

3. *There exists a configuration κ of $\mathcal{U}_{\mathcal{P}}$, such that $\mathcal{A}$ and κ possess extensions that are alarm-isomorphic.*

Assuming, for $\mathcal{A}$, a set of places disjoint from that of $\mathcal{U}_{\mathcal{P}}$, aims at reflecting that alarm patterns vehicle no information regarding hidden states of the original net. This justifies condition 2. Concerning condition 3 the allowed discrepancy between κ and $\mathcal{A}$ formalizes the possible loss of some causalities (e.g., due to independent and non synchronized sensors), and the possible adding of other ones (e.g., when sensors record their alarms in sequence). The key fact is that the information about the concurrency of events produced by the system cannot be observed by the supervisor. To refer to our context of diagnosis, we say that κ *can explain* $\mathcal{A}$. For $\mathcal{A}$ a given alarm pattern of $\mathcal{P}$, we denote by

$$\mathbf{diag}(\mathcal{A}) \tag{3}$$

the set of configurations κ of $\mathcal{U}_{\mathcal{P}}$, satisfying the conditions 1,2,3 of definition 1. In the next subsection, we propose an adequate data structure to represent the set $\mathbf{diag}(\mathcal{A})$ in a compact way, we call it a *diagnosis net.*

[2] Recall that the labeled partial order $(X, \preceq)$ is an *extension* of labeled partial order $(X', \preceq')$ if labeled sets X and X' are isomorphic, and $\preceq \supseteq \preceq'$ holds. When $(X, \preceq)$ is a total order, we call it a *linear* extension of $(X', \preceq')$.

3 Diagnosis nets: expressing asynchronous diagnosis by means of unfoldings

In this section, we provide explicit formulas for the solution of asynchronous diagnosis, in the form of suitable unfoldings.

A first natural idea is to represent $\mathbf{diag}(\mathcal{A})$ by the minimal subnet of unfolding $\mathcal{U}_{\mathcal{P}}$, containing all configurations $\in$ $\mathbf{diag}(\mathcal{A})$, we denote it by $\mathcal{U}_{\mathcal{P}}(\mathcal{A})$. Subnet $\mathcal{U}_{\mathcal{P}}(\mathcal{A})$ inherits canonically by restriction, the causality, conflict, and concurrence relations defined on $\mathcal{U}_{\mathcal{P}}$. Net $\mathcal{U}_{\mathcal{P}}(\mathcal{A})$ contains all configurations belonging to $\mathbf{diag}(\mathcal{A})$, but unfortunately it also contains undesirable maximal configurations *not* belonging to $\mathbf{diag}(\mathcal{A})$, as Fig. 5 reveals.

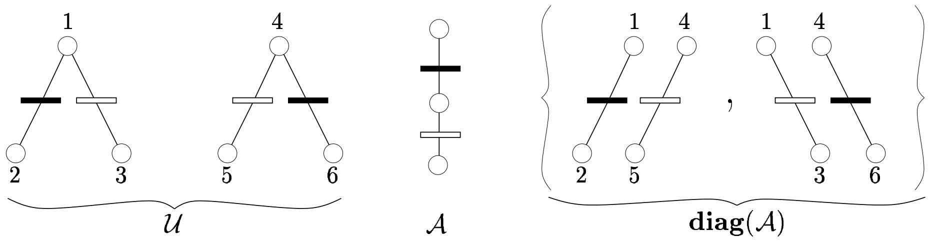
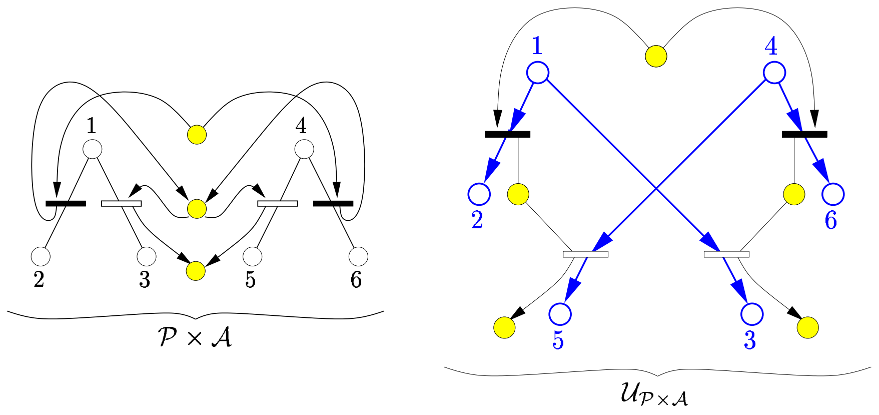


Figure 5: Example 2. Showing $\mathcal{U}, \mathcal{A}$, and $\mathbf{diag}(\mathcal{A})$. Note that $\mathcal{U}_{\mathcal{P}}(\mathcal{A}) = \mathcal{U}$.

In this figure, we show an unfolding $\mathcal{U}$ on the left hand side. In the middle, we show a possible associated alarm pattern $\mathcal{A}$. Alarm labels are figured by colors (black and white). The set $\mathbf{diag}(\mathcal{A})$ is shown on the right hand side, it comprises two configurations. Unfortunately the minimal subnet $\mathcal{U}_{\mathcal{P}}(\mathcal{A})$ of the original unfolding $\mathcal{U}$ which contains $\mathbf{diag}(\mathcal{A})$, is indeed identical to $\mathcal{U}$! Undesirable configurations are $\{(1, t_{12}, 2), (4, t_{46}, 6)\}$ and $\{(1, t_{13}, 3), (4, t_{45}, 5)\}$ (in these statements, t_{12} denotes the transition separating states 1 and 2). But configuration $\{(1, t_{12}, 2), (4, t_{46}, 6)\}$ is such that its two transitions t_{12}, t_{46} explain the same alarm event in $\mathcal{A}$. And the same holds for the other undesirable configuration.

Fig. 6 suggests an alternative solution, using the product $\mathcal{P} \times \mathcal{A}$ of $\mathcal{P}$ and $\mathcal{A}$, seen as labeled nets with respective labels λ and $\lambda_{\mathcal{A}}$ (see subsection 2.3 for these notations). The unfolding $\mathcal{U}_{\mathcal{P}\times\mathcal{A}}$ is shown. The projection, on the set of nodes labelled by nodes from $\mathcal{P}$, is depicted using larger arrows. The reader can verify that the corresponding set of maximal configurations coincides with $\mathbf{diag}(\mathcal{A})$. This suggests that $\mathcal{U}_{\mathcal{P}\times\mathcal{A}}$ is an appropriate representation of $\mathbf{diag}(\mathcal{A})$. We formalize this in the theorem to follow. We use the notations from subsections 2.1 and 2.3, and we need a few more notations.

For $\mathcal{P} = (P, T, \rightarrow)$ a net and X a subset of its nodes, $\mathcal{P}_{|X}$ denotes the restriction of $\mathcal{P}$ to X, defined as $\mathcal{P}_{|X} \triangleq (P \cap X, T \cap X, \rightarrow_{|X})$, where the flow relation $\rightarrow_{|X}$ is defined as the restriction, to $X \times X$, of the flow relation $\rightarrow \subseteq (P \times T) \cup (T \times P)$ given on $\mathcal{P}$. Be careful that we restrict the flow relation, not its transitive closure. For $\mathcal{P} = (P, T, \rightarrow, M_0)$ a Petri net, $\mathcal{P}'$ a sub-net of $\mathcal{P}$ with place set P', and $\mathcal{U} = (B, E, \rightarrow, \varphi)$ a sub-net of the unfolding $\mathcal{U}_{\mathcal{P}}$, we denote by $\mathbf{proj}_{\mathcal{P}'}(\mathcal{U})$ the labeled occurrence net $\mathbf{proj}_{\mathcal{P}'}(\mathcal{U}) \triangleq \mathcal{U}_{|\varphi^{-1}(P') \cup E}$, obtained by restricting $\mathcal{U}$ to the set of conditions labelled by places from $\mathcal{P}'$, and restricting the



1
4
2
3
5
6
$\mathcal{P} \times \mathcal{A}$
1
4
2
6
5
3
$\mathcal{U}_{\mathcal{P}\times\mathcal{A}}$

Figure 6: Example 2. Representing $\mathbf{diag}(\mathcal{A})$ by $\mathcal{U}_{\mathcal{P}\times\mathcal{A}}$.

flow relation accordingly. Finally, for $\mathcal{O}$ an occurrence net, we denote by **config** $(\mathcal{O})$ the set of all its configurations.

Theorem 1 *Let $\mathcal{U}_\mathcal{P}$ be the unfolding of some Petri net $\mathcal{P}$, $\mathcal{A}$ an associated alarm pattern, and let* $\mathbf{diag}(\mathcal{A})$ *be defined as in (3). Consider the unfolding $\mathcal{U}_{\mathcal{P}\times\mathcal{A}} = (\overline{B}, \overline{E}, \rightarrow, \overline{\varphi})$, and its associated projections* $\mathbf{proj}_\mathcal{P}(.)$ *and* $\mathbf{proj}_\mathcal{A}(.)$*. Then, $\kappa \in$* $\mathbf{diag}(\mathcal{A})$ *iff:*

$$\exists \overline{\kappa} \in \mathbf{config}\,(\mathcal{U}_{\mathcal{P}\times\mathcal{A}}) \quad : \quad \mathbf{proj}_\mathcal{P}(\overline{\kappa}) = \kappa \text{ and } \mathbf{proj}_\mathcal{A}(\overline{\kappa}) = \mathcal{A}. \tag{4}$$

Note that every $\overline{\kappa}$ satisfying (4) must be a maximal configuration of $\mathcal{U}_{\mathcal{P}\times\mathcal{A}}$. Theorem 1 expresses that $\mathcal{U}_{\mathcal{P}\times\mathcal{A}}$ is an adequate representation of $\mathbf{diag}(\mathcal{A})$, we call it a *diagnosis net.*

Proof: We first prove the *if* part. Let $\overline{\kappa}$ be a configuration of $\mathcal{U}_{\mathcal{P}\times\mathcal{A}}$ such that $\mathbf{proj}_\mathcal{A}(\overline{\kappa}) = \mathcal{A}$, and define $\kappa = \mathbf{proj}_\mathcal{P}(\overline{\kappa})$. By definition of net extensions (cf. definition 1 and above), $\overline{\kappa}$ is an extension of both κ and $\mathcal{A}$. Hence, by definition 1, $\kappa \in \mathbf{diag}(\mathcal{A})$. This was the easy part.

We then prove the *only if* part. Select an arbitrary $\kappa \in \mathbf{diag}(\mathcal{A})$. We need to show the existence of a $\overline{\kappa}$ satisfying (4). Since κ and $\mathcal{A}$ possess alarm-isomorphic extensions, there exists a bijection ψ, from $E_\mathcal{A}$ (the set of events of $\mathcal{A}$), onto the set of events of κ, such that ψ preserves the alarm labels. Pick e, an event of $\mathcal{A}$, and set

$$f \quad = \quad \psi(e). \tag{5}$$

By definition 1, ${}^\bullet e \cup \{e\} \cup e^\bullet$ and ${}^\bullet f \cup \{f\} \cup f^\bullet$ possess alarm-isomorphic extensions—they do not possess contradictory flow relations. Hence we have

$$\begin{aligned} (\kappa \times \mathcal{A})_{|\,{}^\bullet(f,e)\cup\{(f,e)\}\cup(f,e)^\bullet} &= \kappa_{|\,{}^\bullet f\cup\{f\}\cup f^\bullet} \times \mathcal{A}_{|\,{}^\bullet e\cup\{e\}\cup e^\bullet} && (6)\\ &= (\kappa \cup \mathcal{A})_{|\,{}^\bullet f\cup{}^\bullet e\cup\{(f,e)\}\cup f^\bullet\cup e^\bullet}\,. && (7) \end{aligned}$$

where (6) follows from the definition of the product of labeled nets, and (7) results from the fact that ${}^\bullet e \cup \{e\} \cup e^\bullet$ and ${}^\bullet f \cup \{f\} \cup f^\bullet$ possess alarm-isomorphic extensions.

Let $[\psi] \triangleq \{(f,e) : f = \psi(e)\}$ be the graph of ψ. Denote by B_κ and E_κ the two sets of conditions and events of κ, respectively. Denote by $\kappa \times_\psi \mathcal{A}$ the restriction of $\kappa \times \mathcal{A}$ to $(B_\kappa \cup B_\mathcal{A}) \cup [\psi]$, i.e.,

$$\kappa \times_\psi \mathcal{A} \;\triangleq\; (\kappa \times \mathcal{A})_{\,|\,(B_\kappa \cup B_\mathcal{A})\cup[\psi]}$$

We will show that $\overline{\kappa} = \kappa \times_\psi \mathcal{A}$ satisfies (4). To this end, we first show that the flow relation on $\kappa \times_\psi \mathcal{A}$ unfolds the flow relation on $\mathcal{P} \times \mathcal{A}$, therefore showing that it is a subnet of the unfolding $\mathcal{U}_{\mathcal{P}\times\mathcal{A}}$.

Using the notations from (2) and definition 1, define a map

$$\varphi_\kappa \;:\; (B_\kappa \cup B_\mathcal{A}) \cup [\psi] \;\longmapsto\; (P \cup B_\mathcal{A}) \cup (T \times E_\mathcal{A}) \tag{8}$$

as follows: the restriction of φ_κ to B_κ is equal to φ, the restriction of φ_κ to $B_\mathcal{A}$ is the identity, and for $(f,e) \in [\psi]$, set $\varphi_\kappa(f,e) = (\varphi(f), e)$. The so defined map φ_κ is an homomorphism from net $\kappa \times_\psi \mathcal{A}$ to the net $\mathcal{P} \times \mathcal{A}$. In addition, φ_κ is such that, for $\bar{e}, \bar{e}'$ two events of $\kappa \times_\psi \mathcal{A}$, ${}^\bullet\bar{e} = {}^\bullet\bar{e}'$ and $\varphi_\kappa(\bar{e}) = \varphi_\kappa(\bar{e}')$ together imply $\bar{e} = \bar{e}'$. Hence, by the universal property of branching processes and their associated homomorphism, $\kappa \times_\psi \mathcal{A}$ identifies with a subnet of $\mathcal{U}_{\mathcal{P}\times\mathcal{A}}$, and φ_κ must be the restriction, to $(B_\kappa \cup B_\mathcal{A}) \cup [\psi]$, of the homomorphism $\overline{\varphi}$ from the unfolding $\mathcal{U}_{\mathcal{P}\times\mathcal{A}}$ onto $\mathcal{P} \times \mathcal{A}$.

This, and (6,7), together imply that

$$\begin{array}{rcl} \mathbf{proj}_\mathcal{P}\Big((\kappa \times_\psi \mathcal{A})_{\,|\,{}^\bullet f \cup {}^\bullet e \cup \{(f,e)\} \cup f^\bullet \cup e^\bullet}\Big) & = & \kappa_{\,|\,{}^\bullet f \cup \{f\} \cup f^\bullet}\,, \\ \mathbf{proj}_\mathcal{A}\Big((\kappa \times_\psi \mathcal{A})_{\,|\,{}^\bullet f \cup {}^\bullet e \cup \{(f,e)\} \cup f^\bullet \cup e^\bullet}\Big) & = & \mathcal{A}_{\,|\,{}^\bullet e \cup \{e\} \cup e^\bullet}\,, \end{array} \tag{9}$$

where equalities in (9) hold up to a renaming of event (f,e) by f or e alone. Consider again (9). Since it holds for any e, we get:

$$\mathbf{proj}_\mathcal{P}(\kappa \times_\psi \mathcal{A}) = \kappa \quad , \quad \mathbf{proj}_\mathcal{A}(\kappa \times_\psi \mathcal{A}) = \mathcal{A}, \tag{10}$$

which implies that $\kappa \times_\psi \mathcal{A}$ is causally closed and terminates at a cut of $\mathcal{U}_{\mathcal{P}\times\mathcal{A}}$. On the other hand, $\kappa \times_\psi \mathcal{A}$ is clearly conflict free, therefore

$$\overline{\kappa} \;\triangleq\; \kappa \times_\psi \mathcal{A} \text{ is a configuration of } \mathcal{U}_{\mathcal{P}\times\mathcal{A}}. \tag{11}$$

(10) and (11) together prove the *only if* part of (4). ◇

Remark. Theorem 1 assumes the knowledge of the initial marking M_0 for Petri net $\mathcal{P}$. When only a set $\mathcal{M}_0$ of possible initial markings is known instead, simply augment $(P, T, \rightarrow)$ as follows. Add some additional place p_0 not belonging to P, for each possible initial marking $M_0 \in \mathcal{M}_0$ add one transition t_{M_0} to T, and add the branches $p_o \rightarrow t_{M_0} \rightarrow M_0$ to the flow relation. And apply theorem 1 to the so augmented Petri net.

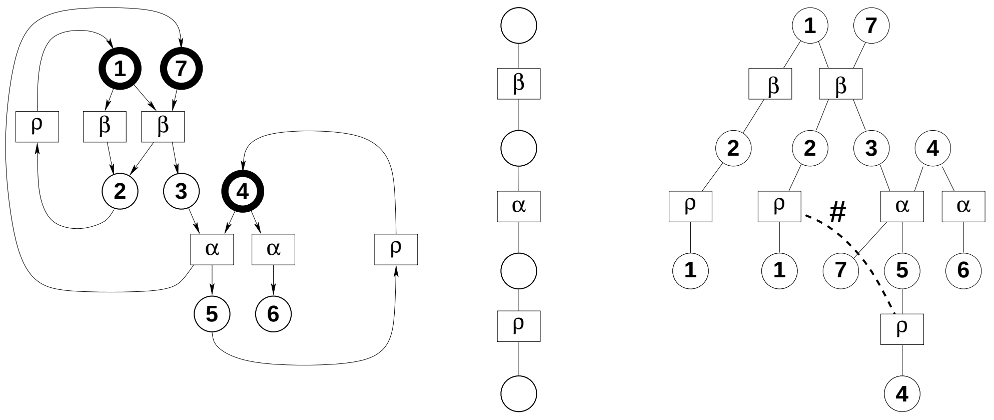


Figure 7: Example 1, diagnosis net, an illustration of theorem 1.

Example 1, illustration of diagnosis nets, and comparison with the use of the marking graph. We show in Fig. 7 an illustration of theorem 1. In this figure we show the Petri net $\mathcal{P}$ of Fig. 3 (left), an associated alarm pattern $\mathcal{A}$ (middle), and the net $\mathcal{U}_{\mathcal{P}\times\mathcal{A}}$, restricted to its nodes labeled by nodes from $\mathcal{P}$ (right). We show in dashed-thick the additional conflict relation between the two otherwise concurrent events labeled by the same alarm ρ. This conflict is inherited from the sharing of a common condition—not shown—belonging to $\mathcal{A}$. It is easily checked that $\mathbf{diag}(\mathcal{A})$ is adequately represented by this diagram. Note that the restriction, to its events, of this data structure, is an *event structure* according to Winskel's definition [30][37].

Four alternative explanations are delivered by this diagnosis, this reflects the ambiguity resulting from asynchrony in this example. Explanation 1: component 1 gets in its faulty state without causing damage to component 2, and then gets self-repaired; independently, component 2 gets into its fatal faulty state 6 ($(\beta \prec \rho) \perp\!\!\!\perp \alpha$). Explanation 2: component 1 gets in its faulty state while causing damage to component 2, and then gets self-repaired; independently, component 2 gets into its fatal faulty state 6 (again $(\beta \prec \rho) \perp\!\!\!\perp \alpha$). Explanation 3: component 1 gets in its faulty state while causing damage to component 2; consequently, component 2 fails to delivers its service and gets into its state 5, where it subsequently gets self-repaired ($\beta \prec \rho \prec \alpha$). Explanation 4: component 1 gets in its faulty state while causing damage to component 2; consequently, component 2 fails to delivers its service; independently component 1 gets self-repaired ($\beta \prec (\rho \perp\!\!\!\perp \alpha)$).

Fig. 8 compares diagnosis nets with the use of marking graphs. The reader is referred again to our running example 1 (shown in Fig. 7), call it $\mathcal{P}$. In the first diagram we show the *marking graph* of $\mathcal{P}$, denoted by $\mathcal{M}(\mathcal{P})$. It is a labeled Petri net whose places are labeled by the reachable markings of $\mathcal{P}$, shown by the combination of the places composing the different markings. We identify the places of the marking graph $\mathcal{M}(\mathcal{P})$ with the markings of $\mathcal{P}$. Then, $M[t\rangle M'$ in $\mathcal{P}$ iff $M \to \tau \to M'$ in $\mathcal{M}(\mathcal{P})$. Transition τ of $\mathcal{M}(\mathcal{P})$ is then labeled by transition t of $\mathcal{P}$. In Fig. 8 we have labeled instead the transitions τ of $\mathcal{M}(\mathcal{P})$ by the alarm labels $\lambda(t) = \alpha, \beta, \rho$ of the associated transitions t from $\mathcal{P}$.

The pre/postset of each transition of $\mathcal{M}(\mathcal{P})$ is a singleton, hence there is no concur-

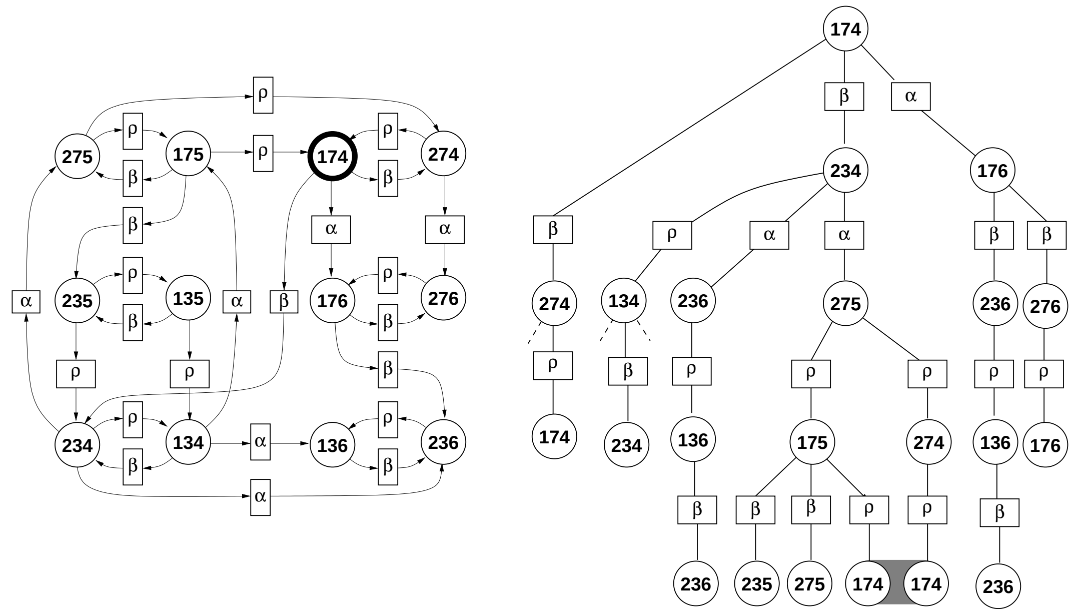


Figure 8: Marking graph of example 1 (left), and unfolding (right).

rency, and $\mathcal{M}(\mathcal{P})$ models an automaton. Note the diamond composed of the two branches $275 \to \rho \to 175 \to \rho \to 174$ and $275 \to \rho \to 274 \to \rho \to 174$, it represents the two possible interleavings of the concurrent transitions labeled by ρ in $\mathcal{P}$.

We can still regard $\mathcal{M}(\mathcal{P})$ as a Petri net, and consider its unfolding $\mathcal{U}_{\mathcal{M}(\mathcal{P})}$, shown in part in the second diagram (some flow relations are sketched in dashed, to save space). Now, we can safely merge the two conditions labeled by 174 in the bottom of this diagram. The reasons for this are the following: 1/ they are both labeled by the same state (namely 174), hence they possess identical continuations, and, 2/ their causal closures is labeled by the same alarm sequence $\beta, \alpha, \rho, \rho$, i.e., explain the same sequences of alarms. Merging the two conditions labeled by 174 in the bottom of the second diagram yields a *lattice,* i.e., a labeled net with branching and joining conditions but no circuit, we denote it by $\mathcal{L}_{\mathcal{M}(\mathcal{P})}$. Lattices are not new, they are the data structures used when applying the Viterbi algorithm for maximum likelihood estimation of hidden state sequences in stochastic automata.

Being linear and not branching any more, $\mathcal{L}_{\mathcal{M}(\mathcal{P})}$ is a more compact data structure than the unfolding $\mathcal{U}_{\mathcal{M}(\mathcal{P})}$. The reason for merging the two places labeled by 174 in $\mathcal{U}_{\mathcal{M}(\mathcal{P})}$ is the diamond occuring in $\mathcal{M}(\mathcal{P})$. But this diamond manifests the concurrency of the two self-repairing transitions, and the unfolding $\mathcal{U}_{\mathcal{P}}$ of $\mathcal{P}$, shown in Fig. 2, already handles this properly: the marking 174 is not duplicated in $\mathcal{U}_{\mathcal{P}}$, unlike in $\mathcal{U}_{\mathcal{M}(\mathcal{P})}$. In fact, this lattice corresponds to a prefix of the unfolding shown in Fig. 2. The unfolding of Fig. 2 is more compact, but in turn, building co-sets requires some processing, whereas this requires no processing for the unfolding of Fig. 8, since co-sets are just places. Therefore, for applications in which memory constraints prevail over processing speed, unfoldings should be preferred.

The generalization of lattices to Petri nets is of interest, and their definition and use for diagnosis will be investigated in the extended version of this paper.

4 Discussion

A net unfolding approach to on-line asynchronous diagnosis was presented. This true concurrency approach is suitable to distributed systems in which no global state and no global time is available, and therefore a partial order model of time is considered. In the present paper, our basic tool was the net *unfolding,* a branching structure suitable to represent the set of configurations of a Petri net using an asynchronous semantics with local states and partially ordered time. Diagnosis nets were introduced as a way to encode all solutions of a diagnosis problem. They avoid the wellknown state explosion problem, that typically results from having concurrent components in a distributed system interacting asynchronously. Whereas state explosion is kept under control, the computing cost of performing the diagnosis on-line increases (due to the need to compute co-sets); but this is typically a preferred tradeoff for the diagnosis of complex asynchronous systems with concurrency.

It is worth saying what this paper does *not* consider. We do not follow a diagnoser approach. One can view a diagnoser as a "compiled" algorithm for diagnosis. It consists in pre-computing a finite state machine which accepts alarm events, and has states labeled by, e.g., visited faults. In contrast, our approach can be seen as an "interpreted" one, since our diagnosis nets are computed, on-line, by using only the original Petri net structure. Also, we did not investigate issues of diagnosability. Diagnosers for unbounded asychronous diagnosis and related diagnosability issues have not been considered in the literature, at least to our knowledge. We believe this could be performed by using so-called *complete prefixes* of the unfolding, see [15][16].

Complexity issues have not been addressed. However, the following pragmatic argument can be given to justify the use of unfoldings. Complete prefixes of unfoldings have been used for model checking, an area in which practical complexity is of paramount importance [29][14][15][16]; in particular, the more parallelism there is in the applicatio, the more is gained from the partial order representation.

Various extensions of this work are under progress. Our target application—fault management in telecommunications networks—typically exhibits a great deal of ambiguity. Hence it is of interest to return (the) most likely explanation(s). Probabilistic versions of the present work have been developed for this purpose, see [2][3][20], and [5] for a theory of corresponding stochastic processes.

Then, this study is clearly an intermediate step toward *distributed* diagnosis, in which diagnosis is perfomed jointly by a network of supervisors communicating asynchronously—see [17] for a first attempt toward distributed diagnosis, and also [19][18].

The robustness of our algorithms against alarm losses, or more generally the failure to communicate, needs to be investigated. Also, due to the systems complexity, there is little hope indeed, that an exact model can be provided, hence we need to develop diagnosis methods that work based on an incomplete model, i.e., a model not able to explain all observed behaviours.

Last but not least, getting the system model itself is a bottleneck, for complex distributed systems such as, e.g., telecommunications network management systems. The issue of how to partially automatize the model construction is investigated in [28].

References

[1] Extended version of this paper
http://www.irisa.fr/sigma2/benveniste/pub/B_al_asdiag_2001.html

[2] 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.

[3] A. Aghasaryan, E. Fabre, A. Benveniste, R. Boubour, C. Jard. Fault Detection and Diagnosis in Distributed Systems : an Approach by Partially Stochastic Petri nets, *Discrete Event Dynamic Systems: theory and application,* special issue on Hybrid Systems, vol. 8, pp. 203-231, June 98.

[4] P. Baroni, G. Lamperti, P. Pogliano, and M. Zanella. Diagnosis of large active systems. *Artificial Intelligence* 110: 135-183, 1999.

[5] A. Benveniste, E. Fabre, and S. Haar. "Markov nets: probabilistic models for distributed and concurrent systems". Irisa Research Report 1415, September 2001, submitted for publication, http://www.irisa.fr/sigma2/benveniste/pub/BlGFH2000.html

[6] 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.

[7] A.T. Bouloutas, G. Hart, and M. Schwartz. Two extensions of the Viterbi algorithm. *IEEE Trans. on Information Theory,* 37(2):430–436, March 1991.

[8] A.T. Bouloutas, S. Calo, and A. Finkel. Alarm correlation and fault identification in communication networks. *IEEE Trans. on Communications,* 42(2/3/4), 1994.

[9] C. Cassandras and S. Lafortune. *Introduction to discrete event systems.* Kluwer Academic Publishers, 1999.

[10] R. Debouk, S. Lafortune, and D. Teneketzis. Coordinated decentralized protocols for failure diagnosis of discrete event systems. *Discrete Event Dynamic Systems: theory and application.* 10(1/2), 33-86, 2000.

[11] R. Debouk, S. Lafortune, and D. Teneketzis. On the effect of communication delays in failure diagnosis of decentralized discrete event systems. Control group report CGR00-04, Univ. of Michigan at Ann Arbor, submitted for publication, 2001.

[12] J. Desel, and J. Esparza. *Free Choice Petri Nets.* Cambridge University Press, 1995.

[13] J. Engelfriet. *Branching Processes of Petri Nets.* Acta Informatica 28, 1991, pp 575–591.

[14] J. Esparza. Model Checking Using Net Unfoldings. *Sci. of Comp. Prog.* **23**:151–195, 1994.

[15] J. Esparza, S. Römer, and W. Vogler. An improvement of McMillan's unfolding algorithm. In T. Margaria and B. Steffen Eds., *Proc. of TACACS'96,* LNCS 1055, 87-106, 1996. Extended version to appear in *Formal Methods in System Design,* 2000.

[16] J. Esparza, and S. Römer. An unfolding algorithm for synchronous products of transition systems, in *proceedings of CONCUR'99,* LNCS 1664, Springer Verlag, 1999.

[17] E. Fabre, A. Benveniste, C. Jard, L. Ricker, and M. Smith. Distributed state reconstruction for discrete event systems. Proc. of the *2000 IEEE Control and Decision Conference (CDC'2000),* Sydney, Dec. 2000.

[18] E. Fabre, A. Benveniste, C. Jard, and M. Smith. Diagnosis of distributed discrete event systems, a net unfolding approach. Preprint, February 2001. http://www.irisa.fr/sigma2/benveniste/pub/F&al2001.html

[19] E. Fabre, A. Benveniste, C. Jard. Distributed diagnosis for large discrete event dynamic systems. In *Proc of the IFAC congress,* Jul. 2002.

[20] E. Fabre, A. Benveniste, C. Jard. Distributed diagnosis for bayesian networks of dynamic systems, *in preparation.*

[21] R.G. Gardner, and D. Harle. Methods and systems for alarm correlation. In *GlobeCom 96,* London, November 1996.

[22] C.N. Hadjicostis, and G.C. Verghese. Monitoring discrete event systems using Petri net embeddings. in *Proc. of Application and theory of Petri nets 1999,* 188–208.

[23] A. Giua. —PN state estimators based on event observation. *Proc. 36th Int. Conf. on Decision and Control,* San Diego, Ca, 4-86–4091, 1997.

[24] A. Giua, and C. Seatzu. Observability of Place/Transition Nets. Preprint, 2001.

[25] K.X. He and M.D. Lemmon. Liveness verification of discrete-event systems modeled by n-safe Petri nets. in *Proc. of the 21st Int. Conf. on Application and Theory of Petri Nets,* Danmark, June 2000.

[26] K.X. He and M.D. Lemmon. On the existence of liveness-enforcing supervisory policies of discrete-event systems modeled by n-safe Petri nets. in *Proc. of IFAC'2000 Conf. on Control Systems Design,* special session on Petri nets, Slovakia, June 2000.

[27] I. Katsela, A.T. Bouloutas, and S. Calo. Centralized vs distributed fault localisation. *Integrated Network Management IV,* A.S. Sethi, Y. Raynaud, and F. Faure-Vincent, Eds. Chapman and Hall, 251-261, 1995.

[28] MAGDA project. Paper on modeling, to be available on the MAGDA project web page http://magda.elibel.tm.fr/

[29] K. McMillan. Using Unfoldings to avoid the state explosion problem in the verification of asynchronous circuits. In: *4th Workshop on Computer Aided Verification*, pp. 164–174, 1992.

[30] M. Nielsen, G. Plotkin, and G. Winskel. Petri nets, event structures, and domains. Part I. *Theoretical Computer Science* **13**:85–108, 1981.

[31] W. Reisig. *Petri nets.* Springer Verlag, 1985.

[32] G. Rozenberg and J. Engelfriet. Elementary Net Systems. In: *Lectures on Petri Nets I: Basic Models.* LNCS **1491**, pp. 12–121, Springer, 1998.

[33] A. Sahraoui, H. Atabakhche, M. Courvoisier, and R. Valette. Joining Petri nets and knowledge-based systems for monitoring purposes. *Proc. of the IEEE Int. Conf. on Robotics Automation,* 1160–1165, 1987.

[34] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis. Diagnosability of discrete-event systems. *IEEE Trans. Autom. Control* 40(9), 1555-1575, 1995.

[35] R. Sengupta. Diagnosis and communications in distributed systems. In *Proc. of* WODES *1998, international Workshop On Discrete Event Systems,* 144-151, IEE, London, England, 1998.

[36] S. Tripakis. Undecidable problems of decentralized observation and control. In *Proc. of the 40th IEEE Conf. on Decision and Control,* Orlando, Dec. 2001.

[37] G. Winskel. Event structures. In *Advances in Petri nets,* LNCS vol. 255, 325–392, Springer Verlag, 1987.