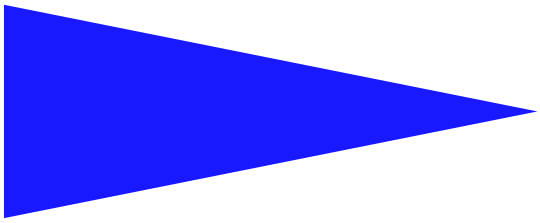
I R I S A
INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTÈMES ALÉATOIRES

PUBLICATION
INTERNE
N° ???



DIAGNOSIS OF DISTRIBUTED DISCRETE EVENT SYSTEMS, A NET UNFOLDING APPROACH

ERIC FABRE, ALBERT BENVENISTE, CLAUDE JARD,
MARK SMITH

ISSN 1166-8687

IRISA
CAMPUS UNIVERSITAIRE DE BEAULIEU - 35042 RENNES CEDEX - FRANCE

IRISA

INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTÈMES ALÉATOIRES
Campus de Beaulieu – 35042 Rennes Cedex – France
Tél. : (33) 02 99 84 71 00 – Fax : (33) 02 99 84 71 71
http://www.irisa.fr

Diagnosis of distributed discrete event systems, a net unfolding approach *

Eric Fabre, Albert Benveniste, Claude Jard** , Mark Smith ***

Thème 4 — Simulation et optimisation
de systèmes complexes
Projets Sigma2, Pampa

Publication interne n ° ??? — Février 2001 — 53 pages

Abstract: In this paper we formulate distributed diagnosis by means of hidden state history reconstruction, from alarm observations. 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, in which local events are ordered if they are either generated on the same site, or related via some causality relation. 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.

Key-words: Distributed diagnosis, discrete event systems, Petri nets, unfoldings, alarm correlation.

(Résumé : tsvp)

* This work is 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. E.F., A.B. are with Inria, C.J. is with CNRS. Corresponding author Albert.Benveniste@irisa.fr
*** Bell Laboratories, 600 Mountain Avenue, Murray Hill, NJ 07974, USA. M.S. participated to this work when visiting IRISA.

Centre National de la Recherche Scientifique
(UPRESSA 6074) Université de Rennes 1 – Insa de Rennes

Institut National de Recherche en Informatique
et en Automatique – unité de recherche de Rennes

Diagnostic de systèmes discrets répartis, une approche par dépliages

Résumé : Dans cet article nous étudions le problème du diagnostic pour des systèmes répartis. Ce problème est formulé comme un problème de reconstruction de trajactoire d'état à partir des alarmes observées. Nous adoptons un point de vue dit de la "concurrence vraie", ce qui signifie que nous ne manipulons jamais d'états globaux, et que nous utilisons un temps qui a la structure d'un ordre partiel. Notre outil principal est le concept de *dépliage* de réseau de Petri. Nous étudions un certain nombre de variantes de ce problème. Cette étude est motivée par le cas de la corrélation d'alarmes en gestion de réseaux.

Mots clés : Diagnostic réparti, systèmes discrets, réseaux de Petri, dépliage, corrélation d'alarmes.

Contents

1 Introduction **4**

2 Tiles and systems **8**
2.1 Tiles 9
2.2 Systems and their parallel composition 9
2.3 From systems to Petri nets 10

3 Some background on Petri Nets and their unfoldings **11**
3.1 Petri Nets and their synchronous products 11
3.2 Representing the runs of a Petri net via unfoldings 12

4 Diagnosis in the framework of Petri Nets **14**
4.1 Problem statement 15
4.2 Diagnosis nets 16

5 Distributed diagnosis **20**
5.1 Problem statement 20
5.2 Architecture of distributed diagnosis 21

6 Algorithms **27**
6.1 Centralized diagnosis 28
6.2 Centralized diagnosis, an optimized version 28
6.3 Distributed diagnosis, the key part 31
6.4 Implementing distributed diagnosis 32

7 Some useful extensions **34**
7.1 Dynamic instanciation of transitions 34
7.2 Loss of alarms 35
7.3 Failure to communicate 37
7.4 Incomplete modelling 38

8 Discussion **38**

A Appendix: implementation details **40**
A.1 Centralized on-line diagnosis of alarm patterns, for a totally ordered alarm pattern 40
A.2 Distributed on-line diagnosis of alarm patterns 40

B Figures illustrating the different notions **44**

1 Introduction

The diagnosis of large, distributed systems is a task of growing importance today. For instance, the increasing complexity and openness of telecommunication networks makes network diagnosis one of the challenging tasks in network management today — network diagnosis is called *event correlation* in this area.

Related work. Fault diagnosis in discrete event systems has attracted a significant attention. We refer the reader to [1][2] for an overview of the litterature and introduction to the subject.

Decentralised diagnosis has been identified as a useful approach for the diagnosis of complex systems, in which alarms are sensed within different components of the system. Decentralised diagnosis has been considered by several authors. A thorough study is performed in [2], including both algorithms and their diagnosability properties. In this reference, the notion of a diagnoser is used, meaning that the solution is formulated in terms of a set of communicating machines that have their states labelled by sets of faults, and react to alarm observations and communications. Also, the language oriented framework of Wonham and Ramadge is used [3], and the systems architecture is that of communicating automata, with a "synchronous" communication based on a global time, as revealed by the assumption "A6" therein. In [4], a different approach is proposed, more in the form of a simulation guided by the observed alarms, again for a model of communicating automata. The solution proposed offers a first attempt to handle the problem of state explosion due to the interleaving of the events involving the different components.

Event correlation in network management is probably the main example of a complex system subject to diagnosis. This area is the subject of a vast litterature, and a number of commercial products are available. We refer the reader to [5] for a survey. There are two main frameworks for most of the 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 in some way or another. The articles [6][7][8] belongs to this family. The authors formulate diagnosis as an optimisation problem, this has some relation with the present approach. The case of event correlation in network management also motivated the series of papers [9][10][11], on which the present paper relies.

Diagnosis architectures. In this paper, motivated by the example of telecommunication network diagnosis and management, we focus on diagnosis algorithms that are suitable to truly distributed, asynchronous, systems. Figure 1 illustrates our purpose by showing three samples of architectures, ordered from left to right and from top to bottom. The first architecture is centralized: one system is globally monitored by one sensor (or, equivalently, one set of synchronized sensors), and one global supervisor performs diagnosis. In the second architecture, the system is distributed. It has several local sensors, each of them having only a partial view of the overall system, the sensors are not synchronized and a global supervisor

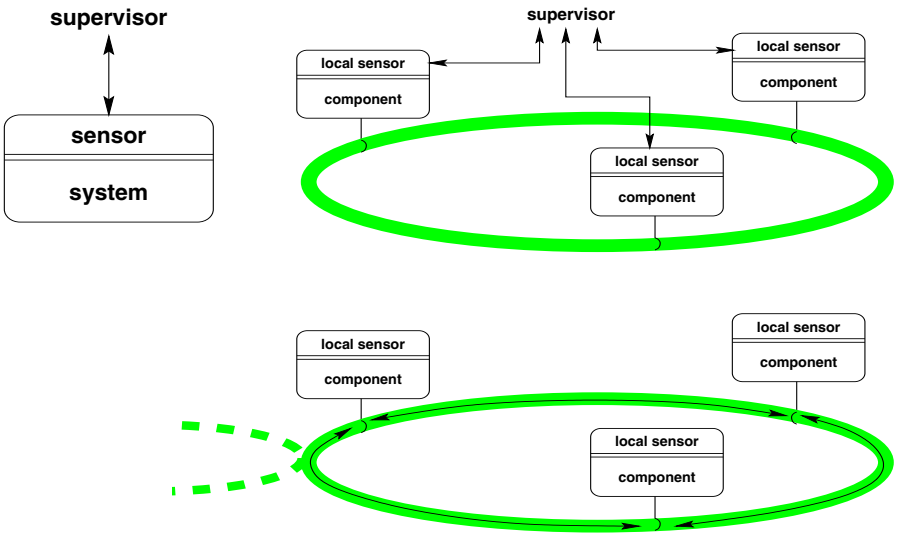


Figure 1: Diagnosis architectures: centralized, decentralized, and distributed.

performs diagnosis — this is the typical architecture today in telecommunication systems. In the third architecture, the system and its diagnosis are both distributed, and there is no overall supervisor, hence the local sensors cooperate to build a coherent view of the overall system. In addition, the system may be open, i.e., it interacts with some partially unknown environment — typically, another telecommunication network, shown in dashed in the figure. In this case, only incomplete knowledge of the system is available. Finally (this is not shown in the picture) the system itself may be dynamic, meaning that its behaviour changes dynamically, due to reconfigurations. In this paper we focus on the third case.

Modelling approaches. In this paper we follow an approach based on partial order models of time and making no use of a global notion of state. This approach was introduced in [9][10][11] and is motivated by the figures 2 and 3. Figure 2 depicts a typical fault propagation in a SDH/SONET network management system. The different boxes are network elements, and the different links correspond to the different layers in the SDH/SONET hierarchy. The fault shown is a fault in STM1 port, resulting in the appearance of a symmetric fault at the distant extremity of the link, both faults propagating upward the hierarchy. The architecture shown for the management system has a distributed, asynchronous, set of sensors, but a central supervisor which collects all alarms (there are many of them!). The resulting chronogram of faults and associated emitted alarms is depicted in figure 3, first diagram. The four network elements are shown. In each element, the different lines figure the different layers in the hierarchy. Finally, the observed part of this pattern is shown in figure 3, second diagram. Note that, while events collected on the same sensor

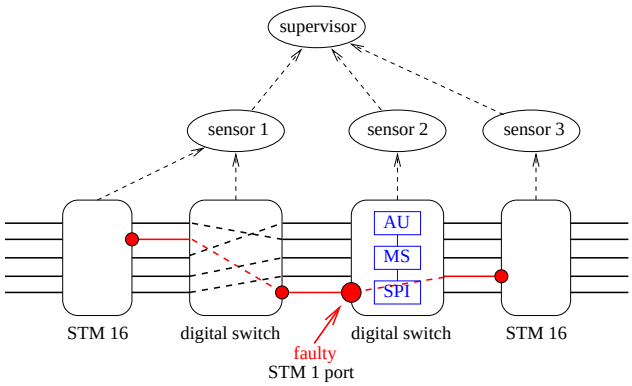


Figure 2: The example of SDH/SONET networks.

are ordered in time, this is not the case for events collected on different sensors. Those events are mutually ordered only when some causality relationship exists, for instance some fault is propagated, or some alarm is broadcast through the network, resulting in a causality relation between its emission and corresponding reception. From the above discussion, the following requirements for our modelling approach emerge, namely :

- a partial order model of time shall be used, time shall capture 1/ the local time at each local sensor, and possibly, 2/ causal relations between the distributed events ;
- no global state shall be utilized.

Such an approach was introduced and discussed in details in [11]. But the so-called "Viterbi puzzle" introduced in the latter reference was not completely specified. In particular, the data structure needed to keep track of *all* solutions to the diagnosis problem was not described, only the way *one* solution should be built was discussed.

In this paper we solve the whole problem and provide one possible data structure for the case in which solutions to the diagnosis problem are built *on-line* while alarms are collected (clearly, this is not the only relevant approach, off-line algorithms are also relevant, see [12]). Paper [13] proposes an approach based on synchronizing automata. States for each local automaton are handled in an enumerated way, but the global, product automaton is never built and the local diagnosers work asynchronously in cooperation toward building the overall diagnosis. In this paper we adopt a net unfolding approach [14][15][16]. Net unfoldings were introduced by D. McMillan to allow reachability analyses in an efficient way for Petri nets [14]. They were generalized to other models of concurrency in [16], including synchronized automata (called "synchronous products of transition systems" in the above reference). Net unfoldings are not wellknown in the control community, they have been used for supervisory control in [17][18]. Finally, net unfoldings were recognized a useful structure for proof systems dealing with products of automata with enumerated states [14]. Net unfoldings support useful concepts such as

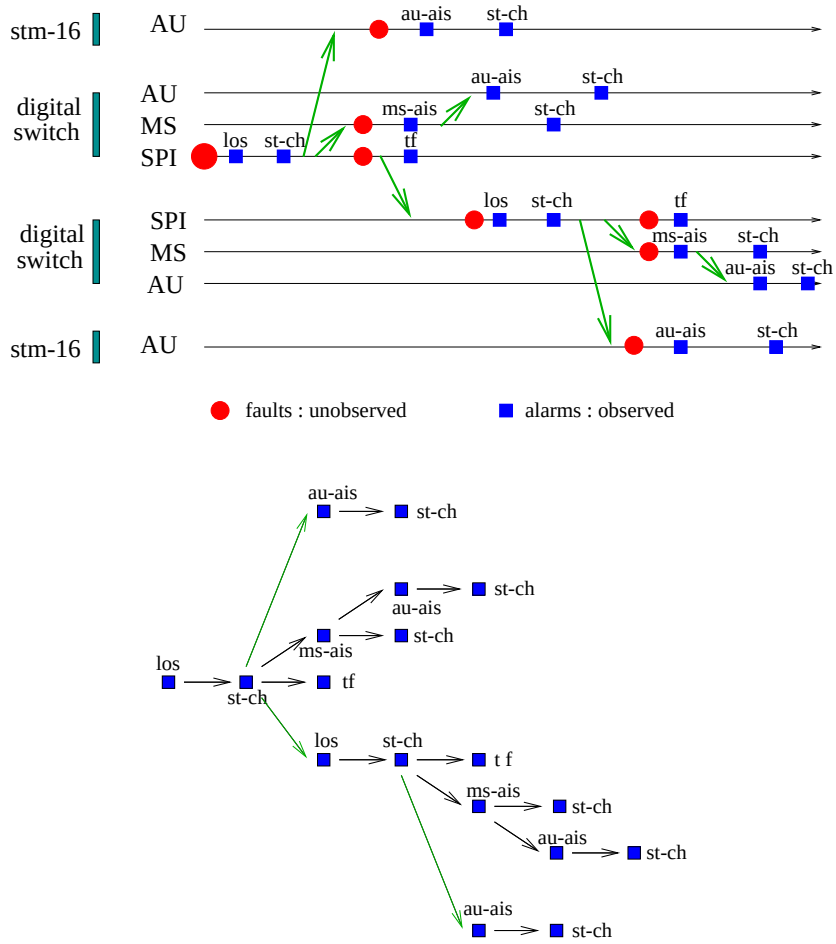


Figure 3: A chronogram of fault and alarm propagation, corresponding observed part.

- *causality,* resulting in partial orders of events ;
- *concurrency,* in order to allow considering local states only, never global states ;
- *conflict,* to model branching histories, i.e., different histories sharing some prefix ; hence conflict relations will allow us to handle the set of all solutions to the diagnosis problem in an efficient way.

From systems to their models. When dealing with model based diagnosis, an underlying model is always assumed to be given. Unfortunately, not paying attention to the

feasibility of this task may result in a non practical solution. Clearly, getting an appropriate model for the fault and alarm propagation in SDH/SONET telecommunications network is a formidable challenge by itself. The available information typically has the following form :

- Components of the system and their relations are described, in the form of an object oriented model involving classes and their generic inheritance and other relations, and associated diagrams involving deployed instances. No precise behaviour, but only architectures are described at this point.
- Various information regarding components may be available in the form of state transition diagrams showing the behaviour of generic components, at some level of abstraction.
- Information regarding the interaction between components are available in the form of scenarios involving the exchanged messages, and so are typical fault and alarms propagation informations.

Actual instanciation of the classes would then result in some instanciation of the generic components, together with their interconnections, and this would yield the desired model. The important point is that such a construction can be partially automatized, once the classes have been defined. As for describing the behaviour of the different objects, we assume that a *transition relation* is attached to each class as an additional method. This transition relation relates a finite set of previous input and state variables to the corresponding set of current output and state variables of each considered object. State variables of these objects can be either local or shared. Since objects have knowledge of only part of the variables, these transition relations relate *partial* states. Such partial transition relations are called *tiles* in this paper, to refer to puzzle games, a metaphor already used in [11].

Organization of the paper. The paper is organized as follows. Our basic model of tiles and systems is introduced in section 2, and we lift this model to a Petri net framework. Petri net and their unfoldings are presented in section 3. Diagnosis is discussed in sections 4 and 5, which constitute the core of this paper. It is formulated as the reconstruction of an unfolding from alarm observations, this unfolding encodes all solutions to our diagnosis problem. Centralized diagnosis is discussed in section 4, and distributed diagnosis is discussed in section 5. Corresponding algorithms are given in section 6, with additional details in appendix A. Useful extensions are considered in section 7, this involves the case in which some alarms are lost and the distributed system may fail to communicate, the case of dynamic instanciation of tiles, and the case of incomplete modelling. Finally we draw some conclusions and perspectives.

2 Tiles and systems

In this section we introduce our mathematical framework. Tiles correspond to partial transitions and systems are defined as a collection of tiles.

2.1 Tiles

We assume a vocabulary $\mathcal{V}$ which is a set of typed variables. All types are assumed finite. States (or partial states) will be described as (partial) valuations of the variables. Formally, we define a *state* x to be a type-consistent interpretation of $\mathcal{V}$, assigning to the set of all variables, a value for it over its domain. We denote by X the set of all states. For a subset of variables $V \subseteq \mathcal{V}$, we define a V-state, or simply partial state, to be a type-consistent interpretation of V. Thus a V-state x assigns to the set V a value x_V for it over its domain. Also, for $v \in V$ a variable, we denote by $v(x)$ its interpretation by state x, and by $v(x_V)$ its interpretation by V-state x_V. Note that $v(x_V) = v(x) = x_v$.

We shall consider partial transitions relating partial states, very much in the same way transitions relate states in standard automata. These partial transitions will be referred to as *tiles* in the sequel. Formally, a *tile* is a 4-tuple

$$\tau \quad = \quad \langle V, x_V^-, \alpha, x_V \rangle \tag{1}$$

where

- $V \subset \mathcal{V}$ is a subset of variables, and
- (x_V^-, α, x_V) is a partial transition, relating the previous V-state x_V^- to the current V-state x_V, and emitting event α where α ranges over some set A of possible event labels. Due to our particular area of interest, these events will be frequently referred to as *alarms* in the sequel.

For τ a tile, we shall sometimes denote by V_τ its set of variables.

2.2 Systems and their parallel composition

A *system* is a triple $\Sigma = \langle V, X_0, \mathcal{T} \rangle$, where

- $V \subset \mathcal{V}$ is a finite set of variables,
- X_0 is a set of initial states, and
- $\mathcal{T}$ is a finite set of (prototype) tiles and $V = \cup_{\tau \in T} V_\tau$.

The interleaved sequence of states and alarms

$$x_0, \alpha_1, x_1, \alpha_2, x_2, \ldots, \alpha_k, x_k, \ldots$$

is a *run* of system Σ if $x_0 \in X_0$ and, for each $k > 0$, there exists $\tau = \langle V_\tau, x_{V_\tau}^-, \alpha, x_{V_\tau} \rangle \in \mathcal{T}$ such that,

$$\begin{aligned} \forall v \in V_\tau \quad &: \quad v(x_{k-1}) = v(x_{V_\tau}^-), \alpha_k = \alpha, v(x_k) = v(x_{V_\tau}), \\ \text{and, } \forall v \notin V_\tau \quad &: \quad v(x_{k-1}) = v(x_k). \end{aligned}$$

Since tiles define partial transitions, it may be the case that two successive tiles of the considered run, say τ and τ', involve disjoint sets of variables. In this case, exchanging τ and τ' in the considered run yields another run. This new run is said to be equivalent to the first one, up to an interleaving. The transitive closure of the so defined relation is an equivalence relation. Since we advocate a local view of state and time, it is advisable not to distinguish runs that are equivalent up to an interleaving. Partial order semantics is the classical answer to this request, we shall present it in the following section, using a Petri net framework.

This being said, centralized diagnosis is formulated as follows: given an alarm sequence $\alpha_1, \alpha_2, \ldots$, reconstruct the set of all runs that have this alarm sequence associated with them.

To develop a framework of distributed diagnosis, we need to formalize what we mean by a distributed system. To this end we equip the set of systems with a parallel composition. For $\Sigma_i = \langle V_i, X_0^i, \mathcal{T}_i \rangle$, $i = 1, 2$ two systems, their parallel composition $\Sigma_1 \parallel \Sigma_2$ is defined as follows:

$$\Sigma_1 \parallel \Sigma_2 \quad \stackrel{\Delta}{=} \quad \langle V_1 \cup V_2, X_0^1 \cap X_0^2, \mathcal{T}_1 \cup \mathcal{T}_2 \rangle \tag{2}$$

Parallel composition is commutative and associative. Distributed diagnosis is formulated as follows. We are given a compound system $\Sigma = \|_{I \in I} \Sigma_i$. Each run of Σ projects onto a local run for each component Σ_i. Each component has its own supervisor, which collects the local alarms of each local run. The different supervisors work concurrently, aiming at constructing, jointly, the set of all runs of Σ which can generate the locally observed alarm sequences.

2.3 From systems to Petri nets

We shall lift our framework into the more classical framework of Petri nets. This will allow us to use corresponding background to develop our approach. At this stage, we assume the reader familiar with Petri nets. If this fails to be the case, or if the reader feels uncomfortable with some notations, she or he is referred to the next section, where all material we need is exposed. We are given a system $\Sigma = \langle V, X_0, \mathcal{T} \rangle$. We shall represent Σ as a labelled Petri net $\mathcal{N}$ defined as follows:

- The places of $\mathcal{N}$ have the form $p = (v, x_v)$, where v ranges over V, and x_v ranges over the domain of v.
- The transitions of $\mathcal{N}$ are just the tiles $\tau \in \mathcal{T}$. For $\tau = \langle V, x_V^-, \alpha, x_V \rangle$ a tile, we associate to it its alarm component α as a label, and we write $\lambda(\tau) = \alpha$.
- Arrow $(v, x_v) \to \tau$ belongs to $\mathcal{N}$ iff $1/\ v \in V_\tau$, and $2/\ v(x_{V_\tau}^-) = x_v$. Similarly, arrow $\tau \to (v', x_{v'})$ belongs to $\mathcal{N}$ iff $1/\ v' \in V_\tau$, and $2/\ v'(x_{V_\tau}) = x_{v'}$.

The so defined net $\mathcal{N}$ is a net of capacity one. This lifting allows us to cast our problem in the framework of Petri nets, which we detail now. The mapping of tiles to transitions is illustrated on figure 4 at the end of this paper.

3 Some background on Petri Nets and their unfoldings

We shall use some basic notions related to Petri Nets and their unfoldings, we shamelessly borrow significant parts of this presentation from [15][16]. For the basic definitions on Petri net, we refer the reader to [19][3]. Unfoldings are a way to encode the reachable markings of a Petri net by means of a special acyclic Petri net called an occurrence net. A key point is that it is enough to regard this occurrence net as a directed graph in order to compute the reachable markings of the original Petri net. We now collect the notions we need on Petri net and occurrence nets.

3.1 Petri Nets and their synchronous products

A *Petri net* consists of a set of *places,* graphically represented by circles and generically denoted by p, a set of *transitions,* graphically represented by bars and generically denoted by t, and a flow relation assigning to each place (transition) a set of input and a set of output transitions (places). The flow relation is graphically represented by arrows leading from places to transitions and from transitions to places. Places and transitions are called *nodes,* generically denoted by n. For n a node, the set of its input and output nodes is denoted by ${}^\bullet n$ and $n^\bullet$, respectively. A place of a net can hold tokens, and a map assigning to each place a number of tokens is called a *marking.* In this paper we consider only Petri nets of *capacity one.* In such Petri nets, places hold zero or one token. If at a given marking all the input places of a transition hold a token and all the places which are outputs and not inputs are empty, then the transition can *occur,* which leads to a new marking obtained by removing one token from each input place and adding one token to each output place. An *occurrence sequence* is a sequence of transitions that can occur in the order specified by the sequence. A Petri net $\mathcal{N} = \{P, P_0, T, \rightarrow\}$ is characterized by its set P of places, its set T of transitions, its flow relation $\rightarrow \subset (P \times T) \cup (T \times P)$, and the subset $P_0 \subseteq P$ which composes the *initial marking* of the net. A *labelling* of Petri net $\mathcal{N}$ is a map $\lambda : T \longmapsto A$, where A is some finite alphabet. A Petri net equipped with a labelling is called a labelled Petri net.

Definition 1 (product of Petri nets) *The* synchronous product $\mathcal{N}_1 \times \mathcal{N}_2$ *of the two labelled Petri nets* $\mathcal{N}_i = \{P_i, P_{0,i}, T_i, \rightarrow_i, \lambda_i\}$, $i = 1, 2$, *is a labelled Petri net defined as follows :*

$$\begin{aligned} \mathcal{N}_1 \times \mathcal{N}_2 \quad \stackrel{\Delta}{=} \quad & \{P, P_0, T, \rightarrow, \lambda\} \text{ , where } P = P_1 \cup P_2,\ P_0 = P_{0,1} \cup P_{0,2} \text{ ,} \\ & \text{and } t \in T \text{ iff :} \end{aligned} \tag{3}$$

case (i) : $\exists t_i \in T_i$ *for some* $i = 1, 2$, *such that*

$$\begin{aligned} \lambda(t) = \lambda_i(t_i) \quad & \in \quad A_i \setminus A_j \text{ , } j \neq i \text{ , and} \\ {}^\bullet t = {}^\bullet t_i \quad & , \quad t^\bullet = t_i^\bullet \end{aligned}$$

case (ii) : *for each* $i = 1, 2$, $\exists t_i \in T_i$ *such that*

$$\begin{aligned} \lambda(t) \quad & = \quad \lambda_1(t_1) = \lambda_2(t_2) \text{ , and} \\ {}^\bullet t = {}^\bullet t_1 \cup {}^\bullet t_2 \quad & , \quad t^\bullet = t_1^\bullet \cup t_2^\bullet \end{aligned}$$

The synchronous product is associative and commutative. In case (i) only one net fires a transition and this transition has a private label, while the two Petri nets synchronise on transitions with identical labels in case (ii). Note that the two Petri nets can have shared places, this deviates from the notion of synchronous product used in [16].

A link with the parallel composition of systems. Let us establish a link between this notion of a synchronous product of Petri nets, and our previous definition (2) for the parallel composition of systems. Consider two systems Σ_1 and Σ_2, and assume they share variables ($V_1 \cap V_2 \neq \emptyset$) but they have distinct sets of alarms, this is the case of interest for practical distributed diagnosis applications. Then corresponding Petri nets $\mathcal{N}_1$ and $\mathcal{N}_2$ have shared places, but distinct labels for their events, and therefore, when considering their synchronous product $\mathcal{N}_1 \times \mathcal{N}_2$, only case (i) applies. As a consequence, in this case, *product net $\mathcal{N}_1 \times \mathcal{N}_2$ is the net associated with the composed system $\Sigma_1 \| \Sigma_2$.*

A link with the diagnosis problem. Consider the case in which net $\mathcal{N}_2$ has its event label set contained in that of net $\mathcal{N}_1$, meaning that events produced by $\mathcal{N}_2$ "could have been" produced by $\mathcal{N}_1$. On the other hand, we assume that $\mathcal{N}_1$ and $\mathcal{N}_2$ have distinct places, say, places of $\mathcal{N}_2$ are dummy and have no particular concrete signification. When computing the synchronous product $\mathcal{N}_1 \times \mathcal{N}_2$, then only case (ii) of synchronizing events applies, and this product will in particular compute those behaviours of $\mathcal{N}_1$ which can "explain" the observed behaviours of $\mathcal{N}_2$. This is very similar to the diagnosis problem we shall investigate later.

3.2 Representing the runs of a Petri net via unfoldings

In this subsection we consider the problem of representing all the runs of a Petri net, using the notion of occurrence net and unfolding.

Occurrence nets – definition, terminology, and notations

Given two nodes n and n' (place or transition) of a Petri net, we say that n *causes* n', written $n \preceq n'$, if either $n' = n$ or there is a path of arrows from n to n'. We say that n and n' are in *conflict,* written $n\#n'$, if there is a place m, different from n and n', from which one can reach n and n', exiting m by different arrows. Finally we say that n and n' are *concurrent,* written $n \perp\!\!\!\perp n'$, if neither $n \preceq n'$, nor $n \preceq n'$, nor $n\#n'$ hold. For n a node, we write $\perp\!\!\!\perp(n)$ to denote the set of nodes that are concurrent with n. An *occurrence net* is a Petri net satisfying the following properties:

1. the net, seen as a directed graph, has no circuit;
2. every place has at most one input transition;
3. no node is in self-conflict, i.e., $n\#n$ does not hold.

Occurrence nets can be infinite. We restrict ourselves to those in which every event has at least one input place, and in which the arrows cannot be followed backward infinitely from any point (this is referred to as *well-foundedness*). It follows that, by following the arrows backward we eventually reach a place without predecessors, these are the *minimal places* of the occurrence net.

To distinguish occurrence nets from other Petri nets, we shall use specific terminology and notations, mostly borrowed from [15]. Occurrence nets are generically denoted by $\mathcal{U}$, their places are called *conditions,* they are generically denoted by b, and the set of conditions is denoted B. Also, their transitions are called *events,* they are generically denoted by e, and the set of events is denoted by E.

A *cut* of an occurrence net is a set of conditions $\mathbf{c}$ satisfying the following two properties: $\mathbf{c}$ is a *co-set* (any two elements of $\mathbf{c}$ are concurrent), and $\mathbf{c}$ is maximal (it is not properly included in any other co-set). A *configuration* is a set of nodes κ satisfying the following two properties: κ is causally closed (if $n \in \kappa$ and $n' \prec n$, then $n' \in \kappa$), conflict-free (no two nodes of κ are in conflict), and, when seen as a set of nodes, a configuration is a union of cuts. Furthermore we require for convenience that all maximal nodes (if any) of configurations shall be conditions. Figure 5 shows an example of a configuration, figure 6 depicts different cuts for the configuration of figure 5, and figure 7 shows an example of an occurrence net.

Unfoldings

For $\mathcal{N}$ a Petri net, its unfolding we introduce now represents the set of all its runs. To define the unfolding, we shall first associate to $\mathcal{N}$ a set of labelled occurrence nets, called the *branching processes* of $\mathcal{N}$. The conditions of these nets are labelled with places of $\mathcal{N}$, their events are labelled with transitions of $\mathcal{N}$, the flow relation of $\mathcal{N}$ is "unfolded" into that of the branching process, and the branching process "starts" at the initial marking of $\mathcal{N}$. All this is formalized next.

Definition 2 (branching process) *A* branching process *of a Petri net* $\mathcal{N} = \{P, P_0, T, \rightarrow\}$ *is a labelled occurrence net* $\mathcal{U} = \{B, E, \rightarrow, \lambda\}$*, where the labelling function* λ*, with domain* $B \cup E$*, satisfies the following properties:*

(i) $\lambda(B) \subseteq P$ *and* $\lambda(E) \subseteq T$*.*

(ii) $\forall e \in E$*, the restriction of* λ *to* ${}^\bullet e$ *is a bijection between* ${}^\bullet e$ *and* ${}^\bullet\lambda(e)$*, and similarly for* $e^\bullet$ *and* $\lambda(e)^\bullet$*.*

(iii) The restriction of λ *to* $min(\mathcal{U})$*, seen as a directed graph, is a bijection between* $min(\mathcal{U})$ *and* P_0*.*

(iv) $\forall e_1, e_2 \in E$*,* ${}^\bullet e_1 = {}^\bullet e_2$ *and* $\lambda(e_1) = \lambda(e_2)$ *together imply* $e_1 = e_2$*.*

Condition *(iv)* expresses that the unfolding of the flow relation is performed without duplication. The set of all branching processes is uniquely defined, up to an isomorphism (i.e.,

a renaming of the conditions and events). Branching processes are partially ordered by inclusion. The union of a finite or infinite set of branching processes of $\mathcal{N}$ is also a branching process of $\mathcal{N}$, where union of branching processes is defined componentwise on places and events. Therefore the union of all branching processes of $\mathcal{N}$ is also a branching process,

$$\text{we call it the } \textbf{\textit{unfolding}} \text{ of } \mathcal{N}, \text{ and denote it by } \quad \mathcal{U}_{\mathcal{N}}\,. \tag{4}$$

This notion is illustrated in figures 9 and 10.

4 Diagnosis in the framework of Petri Nets

We are given the following objects.

- A labelled Petri net $\mathcal{N} = \{P, P_0, T, \rightarrow, \lambda_\alpha\}$, where the range of the labelling map λ_α is the set of possible *alarms,* denoted by A, and
- its unfolding $\mathcal{U}_{\mathcal{N}} = \{B, E, \rightarrow, \lambda\}$, where the different objects have been introduced in subsection 3.2.

Note the following chain of labelling maps:

$$\underbrace{E}_{\text{events}} \xrightarrow{\lambda} \underbrace{T}_{\text{transitions}} \xrightarrow{\lambda_\alpha} \underbrace{A}_{\text{alarms}} \quad : \quad e \longmapsto \lambda(e) \longmapsto \lambda_\alpha(\lambda(e)) \;\triangleq\; \Lambda_\alpha(e)\,, \tag{5}$$

which defines the *alarm label* of event e (or "alarm", for short, when no confusion can occur), we denote it by $\Lambda_\alpha(e)$.

We assume some run of net $\mathcal{N}$, i.e., some configuration κ of its unfolding $\mathcal{U}$. And we assume that we only observe the *alarm events* of configuration κ, meaning that, for event $e \in \kappa$, we only observe $\Lambda_\alpha(e)$. Our task consists in reconstructing all configurations of the unfolding of the considered net, which could give raise to the above mentioned observations. We rephraze this by saying that the desired configurations "explain these observations", and we interpret this as a *diagnosis problem.* To gain an intuition of this, just imagine that we try to recover the diagram on the top of figure 3, from observing the diagram on the bottom of the same figure.

As said before, only alarms are observed, but we also need to specify which information regarding *causalities* between the different alarm events our observation system can capture:

(a) For the simplest design, we assume that we have a single sensor observing sequentially some interleaving $\alpha_1, \alpha_2, \ldots$ ($\alpha_i \in A$), of the alarm events belonging to the considered configuration. We assume that it is not the case that $i' < i$ and α_i caused $\alpha_{i'}$, i.e., the observed interleaving is consistent with the causalities specified by the underlying Petri net.

(b) For a more sophisticated design, we assume that we have a finite set of sensors labelled by $j \in J$. These sensors work concurrently and independently, and sensor j observes

sequentially some interleaving $\alpha_1^j, \alpha_2^j, \ldots$ ($\alpha_i^j \in A$), of the alarm events relevant to its site. Again, we assume that, locally to each site, the observed interleaving is consistent with the underlying causalities.

(c) More generally, we assume that we have some sensoring system able to observe alarm events, partially ordered in a way which does not contradict the way they were causally produced.

Case (a) is typical of a diagnosis using a centralized supervisor, in which alarm events are collected in sequence by respecting causalities — a typical architecture in network management systems today. Case (b) is an extension of (a) to diagnosis via distributed supervision — the preferred architecture for future network management systems in which heterogeneous network management systems will cooperate. Finally, case (c) subsumes case (b) to its important assumptions for our purpose. All this is formalized next.

4.1 Problem statement

To formalize the above discussed partial capture of causalities, we need the notion of *extension* of an occurrence net.

Definition 3 (extension of an occurrence net) *Given a labelled occurrence net, we call an* extension *of it any labelled occurrence net obtained by adding, to this net, conditions and flow relations but not events.*

An occurrence net induces a labelled partial order on the set of its events. Extending this occurrence net according to definition 3 induces an extension of this labelled partial order [1].

Then, we need to formalize what we mean by "observing alarms only". To understand why the following framework is required, the reader is referred to (5) and the related explanations. We consider *alarm labelled* occurrence nets of the form $\mathcal{U} = \{B, E, \rightarrow, \Lambda_\alpha\}$, in which the labelling map $\Lambda_\alpha(e), e \in E$, takes its values in the set A of possible alarms.

Definition 4 (alarm-isomorphic occurrence nets) *Two alarm labelled occurrence nets $\mathcal{U} = \{B, E, \rightarrow, \Lambda_\alpha\}$ and $\mathcal{U}' = \{B', E', \rightarrow', \Lambda'_\alpha\}$ 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 $\forall e \in E : \Lambda'_\alpha(\psi(e)) = \Lambda_\alpha(e)$.*

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 5 (alarm pattern) *Consider a labelled net $\mathcal{N}$ and its unfolding $\mathcal{U}_\mathcal{N}$. A labelled occurrence net $\mathcal{A}$ is called an* alarm pattern of $\mathcal{N}$ *if:*

[1] Recall that the labelled partial order $(X, \preceq)$ is an *extension* of labelled partial order $(X', \preceq')$ if labelled 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 $(X', \preceq')$.

1. *Its labelling map takes its value in alphabet A of alarms,*
2. *$\mathcal{A}$ is itself a configuration (it is conflict free), its set of conditions is disjoint from that of $\mathcal{U}_{\mathcal{N}}$, and*
3. *There exists a configuration κ of $\mathcal{U}_{\mathcal{N}}$ such that $\mathcal{A}$ and κ possess extensions that are alarm-isomorphic.*

Assuming for $\mathcal{A}$ a set of places disjoint from that of $\mathcal{U}_{\mathcal{N}}$ aims at reflecting that alarm patterns vehicle no information regarding hidden states of the original net. This justifies condition 2. Keeping in mind definitions 3 and 4 and the cases (a,b,c) of the discussion before, condition 3 expresses that κ can explain $\mathcal{A}$. For instance: in case (a), $\mathcal{A}$ itself is an extension of κ; in case (b) each local part of $\mathcal{A}$ is an extension of κ, but causalities relating alarm events received by different sensors are lost; case (c) expresses that κ and $\mathcal{A}$ possess compatible partial orders.

For $\mathcal{A}$ a given alarm pattern of $\mathcal{N}$, we denote by

$$\mathsf{diagnosis}\,(\mathcal{A}) \tag{6}$$

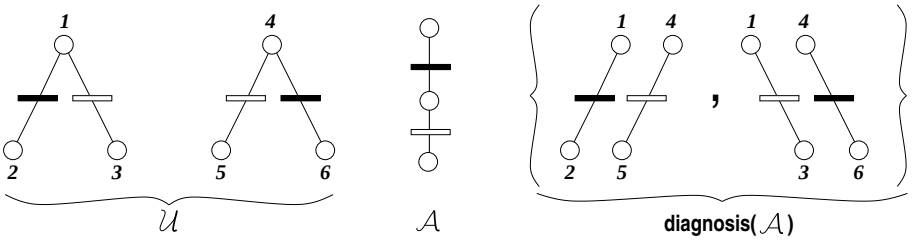
the set of configurations κ of $\mathcal{U}_{\mathcal{N}}$, satisfying the conditions 1,2,3 of definition 5. In the next subsection, we propose an adequate data structure to represent the set $\mathsf{diagnosis}\,(\mathcal{A})$, we call it a *diagnosis net.*

4.2 Diagnosis nets

In occurrence nets, configurations are causally closed and conflict free unions of cuts. Hence configurations are conveniently characterized using causality and conflict relations. Therefore, a first natural idea is to represent $\mathsf{diagnosis}\,(\mathcal{A})$ by

$$\begin{array}{c}\text{the minimal subnet of unfolding } \mathcal{U}_{\mathcal{N}}\text{, containing}\\ \text{all configurations} \in \mathsf{diagnosis}\,(\mathcal{A})\text{; we denote it by } \mathcal{U}_{\mathcal{N}}(\mathcal{A}).\end{array} \tag{7}$$

Subnet $\mathcal{U}_{\mathcal{N}}(\mathcal{A})$ inherits canonically by restriction, of the causality, conflict, and concurrence relations defined on $\mathcal{U}_{\mathcal{N}}$. Net $\mathcal{U}_{\mathcal{N}}(\mathcal{A})$ contains all configurations belonging to $\mathsf{diagnosis}\,(\mathcal{A})$, but unfortunately it also contains undesirable maximal configurations *not* belonging to $\mathsf{diagnosis}\,(\mathcal{A})$, as the following picture shows:



In this picture, 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 $\mathsf{diagnosis}(\mathcal{A})$ is shown on the right hand side, it comprises two configurations. Unfortunately the minimal subnet $\mathcal{U}_{\mathcal{N}}(\mathcal{A})$ of the original unfolding $\mathcal{U}$ which contains $\mathsf{diagnosis}(\mathcal{A})$, 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. The idea is to turn the relation "explain the same alarm event" into a conflict relation, since this would prevent $\{(1,t_{12},2),(4,t_{46},6)\}$ and $\{(1,t_{13},3),(4,t_{45},5)\}$ from being configurations in this case! Let us formalize this idea.

Referring to the definition 5 of alarm patterns, for every $\kappa \in \mathsf{diagnosis}(\mathcal{A})$, denote by ψ_κ the alarm-isomorphism mapping the events of κ onto those of $\mathcal{A}$. For a an event of an alarm pattern $\mathcal{A}$, we denote by $\mathsf{diagnosis}(a)$ the set $\{\psi_\kappa^{-1}(a) : \kappa \in \mathsf{diagnosis}(\mathcal{A})\}$, it is the set of events belonging to $\mathsf{diagnosis}(\mathcal{A})$ which explain a. By abuse of notation, we also denote by $\mathsf{diagnosis}(a)$ the corresponding set of events in subnet $\mathcal{U}_{\mathcal{N}}(\mathcal{A})$. We define the following $\mathcal{A}$-conflict relation on subnet $\mathcal{U}_{\mathcal{N}}(\mathcal{A})$ as follows, we write it $\#_{\mathcal{A}}$:

Definition 6 ($\mathcal{A}$-conflict relation) *Relation $\#_{\mathcal{A}}$ is the weakest relation satisfying the following three conditions:*

1. *If $p\#q$, then $p\#_{\mathcal{A}}q$, i.e., $\#_{\mathcal{A}}$ is stronger than $\#$;*
2. *If $\exists a \in \mathcal{A}$ such that $t \in \mathsf{diagnosis}(a)$ and $t' \in \mathsf{diagnosis}(a)$, then $t\#_{\mathcal{A}}t'$, i.e., sharing an alarm event is a source of conflict;*
3. *If $p\#_{\mathcal{A}}q$ and $p \preceq p'$, $q \preceq q'$, then $p'\#_{\mathcal{A}}q'$, i.e., $\#_{\mathcal{A}}$ is causally closed.*

We equip the subnet $\mathcal{U}_{\mathcal{N}}(\mathcal{A})$ of nodes with the causality relation $\preceq$ inherited from the original unfolding $\mathcal{U}_{\mathcal{N}}$, together with the above defined reinforced conflict relation $\#_{\mathcal{A}}$. Finally, we write $p \perp\!\!\!\perp_{\mathcal{A}} q$ if neither $p \preceq q$, nor $q \preceq p$, nor $p\#_{\mathcal{A}}q$ holds. The following theorem holds:

Theorem 1 *Let $\mathcal{U}_{\mathcal{N}}$ be the unfolding of some Petri net $\mathcal{N}$, $\mathcal{A}$ an associated alarm pattern, and let $\mathsf{diagnosis}(\mathcal{A})$ be defined as in (6). Equip $\mathcal{U}_{\mathcal{N}}(\mathcal{A})$ with the reinforced conflict relation $\#_{\mathcal{A}}$ introduced in definition 6. Then the triple $\mathcal{U}_{\mathcal{N},\mathcal{A}} = \{\mathcal{U}_{\mathcal{N}}(\mathcal{A}),\preceq,\#_{\mathcal{A}}\}$ is an adequate representation of $\mathsf{diagnosis}(\mathcal{A})$, as it possesses exactly $\mathsf{diagnosis}(\mathcal{A})$ as its set of maximal configurations, we write this*

$$\mathsf{diagnosis}(\mathcal{A}) \quad \sim \quad \mathcal{U}_{\mathcal{N},\mathcal{A}}$$

Proof: it is organized into several steps.

1. *Every $\kappa \in \mathsf{diagnosis}(\mathcal{A})$ is a configuration of $\mathcal{U}_{\mathcal{N},\mathcal{A}}$.*

   By definition of $\mathsf{diagnosis}(\mathcal{A})$, no two nodes of κ explain the same alarm event of $\mathcal{A}$. On the other hand, κ, being a configuration of $\mathcal{U}_{\mathcal{N}}(\mathcal{A})$ equipped with its original conflict

relation $\#$, is causally closed. Hence no two nodes of κ are causally related to nodes explaining the same alarm event $a \in \mathcal{A}$. Since κ is already a configuration of $\mathcal{U}_{\mathcal{N}}(\mathcal{A})$ equipped with its original conflict relation $\#$, conditions 2 and 3 of definition 6 imply that κ is a configuration of $\mathcal{U}_{\mathcal{N},\mathcal{A}}$.

2. Every $\kappa \in \mathsf{diagnosis}\,(\mathcal{A})$ is a *maximal* configuration of $\mathcal{U}_{\mathcal{N},\mathcal{A}}$.

   Assume this is not true, then there exists some $\kappa \in \mathsf{diagnosis}\,(\mathcal{A})$ which is not a maximal configuration of $\mathcal{U}_{\mathcal{N},\mathcal{A}}$. Hence $\kappa \subset \kappa', \kappa \neq \kappa'$, for some maximal configuration $\kappa' \in \mathcal{U}_{\mathcal{N},\mathcal{A}}$. Then two cases can occur. Either the suffix $\kappa' \setminus \kappa$ is not a subset of any element of $\mathsf{diagnosis}\,(\mathcal{A})$, hence it can be removed from $\mathcal{U}_{\mathcal{N}}(\mathcal{A})$, therefore contradicting the minimality of $\mathcal{U}_{\mathcal{N}}(\mathcal{A})$. Or some node of $\kappa' \setminus \kappa$ belongs to some element of $\mathsf{diagnosis}\,(\mathcal{A})$, but then it explains some alarm event of $\mathcal{A}$, hence it must be in conflict with the node of κ explaining the same alarm event of $\mathcal{A}$, this contradicts the fact that κ' is a configuration of $\mathcal{U}_{\mathcal{N},\mathcal{A}}$, equipped with the reinforced conflict relation $\#_{\mathcal{A}}$.

3. *Every maximal configuration of* $\mathcal{U}_{\mathcal{N},\mathcal{A}}$ *is an element of* $\mathsf{diagnosis}\,(\mathcal{A})$.

   Assume there exists some maximal configuration κ of $\mathcal{U}_{\mathcal{N},\mathcal{A}}$ which does not belong to $\mathsf{diagnosis}\,(\mathcal{A})$. We have already seen that κ cannot contain an element of $\mathsf{diagnosis}\,(\mathcal{A})$. Then the following cases can occur. Either κ has some suffix which has empty intersection with every element of $\mathsf{diagnosis}\,(\mathcal{A})$, hence it can be removed from $\mathcal{U}_{\mathcal{N}}(\mathcal{A})$, therefore contradicting again the minimality of $\mathcal{U}_{\mathcal{N}}(\mathcal{A})$. Or κ is a union of subconfigurations of elements of $\mathsf{diagnosis}\,(\mathcal{A})$. Then either two different nodes of κ explain the same alarm event, and this prevents κ from being a configuration with respect to the enhanced conflict relation $\#_{\mathcal{A}}$, or no such two nodes exist, and then κ must be a strict subconfiguration of some element of $\mathsf{diagnosis}\,(\mathcal{A})$, hence it is not a maximal configuration of $\mathcal{U}_{\mathcal{N},\mathcal{A}}$. ◇

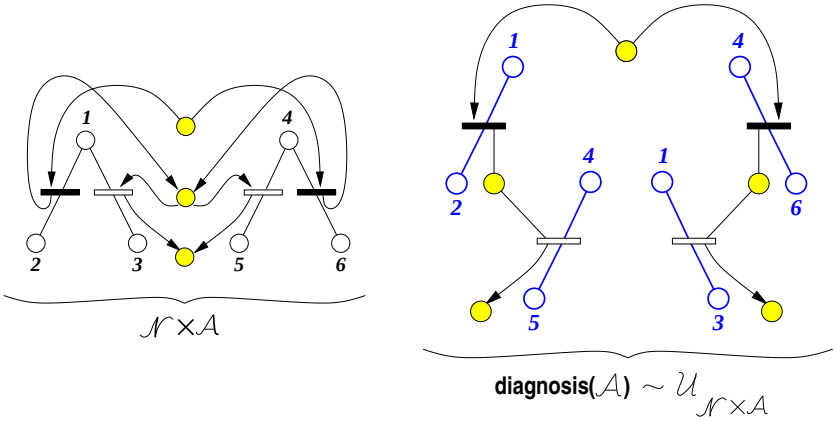
The structure $\mathcal{U}_{\mathcal{N},\mathcal{A}}$ introduced in theorem 1 is not a proper occurrence net: its conflict relation is not inherited from the topology of the underlying bipartite graph, it is rather an extension of the conflict relation associated to $\mathcal{U}_{\mathcal{N}}(\mathcal{A})$ seen as a bipartite graph. But the incremental construction of an unfolding requires anyway to maintain explicitly the concurrence/conflict relations by appropriate pointers, hence there is indeed no additional burden in computing this special data structure.

However, structure $\mathcal{U}_{\mathcal{N},\mathcal{A}}$ lacks algebraic properties, hence it is difficult reasoning with it for the case of distributed diagnosis we discuss hereafter in subsection 5.1. Therefore we shall introduce a less compact, but more elegant representation of $\mathsf{diagnosis}\,(\mathcal{A})$, amenable of algebraic manipulations that will be instrumental in handling the distributed case.

Theorem 2 *Let* $\mathcal{N}$, $\mathcal{U}_{\mathcal{N}}$, $\mathcal{A}$, *and* $\mathsf{diagnosis}\,(\mathcal{A})$ *be as in theorem 1, and consider the unfolding* $\mathcal{U}_{\mathcal{N}\times\mathcal{A}}$. *Then erasing, in the set of all configurations of* $\mathcal{U}_{\mathcal{N}\times\mathcal{A}}$, *the conditions labelled with nodes from* $\mathcal{A}$, *yields the set* $\cup_{\mathcal{A}' \preceq \mathcal{A}}\ \mathsf{diagnosis}\,(\mathcal{A}')$, *where* $\mathcal{A}'$ *ranges over the set of the prefixes of* $\mathcal{A}$. *Hence* $\mathcal{U}_{\mathcal{N}\times\mathcal{A}}$ *is an adequate representation of* $\cup_{\mathcal{A}'}\ \mathsf{diagnosis}\,(\mathcal{A}')$, *written*

$$\bigcup_{\mathcal{A}' \preceq \mathcal{A}} \mathsf{diagnosis}\,(\mathcal{A}') \quad \sim \quad \mathcal{U}_{\mathcal{N}\times\mathcal{A}}\,.$$

This theorem is illustrated in the following picture, which continues our running example. The reader should compare this figure with the preceding one, and note that the restriction, to the nodes of $\mathcal{N}$, of unfolding $\mathcal{U}_{\mathcal{N}\times\mathcal{A}}$, has exactly $\mathsf{diagnosis}\,(\mathcal{A})$ as its set of configurations :



Proof of theorem 2 : it relies on the following claims.

1. *Let κ be a configuration of $\mathcal{U}_{\mathcal{N}\times\mathcal{A}}$; then erasing, in κ, the conditions labelled by nodes from $\mathcal{A}$, yields an element of* $\mathsf{diagnosis}\,(\mathcal{A}')$*, where $\mathcal{A}'$ is obtained by erasing, in κ, the conditions labelled by nodes from $\mathcal{N}$.*

   To prove this claim, we use in detail the definition 1 of the product of Petri nets, and the definition 2 of branching processes. Since $\mathcal{N}$ and $\mathcal{A}$ possess the same labelling alphabet A, only case (ii) of definition 1 applies, i.e., both $\mathcal{N}$ and $\mathcal{A}$ must synchronize at each transition of their product $\mathcal{N}\times\mathcal{A}$. Using this remark and conditions (iii) and (ii) of definition 2, erasing, in κ, the conditions labelled by places from $\mathcal{N}$, yields a configuration of $\mathcal{U}_{\mathcal{A}} = \mathcal{A}$ (we use the obvious fact that the unfolding of an occurrence net is this net itself). But the configurations of $\mathcal{A}$ are just its prefixes, hence the so obtained configuration is some prefix, $\mathcal{A}'$, of $\mathcal{A}$. Using again the fact that all transitions of $\mathcal{N}\times\mathcal{A}$ are synchronizing transitions, erasing, in κ, the conditions labelled by conditions from $\mathcal{A}$, yields a configuration of $\mathcal{U}_{\mathcal{N}}$. By definition 5 of alarm patterns, this configuration is an element of $\mathsf{diagnosis}\,(\mathcal{A}')$.

2. *For $\mathcal{A}'$ an arbitrary prefix of $\mathcal{A}$, any element of* $\mathsf{diagnosis}\,(\mathcal{A}')$ *is obtained by erasing, in some configuration κ of $\mathcal{U}_{\mathcal{N}\times\mathcal{A}}$, the conditions that are labelled by nodes from $\mathcal{A}'$.*

   It is enough to prove the above claim for the special case $\mathcal{A}' = \mathcal{A}$. By theorem 1, we know that $\mathsf{diagnosis}\,(\mathcal{A})$ coincides with the set of maximal configurations of the structure $\mathcal{U}_{\mathcal{N},\mathcal{A}}$. From the definition 6 of enhanced conflict relation $\#_{\mathcal{A}}$, we immediately see that any maximal configuration of $\mathcal{U}_{\mathcal{N},\mathcal{A}}$ is obtained by erasing, in some configuration κ of $\mathcal{U}_{\mathcal{N}\times\mathcal{A}}$, the conditions that are labelled by nodes from $\mathcal{A}'$. This proves the claim, and finishes the proof of the theorem. ◇

The data structures introduced in theorems 1,2 are generically called ***diagnosis nets*** in the sequel. We are now ready to investigate the distributed case.

5 Distributed diagnosis

5.1 Problem statement

Here we assume several systems running concurrently, and sharing some of their hidden state variables. Each system would have its own local sensoring system. Different sensoring systems are concurrent, and share no information. Now, we cast again distributed diagnosis in our Petri net framework.

Throughout this section we consider a product net of the form

$$\mathcal{N} \quad = \quad \prod_{i \in I} \mathcal{N}_i \,, \; I \text{ a set.}$$

where $\prod$ refers to the synchronous product of nets. Due to subsections 2.2, 2.3, and the remark at the end of subsection 3.1, we assume the following:

$$\begin{array}{l} \text{labelled nets } \mathcal{N}_i \text{ have pairwise distinct label sets for their events,} \\ \text{but they can share places.} \end{array} \tag{8}$$

We need to generalize the problem statement of subsection 4.1 to the distributed case.

Definition 7 (distributed alarm pattern) *Given the family $\mathcal{N}_i, i \in I$, a* distributed alarm pattern *for $\prod_{i \in I} \mathcal{N}_i$ consists of a family $\mathcal{A}_i, i \in I$ of nets having the following properties:*

1. *$\forall i \in I$, $\mathcal{A}_i$ is an alarm pattern of $\mathcal{N}_i$,*
2. *the sets of places of $\mathcal{A}_i$ are both pairwise disjoint and disjoint from that of $\prod_{i \in I} \mathcal{N}_i$.*

Again, assuming pairwise disjoint sets of places reflects the assumption that sensors cannot capture causalities relating events occurring at different sites. Set

$$\mathcal{A} \quad \triangleq \quad \prod_{i \in I} \mathcal{A}_i \,.$$

Then $\mathcal{A}$ satisfies conditions 2,3 which define alarm patterns in subsection 4.1 on (centralized) diagnosis. Hence we can consider $\mathsf{diagnosis}\,(\mathcal{A})$ and we define

$$\mathsf{diagnosis}\,(\mathcal{A}_i, i \in I) \quad \triangleq \quad \mathsf{diagnosis}\,(\mathcal{A}) \;. \tag{9}$$

Therefore we already know how to represent it: apply theorem 1, which yields the following respective representations for $\mathsf{diagnosis}\,(\mathcal{A}_i, i \in I)$, namely

$$\mathcal{U}_{\mathcal{N},\mathcal{A}} \quad = \quad \{\mathcal{U}_{\mathcal{N}}(\mathcal{A}), \preceq, \#_{\mathcal{A}}\} \,,$$

by using theorem 1, or alternatively

$$\mathcal{U}_{\mathcal{N}\times\mathcal{A}} \tag{10}$$

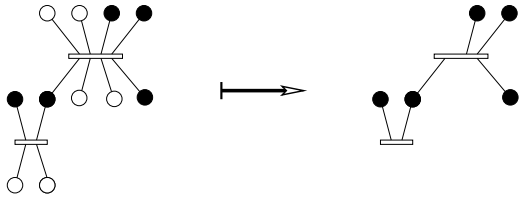
by using theorem 2. Due to its simpler algebraic structure, unless otherwise specified, we shall work with the latter representation in the sequel. To prepare for a distributed computing of $\mathsf{diagnosis}\,(\mathcal{A}_i, i \in I)$, we shall now analyse the structure of diagnosis nets for the distributed case.

5.2 Architecture of distributed diagnosis

In this subsection we prepare for the design of algorithms for distributed diagnosis. We describe it first using the metaphor of puzzle games. Roughly speaking, we assume each local sensor has a *player* associated with it. We wish to regard the diagnosis problem as a cooperative multi-player game in which each player monitors his own sensor, and cooperates with other players at constructing the overall diagnosis, on-line. We like to see the diagnosis net under construction as a "puzzle" made of tiles glued together. Each player has his own private subset of tiles, to reflect that the whole model is the product of local Petri net models and each player only knows his local model. Since the different players cooperate at building an overall puzzle, each given player can put tiles on current puzzle boudaries that were constructed by himself (local continuation) or by other players (cooperative continuation). Finally, we require that the players can cooperate at this puzzle game like "agents" working in a fully asynchronous way, without the need for any particular synchronization service provided by the underlying communication infrastructure. Let us formalize this.

Some further material on Petri nets and their unfoldings

We need some further material on Petri nets and their unfoldings. We define the projection of a Petri net $\mathcal{N} = \{P, P_0, T, \rightarrow, \lambda\}$ onto a subset $P' \subset P$ of its places, this is depicted below, where black patches refer to places belonging to P':



and is formalized as follows:

$$\mathbf{proj}_{P'}\,(\mathcal{N}) \quad = \quad \{P', P_0 \cap P', T', \rightarrow_{P'}, \lambda'\}\,, \tag{11}$$

where

$$T' \quad = \quad T \setminus \{t : ({}^\bullet t \cup t^\bullet) \cap P' = \emptyset\}\,, \tag{12}$$

$\rightarrow_{P'}$ is the restriction to $(P' \times T') \cup (T' \times P')$ of the flow relation $\rightarrow$, and λ' is the restriction to T' of the labelling map λ. To ensure the consistency of the definition (12) for T', we assume that the pair $(\mathcal{N}, P')$ satisfies:

$$\forall t \in T \quad : \quad \begin{cases} \text{either } {}^\bullet t \cap P' \neq \emptyset\,, \\ \text{or } ({}^\bullet t \cup t^\bullet) \cap P' = \emptyset\,, \end{cases} \tag{13}$$

so that no transition of $\mathbf{proj}_{P'}(\mathcal{N})$ has an empty preset — note that this also allows us to avoid problems in defining the unfolding of $\mathbf{proj}_{P'}(\mathcal{N})$. Set $P'' = P \setminus P'$. If both pairs $(\mathcal{N}, P')$ and $(\mathcal{N}, P'')$ satisfy condition (13), then the following decomposition holds:

$$\mathcal{N} \quad = \quad \mathbf{proj}_{P'}(\mathcal{N}) \times \mathbf{proj}_{P''}(\mathcal{N}) \tag{14}$$

Then we will need the following combinator on Petri nets (recall that Petri nets can share places, not only transitions):

$$\mathcal{N} \sqcap \mathcal{N}' \quad \stackrel{\Delta}{=} \quad \mathbf{proj}_{P \cap P'}(\mathcal{N}) \times \mathbf{proj}_{P \cap P'}(\mathcal{N}')\,. \tag{15}$$

where P, P' denote the respective sets of places for Petri nets $\mathcal{N}, \mathcal{N}'$. Using these operators, formula (14) implies the following result:

Lemma 1 *for $\mathcal{N}$ and $\mathcal{N}'$ two nets, the following formula holds:*

$$\mathcal{N} \times \mathcal{N}' \quad = \quad \mathcal{N}_{|\mathsf{oc}} \times (\mathcal{N} \sqcap \mathcal{N}') \times \mathcal{N}'_{|\mathsf{oc}}\,, \tag{16}$$

where

$$\mathcal{N}_{|\mathsf{oc}} = \mathbf{proj}_{P \setminus P'}(\mathcal{N}) \quad , \quad \mathcal{N}'_{|\mathsf{oc}} = \mathbf{proj}_{P' \setminus P}(\mathcal{N}')\,.$$

The interest of decomposition (16) lies in the following two facts: on the one hand $\mathcal{N}_{|\mathsf{oc}}$, $(\mathcal{N} \sqcap \mathcal{N}')$, and $\mathcal{N}'_{|\mathsf{oc}}$ possess pairwise disjoint sets of places, hence they can only synchronize on their shared events; and on the other hand, the remaining operation $\mathcal{N} \sqcap \mathcal{N}'$ involves two nets with identical sets of places. In addition, if nets $\mathcal{N}$ and $\mathcal{N}'$ have disjoint sets of events, then operation $\mathcal{N} \sqcap \mathcal{N}'$ involves two nets with identical sets of places and no shared events. Thus decomposition (16) provides a separation of concerns.

Distributed diagnosis with two players, a decomposition formula

We consider two Petri nets $\mathcal{N}'$ and $\mathcal{N}''$ such that $P' \cap P'' \neq \emptyset$ and $T' \cap T'' = \emptyset$, and associated alarm patterns $\mathcal{A}'$ and $\mathcal{A}''$, following the general scheme of distributed diagnosis as discussed in subsection 5.1. In particular, $\mathcal{A}'$ and $\mathcal{A}''$ have disjoint sets of conditions and events. The diagnosis consists in computing the unfolding

$$\mathcal{U}_{\mathcal{N} \times \mathcal{A}} \quad = \quad \mathcal{U}_{(\mathcal{N}' \times \mathcal{N}'') \times (\mathcal{A}' \times \mathcal{A}'')}\,.$$

Using lemma 1, decompose

$$\begin{array}{rcccccc} \mathcal{N}\times\mathcal{A} & = & (\mathcal{N}'\times\mathcal{A}') & \times & (\mathcal{N}''\times\mathcal{A}'') & & \\ & = & (\mathcal{N}'\times\mathcal{A}')_{|\mathsf{loc}} & \times & (\mathcal{N}'\sqcap\mathcal{N}'') & \times & (\mathcal{N}''\times\mathcal{A}'')_{|\mathsf{loc}} \\ & = & (\mathcal{N}'_{\mathsf{loc}}\times\mathcal{A}') & \times & (\mathcal{N}'\sqcap\mathcal{N}'') & \times & (\mathcal{N}''_{\mathsf{loc}}\times\mathcal{A}'') \end{array}$$

(note that alarm patterns $\mathcal{A}'$ and $\mathcal{A}''$ possess only private places), hence we need to compute the unfolding:

$$\mathcal{U}_{\mathcal{N}\times\mathcal{A}} \;=\; \mathcal{U}\,\underbrace{(\mathcal{N}'_{\mathsf{loc}}\times\mathcal{A}')}_{\mathsf{loc}'}\times\underbrace{(\mathcal{N}'\sqcap\mathcal{N}'')}_{\mathsf{interaction}}\times\underbrace{(\mathcal{N}''_{\mathsf{loc}}\times\mathcal{A}'')}_{\mathsf{loc}''}\;. \tag{17}$$

In (17) the term loc' (resp. loc'') is local to player play' (resp. play"). The term interaction concentrates the interaction between the two players. Therefore player play' deals with $\mathsf{loc}'\times\mathsf{interaction}$, and symmetrically for player play". This is reflected by the following structure for the unfolding (17): referring to the definition 2, if

$$\mathcal{U}_{\mathcal{N}\times\mathcal{A}} \;=\; \{B,E,\rightarrow,\lambda\}$$

then:

- The set B of conditions decomposes as

$$B \;=\; B_{|\mathsf{loc}'}\uplus B_{|\mathsf{interaction}}\uplus B_{|\mathsf{loc}''} \tag{18}$$

  where $B_{|\mathsf{loc}'}$ is the set of private conditions for player play' (and similarly for $B_{|\mathsf{loc}''}$), and $B_{|\mathsf{interaction}}$ is the set of conditions which must be known from both players.

- The set E of events decomposes as

$$E \;=\; E_{|\mathsf{loc}'}\uplus E_{|\mathsf{loc}''} \tag{19}$$

  where $E_{|\mathsf{loc}'}$ is the set of private events for player play' (and similarly for $E_{|\mathsf{loc}''}$).

- The flow relation handled by player play' is defined on

$$((B_{|\mathsf{loc}'}\uplus B_{|\mathsf{interaction}})\times E_{|\mathsf{loc}'})\;\cup\;(E_{|\mathsf{loc}'}\times(B_{|\mathsf{loc}'}\uplus B_{|\mathsf{interaction}}))\,, \tag{20}$$

  and similarly for player play".

Conditional independence

To prepare for distributed diagnosis with two players, we first show that concurrence and conflict relations can be recovered from neighbouring interaction involving only Petri nets in a direct interaction. We formalize this using the notion of *conditional independence.* Consider three Petri nets $\mathcal{N}_-,\mathcal{N}_0,\mathcal{N}_+$ such that:

(a) $\mathcal{N}_0, \mathcal{N}_-$ on the one hand, and $\mathcal{N}_0, \mathcal{N}_+$ on the other hand, have shared places and transitions,

(b) but $\mathcal{N}_-, \mathcal{N}_+$ have disjoint sets of places and transitions.

We say that the Petri nets $\mathcal{N}_-$ and $\mathcal{N}_+$ are *conditionally independent, given* $\mathcal{N}_0$. Hence pairs $(\mathcal{N}_0, \mathcal{N}_-)$ and $(\mathcal{N}_0, \mathcal{N}_+)$ directly interact, whereas $(\mathcal{N}_-, \mathcal{N}_+)$ interact only indirectly, through $\mathcal{N}_0$. Consider the product Petri net $\mathcal{N} = \mathcal{N}_- \times \mathcal{N}_0 \times \mathcal{N}_+$ and its unfolding $\mathcal{U}_\mathcal{N}$. We shall use the following notations. In the rest of this subsection, the term *"labelled by* $\mathcal{N}_0$*"* will mean for short: "labelled by nodes belonging to $\mathcal{N}_0$", and similarly for $\mathcal{N}_-, \mathcal{N}_+$.

For n_0, n_0' two nodes of occurrence net $\mathcal{U}_\mathcal{N}$ that are labelled by $\mathcal{N}_0$, we write

$$n_0 \quad \preceq_0 \quad n_0' \tag{21}$$

if the path of arrows from n_0 to n_0' in $\mathcal{U}_\mathcal{N}$ visits only nodes that are labelled by $\mathcal{N}_0$. In other words, relation $\preceq_0$ is the part of the causality relation on $\mathcal{U}_\mathcal{N}$ which is locally caused by Petri net $\mathcal{N}_0$. Local causality relations $\preceq_-$ and $\preceq_+$ are defined accordingly.

For n_0', n_0'' two nodes of occurrence net $\mathcal{U}_\mathcal{N}$ that are labelled by $\mathcal{N}_0$, we write

$$n_0' \quad \#_0 \quad n_0'' \tag{22}$$

if $n_0' \neq n_0''$ if there is a condition $b_0 \in \mathcal{U}_\mathcal{N}$ labelled by $\mathcal{N}_0$ and different from n_0' and n_0'', from which one can reach n_0' and n_0'' through $\mathcal{U}_\mathcal{N}$, exiting b_0 by different arrows. In other words, $n_0' \ \#_0 \ n_0''$ holds if the source of the conflict is labelled by $\mathcal{N}_0$. Local conflict relations $\#_-$ and $\#_+$ are defined accordingly.

Finally, $\preceq_{|\mathcal{N}_0}$ denotes the restriction, to the set of nodes labelled by $\mathcal{N}_0$, of the causality relation on $\mathcal{U}_\mathcal{N}$, i.e., for two nodes n, n':

$$n \preceq_{|\mathcal{N}_0} n' \quad \text{iff} \quad n \preceq n' \text{ and } n, n' \text{ are labelled by } \mathcal{N}_0\,. \tag{23}$$

Restricted conflict relation $\#_{|\mathcal{N}_0}$ is defined in the same way. And similar notations hold for the two other Petri nets $\mathcal{N}_-$ and $\mathcal{N}_+$. Also, $\preceq_{|\mathcal{N}_- \cap \mathcal{N}_0}$ denotes the restriction of the causality relation $\preceq$ to the set of nodes labelled by both $\mathcal{N}_-$ and $\mathcal{N}_0$, and so on.

Theorem 3 (a useful factorization) *Consider three Petri nets* $\mathcal{N}_-, \mathcal{N}_0, \mathcal{N}_+$ *such that* $\mathcal{N}_-$ *and* $\mathcal{N}_+$ *are conditionally independent given* $\mathcal{N}_0$*. Then,*

1. *the relation* $\preceq_{|\mathcal{N}_-}$ *is the transitive closure of the relation* $\preceq_- \cup \preceq_{|\mathcal{N}_- \cap \mathcal{N}_0}$*, and*
2. *the relation* $\#_{|\mathcal{N}_-}$ *is the closure, through causality relation* $\preceq_{|\mathcal{N}_-}$*, of the relation* $\#_- \cup \#_{|\mathcal{N}_- \cap \mathcal{N}_0}$*.*

Proof: Point 1 is immediate, by inspection of the graph of the unfolding $\mathcal{U}_{\mathcal{N}_- \times \mathcal{N}_0 \times \mathcal{N}_+}$. To show point 2, recall the following. Let $b^\#$ be a source of conflict in the unfolding $\mathcal{U}_\mathcal{N}$, and let n_-' and n_-'' be two different nodes labelled by $\mathcal{N}_-$, which can be reached, within $\mathcal{U}_\mathcal{N}$, from $b^\#$ by exiting $b^\#$ via two different arrows. Hence we have $n_-' \ \#_{|\mathcal{N}_-} \ n_-''$, by definition. We consider the following three cases.

$\lambda(b^{\#}) \in \mathcal{N}_-$ **:** then the pair (n'_-, n''_-) belongs to the closure of relation $\#_{|\mathcal{N}_-}$ by causality relation $\preceq_{|\mathcal{N}_-}$, by definition.

$\lambda(b^{\#}) \in \mathcal{N}_0$ **:** consider a path from $b^{\#}$ to n'_-, and denote by n'_{0-} the node labelled by nodes shared by $\mathcal{N}_-$ and $\mathcal{N}_0$ which is closest to n'_- in this path, i.e., the last node labelled by $\mathcal{N}_0$ before reaching n'_-. Node n''_{0-} is defined similarly. From the preceding point, we already know that $n'_{0-} \#_0 n''_{0-}$. By construction, we have $n'_- \succ_- n'_{0-}$, whence $n'_- \succ_{|\mathcal{N}_-} n'_{0-}$ by using statement 1 of the theorem. Hence the pair (n'_-, n''_-) belongs to the closure of relation $\#_{|\mathcal{N}_- \cap \mathcal{N}_0}$ by causality relation $\preceq_{|\mathcal{N}_-}$.

$\lambda(b^{\#}) \in \mathcal{N}_+$ **:** the same pair (n'_{0-}, n''_{0-}) as before still belongs to $\#_{|\mathcal{N}_0}$, and we finish the proof as in the preceding case. ◇

Theorem 3 is used as follows.

Corollary 1 *For the construction of the restriction, to the nodes labelled by $\mathcal{N}_-$ or $\mathcal{N}_0$, of the causality and conflict relations on $\mathcal{U}_{\mathcal{N}}$, it is enough to maintain 1/ the local conflict and causality relations $\#_-$ and $\preceq_-$ due to $\mathcal{N}_-$, together with 2/ the conflict relation $\#_{|\mathcal{N}_- \cap \mathcal{N}_0}$ and the transitive reduction of the causality relation $\preceq_{|\mathcal{N}_- \cap \mathcal{N}_0}$.*

Direct cooperation between players play_- and play_+ is not required: the construction of the global unfolding factorizes into the two local cooperations between play_- and play_0 on the one hand, and play_0 and play_+ on the other hand.

If we reuse this remark for the construction of the diagnosis net of formula (17), it means that players play' and play'' only interact via their interaction part.

Distributed diagnosis in the case of several players in a tree-structured interaction

Now we turn to the general setup of subsection 5.1: we have a set $\mathcal{N}_i, i \in I$ of Petri nets, and corresponding alarm patterns $\mathcal{A}_i$. We define the (nondirected) *interaction graph* $\mathcal{I}$ as follows:

$$(i,j) \in \mathcal{I} \quad \text{iff} \quad P_i \cap P_j \neq \emptyset\,, \tag{24}$$

i.e., we draw a branch (i,j) iff the two nets $\mathcal{N}_i$ and $\mathcal{N}_j$ possess shared places. Pick a branch $(j_\star, k_\star) \in \mathcal{I}$. Assume that we can partition set I into

$$I \;=\; J \uplus \{j_\star\} \uplus \{k_\star\} \uplus K\,, \tag{25}$$

and the subsets J and K are separated by the branch $(j_\star, k_\star)$, meaning that there is no branch in the graph $\mathcal{I}$ linking two nodes sitting on J and K, respectively. Set

$$\mathcal{N}' = \left(\prod_{j \in J} \mathcal{N}_j\right) \times \mathcal{N}_{j_\star} \quad , \quad \mathcal{N}'' = \mathcal{N}_{k_\star} \times \left(\prod_{k \in K} \mathcal{N}_k\right)$$

Applying decomposition (16) successively to $\mathcal{N}$ and $\mathcal{N}_{j_\star} \times \mathcal{N}_{k_\star}$, and using notations therein, we have

$$\begin{array}{rcccccc} \mathcal{N} & = & \mathcal{N}' \times \mathcal{N}'' & = & \mathcal{N}'_{\mathsf{loc}} & \times & (\mathcal{N}' \sqcap \mathcal{N}'') & \times & \mathcal{N}'_{\mathsf{loc}} \\ & & \mathcal{N}_{j_\star} \times \mathcal{N}_{k_\star} & = & (\mathcal{N}_{j_\star})_{\mathsf{loc}} & \times & (\mathcal{N}_{j_\star} \sqcap \mathcal{N}_{k_\star}) & \times & (\mathcal{N}_{k_\star})_{\mathsf{loc}} \end{array}$$

and the following relations hold:

$$\mathcal{N}' \sqcap \mathcal{N}'' = \mathcal{N}_{j_\star} \sqcap \mathcal{N}_{k_\star} \tag{26}$$

$$\mathcal{N}'_{\mathsf{loc}} = \left(\prod_{j \in J} \mathcal{N}_j\right) \times (\mathcal{N}_{j_\star})_{\mathsf{loc}} \quad , \quad \mathcal{N}''_{\mathsf{loc}} = (\mathcal{N}_{k_\star})_{\mathsf{loc}} \times \left(\prod_{k \in K} \mathcal{N}_k\right)$$

Formulas (26) express that, in order to compute the interaction between $\mathcal{N}'$ and $\mathcal{N}''$, it is enough to compute the interaction between the adjacent components $\mathcal{N}_{j_\star}$ and $\mathcal{N}_{k_\star}$.

If the graph $\mathcal{I}$ is indeed a *tree,* then decomposition (25) holds for any branch $(j,k) \in \mathcal{I}$, and the same reasoning as above applies inductively to $\mathcal{N}'_{\mathsf{loc}} = (\prod_{j \in J} \mathcal{N}_j) \times (\mathcal{N}_{j_\star})_{\mathsf{loc}}$ and $\mathcal{N}''_{\mathsf{loc}} = (\mathcal{N}_{k_\star})_{\mathsf{loc}} \times (\prod_{k \in K} \mathcal{N}_k)$, separately. To summarize, for the case of several players in a tree-structured interaction, it is enough that each player handles his interaction with his neighbours in a local manner. More precisely, we distinguish some index belonging to I and denote it by 0, and let $I_0 \subseteq I$ be the set of the neighbours of 0 in graph $\mathcal{I}$, player 0 needs to perform the following:

Specification 1 [tree-structured multiplayer distributed diagnosis]

(a) Let $P_{0,\mathsf{loc}}$ be the set of private places of $\mathcal{N}_0$, and define $\mathcal{N}_{0,\mathsf{loc}}$ accordingly. Player 0 has to compute locally the contribution of $\mathcal{N}_{0,\mathsf{loc}} \times \mathcal{A}_0$ to the unfolding.

(b) For each $j \in I_0$, player 0 needs to cooperate (only) with player j in computing the contribution of $(\mathcal{N}_0 \sqcap \mathcal{N}_j)$ to the unfolding.

(c) Player 0 has to to compute locally the contribution of the product $\mathcal{N}_{0,\mathsf{loc}} \times \mathcal{A}_0 \times \prod_{j \in I_0} (\mathcal{N}_0 \sqcap \mathcal{N}_j)$ to the unfolding.

NOTA. The assumption that the interaction graph is a tree is important. Assume for instance $\mathcal{I}$ is a cycle $0, 1, \ldots, K, 0$. Then the restriction, to the nodes labelled by $\mathcal{N}_0$, of the conflict relation on $\mathcal{U}_\mathcal{N}$, cannot be deduced solely from $\#_0$, $\#_{|\mathcal{N}_1}$, and $\#_{|\mathcal{N}_K}$. For example, it is possible that two nodes of $\mathcal{U}_\mathcal{N}$ labelled by $\mathcal{N}_0$ are reached, by starting from the same condition labelled by $\mathcal{N}_k$ for some $0 < k < K$, via two paths, respectively traversing nodes labelled by $\mathcal{N}_{k-1}, \mathcal{N}_{k-2}, \ldots, \mathcal{N}_1, \mathcal{N}_0$ and $\mathcal{N}_{k+1}, \mathcal{N}_{k+2}, \ldots, \mathcal{N}_K, \mathcal{N}_0$. In this case, the conflict relation cannot be locally computed and communicated along the different components, it is truly global.

6 Algorithms

In this section, we specify the algorithms that implement the three tasks of specification 1. We first describe the framework we use for such a specification, based on rewriting rules.

First, we state an inductive construction of the unfolding of a Petri net, as defined in (4) and definition 2 (this construction is borrowed from [16]). In this construction, we make consistent use of the following notations. The conditions of the unfolding have the form (e,p), where e is an event of the unfolding and p a place of the Petri net in consideration; the label of condition (e,p) is p (written $\lambda(e,p) = p$), and its unique input event is e. Conditions $(\perp,p)$ are those having no input event, i.e., the distinguished symbol $\perp$ is used for the minimal conditions of the occurrence net. Similarly, events of the unfolding have the form (X,t), where X is a co-set of conditions belonging to the unfolding, and t is a transition of the Petri net in consideration; the label of event (X,t) is t (written $\lambda(X,t) = t$), and its set of input conditions is X.

In the sequel we make use of these notations and represent a branching process as a pair (B,E) of conditions and events. The set of *branching processes* of $\mathcal{N} = \{P, P_0, T, \rightarrow\}$ can be inductively constructed as follows:

- The following term is a branching process of $\mathcal{N}$:

$$(\{(\perp,p), p \in P_0\}, \emptyset)\ . \tag{27}$$

- If (B,E) is a branching process, $t \in T$ an event of $\mathcal{N}$, and $X \supseteq B$ a co-set labelled by ${}^\bullet t$, then the following term is also a branching process of $\mathcal{N}$:

$$(\, B \cup \{\, (e,p) \,|\, p \in t^\bullet \,\}, E \cup \{e\}\,), \quad \text{where} \quad e = (X,t)\,. \tag{28}$$

If $e \notin E$ we call e a *possible extension* of (B,E), written $(B,E) \odot e$, we denote the corresponding extended branching process by $(B,E) \bullet e$ and call it a *continuation* of (B,E) by e. This inductive construction is restated, for convenience, in the following form of a rewriting rule:

$$\frac{\text{precondition}}{\text{current branching process} \ \vdash \ \text{continuation}}$$

Using this notation, rule (28) for branching process continuation rewrites as follows:

$$\frac{X \text{ co-set of } B\,,\ \lambda(X) = {}^\bullet t}{\begin{array}{c} e = (X,t) \ \text{ in :} \\ (B,E) \ \vdash \ (\, B \cup \{\, (e,p) \,|\, p \in t^\bullet \,\}, E \cup \{e\}\,) \end{array}} \tag{29}$$

In the subsections to follow, we specify tasks (a),(b),(c) using similar notations. Note that task (a) coincides with the centralized diagnosis. Before discussing these three tasks, we shall first specify the algorithm implementing the more generic task of computing the unfolding $\mathcal{U}_{\mathcal{N}_1 \times \mathcal{N}_2}$, for two nets $\mathcal{N}_1$ and $\mathcal{N}_2$ having no shared places.

NOTATION. To shorten the rewriting rules, we shall discard the statement "X co-set of B" in the sequel. Be careful that checking such a condition requires knowing the concurrence relation, or, equivalently, the causality and conflict relations. It is therefore a nontrivial task in the distributed case (corollary 1).

6.1 Centralized diagnosis

More precisely, we discuss

> the computation of the unfolding $\mathcal{U}_{\mathcal{N}_1 \times \mathcal{N}_2}$, for two nets $\mathcal{N}_1$ and $\mathcal{N}_2$ having no shared places.

This generic algorithm can serve as an implementation of both tasks (a) and (c). Note that these tasks are local to player $i_\star$. We have two nets $\mathcal{N}_1$ and $\mathcal{N}_2$: p_i (resp. t_i) shall denote generically a place (resp. event) of net i, and the labelling map for events is denoted by $\lambda(.)$ for both the two nets and the unfolding under construction. Using these notations, we have the following rules for constructing $\mathcal{U}_{\mathcal{N}_1 \times \mathcal{N}_2}$:

$$\frac{\lambda(t_i) \text{ is private, and } \lambda(X_i) = {}^\bullet t_i}{\begin{array}{c} e_i = (X_i, t_i) \ \text{ in :} \\ (B,E) \ \vdash \ (B \cup \{\, (e_i, p_i) \,|\, p_i \in t_i^\bullet \,\}, E \cup \{e_i\}) \end{array}} \tag{30}$$

$$\frac{\lambda(t_1) = \lambda(t_2) \stackrel{\Delta}{=} \lambda \ , \ \forall i = 1,2 \ : \ \lambda(X_i) = {}^\bullet t_i}{\begin{array}{c} e = (X_1 \cup X_2, t) \ , \ \lambda(t) = \lambda \ \text{ in :} \\ (B,E) \ \vdash \ (B \cup \{\, (e,p) \,|\, p \in t_1^\bullet \cup t_2^\bullet \,\}, E \cup \{e\}) \end{array}} \tag{31}$$

Rule (30) performs a local continuation involving a single component, whereas rule (31) performs a synchronized continuation.

Thanks to theorem 2, the above rules (30,31) implement the computation of the unfolding $\mathcal{U}_{\mathcal{N} \times \mathcal{A}}$, and hence solve the centralized diagnosis problem. The resulting algorithm can be run "on-line", meaning that the unfolding $\mathcal{U}_{\mathcal{N} \times \mathcal{A}}$ can be updated using rules (30,31) each time a new alarm is collected. Since continuations can occur from any node in the current status of the unfolding, the whole unfolding must be continuously maintained. Of course this data structure is of rapidly increasing complexity, and this makes the general algorithm based on the above rule quite cumbersome.

However, for centralized diagnosis, it is a natural assumption that data are sequentially collected at a sensor, in an order which does not contradict the causality relations due to the underlying Petri nets, see the discussion in the beginning of section 4. In the next subsection we shall take advantage of this and provide in this case a much more efficient algorithm.

6.2 Centralized diagnosis, an optimized version

More precisely, we investigate

the computation of the diagnosis net $\mathcal{U}_{\mathcal{N},\mathcal{A}}$, for $\mathcal{A}$ a totally ordered alarm pattern of $\mathcal{N}$.

In this subsection we show how to compute the special structure $\mathcal{U}_{\mathcal{N},\mathcal{A}}$, in the particular case considered. We proceed by starting from the unfolding $\mathcal{U}_{\mathcal{N}\times\mathcal{A}}$, and we use theorem 2 which relates this unfolding to $\mathcal{U}_{\mathcal{N},\mathcal{A}}$. Also, our objective here is to refine the rules to take advantage of the particular assumptions and derive some optimizations. This is performed in several steps.

1. **On-line computation of $\mathcal{U}_{\mathcal{N}\times\mathcal{A}}$.** We first rewrite rules (30,31) for computing $\mathcal{U}_{\mathcal{N}\times\mathcal{A}}$. Since $\mathcal{A}$ is an alarm pattern of $\mathcal{N}$, it has no private event and only the "synchronizing" rule (31) applies. The latter rule specializes as follows:

$$\frac{\lambda(t) = \lambda(a) \stackrel{\Delta}{=} \alpha \ , \ \lambda(X) = {}^\bullet t \ , \ \lambda(X^\alpha) = {}^\bullet a}{\begin{array}{c} e = (X \uplus X^\alpha, t) \ \text{ in :} \\ (B,E) \ \vdash \ (\, B \cup \{\, (e,p) \,|\, p \in t^\bullet \uplus a^\bullet \,\} \,, E \cup \{e\}\,) \end{array}} \tag{32}$$

Now, since $\mathcal{A}$ is totally ordered, we can simplify the precondition $\lambda(X^\alpha) = {}^\bullet a$ and the postcondition. Simply write

$$\begin{array}{lcl} \mathcal{A} & = & (B,E)\,, \text{ where} \\ B & = & (\bot,1),((1,\alpha_1),2),\ldots,((n,\alpha_n),n),\ldots \\ E & = & (1,\alpha_1),(2,\alpha_2),\ldots,(n,\alpha_n),\ldots, \text{ and } \lambda((n,\alpha_n)) = \alpha_n\,, \end{array}$$

and rewrite (32) as follows:

$$\frac{\lambda(t) = \lambda((n,\alpha_n)) = \alpha_n \ , \ \lambda(X) = {}^\bullet t}{\begin{array}{c} e = (X \uplus \{n\}, t) \ \text{ in :} \\ (B,E \ \vdash \ (\, B \cup \{\, (e,p) \,|\, p \in t^\bullet \,\} \cup \{(e,n+1)\}\,,\, E \cup \{e\}\,) \end{array}} \tag{33}$$

The presence of index n suggests to consider an *on-line* version of this algorithm. This consists in applying the algorithm in the following way:

$$[\,\forall n = 1,2,\ldots [\,\forall t \ \ : \ \ \mathbf{R}_{(33)}(n,t)\,]\,], \tag{34}$$

where $\mathbf{R}_{(33)}(n,t)$ denotes rule (33), for n and t seen as parameters. Then the on-line version is just the inner part of (34):

$$[\,\forall t \ \ : \ \ \mathbf{R}_{(33)}(\mathit{current},t)\,], \tag{35}$$

where "*current*" denotes the current value for index n. The continuations performed by applying rule (35) are called the *fresh continuations*.

2. **Computing really** diagnosis$(\mathcal{A})$**.** Now, reading carefully theorems 1 and 2 reveals that we need in fact to compute $\mathcal{U}_{\mathcal{N},\mathcal{A}}$, not $\mathcal{U}_{\mathcal{N}\times\mathcal{A}}$. In $\mathcal{U}_{\mathcal{N}\times\mathcal{A}}$, some configurations

explain only prefixes of alarm pattern $\mathcal{A}$, not the full alarm pattern. These configurations must be removed while computing $\mathcal{U}_{\mathcal{N}\times\mathcal{A}}$. To kill the nodes that cannot explain the whole alarm pattern, but only prefixes of it, we only need, after having applied rule $[\,\forall t : \mathbf{R}_{(33)}(n,t)\,]$, to discard those nodes that cannot explain the current alarm n: since $\mathcal{A}$ is totally ordered, such nodes will never be able to explain alarm n. Therefore, call E_n the set of events that have been added to E while applying rule $[\,\forall t : \mathbf{R}_{(33)}(n,t)\,]$, and let "$\#(E_n)$ in (B,E)" denote the set of nodes, which belong to (B,E) and are in conflict with every node of E_n. We claim the following:

$$\begin{array}{l}\textit{the nodes belonging to "}\ \#(E_n)\ \textit{in}\ (B,E)\textit{"}\\ \textit{will never explain alarm } n.\end{array} \tag{36}$$

Accordingly, the following postprocessing is applied after $[\,\forall t : \mathbf{R}_{(33)}(n,t)\,]$:

$$\mathbf{post}\,\mathbf{R}_{(33)}(n) \quad : \quad (B,E) \;\longrightarrow\; (B,E) \setminus (\,\#(E_n) \text{ in } (B,E)\,) \tag{37}$$

It remains to justify claim (36). By definition of E_n, no *immediate* continuation of E_n can explain the nth alarm event. Assume further continuing (B,E) can *later* explain the nth alarm event. This means that there exists a subsequent alarm event, with index $m > n$, and some configuration κ of (B,E), which 1/ continues (B,E), 2/ can explain the nth alarm event, and 3/ such that $e_n \succ e_m$ in κ (with obvious notations). The conjunction of $e_m \succ e_n$ in $\mathcal{A}$, and $e_n \succ e_m$ in κ, contradicts condition 3 of definition 5. This justifies claim (36).

3. **Making use of theorem 1.** Also, a representation equivalent to $\mathcal{U}_{\mathcal{N},\mathcal{A}}$, but having the form of a net, is obtained by simply removing, from $\mathcal{U}_{\mathcal{N}\times\mathcal{A}}$, the causalities that are solely inherited from alarm pattern $\mathcal{A}$, and keeping only the so inherited conflict relations. To this end, rule $\mathbf{R}_{(33)}(n,t)$ is rewritten as follows:

$$\mathbf{R}_{(33)}(n,t) \quad : \quad \frac{\lambda(t) = \lambda((n,\alpha_n)) = \alpha_n \ ,\ \lambda(X) = {}^\bullet t}{\begin{array}{l} e = (X \uplus \{n\}, t) \ \text{ in :}\\ (B,E) \ \vdash\ (\,B \cup \{\,(e,p)\,|\,p \in t^\bullet\,\}\ ,\ E \cup \{e\}\,)\end{array}} \tag{38}$$

Compare with (33): we have removed the causalities from e to index $n+1$, this were the causalities inherited from the totally ordered alarm pattern. On the other hand, the causalities from n to e are kept, this encodes the reinforced conflict relation $\#_{\mathcal{A}}$ introduced in theorem 1. At this stage, rule (35) rewrites:

$$[\,\forall t\ :\ \mathbf{R}_{(33)}(\mathit{current},t)\,] \quad ; \quad \mathbf{post}\,\mathbf{R}_{(33)}(\mathit{current}) \tag{39}$$

with the form (38) for $\mathbf{R}_{(33)}(n,t)$.

4. **Optimizing.** We can still optimize rule (39) by noting that, in the term (B,E) resulting from aplying this rule, not all places from B can serve for future continuations of (B,E). Denote by $\perp\!\!\!\perp(E_n)$ the maximal places belonging to (B,E) that are concurrent

with *some* event belonging to (E_n) (note the difference with the former definition of $\#(E_n)$). Then we claim that

$$\begin{array}{l}\textit{only the nodes belonging to } E_n^\bullet \cup \perp\!\!\!\perp(E_n) \textit{ in } (B,E),\\ \textit{can serve for future continuations of } (B,E).\end{array} \tag{40}$$

Using claim (40), rule (38) rewrites as follows, note the modification of the pre-condition:

$$\mathbf{R}^{\mathbf{opt}}_{(33)}(n,t) \quad : \quad \frac{\begin{array}{c}X \subseteq E_{n-1}^\bullet \cup \perp\!\!\!\perp(E_{n-1}) \ \text{ in } (B,E)\\ \lambda(t) = \lambda((n,\alpha_n)) = \alpha_n \ , \ \lambda(X) = {}^\bullet t\end{array}}{\begin{array}{c}e = (X \uplus \{n\}, t) \ \text{ in :}\\ (B,E) \ \vdash \ (\, B \cup \{\, (e,p) \,|\, p \in t^\bullet \,\} \ , \ E \cup \{e\}\,)\end{array}} \tag{41}$$

This pruning mechanism is illustrated in figure 8. Now it remain to justify claim (40). To this end, consider the $n+1$st alarm event, and assume there exists some continuation e of (B,E), e explains the $n+1$st alarm event, but it is not the case that e can be concatenated to (B,E) by using the extended precondition (41). Then it must be the case that e has in its prefix some place q strictly anterior to E_n for the causality relation $\preceq$. Hence we must have $e \in \#(E_n)$, and we derive a contradiction as in the proof of claim (36).

In appendix A, we give details on the implementation of these algorithms. In particular we give a variation of the above optimized on-line version. Then we also give an optimized version for specification 1, when each local alarm pattern is totally ordered.

6.3 Distributed diagnosis, the key part

Here we discuss

> the *distributed* computation of the unfolding $\mathcal{U}_{\mathcal{N}_1 \times \mathcal{N}_2}$ for two nets $\mathcal{N}_1$ and $\mathcal{N}_2$ having identical sets of places, but disjoint sets of events (whence $\mathcal{N}_1 \times \mathcal{N}_2 = \mathcal{N}_1 \sqcap \mathcal{N}_2$)

The distributed computation of $\mathcal{U}_{\mathcal{N}_1 \sqcap \mathcal{N}_2}$ in this case is a very simplified version of task (b) of specification 1. We discuss it here for didactic purposes. Also we shall reuse the corresponding rules for implementing task (b) of specification 1, see subsection 6.4.

Consider two nets $\mathcal{N}_1$ and $\mathcal{N}_2$ as before. Events of $\mathcal{N}_i$ are generically denoted by t_i. The two rules for player play_1 are given next:

$$\frac{\lambda(X_1) = {}^\bullet t_1}{\begin{array}{c}e_1 = (X_1, t_1) \ \text{ in :}\\ (B_1,E_1) \ \vdash \ (\, B_1 \cup \{\, (e_1,p_1) \,|\, p_1 \in t_1^\bullet \,\}, E_1 \cup \{e_1\}\,) \ ;\\ \forall p_1 \in t_1^\bullet, \mathsf{emit}\ (e_1,p_1)\end{array}} \tag{42}$$

$$\frac{\mathsf{receive}\ (e_2,p_2)}{(B_1,E_1) \ \vdash \ (\, B_1 \cup \{\, (e_2,p_2)\}, E_1 \cup \{e_2\}\,)} \tag{43}$$

Rule (42) is the replica of (29), it corresponds to player play_1 performing a continuation based on a local event t_1. Rule (43) corresponds to play_1 taking into account a continuation performed by player play_2.

The two players apply these rules in parallel, repeatedly and asynchronously. Note that the emission in (42) is non blocking. Assume a model of communication medium in which messages are not lost, but the ordering of message can be modified — this is a weak form of a reliable communication medium. For $i, j = 1, 2$ and $i \neq j$, denote by (B_{ij}, E_{ij}) the set of continuations (e_j, p_j) that have been sent by player play_j but not received yet by player play_i. The following result is immediate, but it is at the same time important:

Theorem 4 *We have*

$$(B_{ij}, E_{ij}) \subset (B_j, E_j)\,,$$

and the following balance equation holds:

$$(B_1, E_1) \cup (B_{12}, E_{12}) \quad = \quad (B_2, E_2) \cup (B_{21}, E_{21})\,. \tag{44}$$

Theorem 4 formalizes the consistency of the distributed algorithm for the type of communication medium we have considered. In particular, if no message is in the process of being communicated, then $(B_{12}, E_{12}) = (B_{21}, E_{21}) = \emptyset$, whence $(B_1, E_1) = (B_2, E_2)$ follows, meaning that the two players have a consistent view of the unfolding in this case.

6.4 Implementing distributed diagnosis

Here we adapt and assembly the primitives consisting of the rules (30,31) for implementing tasks (a) and (c), and (42,43) for implementing task (b) of specification 1. In writing these rules we take into account the particular form (18,19,20) for the diagnosis net in the distributed case, and we make consistent use of theorem 3 and corollary 1.

We give the rules for player play_0, using notations from specification 1. This player constructs a nested set of branching processes, denoted by (B, E), and (B, E) decomposes as follows:

- The set B of its conditions decomposes as

$$B \quad = \quad B_0 \uplus A_0 \uplus B_{I_0} \tag{45}$$

  where B_0 are the local conditions labelled by Petri net $\mathcal{N}_{0,\mathsf{loc}}$, A_0 are local conditions labelled by the local alarm pattern $\mathcal{A}_0$, B_j is the set of conditions labelled by places of $\mathcal{N}_0$ that are shared with Petri net $\mathcal{N}_j$, and $B_{I_0} = \uplus_{j \in I_0} B_j$ are the interacting conditions. Co-sets of B_0, A_0, B_{I_0}, are generically written X_0, X_0^{α}, and X_{I_0}, respectively, and we write $X_j \stackrel{\Delta}{=} X_{I_0} \cap B_j$.

- The set E of its events decomposes as

$$E \quad = \quad E_0 \uplus E_{I_0} \tag{46}$$

where E_0 is the set of local events, labelled by a synchronized transition of Petri net $\mathcal{N}_0$ and alarm pattern $\mathcal{A}_0$, and E_{I_0} is the set of imported events, i.e., events that have been constructed by some other player $j \in I_0$.

- Synchronized transitions of Petri net $\mathcal{N}_0$ and alarm pattern $\mathcal{A}_0$ are written in the form $t_0 \times a_0$, to indicate that they synchronize transition t_0 belonging to Petri net $\mathcal{N}_0$ and event a_0 belonging to alarm pattern $\mathcal{A}_0$.

According to corollary 1, player play_0 needs to transmit, to each neighbour play_j, the restriction to $\mathcal{N}_j$ of the transitive reduction of the causality relation $\preceq_{|\mathcal{N}_0}$, and the restriction to $\mathcal{N}_j$ of the conflict relation $\#_{|\mathcal{N}_0}$. Hence, in adapting rule (42) to the full specification 1, we need to adapt the "emit" statement to take this point into account.

1. We first construct the transitive reduction of the restricted causality relation $\preceq_{|\mathcal{N}_0 \cap \mathcal{N}_j}$. Consider $e \in E_0$ a local event such that some nodes belonging to its postset $e^\bullet$ are labelled by nodes shared by both $\mathcal{N}_0$ and $\mathcal{N}_j$. This event can give raise to a continuation by player play_j and therefore player play_0 needs to forward the information that play_j needs to perform such a continuation. For e an event as before, denote by

$$ {}^\star e_{|j} \ , \ j \neq 0 \tag{47}$$

the *star-preset of* e *in* $\mathcal{N}_j$, it is a set of conditions defined as follows:

   (a) start from event e; follow backward the flow relation in the unfolding under construction, until either nodes labelled by $\mathcal{N}_j$ are reached, or a node labelled by $\mathcal{N}_{j'}, j' \neq j$ is reached, or a minimal node labelled by $\mathcal{N}_0$ is reached.

   (b) If a non empty set of nodes labelled by $\mathcal{N}_j$ is reached, then ${}^\star e_{|j}$ denotes this set of nodes, it is a co-set of the unfolding under construction. Otherwise, ${}^\star e_{|j}$ is a new condition forwarded to play_j, having a distinguished label "0" to indicate the origin of the condition.

   Finally, if the above event e has the form $e = (X, t)$ where X is a co-set of B labelled by the preset of transition t, then we define

$$ e_{|j} \quad \stackrel{\Delta}{=} \quad ({}^\star e_{|j}, t) \ , \tag{48}$$

   i.e., $e_{|j}$ is the pair obtained by replacing co-set X by the star-preset of of e in $\mathcal{N}_j$. The preset of $e_{|j}$ is known to player play_j hence play_j can use $e_{|j}$ and its postset, for local continuation. So far for the transmission, to play_j, of the restricted causality relation $\preceq_{|\mathcal{N}_0 \cap \mathcal{N}_j}$.

2. Next, we need to transmit the restricted conflict relation $\#_{|\mathcal{N}_0 \cap \mathcal{N}_j}$. Referring to case (1b) above, two situations occur:

   (a) The star preset ${}^\star e_{|j}$ is labelled by $\mathcal{N}_j$. Then no additional conflict information needs to be forwarded, since it is entirely encoded in the topology of the local part of the unfolding under construction, seen as a graph.

(b) The star preset ${}^\star e_{|j}$ is composed of "dummy" conditions. Then the conflict information has to be forwarded explicitly. This is performed by marking the emitted event $e_{|j}$ with a minimal description of the set of dummy conditions, labelled by "0", which are in conflict with $e_{|j}$.

In the following, we assume that $e_{|j}$ is equipped with this additional marking whenever needed.

Using these notations, the resulting rules are the following:

$$\frac{\lambda_\alpha(t_0) = \lambda_\alpha(a_0) \stackrel{\Delta}{=} \lambda_\alpha(t)\,,\ t = t_0 \times a_0\,,\ (X_0 \uplus X_{I_0}) = {}^\bullet t_0\,,\ X_0^\alpha = {}^\bullet a_0}{\begin{array}{c} e = (X_0 \uplus X_0^\alpha \uplus X_{I_0}, t) \ \text{ in :} \\ (B_0 \uplus A_0 \uplus B_{I_0}\,,\ E_0) \\ \top \\ (B_0 \uplus A_0 \uplus B_{I_0} \cup \{(e,p) | p \in t_0^\bullet\} \cup \{(e,\beta) | \beta \in a_0^\bullet\}\,,\ E_0 \cup \{e\})\ ; \\ [\ \forall p \mid p \in (t_0^\bullet \cap P_j)\,,\ j \in I_0\ ]\ :\ \mathsf{emit}\ \ (e_{|j}, p)\ \ \mathsf{to\ player}\ j \end{array}} \tag{49}$$

where $e_{|j}$ is defined in (47,48). Rule (49) corresponds to player play_0 performing a continuation of the current branching process. This results in 1/ extending the set of local conditions B_0, the set of alarm conditions A_0, and the set of interaction conditions B_{I_0}, 2/ extending the set of local events E_0, and 3/ emitting the interacting continuations to players belonging to the neighbour of play_0. Note that the emitted continuation involves only objects that are shared with player play_j. This rule is obtained by combining the rules (31) and (42).

The next rule is the symmetric one, in which interacting continuations are received from some neighbouring player, it corresponds to rule (43):

$$\frac{\mathsf{receive}\ (e_j, p_j)\ \mathsf{from\ player}\ j}{(B_{I_0}\,,\ E_{I_0})\ \vdash\ (B_{I_0} \cup \{(e_j, p_j)\}\,,\ E_{I_0} \cup \{e_j\})} \tag{50}$$

Rule (50) performs a continuation of the set B_{I_0} of interaction conditions and of the set E_{I_0} of imported events. The distributed diagnosis algorithm is illustrated, for two players, in figure 13. The same remark holds, regarding the nature of the communications, as in the subsection 6.3.

7 Some useful extensions

In this section we discuss the various extensions needed to encompass the cases listed in the abstract.

7.1 Dynamic instanciation of transitions

We first discuss this in within our original framework of systems and tiles, as introduced in section 2. The idea is that the system $\Sigma = \langle V, X_0, \mathcal{T} \rangle$ itself becomes dynamic. Therefore we

modify the definition (1) of tiles as follows. A *dynamic tile* is a 5-tuple

$$\tau \quad = \quad \langle V^-, V \,;\, x^-_{V^-}, \alpha, x_V \rangle \tag{51}$$

where

- $V^-, V \subset \mathcal{V}$ are finite subsets of variables. V^- is the set of *previous* variables, and V is the set of *current* variables. The important fact is that we can have $V \neq V^-$, meaning that variables can be created or destroyed, dynamically. Accordingly, the set $\mathcal{V}$ of all variables is now possibly infinite (but countable).
- The triple $(x^-_{V^-}, \alpha, x_V)$ is a partial transition, relating the previous V^--state $x^-_{V^-}$ to the current V-state x_V, and emitting event α where α ranges over some set A of possible event labels.

We now cast this situation in our Petri net framework, by following mutatis mutandis the reasoning of subsection 2.3. Repeating the reasoning of the latter subsection, we obtain the following class of Petri nets.

Definition 8 (locally finite Petri nets) *Let $\mathcal{N} = \{P, P_0, T, \rightarrow, \lambda\}$ be a Petri net such that: the sets of places P and transitions T are* infinite, *but the initial marking P_0 is* finite, *and each* finite *$X \subset P$ and $S \subset T$ have* finite *postsets $X^\bullet$ and $S^\bullet$, respectively. A Petri net satisfying the above conditions is called* locally finite.

This being said, the analysis of sections 4 and 5, and the algorithms of section 6 extend without modification to locally finite Petri nets and their unfoldings.

7.2 Loss of alarms

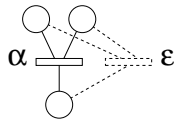
We consider a situation in which some alarms may be lost. This is modelled as follows. Referring again to section 2, tiles having alarms that may be subject to losses are modelled as follows:

$$\tau \quad = \quad \langle V, x^-_V, \alpha \vee \varepsilon, x_V \rangle \tag{52}$$

meaning that the tile may emit either alarm α, or keep silent (symbol ε) due to the loss of its emitted alarm. Duplicate tile (52) into two different tiles, namely

$$\tau_\alpha = \langle V, x^-_V, \alpha, x_V \rangle \quad , \quad \tau_\varepsilon = \langle V, x^-_V, \varepsilon, x_V \rangle \tag{53}$$

This yields a new type of *silent* tile τ_ε. Casting this situation into our Petri net framework, we consider a labelled net $\mathcal{N} = \{P, T, \rightarrow, \lambda\}$ and its unfolding $\mathcal{U}_\mathcal{N}$. Referring to our objective of modelling the loss of alarms, the idea is that the transitions which are labelled by alarms subject to loss are duplicated, with their copy having a silent transition, this is illustrated below:



The resulting net has its set of transitions partitioned into the set of *observed* and that of *silent* transitions. We can handle this in two different ways we detail next.

First approach : modifying the definition of alarm patterns

We modify the definition 5 for alarm patterns to allow for an arbitrary number of alarm losses between two successive observed alarms. We first need to define the *silence closure* κ_ε of a configuration κ of the unfolding $\mathcal{U}_\mathcal{N}$. It is obtained by applying iteratively, to κ, the following transformation, until convergence occurs (this happens in finitely many steps) :

1. Set $\kappa_{\text{curr}} := \kappa$;
2. Until no event remains in κ_{curr} which is labelled by ε, do :
   (a) pick an event $e \in \kappa_{\text{curr}}$ labelled by the silent alarm ε,
   (b) for each $e' \in {}^{\bullet\bullet}e$, add $e^\bullet$ to $e'^\bullet$,
   (c) for each $e'' \in e^{\bullet\bullet}$, add ${}^\bullet e$ to ${}^\bullet e''$,
   (d) remove e from κ_{curr} ;
3. Return $\kappa_\varepsilon := \kappa_{\text{curr}}$.

Then, definition 5 is modified as follows :

Definition 9 (alarm pattern, with loss of alarms) *Consider a labelled net $\mathcal{N}$ and its unfolding $\mathcal{U}_\mathcal{N}$. A labelled occurrence net $\mathcal{A}$ is called an* alarm pattern of $\mathcal{N}$ *if:*

1. *Its labelling map takes its values in alphabet A of alarms.*
2. *$\mathcal{A}$ is itself a configuration (it is conflict free), and its set of places is disjoint from that of $\mathcal{U}_\mathcal{N}$, and*
3. *there exists a configuration κ of $\mathcal{U}_\mathcal{N}$, such that its silence compression κ_ε and $\mathcal{A}$ possess extensions that are event-isomorphic.*

Since label ε is now private to unfolding $\mathcal{U}_\mathcal{N}$, the diagnosis still consists in computing the product $\mathcal{U}_{\mathcal{N}\times\mathcal{A}}$. With this in mind, subsections 4.2 to 6.3 extend as such, except for subsection 6.2, which does not extend to this new situation. The reason is that, whereas alarm pattern $\mathcal{A}$ is a totally ordered configuration, the unfolding $\mathcal{U}_\mathcal{N}$ has ε as a *private* label, something preventing from applying the results of subsection 6.2. The same negative remark holds for the detailed distributed algorithm of appendix A.2.

Second approach : introducing macro-tiles/transitions

The idea is that silent tiles of formula (53) shall be concatenated together to build *macro-tiles,* which must terminate by a non silent tile. We formalize this construction now, in our Petri net framework. Macro-transitions are obtained as follows. Consider Petri net $\mathcal{N}$, and associate, to each transition of it, an isomorphic silent transition (cf. the figure above). Denote by $\mathcal{N}_\varepsilon$ the resulting Petri net. Let us see Petri net $\mathcal{N}_\varepsilon$ as a bipartite graph. Search in $\mathcal{N}_\varepsilon$ all maximal convex[2] subgraphs $t^{(\varepsilon)}$ of $\mathcal{N}_\varepsilon$, having the following properties: $t^{(\varepsilon)}$ is a union of transitions and their pre– and postsets, all transitions of $t^{(\varepsilon)}$ are linearly ordered, $t^{(\varepsilon)}$ terminates by visible transitions, and other transitions of $t^{(\varepsilon)}$ are all silent. Hence $t^{(\varepsilon)}$ is a linearly ordered chain of transitions. Remove from $t^{(\varepsilon)}$ the possible loops. Collapse $t^{(\varepsilon)}$ into a single transition, by taking the minimal places of $t^{(\varepsilon)}$ as its preset, the last (visible) transition as its unique transition, and its maximal nodes as postset. This makes $t^{(\varepsilon)}$ a new transition. This construction is illustrated in the figure 14. Now each silent transition of $\mathcal{N}_\varepsilon$ is contained in a unique macro-tile $t^{(\varepsilon)}$. Therefore we modify $\mathcal{N}_\varepsilon$ by removing its silent transitions, and adding the above constructed macro-transitions (they are only finitely many of them). In the so modified Petri net, all transitions are visible, and its unfolding is the same as that of the original $\mathcal{N}_\varepsilon$, with silent transitions being collapsed. The bottom line is that we are now back to our standard setting, and all our machinery applies.

All this is satisfactory, however it should be pointed out that the construction of macro-transitions itself raises problems, for the case of distributed diagnosis. Building macro-transitions requires dealing with the global Petri net $\mathcal{N}$ and its silent extension $\mathcal{N}_\varepsilon$. When distributed diagnosis is considered, the two cases may occur. Either alarms subject to losses are only *local* to some given player, and then constructing macro-transitions works fine, based on the above described procedure. Or alarms subject to losses belong to the pre– or post-set of some place shared by two different players. Then, building macro-transitions requires the cooperation of these two players, something we do not consider practical.

To summarize, constructing macro-transitions is a good solution, provided that alarm losses are only *local* to some player. The general case requires further investigation.

7.3 Failure to communicate

Here we investigate the situation in which distributed diagnosis is performed, and some coordination messages exchanged between the different players can be lost. This only affects the distributed part of our procedure, namely the algorithm specified in subsection 6.3.

Assume first the algorithm (42,43) is applied as such and not modified. Then the fundamental balance equation (44) no longer holds, it is replaced by :

$$\begin{aligned} & (B_1, E_1) \cup (B_{12}, E_{12})_{\text{received}} \cup (B_{12}, E_{12})_{\text{lost}} \\ = \; & (B_2, E_2) \cup (B_{21}, E_{21})_{\text{received}} \cup (B_{21}, E_{21})_{\text{lost}} , \end{aligned} \tag{54}$$

[2] $\mathcal{G}$ is said to be a *convex* subgraph of $\mathcal{N}_\varepsilon$ if the following holds : if p, q are two nodes of $\mathcal{G}$ and there is a path from p to q in $\mathcal{N}_\varepsilon$, then this path must also belong to $\mathcal{G}$.

with selfexplanatory notations. The modified balance equation (54) shows that the algorithm (42,43) will not construct spurious configurations, but some actual configurations may be missed.

7.4 Incomplete modelling

By this we mean the following situation, referring again to our tile/system framework. We assume our model is partial, but correct. This means that, in our hypothesized system $\Sigma = \langle V, X_0, \mathcal{T} \rangle$, V are really variables of the system being modelled, and tiles belonging to $\mathcal{T}$ are actual behaviours of it. But it may be the case that other, unmodelled, variables may be needed to describe the behaviour of the system completely, together with associated missing sets of tiles. Casting this situation in our Petri net framework, we arrive at the following.

We assume some "true" underlying Petri net $\mathcal{N} = \{P, T, \rightarrow, \lambda\}$, modelling the exact behaviour of the considered system. Petri net $\mathcal{N}$ is not known, but we assume some hypothesized model $\mathcal{N}_{hyp} = \{P_{hyp}, T_{hyp}, \rightarrow_{hyp}, \lambda_{hyp}\}$. We assume that $P_{hyp} \subset P, T_{hyp} \subset T, \rightarrow_{hyp} \subset \rightarrow$, and labelling map λ_{hyp} is the restriction, to T_{hyp}, of the original labelling map λ. This formalizes the fact that our model is partial, but correct.

Finally, we assume that the range of labelling map λ equals that of λ_{hyp}, plus some distinguished element we denote again by ε. Label ε is used to indicate that it is not seen by the supervisor. Transitions belonging to the true but unknown set T can emit alarms that are observable by the supervisor, but they can also be silent (i.e., they are labelled by ε). Finally, we assume that, when a transition belonging to T_{hyp} occurs, then it must emit an alarm observable by the supervisor.

To summarize, we are in the following situation. There is some true Petri net $\mathcal{N}$. This Petri net can have alarm patterns $\mathcal{A}_\varepsilon$ associated with it, we have used subscript $._\varepsilon$ to indicate that the supervisor observes all alarms but ε. Therefore, we are in the lossy alarms situation analysed in subsection 7.2. But, in addition, we only have partial knowledge of the possible transitions Petri net $\mathcal{N}$ can perform. This is identical to the situation analysed in subsection 7.3. Accordingly, the solution consists in combining the analyses from these two subsections. Details are beyond the scope of this paper, and are not given here.

8 Discussion

A net unfolding approach to on-line distributed diagnosis was presented. Distributed diagnosis was approached by means of hidden state history reconstruction from alarm observations. 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. We considered the following situations: 1/ one supervisor, centralized algorithm, 2/ a distributed architecture of supervisors with local sensors, 3/ the system itself is reconfigurated dynamically, and reconfiguration actions are either known or hidden. Then we sketched how

to handle the case in which the system may fail to communicate or deliver some alarms, or the knowledge of the underlying system model is incomplete.

Some comparison is in order, with the technique presented in [13]. While the present paper uses a Petri net approach, [13] uses a model of automata interacting asynchronously via shared variables. For the single player case, this reduces to a single automaton, and [13] proposes a data structure which turns out to be equivalent to the unfolding of the considered automaton. For the multi-player case, each player handles the state of his own automaton, and the different players cooperate asynchronously at performing the overall continuations. Referring to our framework, this means that each player handles (local) *cuts,* not co-sets as players do in our case. Hence one can view the approach of the present paper as a more fragmented version of the approach of [13], in which concurrency is handled also within each component and not only between different components. This may be an advantage in the case where the model of each local Petri net is itself obtained from smaller models of components, a frequently encountered situation when object oriented techniques are used for modelling. In turn, the approach of [13] encompasses probabilistic models, in which non-determinacy is solved by maximum likelihood techniques. Also, since the global state of an automaton is sufficient to predict its future, trajectories having explained the same alarm patterns and terminating at the same state can be merged for joint continuation. Hence both approaches have their advantages and disavantages.

Implementation issues are numerous. First, we have proposed an on-line approach. This may not be very practical, due to the cost of communications in distributed systems, and it may not be needed. On the other hand, purely batch "post-mortem" diagnosis is generally not sufficient. What seems a reasonable approach is to perform on-line/per-packet diagnosis, for better use of network bandwidth. Second, we have seen the need to consider incomplete modelling. In such case, our algorithm will temporarily loose its ability to explain the current situation. An extreme situation occurs when only finite patterns can be recognised prior to escaping out of the model. In this case our approach merges with the technique of "chronicles" known in the AI community [20].

A Appendix: implementation details

Notations. In the sequel, we shall not use algorithm (27,28) for constructing branching processes. Also we shall not code the conflict and concurrence relations by means of graph properties of the constructed occurrence nets, as it was done before in the section 6 on algorithms. We shall instead maintain a more compact data structure invlving only a suffix of the previously considered occurrence nets, and conflict/concurrence relations will be maintained explicitly.

So we shall abandon the notation (B, E) and use the generic notation $\mathcal{U}$ instead:

$$\mathcal{U} \quad = \quad \{B, E, \prec, \perp\!\!\!\perp, \#\}$$

where B and E are the sets of conditions and events of $\mathcal{U}$, and $\prec, \perp\!\!\!\perp, \#$ are the causality, concurrence, and conflict relations. Notation $\mathcal{U} \odot e$ shall denote that event e is a possible extension of $\mathcal{U}$, and $\mathcal{U} \bullet e$ denotes the corresponding continuation (see the notations introduced at the beginning of section 6 on algorithms). Also, for E a set of events e such that $\mathcal{U} \odot e$ holds, then $\mathcal{U} \bullet E$ shall denote the union of nets $\mathcal{U} \bullet e$ for e ranging over E. For n a node, $\perp\!\!\!\perp(n)$ (resp. $\preceq(n), \#(n)$) shall denote the set of nodes that are concurrent with n (resp. inferior to n, in conflict with n). For X a set of nodes, $\perp\!\!\!\perp(X) = \bigcup_{n \in X} \perp\!\!\!\perp(n)$, and $\preceq(X) = \bigcup_{n \in X} \preceq(n)$, but in contrast $\#(X) = \bigcap_{n \in X} \#(n)$, note the change.

A.1 Centralized on-line diagnosis of alarm patterns, for a totally ordered alarm pattern

We present here a detailed implementation of the algorithm already discussed in subsection 6.2. In the following algorithm, we are given some underlying, *totally ordered,* infinite alarmpattern $\mathcal{A}_\infty$, and all considered alarms are taken from it. On-line diagnosis means that, for $\mathcal{A}$ a prefix of $\mathcal{A}_\infty$, and $\mathcal{A}'$ a continuation of $\mathcal{A}$, we compute $\mathcal{U}_{\mathcal{N},\mathcal{A}'}$ by updating $\mathcal{U}_{\mathcal{N},\mathcal{A}}$. As in subsection 6.2, it will be enough to update some suffix of $\mathcal{U}_{\mathcal{N},\mathcal{A}}$, we shall denote by $\mathcal{G}$ such a sufficient suffix. The resulting algorithm is shown in table A.1. In this and the following algorithms, for X a local variable, X' shall denote its update. According to (57,58), $\#_{\mathcal{A} \bullet a}(E_a) \subseteq \mathcal{U}_{\mathcal{N},\mathcal{A}} \bullet E_a$, but it has *no continuation* so far that can explain $\mathcal{A} \bullet a$, therefore it is removed from $\mathcal{G} \bullet\!\!- a$, and also from $\mathcal{U}_{\mathcal{N},\mathcal{A}} \bullet E_a$. This algorithm is illustrated in figures 11 and 12. This algorithm is derived, from that of subsection 6.2, by not introducing the conditions and events from alarm pattern $\mathcal{A}$, but rather propagating directly the enhanced conflict relation $\#_{\mathcal{A}}$. Clearly, the special structure $\mathcal{U}_{\mathcal{N},\mathcal{A}}$ is totally suitable for this detailed implementation.

A.2 Distributed on-line diagnosis of alarm patterns

We now turn to refine specification 1 for distributed diagnosis. According to this specification, player 0 has to cooperate with his neighbours at computing the contribution of the product $\mathcal{N}_{0,\mathsf{loc}} \times \mathcal{A}_0 \times \prod_{j \in I_0} (\mathcal{N}_0 \sqcap \mathcal{N}_j)$ to the unfolding. Since there is no direct interaction

inputs: $a \in \mathcal{A}_\infty$ such that $\mathcal{A} \odot a$, where $\mathcal{A}$ is a finite prefix of $\mathcal{A}_\infty$.

local: pair $(\mathcal{G}, \mathcal{U}_{\mathcal{N},\mathcal{A}})$, where $\mathcal{G}$ is the sufficient suffix of $\mathcal{U}_{\mathcal{N},\mathcal{A}}$.

1. Set

$$\begin{aligned} E_a &= \{\, e \mid \mathcal{G} \odot e \ \wedge\ e \sim a \,\} \\ E_a^\bullet &= \{\, (e,p) \mid e \in E_a \ \wedge\ p \in \lambda(e)^\bullet \,\} \end{aligned} \tag{55}$$

   where $e \sim a$ means that both events have identical alarm labels, i.e., e can explain a.

2. Construct $\mathcal{G} \bullet E_a \bullet E_a^\bullet$, extend concurrence relation $\perp\!\!\!\perp_{\mathcal{A}}$ by setting:

$$\begin{aligned} \forall e \in E_a \ :\ \perp\!\!\!\perp_{\mathcal{A}\bullet a}(e) &= \bigcap_{q \in {}^\bullet e} \perp\!\!\!\perp_{\mathcal{A}}(q) \\ \forall q \in e^\bullet\,,\ \perp\!\!\!\perp_{\mathcal{A}\bullet a}(q) &= \perp\!\!\!\perp_{\mathcal{A}\bullet a}(e) \bigcup (e^\bullet \setminus \{q\}) \end{aligned} \tag{56}$$

   and the extended conflict relation $\#_{\mathcal{A}\bullet a}$ follows accordingly. At this stage we have a net $\mathcal{G} \bullet E_a \bullet E_a^\bullet$ with its extended causality, concurrence, and conflict relations.

3. Update $\mathcal{G}$:

$$\mathcal{G}' \stackrel{\Delta}{=} \mathcal{G}_{\bullet\bullet} a = E_a^\bullet \cup (\mathcal{G} \cap \perp\!\!\!\perp_{\mathcal{A}\bullet a}(E_a)) \tag{57}$$

   note that $\mathcal{G}_{\bullet\bullet} a$ is a suffix of $\mathcal{U}_{\mathcal{N},\mathcal{A}} \bullet E_a \bullet E_a^\bullet$.

4. Update $\mathcal{U}_{\mathcal{N},\mathcal{A}}$ by putting:

$$\mathcal{U}'_{\mathcal{N},\mathcal{A}} \stackrel{\Delta}{=} \mathcal{U}_{\mathcal{N},\mathcal{A}\bullet a} = \preceq (\mathcal{G}_{\bullet\bullet} a) \tag{58}$$

   where $\preceq (\mathcal{G}_{\bullet\bullet} a)$ denotes the smallest prefix of $\mathcal{U}_{\mathcal{N},\mathcal{A}} \bullet E_a$ containing $\mathcal{G}_{\bullet\bullet} a$.

Table 1: Centralized on-line diagnosis in the case of a totally ordered alarm pattern (optimized algorithm).

between the players labelled j, we collapse them into a single alternative player play', cooperating with play_0 at computing the contribution of the product $\mathcal{N}_{0,\mathsf{loc}} \times \mathcal{A}_0 \times (\mathcal{N}_0 \sqcap \mathcal{N}')$ to the unfolding.

Rewriting $\mathcal{N}_{0,\mathsf{loc}} \times \mathcal{A}_0 \times (\mathcal{N}_0 \sqcap \mathcal{N}')$ as $(\mathcal{N}_{0,\mathsf{loc}} \times (\mathcal{N}_0 \sqcap \mathcal{N}')) \times \mathcal{A}_0 \stackrel{\Delta}{=} \mathcal{M} \times \mathcal{A}_0$ makes this looking like a diagnosis problem and we may think that we already have the solution for this! Unfortunately, $\mathcal{N} \sqcap \mathcal{N}'$ has more behaviors than $\mathcal{N}$, since nets $\mathcal{N}$ and $\mathcal{N}'$ share only

places, but no events. Therefore $\mathcal{A}_0$ is not an alarm pattern of Petri net $\mathcal{M}$, and we must adapt our algorithm A.1.

We give here an optimized version, which relies on the assumption that the local alarm pattern $\mathcal{A}_0$ is totally ordered (but of course, we do not know the interleaving with other alarm patterns). In this case, we also need to construct the special structure $\mathcal{U}_{\mathcal{M},\mathcal{A}}$, similar to the one used in table A.1. However the pruning mechanism of table A.1 cannot be reused as such, since player play' will contribute to $\mathcal{U}_{\mathcal{M},\mathcal{A}}$ by providing continuations not corresponding to any event of $\mathcal{A}_0$. Hence, some alarms from $\mathcal{A}_0$ may not be, at some point, explanable via a local continuation of the $\mathcal{U}_{\mathcal{M},\mathcal{A}}$, but they may become explanable *later,* after player play' has offered suitable continuations.

The resulting algorithm is shown in table A.2. To focus on the optimization aspect, we did not include the aspects related to corollary 1 and the discussion on the star-preset in (47,48) of subsection 6.4. This means that we transmit to the other players the whole continuation and do not compute its proper restriction. The distributed diagnosis algorithm is illustrated, for two players, in figure 13.

inputs: **case 1** $a \in \mathcal{A}_\infty$, such that $\mathcal{A} \bullet a$, where $\mathcal{A}$ is a finite prefix of $\mathcal{A}_0$;
case 2 (e', p'), a continuation, by $\mathcal{N}'$, of the unfolding $\mathcal{U}_{\mathcal{M},\mathcal{A}}$;
case 3 $(e', p')_{\mathsf{blocked}}$, a mark indicating that $(e', p') \in \mathcal{U}_{\mathcal{M},\mathcal{A}}$ is currently blocked for continuation by the environment.

local: $\mathcal{B}$, and $\mathcal{C}$, such that $\mathcal{C} \subseteq \mathcal{B} \subseteq \mathcal{A}$;
$\mathcal{U} \triangleq \mathcal{U}_{\mathcal{M},\mathcal{A}}$, $\mathcal{G}$ sufficient suffix of $\mathcal{U}$, $\mathcal{K}$ a prefix of $\mathcal{G}$.

output: $\delta\mathcal{K}$.

1. **case 1** update $\mathcal{C}' = \mathcal{C} \cup \{a\}$, $\mathcal{A}' = \mathcal{A} \bullet a$;
   case 2 update $\mathcal{G} := \mathcal{G} \bullet (e', p')$, $(e', p') \notin \mathcal{K}$; update
   $$\mathcal{C}' \quad = \quad \mathcal{C} \cup \{a \in \mathcal{B} \mid \exists e : \mathcal{G} \odot e \ \wedge \ e \sim a\}$$
   case 3 if $(e', p') \in \mathcal{K}$, then remove (e', p') from $\mathcal{G}$, and update
   $$\mathcal{B}' \quad = \quad \mathcal{B} \setminus \{a \in \mathcal{B} \mid \neg[\exists e : \mathcal{G} \odot e \ \wedge \ e \sim a\,]\}$$
2. for $a \in \mathcal{C}$, set
   $$\begin{aligned} E_a &= \{\, e \mid \ (\mathcal{G} \setminus \mathcal{K}) \odot e \ \wedge \ e \sim a \,\} \\ E_a^\bullet &= \{\, (e, p) \mid \ e \in E_a \ \wedge \ \ p \in \lambda(e)^\bullet \,\} \end{aligned}$$
3. Update $\mathcal{G}' = \mathcal{G} \bullet E_a \bullet E_a^\bullet$, extend concurrence relation $\perp\!\!\!\perp_{\mathcal{A}}$ by setting:
   $$\begin{aligned} \forall e \in E_a : \perp\!\!\!\perp_{\mathcal{A}}(e) &= \bigcap_{q \in {}^\bullet e} \perp\!\!\!\perp_{\mathcal{A}}(q) \\ \forall q \in e^\bullet, \perp\!\!\!\perp_{\mathcal{A}}(q) &= \perp\!\!\!\perp_{\mathcal{A}}(e) \bigcup (e^\bullet \setminus \{q\}) \end{aligned}$$
   and the extended conflict relation $\#_{\mathcal{A}}$ follows accordingly.
4. Define $\mathcal{K}'$ by
   $$\mathcal{G}' \setminus \mathcal{K}' \quad = \quad E_a^\bullet \ \cup \ (\mathcal{G} \cap \perp\!\!\!\perp_{\mathcal{A} \bullet a}(E_a))$$
   and emit $\delta\mathcal{K} \triangleq \mathcal{K}' \setminus \mathcal{K}$, equipped with relations $\perp\!\!\!\perp_{\mathcal{A}}$ and $\#_{\mathcal{A}}$. Update $\mathcal{U}_{\mathcal{M},\mathcal{A}}$ by putting $\mathcal{U}'_{\mathcal{M},\mathcal{A}} = \preceq(\mathcal{G}')$.

Table 2: Distributed diagnosis for totally ordered local alarm patterns. $\mathcal{K}$ contains the prefix of $\mathcal{G}$ which is currently blocked, for continuation by play_0. $\mathcal{C}$ contains those alarm events that can be currently explained by play_0, and $\mathcal{B}$ contains in addition alarm events that might later become explainable by play_0, after continuation by play'. $\delta\mathcal{K}$ is the subnet which just became blocked for play_0 ($\delta\mathcal{K}$ mirrors input of case 3), for emission toward the other players.

B Figures illustrating the different notions

In these figures, time progresses downwards.

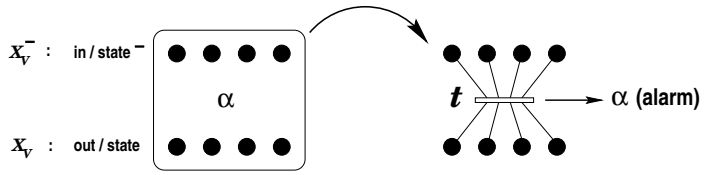


Figure 4: From tiles to transitions. The diagram on the left depicts a tile. Black patches figure valuation of state variables. For V the set of variables of the tile, the patches on the left denote previous state variables $x_v^-, v \in V$, whereas the patches on the right denote current state variables $x_v, v \in V$. On the right hand side, the tile is redrawn as a transition t. The resulting transition fires downward. When firing, it emits an alarm α.

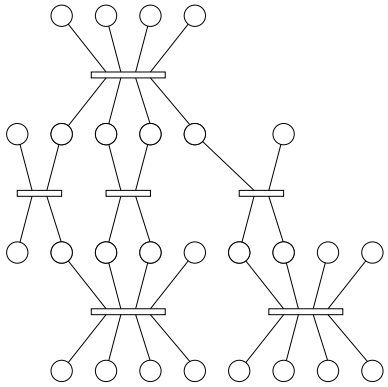


Figure 5: A configuration.

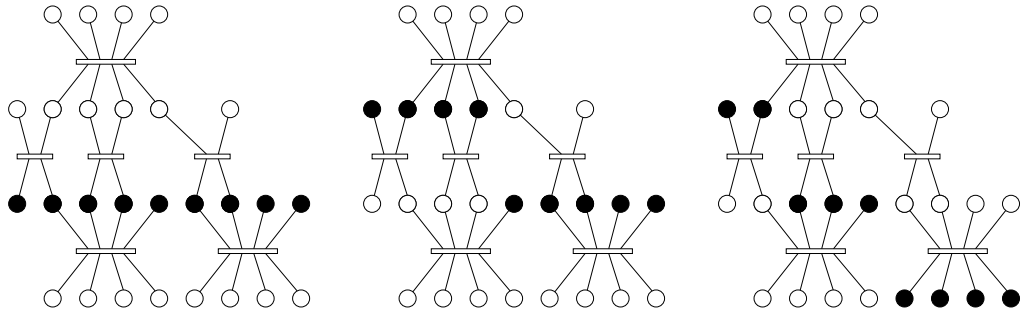


Figure 6: Showing three different cuts for the configuration of figure 5. The cuts are composed of the black patches.

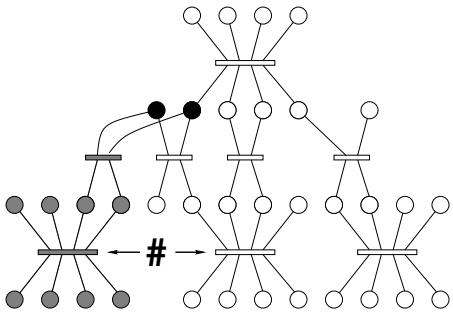


Figure 7: Occurrence net. The source of the conflict is depicted in black. Two different histories branch from this conflict, the grey one and the white one. The conflict is indicated by the # symbol.

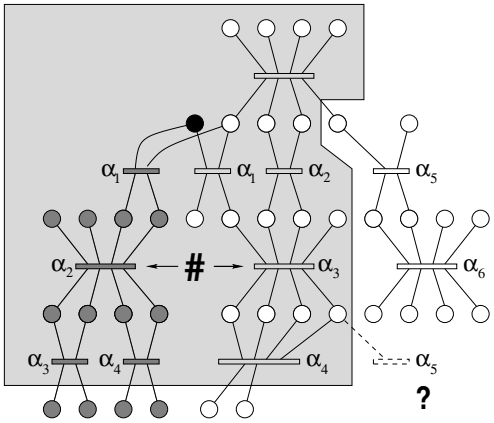


Figure 8: Pruning. The pruning mechanism used in both algorithms of subsection 6.2 and table A.1 is illustrated here, for the case where alarms $\alpha_i, i = 1, 2, 3, 4$ have been processed yet, and next alarm for processing is α_5. No more concatenation can be performed from the light grey zone. In particular, the dashed continuation with a question mark is not possible, as it would give raise to a configuration not explaining α_4.

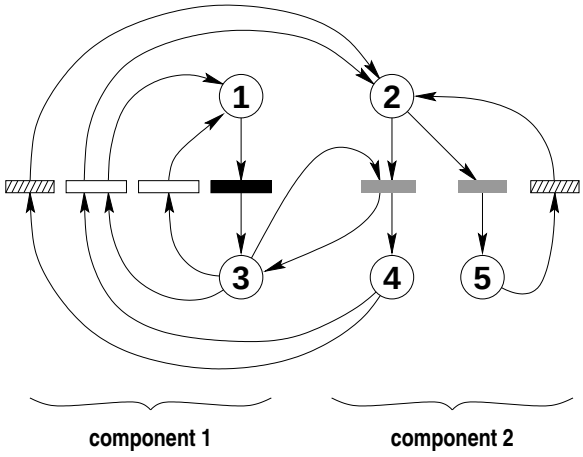


Figure 9: A petri net example. We show two interacting components. The interpretation of this example is the following. Places 1 and 2 are the nominal (safe) states. Places 3 and 5 figure truly faulty states. Place 4, for component 2, indicates a fake faulty state, resulting from the unability of component 2 to deliver its service, due to the failure of component 1. Alarms are attached to transitions. Transitions figured with the same pictogram emit identical alarms, therefore resulting in ambiguities. Self-repair can occur, for each component.

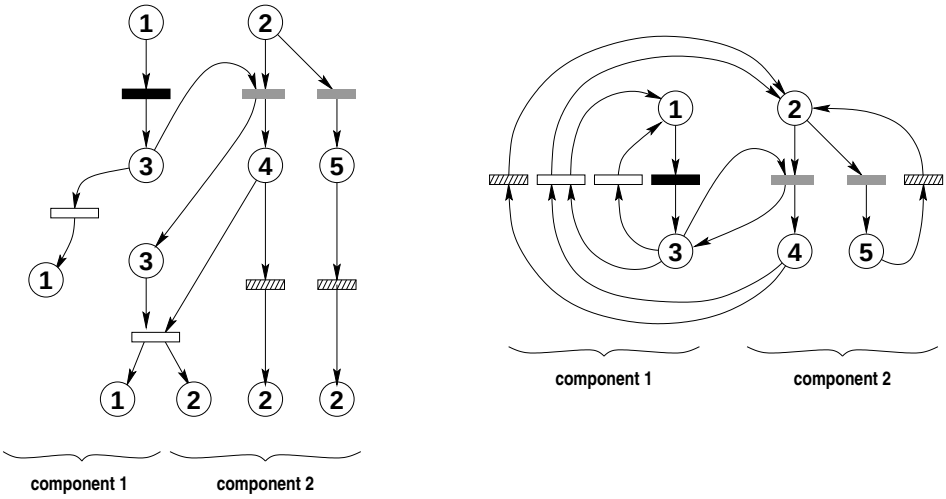


Figure 10: Unfolding the petri net example. We show only a prefix of the unfolding (i.e., a branching process).

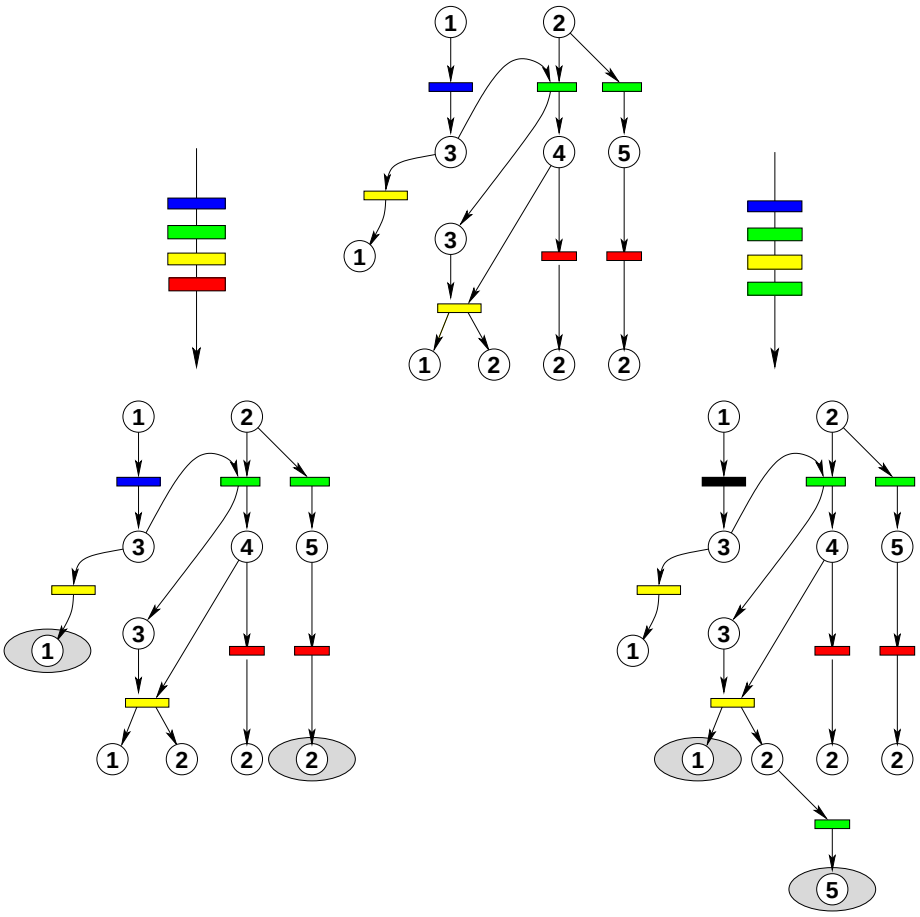


Figure 11: Centralized Diagnosis. The unfolding of figure 10 is shown on the top, for comparison. The two pictures on the bottom show the centralized diagnosis, for two different sequences of observed alarms, repectively depicted on the top of each unfolding. The configurations which explain these two alarm sequences are those terminating at the cuts marked with the light grey ellipsoid patches. The steps leading to this final solution are shown in figure 12.

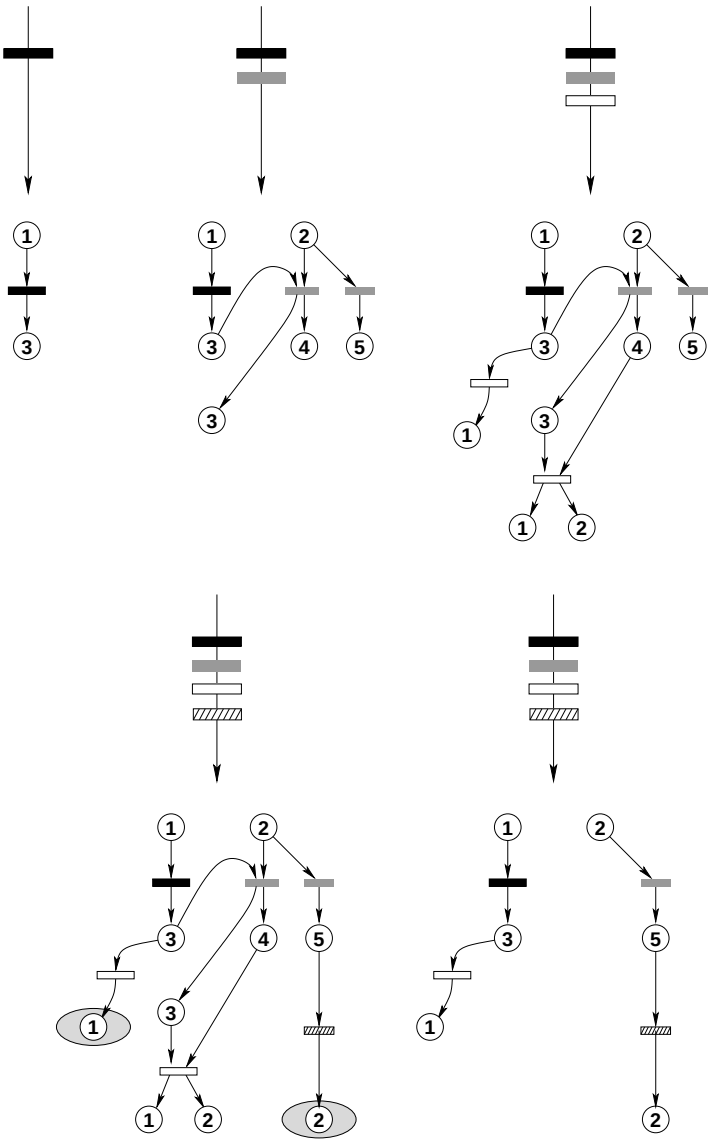


Figure 12: Centralized Diagnosis, the steps of the algorithm. We show how the solution of figure 11, left case, is obtained, on-line while reading the successive alarms. The second and third diagrams exhibit conflits, therefore showing the multiple solutions to the diagnosis problem at these steps. The fourth diagram is that of figure 11. In the last diagram we have backpropagated the pruning, therefore cutting back dead branches for the diagnosis. Note that no conflict is shown, this results from killing back one of the two solutions shown in the third diagram.

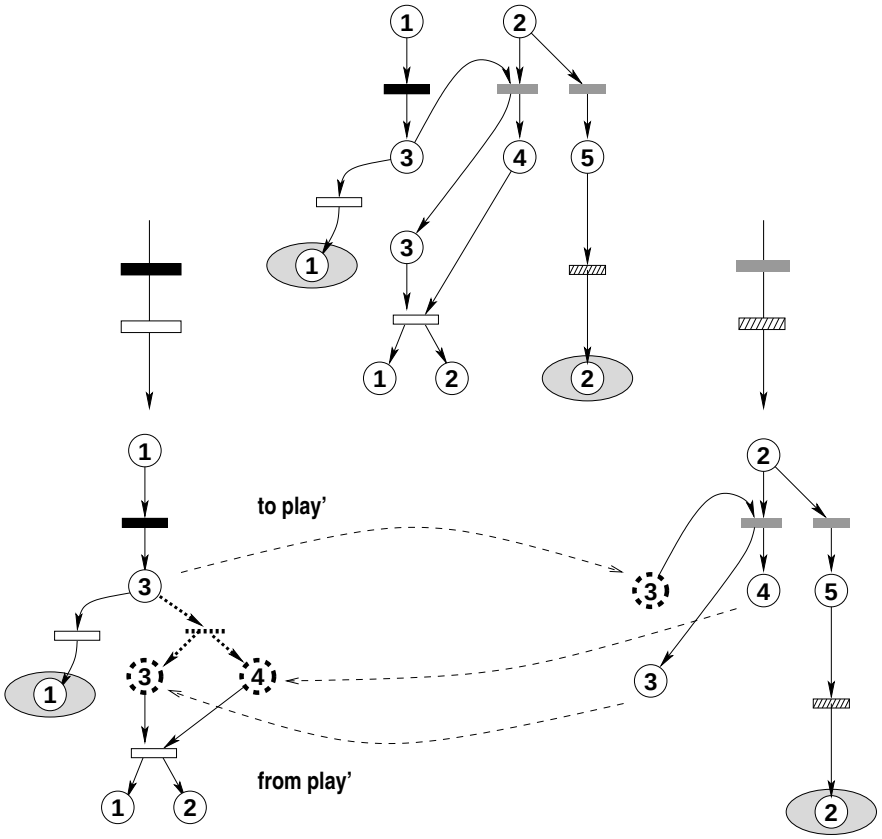


Figure 13: Distributed Diagnosis. The distributed algorithm corresponding to the centralized algorithm of figure 11, is shown here. Player play_0 sits on the left, and player play' on the right. On the top we recall the above centralized algorithm, and the distributed version is depicted on the bottom. The thin dashed arrows show the communications between the two players. The objects received from other players are shown in dashed, boldface. In particular, player play_0 receives the combination of the dark grey dashed patch and the ingoing arrows, also dashed: this is just the abstraction player play' needs from the first top-left transition from the second player, note the dummy transition which has been added to encode concurrency.

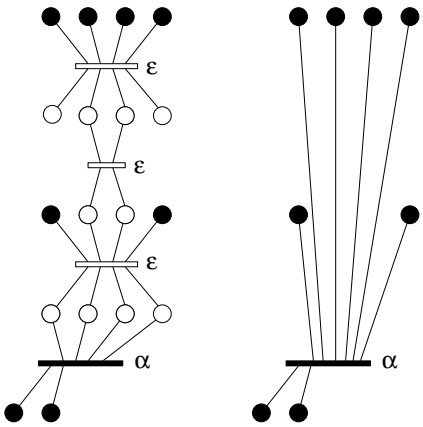


Figure 14: Constructing a macro-tile. The diagram on the left shows a $t^{(\varepsilon)}$ satisfying the requirements listed in subsection 7.2. The visible transition is filled in black. The black places are the minimal or maximal ones. The diagram on the right hand side shows the resulting macro-tile.

References

[1] 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.

[2] 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.

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

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

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

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] Y. Pencolé. Decentralized diagnoser approach: application to telecommunication network. In Proceedings of the International Workshop on Principles of Diagnosis (DX'00), Morelia, Mexico, 2000.

[13] 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.

[14] K. McMillan. *Symbolic Model Cheking: an approach to the state explosion problem.* Kluwer, 1993.

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

[17] 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.

[18] 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.

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

[20] C. Dousson, P. Gaborit, and M. Ghallab. Situation recognition: representation and algorithms. In *Proc. of the 13th IJCAI,* 166-172, 1993.

[21] M. Shapiro, Le Fessant, F. Ferreira, and P. Ferreira. Recent Advances in Distributed Garbage Collection. In *Recent Advances in Distributed Systems,* S. Krakowiak and S. K. Shrivastava Eds., LNCS 1752, 104–126, 2000.
http://www-sor.inria.fr/publi/RAIDGC_lncs1752.html

[22] Michel Raynal, "Algorithmes distribués et protocoles," Eyrolles, 85.