INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Distributed monitoring of concurrent and asynchronous systems—extended version

Albert Benveniste — Stefan Haar — Eric Fabre — Claude Jard

N° 4842 – version 2

version initiale Juillet 2003 – version révisée Juillet 2004

THÈME 1

ISSN 0249-6399

Distributed monitoring of concurrent and asynchronous systems—extended version *

Albert Benveniste , Stefan Haar , Eric Fabre , Claude Jard †

Thème 1 — Systèmes communicants
Projet DistribCom

Rapport de recherche n° 4842 – version 2‡— version initiale Juillet 2003 — version révisée Juillet 2004 61 pages

Abstract: In this paper we study the diagnosis of distributed asynchronous systems with concurrency. Diagnosis is performed by a peer-to-peer distributed architecture of supervisors. Our approach relies on Petri net unfoldings and event structures, as means to manipulate trajectories of systems with concurrency.

This report is an extended version of the paper with same title, which appeared as a plenary address in the *Proceedings of CONCUR'2003*

Key-words: asynchronous, concurrent, distributed, unfoldings, event structures, fault diagnosis, fault management.

* This work was supported by the RNRT project MAGDA2, 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. AB, SH, EF are with INRIA, and CJ is with CNRS

‡ This is a deeply revised version of the original report. The original version is no longer valid and should be replaced by this one.

Unité de recherche INRIA Rennes
IRISA, Campus universitaire de Beaulieu, 35042 Rennes Cedex (France)
Téléphone : +33 2 99 84 71 00 — Télécopie : +33 2 99 84 71 71

Diagnostic réparti de systèmes concurrents et asynchrones

Résumé : On étudie le diagnostic réparti de systèmes concurrents, distribués et asynchrones. L'application cible est la gestion d'alarmes dans les réseaux et services. On utlise les techniques de dépliages de réseaux de Petri et de structure d'événements pour traiter de la concurrence.

Mots-clés : asynchrone, concurrent, réparti, dépliage, structure d'événements, diagnostic de pannes, gestion d'alarmes.

Contents

1 Introduction

In this paper we consider fault diagnosis of distributed and asynchronous Discrete Event Systems (DES). The type of system we consider is depicted in Fig. 1. It consists of a distributed

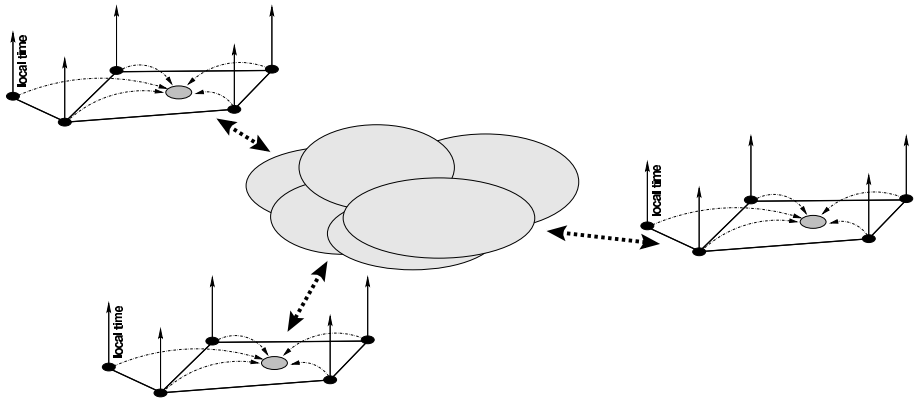

Figure 1: Three domains with cooperating supervisors.

architecture in which each supervisor is in charge of its own domain, and the different supervisors cooperate at constructing a set of *coherent* local views for their respective domains. Each domain is a networked system, in which alarms are collected and processed locally by the supervisor (shown as a larger gray circle). The different domains are interconnected by a network, represented by the "IP-cloud". The situation is summarized as follows:

Requirements 1

1. *The overall system is composed of several subsystems. Each subsystem has its own supervisor.*
2. *The communications between the subsystems, and within each subsystem, are asynchronous.*
3. *Each supervisor has knowledge of the local model of its subsystem, together with relevant information regarding the interface with adjacent subsystems.*
4. *Each supervisor collects data from sensors that are local to its subsystem. Sensors from different sites are not synchronized.*
5. *The duty of each supervisor is to construct a "local projection" of the global diagnosis, for the overall system.*
6. *To this end, the different supervisors act as peers, by exchanging information, asynchronously, with the other supervisors.*

The above requirements were motivated by our application to *distributed fault management in telecommunications networks and services* [2, 3, 4]. Since faults in this context are typically transient, providing explanations in the form of "correlation scenarios" showing the causal relations between faults and alarms, is essential. Therefore, we concentrate on constructing such scenarios, leaving aside the additional questions of fault isolation and diagnosability.

To serve as a background for the reader, here are some figures that are borrowed from our realistic example of Section 2: each subsystem is an asynchronous network of automata, each automaton has a handful of states, and there are from hundreds to thousands of such automata in the network. Each root fault can cause hundreds of correlated alarms that travel throughout each subsystem and are collected by the corresponding local supervisor. Supervised domains may very well be orders of magnitude larger in the future. Thus, scalability is a major concern. It is important that the type of algorithm we develop takes this context into account. Never constructing the overall diagnosis but rather only their local projections, ensures scalability, from subsystems to the overall system. This motivated Requirement 1.5.

In this paper, we address the problem of distributed monitoring of DES according to Requirements 1. Our approach has the following features:

- We follow a model-based approach, that is, our algorithms use a model of the underlying system. In this direction, the *diagnoser* approach by Lafortune *et al.* [26, 27] is a very elegant technique that consists in "pre-computing" all possible diagnoses, for each possible history of events, in the form of an enriched observer of the system for monitoring. Diagnosers provide the fastest on-line algorithms for diagnosis, at the price of excessive memory requirements. Given our context, we do not adopt this approach. We follow instead a more "interpreted" approach, in which the set of all possible "correlation scenarios" relating hidden faults and the observed alarms are computed, on-line. Thus, we trade off speed for memory. The closest approach to ours we know of in this respect is that of Lamperti and Zanella [21].

- An asynchronous network of automata can be seen and handled as a single automaton. However, the resulting automaton has an infinite number of states unless we assume bounded buffering of the communications. Even so, its size is exponentially larger than its components and becomes quickly unacceptable. An important source of reduction in the size of the objects handled consists in taking advantage of the concurrency that is typically exhibited between the different components. To this end, in a given history of the system, events that are not causally related are simply not ordered. Thus, *a history is a partial order of events.*

- Histories can share prefixes. To reduce the data structures handled by the algorithms, it is desirable to represent shared prefixes only once. *Unfoldings* and *event structures* were concepts introduced in the early eighties by Winskel *et al.* [31] for this purpose. An impressive theoretical apparatus has been developed since then [32, 33] to provide proper notions of parallel composition, based on tools from category theory.

Our algorithms represent both the system for supervision, and the "correlation scenarios" relating faults and alarms, by means of unfoldings and event structures. The mathematical framework that comes with this was essential in formally proving our algorithms.

Less importantly, we use safe Petri nets to model of asynchronous systems with concurrency. Executions of safe Petri nets are naturally represented by means of unfoldings or event structures.

- In the algorithms we develop, the different supervisors act as peers, with no overall synchronization or scheduling. They read local alarms, receive messages from and send messages to other peers, using fully asynchronous communications.
- Limitations of our approach are the following:
  - We do not address fault tolerance, i.e., the loss of alarms or communication messages between peers. This extension is a mild one, however.
  - We assume that a model of the system is at hand. Given our context, such a model cannot be constructed by hand. How to construct "automatically" such a model in the context of telecommunications network and service management, is reported in [3].
  - We assume that our model is valid. Overcoming this limitation is a more difficult and less classical problem.
  - Last but not least, we do not address dynamic reconfiguration, i.e., the fact that the system structure itself is subject to changes. Clearly, this is a needed extension for our motivating application. Clearly also, pre-compiled approaches such as that of diagnosers cannot handle this. In contrast, our more "interpreted" approach is better suited at addressing dynamic reconfiguration.

The paper is organized as follows. Our motivating application is described in Section 2. The problem of distributed diagnosis is extensively discussed in Section 3, based on a toy example. In particular, we introduce the architecture of our distributed algorithm and motivate the mathematical apparatus on event structures that we introduce in Section 4. Using this framework, we formally set the problem of distributed diagnosis of asynchronous systems in Section 5. To overcome the sophistication of distributed diagnosis, we structure it into a higher level orchestration based on a small set of primitive operations on event structures. These primitives are introduced and studied in Section 6. Then, the orchestration is presented in Section 7, and the overall algorithms for both off-line and on-line distributed diagnosis are formally analysed. Finally, related work is discussed in Section 8 and conclusions are drawn.

2 Motivating application: fault management in telecommunication networks

This section has been prepared with the help of Armen Aghasaryan[1].

2.1 Overall problem description

Distributed self-management is a key objective in operating large scale infrastructures. Fault management is one of the five classical components of management, and our driving application.

To ensure modularity, network management systems are decomposed into interconnected Network Elements (NE), composed in turn of several Managed Objects (MO). MO's act as peers providing to each other services for overall fault management. Consequently, each MO is equipped with its own fault management function. This involves self-monitoring for possible own internal sources of fault, as well as propagating, to clients of the considered MO, the effect of one of its servers getting disabled.

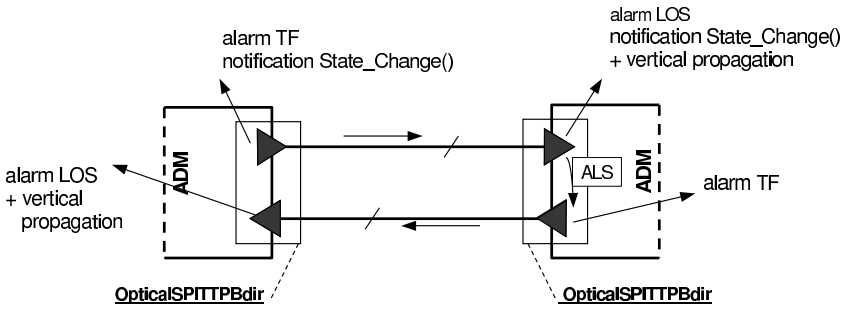

Figure 2: Fault propagation: the case of a laser failure occurring at one extremity of an optical link.

This is illustrated in Fig. 2 for the case of a laser failure occurring at one extremity of an optical link[2]. The scenario is the following. The *source* of the failure is the laser sitting on the top left extremity. Detection of the laser failure *causes* the host MO to get disabled ("notification State_Change"). Consequently, this MO emits a "TF" alarm to the supervisor. *Concurrently,* a Loss of Signal (LOS) is detected at the top right extremity. *Consequently,* state changes, alarms, and propagation of events/messages sent to the MO's sitting at higher levels of the network hierarchy, follow accordingly. This *causes* the disabling of the MO on

[1]Alcatel Research & Innovation, Next Generation Network and Service Management Project, Alcatel R&I, Route de Nozay, Marcoussis, 91461 France

[2]Technical terms and acronyms are those from Wavelength Division Multiplexing (WDM), or SDH/SONET types of networks.

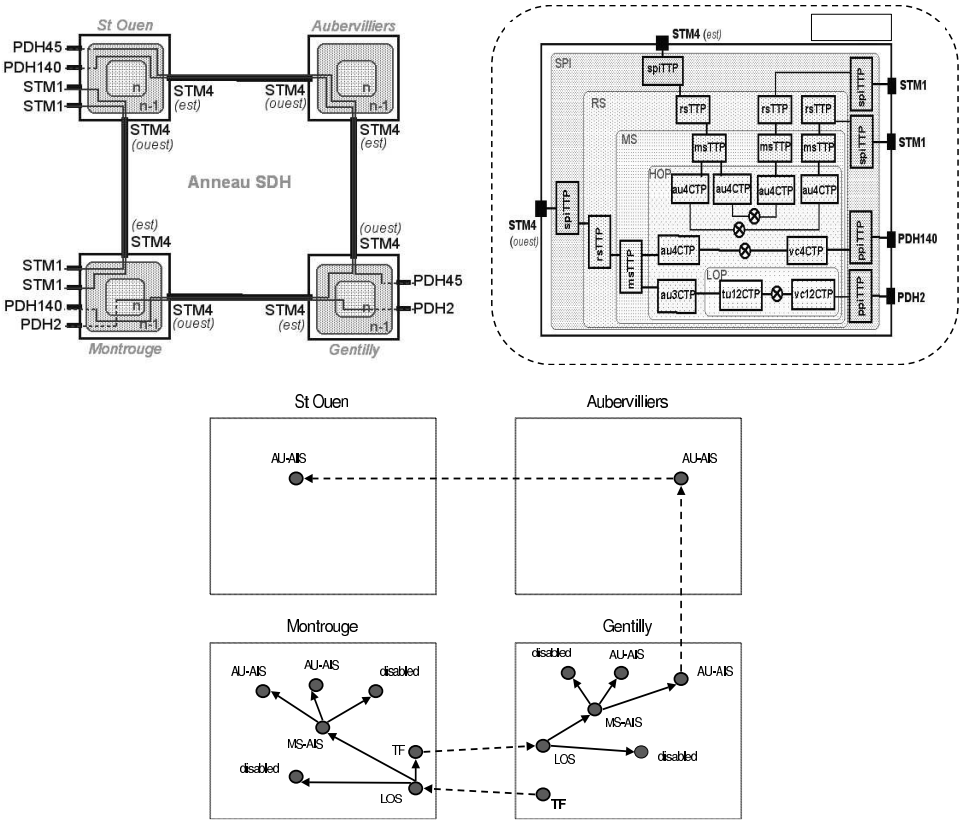

Figure 3: The Paris area SDH/SONET ring (top-left), and a detail of the Montrouge node (top-right). The different levels of the SDH hierarchy are shown: SPI, RS, etc. A fault propagation scenario distributed across the four different sites (bottom). The dashed arrows indicate distant propagation. The cryptic names are SDH/SONET fault labels.

the lower right and stops the emission of light, which *causes* in turn a LOS at the MO on the lower left. Thus, the latter appears as a sort of echo to the primary failure. We have emphasized the terms: *source, causes, concurrently,* to point out fundamental aspects of fault management in networked systems.

As a larger example, Fig. 3-top-left shows the SDH/SONET ring in operation in the Paris area (the locations indicated are suburbs of Paris). A few ports and links are shown. The top-right diagram is a detailed view of the Montrouge node. The nested light to mid gray rectangles represent the different layers in the SDH hierarchy, with the largest one being the physical layer. The different boxes are the MOs, and the links across the different layers are the paths for upward/downward fault propagation. Each MO can be seen as an automaton

reacting to input events/messages, changing its state, and emitting events and alarms to its neighbors, both co-located and distant. Fig. 3-bottom shows a realistic example of a fault propagation scenario distributed across the four different sites.

To summarize, the different supervisors are distributed, and different MO's operate concurrently and asynchronously within each supervisor.

2.2 Safe Petri nets to formalize distributed diagnosis

Our approach to mathematical modeling is illustrated by Fig. 4 and Fig. 5. Fig. 4 represents a

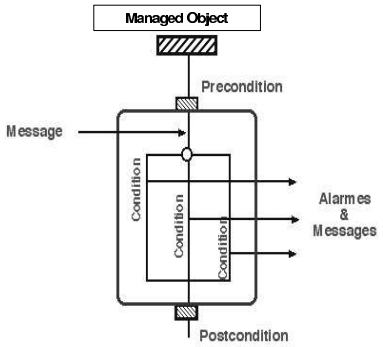

Figure 4: A scenario to capture the generic behavior of a Managed Object.

generic behavior of a MO. This MO possesses an internal state; (pre/post)-conditions refer to this state. MOs exchange messages and alarms; only alarms are visible to the supervisor(s), and other messages typically capture the propagation of hidden faults and state changes. Fig. 5 shows four causally related instances of scenarios of the form given in Fig. 4, which correspond to the fault case shown in Fig. 2. Causality occurs both via hidden messages and state changes.

We shall next represent the above set of scenarios by means of safe Petri nets. Generally speaking, safe Petri nets generalize automata toward distributed and asynchronous systems and offer powerful techniques to manipulate distributed executions of these.

Fig. 6 explains how safe Petri nets can be used to formalize the modeling approach of Fig. 4 and Fig. 5. In Fig. 6, the 1st diagram is a copy of that of Fig. 4. This scenario can be split into three scenarios, one for each different case, defined by the satisfaction of the three different conditions. One of these cases is depicted on the 2nd diagram.

Now, assume that pre/postconditions, conditions, and messages, are expressed in terms of expressions involving a finite number of variables with finite domains. Assigning a value for each variable defines a *state*; the state prior (posterior) to an alarm is called its *pre-state* (*post-state*). To each possible value for each variable we associate a new *place*. The pre-state of the scenario from 2nd diagram is therefore adequately represented by associating

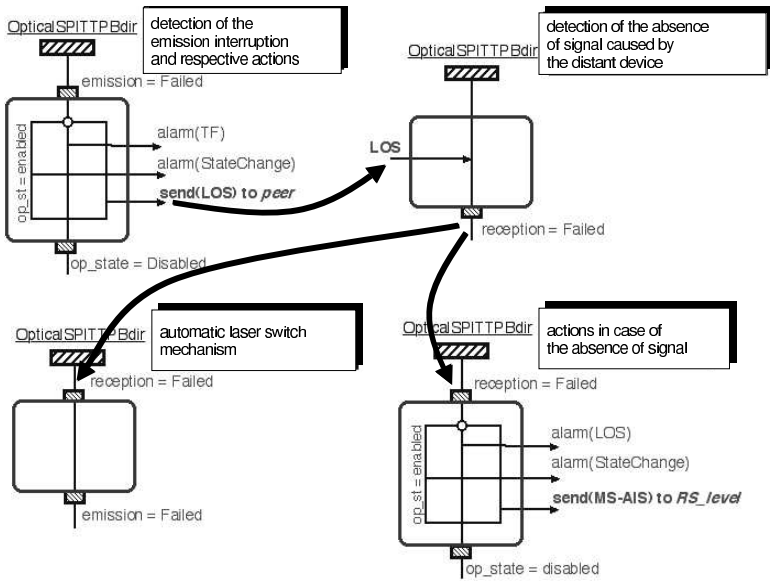

Figure 5: Four causally related instances of scenarios of the form given in Fig. 4, which correspond to the fault case shown in Fig. 2.

one token per variable, and putting each token in the place that represents the value of that

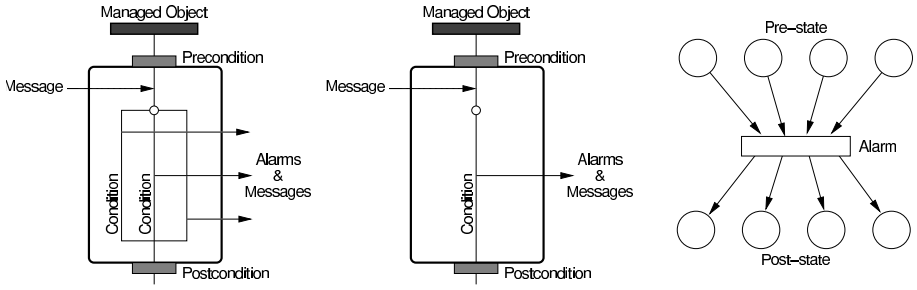

Figure 6: From scenarios of Fig. 5 to Petri nets.

variable in the considered state—in this diagram, we assume that 4 variables are involved.

Now, the scenario shown on the 2nd diagram expresses that, if a certain message is received and the pair consisting of {precondition, condition} is satisfied, then the considered alarm is emitted and some postcondition results. We say that a pre-state is valid for the considered scenario if it satisfies the pair {precondition, condition}.

Represent each triple {pre-state, alarm, post-state}, such that the pre-state is valid for the scenario of the 2nd diagram, in the form of a *transition* with its pre- and post-set, this is shown in the 3rd diagram. Having a token in each place indicates that the four considered variables are in the considered pre-state. Then, the considered alarm is emitted and a certain post-state results that must satisfy the postcondition and implies the emission of certain messages to the other elementary scenarios.

Equivalently, having a token in each place of the pre-set allows the firing of the transition, which results in the emission of the given alarm, the removal of one token from each input place and the adding of one token in each output place. Thus, the 3rd diagram is nothing but one transition of a Petri net. From our explanations, places can hold only 0 or 1 token, since they indicate whether the current value for their associated variable differs from or is equal to the particular value that the considered place represents.

To summarize, we have shown that the type of model of scenarios we used can be translated into *safe Petri nets,* i.e., Petri nets in which places are guaranteed to hold 0 or 1 token. We shall use the latter framework in the remainder of this paper.

3 Discussing distributed diagnosis using a toy example

In this study we consider a distributed system with asynchronous communications and concurrency, both between and within the different subsystems. Several mathematical frameworks could be considered for this purpose, and indeed used to develop our approach. We have chosen *safe Petri nets* as our mathematical framework, for the following reasons: 1/ safe Petri nets are a natural model of systems with local states, asynchronous communications, and internal concurrency, 2/ safe Petri nets can be composed, 3/ unfoldings and event structures have been extensively studied to represent executions of safe Petri nets with concurrency, and 4/ safe Petri nets are a convenient support for the intuition. In this section, we present and discuss a toy illustrative example used throughout the paper. Also, we introduce the minimal mathematical framework on safe Petri nets and their unfoldings that is needed to properly understand this example.

3.1 Prerequisites on safe Petri nets

Basic references are [9, 12, 25]. A *net* is a triple $\mathcal{N} = (P, T, \rightarrow)$, where P and T are disjoint sets of *places* and *transitions*, and $\rightarrow \subseteq (P \times T) \cup (T \times P)$ is the *flow relation.* Let $\preceq$ and $\prec$ denote the reflexive and irreflexive transitive closures of the flow relation $\rightarrow$, respectively. Places and transitions are called *nodes,* generically denoted by x. For $x \in P \cup T$, we denote by ${}^\bullet x = \{y : y \rightarrow x\}$ the *preset* of node x, and by $x^\bullet = \{y : x \rightarrow y\}$ its *post-set.* For $X \subset P \cup T$, we write ${}^\bullet X = \bigcup_{x \in X} {}^\bullet x$ and $X^\bullet = \bigcup_{x \in X} x^\bullet$.

For $\mathcal{N}$ a net, a *marking* of $\mathcal{N}$ is a multi-set M of places, i.e., a map $M : P \mapsto \{0, 1, 2, \ldots\}$. A *Petri net* is a pair $\mathcal{P} = (\mathcal{N}, M_0)$, where $\mathcal{N}$ is a net having finite sets of places and transitions, and M_0 is an *initial* marking. A transition $t \in T$ is *enabled* at marking M if $M(p) > 0$ for every $p \in {}^\bullet t$. Such a transition can *fire*, leading to a new marking $M' =$

$M - {}^\bullet t + t^\bullet$, denoted by $M[t\rangle M'$. Petri net $\mathcal{P}$ is *safe* if $M(P) \subseteq \{0,1\}$ for every reachable marking M. Throughout this paper, we consider only safe Petri nets, hence marking M can be regarded as a subset of places. The *language* $\mathcal{L}_\mathcal{P}$ of labeled Petri net $\mathcal{P}$ is the subset of A^* consisting of the words $\lambda(t_1), \lambda(t_2), \lambda(t_3), \ldots$, where $M_0[t_1\rangle M_1[t_2\rangle M_2[t_3\rangle M_3 \ldots$ ranges over the set of finite firing sequences of $\mathcal{P}$. Note that $\mathcal{L}_\mathcal{P}$ is prefix closed.

For $\mathcal{N} = (P, T, \rightarrow)$ a net, a *labeling* is a map $\lambda : T \mapsto A$, where A is some finite alphabet. A net $\mathcal{N} = (P, T, \rightarrow, \lambda)$ equipped with a labeling λ is called a *labeled net.* For $\mathcal{N}_i = \{P_i, T_i, \rightarrow_i, \lambda_i\}$, $i \in \{1,2\}$, two labeled nets, their *synchronous product* (or simply "product", for short) is defined as follows:

$$
\begin{array}{rcl}
\mathcal{N}_1 \times \mathcal{N}_2 & =_{\text{def}} & (P, T, \rightarrow, \lambda), \text{ where:} \\
P & = & P_1 \uplus P_2, \text{ where } \uplus \text{ denotes the disjoint union} \\
T & = & \left\{ \begin{array}{ll} & \{t =_{\text{def}} t_1 \in T_1 \mid \lambda_1(t_1) \in A_1 \setminus A_2\} \quad \text{(i)} \\ \cup & \{t =_{\text{def}} (t_1, t_2) \in T_1 \times T_2 \mid \lambda_1(t_1) = \lambda_2(t_2)\} \quad \text{(ii)} \\ \cup & \{t =_{\text{def}} t_2 \in T_2 \mid \lambda_2(t_2) \in A_2 \setminus A_1\}, \quad \text{(iii)} \end{array} \right. \\
p \rightarrow t \quad \text{iff} & & \left\{ \begin{array}{ll} p \in P_1 \text{ and } p \rightarrow_1 t_1 & \text{for case (i)} \\ \exists i \in \{1,2\} : p \in P_i \text{ and } p \rightarrow_i t_i & \text{for case (ii)} \\ p \in P_2 \text{ and } p \rightarrow_2 t_2 & \text{for case (iii)} \end{array} \right.
\end{array}
$$

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

For $\mathcal{N}_i = \{P_i, T_i, \rightarrow_i\}$, $i \in \{1,2\}$, two nets such that $T_1 \cap T_2 = \emptyset$, their *parallel composition* is the net

$$\mathcal{N}_1 \,||\, \mathcal{N}_2 \quad =_{\text{def}} \quad (P_1 \cup P_2, T_1 \cup T_2, \rightarrow_1 \cup \rightarrow_2). \tag{1}$$

Petri nets and occurrence nets inherit this notion. For Petri nets, we adopt the convention that the resulting initial marking is equal to $M_{1,0} \cup M_{2,0}$, the union of the two initial markings. Note that any safe Petri net is the parallel composition of its elementary nets consisting of a single transition together with its pre- and post-set.

3.2 Presenting the running example, and the problem

Our running example involves two interacting components. Both components can fail, independently. In addition, the 2nd component uses the services of the 1st one, therefore it fails delivering its service when the 1st component fails. Alarms reported do not distinguish between a true failure and a failure to delivery service due to the other component. Thus, nondeterminism results in the interpretation of alarm messages.

Our example is shown in Fig. 7, in the form of a labeled Petri net with two components interacting via parallel composition (1); these components are numbered 1 and 2. Component 2 uses the services of component 1, and therefore may fail to deliver its service

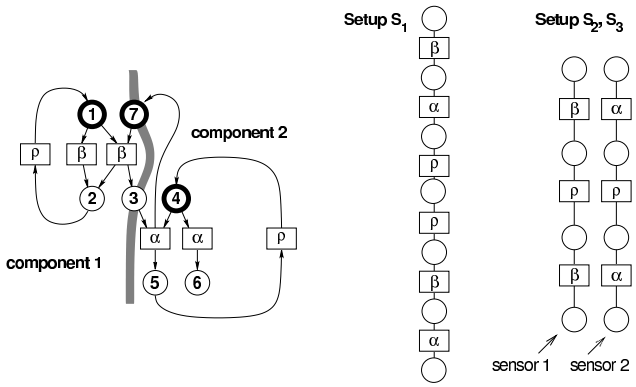

Figure 7: Running example in the form of a Petri net $\mathcal{P}$.

when component 1 is faulty. The two components interact via their shared places 3 and 7, represented by the gray zone; note that this Petri net is safe.

Component 1 has two private states: safe, represented by place 1, and faulty, represented by place 2. Upon entering its faulty state, component 1 emits an alarm β. The fault of component 1 is temporary, thus self-repair is possible and is represented by the label ρ. Component 2 has three private states, represented by places $4, 5, 6$. State 4 is safe, state 6 indicates that component 2 is faulty, and state 5 indicates that component 2 fails to deliver its service, due to the failure of component 1. Fault 6 is permanent and cannot be repaired.

The failure of component 2 caused by a fault of component 1 is modeled by the shared place 3. The monitoring system of component 2 only detects that component 2 fails to deliver its service, it does not distinguish between the different reasons for this. Hence the same alarm α is attached to the two transitions posterior to 4. Since fault 2 of component 1 is temporary, self-repair can also occur for component 2, when in faulty state 5. This self-repair is not synchronized with that of component 1, but bears the same label ρ. Finally, place 7 guarantees that fault propagation, from component 1 to 2, is possible only when the latter is in safe state.

The initial marking consists of the three states $1, 4, 7$. Labels (alarms α, β or self-repair ρ) attached to the different transitions or events, are generically referred to as *alarms* in the sequel.

Three different setups can be considered for diagnosis, assuming that messages are not lost:

Setup $\mathbf{S_1}$: The successive alarms are recorded in sequence by a single supervisor, in charge of fault monitoring. The sensor and communication infrastructure guarantees that causality is respected: for any two alarms such that α causes α', α is recorded before α'.

Setup $\mathbf{S}_2$: Each sensor records its local alarms in sequence, while respecting causality. The different sensors perform independently and asynchronously, and a single supervisor collects the records from the different sensors. Thus any interleaving of the records from different sensors is possible, and causalities among alarms from different sensors are lost.

Setup $\mathbf{S}_3$: The fault monitoring is distributed, with different supervisors cooperating asynchronously. Each supervisor is attached to a component, records its local alarms in sequence, and can exchange supervision messages with the other supervisors, asynchronously.

A simple solution?

For setup $\mathbf{S}_1$, there is a simple solution. Call $\mathcal{A}$ the recorded alarm sequence. Try to fire this sequence in the Petri net from the initial marking. Each time an ambiguity occurs (two transitions may be fired explaining the next event in $\mathcal{A}$), a new copy of the trial (a new Petri net) is instantiated to follow the additional firing sequence. Each time no transition can be fired in a trial to explain a new event, the trial is abandoned. Then, at the end of $\mathcal{A}$, all the behaviors explaining $\mathcal{A}$ have been obtained. Setup $\mathbf{S}_2$ can be handled similarly, by exploring all inter-leavings of the two recorded alarm sequences. However, this direct approach does not represent efficiently the set of all solutions to the diagnosis problem.

In addition, this direct approach does not work for Setup $\mathbf{S}_3$. In this case, no supervisor knows the entire net and no global interleaving of the recorded alarm sequences is available. Maintaining a coherent set of causally related local diagnoses becomes a difficult problem for which no straightforward solution works. The approach we propose in this paper addresses both the Setup $\mathbf{S}_3$ and the efficient representation of all solutions, for all setups. In the next section, we discuss this special representation, called *unfolding*.

3.3 Unfoldings: a data structure to represent all runs

Running example, continued. Fig. 8, 1st diagram, shows a variation of the net $\mathcal{P}$ of Fig. 7. The labels α, β, ρ have been discarded, and transitions are i, ii, iii, iv, v, vi. Places constituting the initial marking are indicated by thick circles.

To allow for a compact representation of all runs of a Petri net, the two following key ideas are used: 1/ represent each run as a partial oder (rather than a sequence) of events, and 2/ represent only once shared prefixes of different runs. This we explain next.

The mechanism of constructing a run of $\mathcal{P}$ in the form of a partial order is illustrated in the 1st and 2nd diagrams. Initialize any run of $\mathcal{P}$ with the three conditions labeled by the initial marking $(1, 7, 4)$. Append to the pair $(1, 7)$ a copy of the transition $(1, 7) \rightarrow i \rightarrow (2, 3)$. Append to the new place labeled 2 a copy of the transition $(2) \rightarrow iii \rightarrow (1)$. Append, to the pair $(3, 4)$, a copy of the transition $(3, 4) \rightarrow iv \rightarrow (7, 5)$ (this is the step shown). We have constructed (the prefix of) a run of $\mathcal{P}$. Now, all runs can be constructed in this way. Different runs can share some prefix.

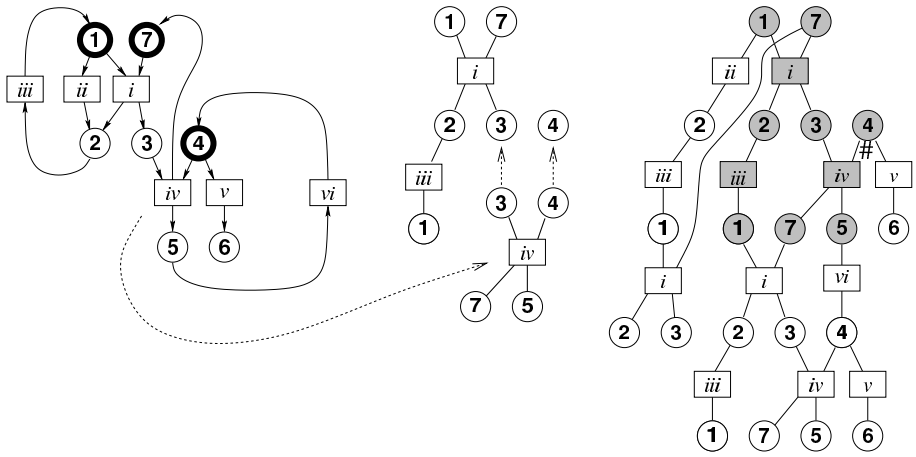

Figure 8: A Petri net (left), and representing its runs in a branching process. Petri nets are drawn by using directed arrows. Since occurrence nets are acyclic, we draw them using non-directed branches to be interpreted as implicitly directed toward bottom. Symbol # on the 3rd diagram indicates a source of conflict.

In the 3rd diagram we show (prefixes of) all runs, by superimposing their shared parts. The gray part of this diagram is a copy of the run shown in the 2nd diagram. The alternative run on the extreme left of this diagram (it involves successive transitions labeled ii, iii, i) shares only its initial places with the run in gray. On the other hand, replacing, in the gray run, the transition labeled iv by the one labeled v yields another run which shares with the gray one its transitions respectively labeled by i and by iii. This 3rd diagram is a *branching process* of $\mathcal{P}$, we denote it by $\mathcal{B}$; it is a net without cycle, in which the preset of any condition contains exactly one event. Nodes of $\mathcal{B}$ are labeled by places/transitions of $\mathcal{P}$ in such a way that the two replicate each other, locally around transitions. Branching processes can be extended, by inductively continuing the process of Fig. 8. The resulting limit is called the *unfolding* of $\mathcal{P}$, denoted by $\mathcal{U}_{\mathcal{P}}$.

Causality, conflict, concurrency. When dealing with unfoldings, to distinguish from the corresponding concepts in Petri nets, we shall from now on refer to *conditions/events* instead of places/transitions. Conditions or events are generically called *nodes*. Since unfoldings represent executions of Petri nets, they satisfy some particular properties:

- *Causality.* Unfoldings possess no cycle. Thus the transitive closure of the $\rightarrow$ relation is a partial order, we denote it by $\preceq$ and call it the *causality* relation. For example, the branch $(1) \rightarrow (ii) \rightarrow (2)$ sitting on the top left of the fourth diagram of Fig. 8 yields the causality $(1) \preceq (2)$. Causality is the proper concept of "time" for executions of Petri nets.

- *Conflict.* Unfoldings are such that the preset of any condition contains exactly one event. However, its post-set can contain two or more different events, as shown by the subnet $(1) \rightarrow (ii, i)$ sitting on the top left of the fourth diagram of Fig. 8. This indicates that the initial condition labeled by (1) can be followed, in one execution, by an event labeled by ii, or, in a different execution, by an event labeled by i. A condition having a post-set with two events or more indicates the branching of different executions, from the considered condition. Conditions or events belonging to different executions are called in *conflict.* The conflict relation is denoted by the symbol $\#$. Clearly, the conflict relation is closed under causality: if $x \# x'$ holds for two nodes, and $x \preceq y, x' \preceq y'$, then $y \# y'$ follows. Thus sources of conflict are important, Fig. 8 shows an example.

- *Concurrency.* Executions are represented by maximal sets of nodes involving no conflict. In an execution, nodes can be either causally related, or *concurrent.* Thus two nodes x, y are concurrent iff none of the following conditions hold: $x \# y$, $x \preceq y$, $y \preceq x$. Thus concurrency is an ancillary relation, derived from knowing both the causality and conflict relations. Concurrent nodes model "independent progress" within an execution. Concurrency is an important concept in distributed systems.

As the above introduced concepts are subtle, we formalized them now.

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

(i) $\forall x \in B \cup E : \neg[x \# x]$ ($\preceq$ is a partial order);
(ii) $\forall x \in B \cup E : \neg[x \prec x]$ ($\preceq$ is a partial order);
(iii) $\forall x \in B \cup E : |\{y : y \prec x\}| < \infty$ ($\preceq$ is well founded);
(iv) $\forall b \in B : |{}^\bullet b| \leq 1$ (each place has at most one input transition).

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

Nodes x and x' are *concurrent,* written $x \perp\!\!\!\perp x'$, if neither $x \preceq x'$, nor $x \preceq x'$, nor $x \# x'$ hold. A *co-set* is a set X of pairwise concurrent conditions. A *configuration* is a sub-net κ of $\mathcal{O}$, which is *conflict-free* (no two nodes are in conflict), *causally closed* (if $x' \preceq x$ and $x \in \kappa$, then $x' \in \kappa$), and contains $\min(\mathcal{O})$. In the sequel, we will only consider well-formed configurations, i.e., configurations κ such that every event contained in κ has its entire post-set also contained in κ—this will not be mentioned any more.

A *homomorphism* from a net $\mathcal{N}$ to a net $\mathcal{N}'$ is a map $\varphi : P \cup T \mapsto P' \cup T'$ such that: (i) $\varphi(P) \subseteq P'$, $\varphi(T) \subseteq T'$, and (ii) for every transition t of $\mathcal{N}$, the restriction of φ to ${}^\bullet t$ is a bijection between ${}^\bullet t$ and ${}^\bullet \varphi(t)$, and the restriction of φ to $t^\bullet$ is a bijection between $t^\bullet$ and

$\varphi(t)^\bullet$. Reverting the dashed curved arrow relating the 1st and 2nd diagrams of Fig. 8 yields an illustration of this notion.

A *branching process* of Petri net $\mathcal{P}$ is a pair $\mathcal{B} = (\mathcal{O}, \varphi)$, where $\mathcal{O}$ is an occurrence net, and φ is a homomorphism from $\mathcal{O}$ to $\mathcal{P}$ regarded as nets, such that: (i) the restriction of φ to $\min(\mathcal{O})$ is a bijection between $\min(\mathcal{O})$ and M_0 (the set of initially marked places), and (ii) for all $e, e' \in E$, ${}^\bullet e = {}^\bullet e'$ and $\varphi(e) = \varphi(e')$ together imply $e = e'$. By abuse of notation, we shall sometimes write $\min(\mathcal{B})$ instead of $\min(\mathcal{O})$. The set of all branching processes of Petri net $\mathcal{P}$ is uniquely defined, up to an isomorphism (i.e., a renaming of the conditions and events), and we shall not distinguish isomorphic branching processes. For $\mathcal{B}, \mathcal{B}'$ two branching processes, $\mathcal{B}'$ is a *prefix* of $\mathcal{B}$, written $\mathcal{B}' \sqsubseteq \mathcal{B}$, if there exists an injective homomorphism ψ from $\mathcal{B}'$ into $\mathcal{B}$, such that $\psi(\min(\mathcal{B}')) = \min(\mathcal{B})$, and the composition $\varphi \circ \psi$ coincides with φ', where $\circ$ denotes the composition of maps. By theorem 23 of [13], there exists (up to an isomorphism) a unique maximum branching process according to $\sqsubseteq$,

$$\text{we denote it by } \mathcal{U}_\mathcal{P} \text{ and call it the } \textit{unfolding} \text{ of } \mathcal{P}. \tag{2}$$

Maximal configurations of $\mathcal{U}_\mathcal{P}$ are called *runs* of $\mathcal{P}$. The unfolding of $\mathcal{P}$ possesses the following universal property: for every occurrence net $\mathcal{O}$, and every homomorphism $\phi : \mathcal{O} \mapsto \mathcal{P}$ such that $\phi(\min(\mathcal{O}) \subseteq M_0$, there exists an injective homomorphism $\iota : \mathcal{O} \mapsto \mathcal{U}_\mathcal{P}$, such that: $\phi = \varphi \circ \iota$, where φ denotes the homomorphism associated to $\mathcal{U}_\mathcal{P}$. This decomposition expresses that $\mathcal{U}_\mathcal{P}$ "maximally unfolds" $\mathcal{P}$. If $\mathcal{P}$ is itself an occurrence net and $M_0 = \min(\mathcal{P})$ holds, then $\mathcal{U}_\mathcal{P}$ identifies with $\mathcal{P}$. Fig. 8 illustrates the incremental construction of the unfolding of a Petri net.

Having this material at hand, in the next subsections we discuss diagnosis under the three setups $\mathbf{S}_1$, $\mathbf{S}_2$, and $\mathbf{S}_3$.

3.4 Asynchronous diagnosis with a single sensor and supervisor

Here we consider setup $\mathbf{S}_1$, and our discussion is supported by Fig. 9 and Fig. 10. The 1st diagram of Fig. 9 is the alarm sequence $\beta, \alpha, \rho, \rho, \beta, \alpha$ recorded at the unique sensor. It is represented by a cycle-free, linear Petri net, whose conditions are not labeled—conditions have no particular meaning, their only purpose is to indicate the ordering of alarms. Denote by $\mathcal{A}' = \beta \to \alpha \to \rho$ the shaded prefix of $\mathcal{A}$.

The 2nd diagram of Fig. 9 shows the net $\mathcal{U}_{\mathcal{A}' \times \mathcal{P}}$, obtained by unfolding the product $\mathcal{A}' \times \mathcal{P}$ using the procedure explained in the figure 8. The net $\mathcal{U}_{\mathcal{A}' \times \mathcal{P}}$ shows how successive transitions of $\mathcal{P}$ synchronize with transitions of $\mathcal{A}'$ having identical label, and therefore explain them. The curved branches of this diagram indicate the contribution of $\mathcal{A}'$ to this unfolding, whereas the straight branches indicate the contribution of $\mathcal{P}$. This unfoldings reveals that three different explanations exist for $\mathcal{A}'$. Note the source of conflict (marked by $\#$) that is attached to a condition labeled by ii; this conflict propagates, by causality, to the conflict between the two events labeled by ρ that is marked by a larger $\#$.

We are not really interested in showing the contribution of $\mathcal{A}'$ to this unfolding. Thus we project it away. The result is shown on the 1st diagram of Fig. 10. The dashed line labeled

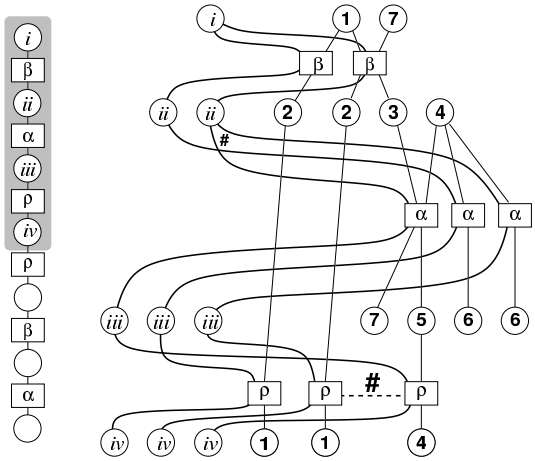

Figure 9: Asynchronous diagnosis with a single sensor: showing an alarm sequence $\mathcal{A}$ (1st diagram) and the explanation of the prefix $\mathcal{A}' = \beta \rightarrow \alpha \rightarrow \rho$ in the form of the unfolding $\mathcal{U}_{\mathcal{A}' \times \mathcal{P}}$ (2nd diagram).

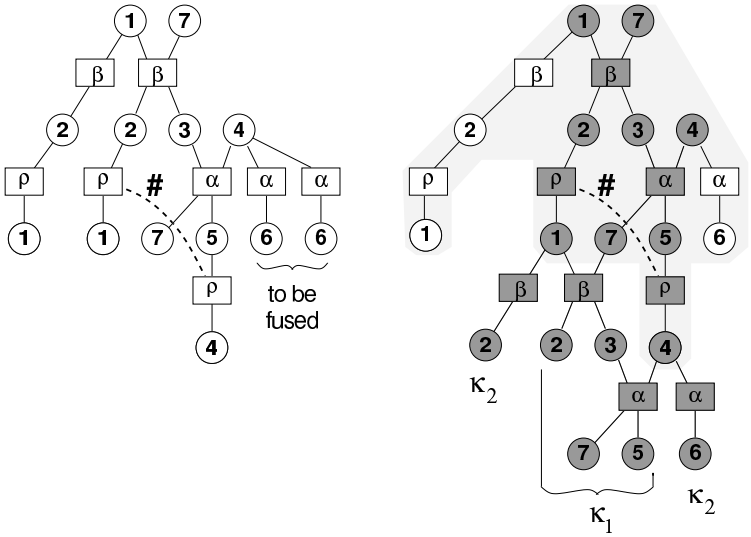

Figure 10: Erasing the places related to the alarm sequence $\mathcal{A}'$ in the 2nd diagram of Fig. 9 yields the 1st diagram of this figure. A full explanation of $\mathcal{A}$ is given in the 2nd diagram of this figure.

$\#$ originates from the corresponding conflict in $\mathcal{U}_{\mathcal{A}'\times\mathcal{P}}$ that is due to two different conditions explaining the same alarm ρ, cf. above. Thus we need to remove, as possible explanations of the prefix, all runs of the 3rd diagram that contain the $\#$-linked pair of events labeled ρ. All remaining runs are valid explanations of the subsequence β, α, ρ. However, the reader will notice the duplicated path $4 \to \alpha \to 6$. As this duplication is unnecessary, we fuse the two isomorphic paths of the form $4 \to \alpha \to 6$. The latter "trimming" operation will be systematically applied from now on when discussing our example.

Finally, the net shown in the 2nd diagram of Fig. 10 contains a prefix consisting of the nodes filled in dark gray. The white nodes correspond to runs that can explain the prefix $\mathcal{A}'$ but not the entire $\mathcal{A}$. The gray prefix is the union of the two runs κ_1 and κ_2 of $\mathcal{P}$, that explain $\mathcal{A}$ entirely, namely [3]:

$$\kappa_1 = \left\{ \begin{array}{ll} & (1,7) \to \beta \to (2,3) \\ \cup & (3,4) \to \alpha \to (7,5) \\ \cup & (2) \to \rho \to (1) \\ \cup & (5) \to \rho \to (4) \\ \cup & (1,7) \to \beta \to (2,3) \\ \cup & (3,4) \to \alpha \to (7,5) \end{array} \right. \qquad \kappa_2 = \left\{ \begin{array}{ll} & (1,7) \to \beta \to (2,3) \\ \cup & (3,4) \to \alpha \to (7,5) \\ \cup & (2) \to \rho \to (1) \\ \cup & (5) \to \rho \to (4) \\ \cup & (1) \to \beta \to (2) \\ \cup & (4) \to \alpha \to (6) \end{array} \right. \tag{3}$$

Warning: a flash forward to *event structures*. The reader is kindly asked to confront the diagrams of Fig. 10 with the formal definition of occurrence nets as provided in section 3.3. She or he will recognize that these diagrams are not occurrence nets: the additional conflict shown on the 1st diagram is not explained by the topological structure of the net, since the two conflicting events share in their past an event, not a condition. The same remark holds for the 2nd diagram.

We kindly ask our gentle reader to wait until Section 4.1, where the adequate notion of *event structure* is introduced to properly encompass the last two diagrams—for the moment, we shall continue to freely use diagrams of this kind.

Finally, referring to the 1st diagram, it seems reasonable to fuse the two isomorphic paths of the form $4 \to \alpha \to 6$. This is indeed what our operation of event structure *trimming* will perform, see Section 4.2.

3.5 Asynchronous diagnosis with two concurrent sensors and a single supervisor

Focus on setup $\mathbf{S}_2$, in which alarms are recorded by two independent sensors, and then collected at a single supervisor for explanation. Fig. 11 shows the same alarm history as in Fig. 9, except that it has been recorded by two independent sensors, respectively attached to each component. The supervisor knows the global model of the system, we recall it in the 1st diagram of Fig. 11.

[3] Strictly speaking, our projection operation creates two respective clones of κ_1 and κ_2 by exchanging, in (3), the two lines explaining the ρ-alarms. But the two resulting pairs of isomorphic configurations are fused by our "trimming" operation, hence we did not show these clones.

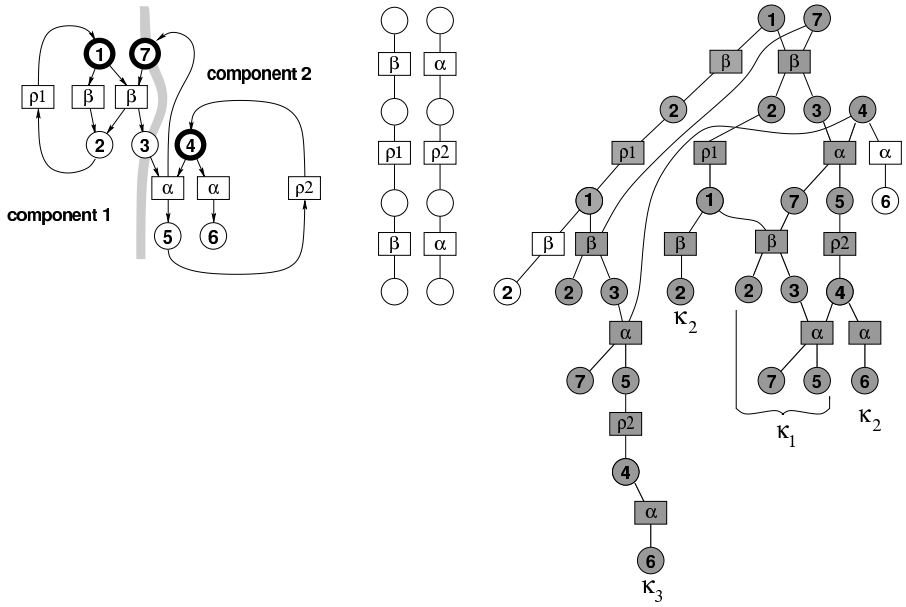

Figure 11: Asynchronous diagnosis with two independent sensors: showing an alarm pattern $\mathcal{A}$ (middle) consisting of two concurrent alarm sequences, and its explanation (right).

The two "repair" actions are now distinguished since they are seen by different sensors, this is why we use different labels: ρ_1, ρ_2. This distinction reduces the ambiguity: in Fig. 11 we suppress the white filled path $(2) \rightarrow \rho \rightarrow (1)$ that occurred in Fig. 10. On the other hand, alarms are recorded as two concurrent sequences, one for each sensor, call the whole an *alarm pattern*. Causalities between alarms from different components are lost. This leads to further ambiguity, as shown by the additional configuration κ_3 that can explain the alarm pattern in Fig. 11, compare with Fig. 10. The valid explanations for the entire alarm pattern are the three configurations κ_1, κ_2 and κ_3 filled in dark gray in the 3rd diagram. To limit the complexity and size of the figures, we will omit the "long" configuration κ_3 in the sequel.

3.6 Distributed diagnosis with two concurrent sensors and supervisors

Consider setup $\mathbf{S}_3$, in which alarms are recorded by two independent sensors, and processed by two local supervisors which can communicate asynchronously. Fig. 12 shows two branching processes, respectively local to each supervisor. For completeness, we have shown the information available to each supervisor. It consists of the local model of the component considered, together with the locally recorded alarm pattern. The process constructed by supervisor 1 involves only events labeled by alarms collected by sensor 1, and places that

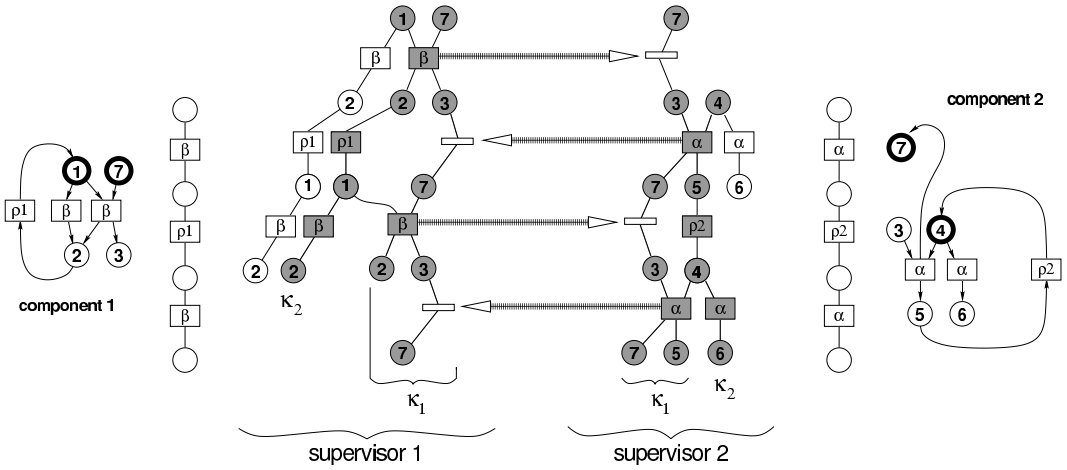

Figure 12: Distributed diagnosis: constructing two coherent local views of the branching process $\mathcal{U}_{\mathcal{P},\mathcal{A}}$ of Fig. 11 by two supervisors cooperating asynchronously (for simplicity, configuration κ_3 of Fig. 11 has been omitted.)

are either local to component 1 (e.g., $1, 2$) or shared (e.g., $3, 7$); and similarly for the process constructed by supervisor 2.

The 3rd diagram of Fig. 11 can be recovered from Fig. 12 in the following way: glue events sitting at opposite extremities of each thick dashed arrow, identify adjacent conditions, and remove the thick dashed arrows. These dashed arrows indicate a communication between the two supervisors, let us detail the first one. The first event labeled by alarm β belongs to component 1, hence this explanation for β has to be found by supervisor 1. Supervisor 1 sends an abstraction of the path $(1,7) \rightarrow \beta \rightarrow (2,3)$ by removing the local conditions $1, 2$ and the label β since the latter do not concern supervisor 2. Thus supervisor 2 receives the path $(7) \rightarrow [] \rightarrow (3)$ to which it can append its local event $(3,4) \rightarrow \alpha \rightarrow (7,5)$; and so on.

Discussion: handling asynchronous communications. The cooperation between the two supervisors needs only asynchronous communication. Each supervisor can simply "emit and forget." Diagnosis can progress concurrently and asynchronously at each supervisor.

For example, supervisor 1 can construct the branch $[1 \rightarrow \beta \rightarrow 2 \rightarrow \rho_1 \rightarrow 1 \rightarrow \beta \rightarrow 2]$ as soon as the corresponding local alarms are collected, without ever synchronizing with supervisor 2. Assume some (finite but possibly unbounded) communication delay between the two supervisors. Consider the explanations of the second occurrence of alarm β by the 1st supervisor (there are three of them). The left most two do not require any synchronization with the supervisor 2. Thus they can be produced as soon as the local alarm sequence β, ρ_1, β has been observed, independently from what supervisor 2 is doing, i.e., *concurrently* with supervisor 2. In contrast, the right most explanation needs to synchronize with su-

pervisor 2, since it waits for the abstraction (3) $\rightarrow$ [] $\rightarrow$ (7) sent by supervisor 2. Thus this third explanation may suffer from some (finite but possibly unbounded) communication delay. However this will not impact the production of the first two explanations. This perfectly illustrates how a concurrency approach allows to handle asynchronous communications. This should be compared with the approaches proposed by Lafortune et al. [11][20] where essentially synchronous communications, from sensors to supervisors and between the different supervisors, is required.

4 Event structures and their use in asynchronous diagnosis

In section 3.4 we announced the need to consider event structures. This section is devoted to their introduction for the purpose of asynchronous diagnosis.

4.1 Prime event structures

Running example, continued. Fig. 13 shows in (a) the 1st diagram of Fig. 10. Focus for

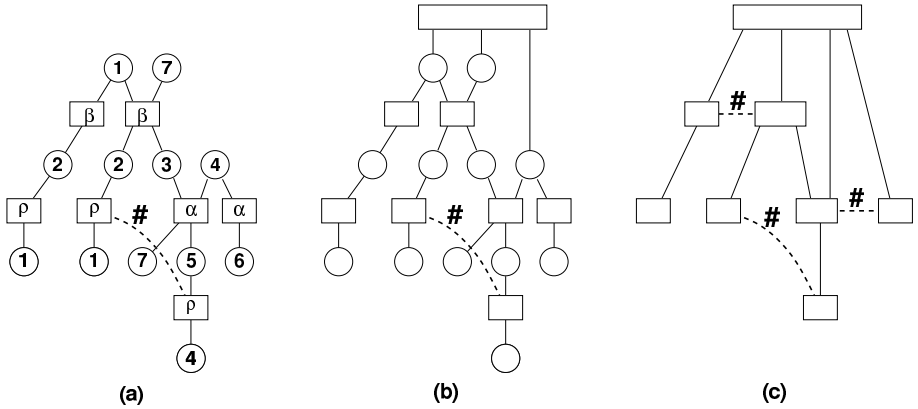

Figure 13: The informal labeled occurrence net (a), taken from Fig. 9, 3rd diagram (conditions are figured by circles and events are figured by boxes). Erasing the labels of events and adding an initial event yields the net (b). The resulting *event structure* is shown in diagram (c).

the moment on the topological structure of this diagram by ignoring labels, and add an initial event: this yields the net (b). In net (b), sources of conflicts are either mentioned explicitly, or inferred from the graph topology by searching for downward branching conditions. This dual way of indicating conflict is not elegant. Thus, we prefer to omit conditions and represent explicitly all sources of conflicts between events—conflict will be inherited by causality. Performing this yields the *event structure* depicted in (c), where the down-going

branches indicate causality, and sources of conflict are explicitly indicated. In this structure, the information regarding labels has been lost. We shall show later how to add it properly to diagram (c). ◇

We are now ready to introduce the mathematics of event structures.

Prime event structures: formal definition. Event structures have been introduced in [23], and further extensively studied by G. Winskel [31, 33] and several authors since then. Several classes of event structures have been proposed, by relaxing the conditions required on the conflict relation and/or exchanging the causality relation for a more general "enabling" relation. Equipping prime event structures with parallel composition has been recognized quite complex. An inductive definition is presented in [10]. Indirect, non inductive, definitions have been proposed by G. Winskel in [33]. F. Vaandrager [30] has proposed a simple direct, non inductive, definition, in categorical style. This definition suits our needs. Here we summarize the results from [30], with minimal changes in the notations.

A *prime event structure*[4] is a triple $\mathcal{E} = (E, \preceq, \#)$, where E is a set of *events*, $\preceq$ is a partial order on E such that for all $e \in E$, the set $\{e' \in E \mid e' \preceq e\}$ is finite, and $\#$ is a symmetric and irreflexive relation on E such that for all $e_1, e_2, e_3 \in E$, $e_1 \# e_2$ and $e_2 \preceq e_3$ imply $e_1 \# e_3$.[5] Each subset of events $F \subseteq E$ induces a *substructure* $\mathcal{E}_{|F} = (F, \preceq_{\mathcal{F}}, \#_{\mathcal{F}})$, by restricting to F the relations $\preceq$ and $\#$.

As usual, we write $e \prec e'$ for $e \preceq e'$ and $e \neq e'$. We write $\lceil e \rceil$ for the set $\{e' \in E | e' \preceq e\}$ and we call it the *configuration generated by* e. For $\mathcal{E} = (E, \preceq, \#)$ an event structure, a subset X of E is called *causally closed* if $e \in X$ implies $\lceil e \rceil \subseteq X$. Subset X is called *conflict-free* if no pair of elements of X are in conflict, i.e., $X \times X \cap \# = \emptyset$. A *configuration* is a causally closed conflict-free subset of E. Each event structure $\mathcal{E} = (E, \preceq, \#)$ induces a *concurrency* relation defined by $e \perp\!\!\!\perp e'$ iff neither $e \preceq e'$ nor $e' \preceq e$ nor $e \# e'$ holds. A subset X of concurrent events is called a *co-set*.

Morphisms. We will use partial functions. We indicate that ψ is a partial function from X to Y by writing $\psi : X \mapsto_\star Y$. The domain of ψ is denoted by $dom(\psi)$. Since $\psi(x)$ may not be defined for $x \in X$, we indicate this by writing $\psi(x) = \star$, thus symbol "$\star$" is to be interpreted as "undefined".

$$\text{For } \psi : X \mapsto_\star Y \text{ and } X' \subseteq X, \text{ set } \psi(X') =_{\text{def}} \{\psi(x) \mid x \in X'\}. \tag{4}$$

A *morphism* from $\mathcal{E}_1$ to $\mathcal{E}_2$ is a partial function $\psi : E_1 \mapsto_\star E_2$ such that:

$$\forall (e_1, e_2) \in E_1 \times E_2 : e_2 \prec_2 \psi(e_1) \Rightarrow \exists e_1' \in E_1, e_1' \prec_1 e_1 \text{ and } \psi(e_1') = e_2 \tag{5}$$

$$\forall e_1, e_1' \in E_1 : \psi(e_1) \#_2 \psi(e_1') \text{ or } \psi(e_1) = \psi(e_1') \Rightarrow e_1 \#_1 e_1' \text{ or } e_1 = e_1' \tag{6}$$

[4] From now on, when referring to prime event structures, we shall omit the term "prime", unless it is required for the point being discussed.

[5] Obviously, restricting an occurrence net to its set of events yields a prime event structure. This is the usual way of associating nets and event structures, and explains the name.

Conditions (5,6) state that morphisms can erase but cannot create causalities and conflicts. Condition (5) can be equivalently reformulated as follows:

$$\forall e_1 \in E_1 : \psi(e_1) \text{ defined } \Rightarrow \lceil \psi(e_1) \rceil \subseteq \psi(\lceil e_1 \rceil) \tag{7}$$

and the following result is proved in [30]:

$$X \text{ is a configuration of } \mathcal{E}_1 \quad \Rightarrow \quad \psi(X) \text{ is a configuration of } \mathcal{E}_2, \tag{8}$$

it shows that morphisms are indeed a natural notion. In [30] it is proved that prime event structures with morphisms of event structures form a category $\underline{\underline{\mathcal{E}}}$ with the usual composition of partial functions as composition and the identity functions on events as identity morphisms.

4.2 Labeled event structures and trimming

As discussed at the end of Section 3, we are mainly interested in event structures originating from net unfoldings. The homomorphism φ mapping unfolding $\mathcal{U}_\mathcal{P}$ to $\mathcal{P}$ yields a natural labeling of the events of $\mathcal{U}_\mathcal{P}$ in terms of transitions of $\mathcal{P}$. Thus, net unfoldings induce naturally event structures in which events are labeled by transitions of $\mathcal{P}$.

However, as seen from the illustrative example of Section 3, interactions between components and supervisors occur via shared places, and diagnosis is naturally expressed in terms of sequences of markings. Therefore transitions of the underlying Petri nets play little role in distributed diagnosis. Hence, we shall rather label events of $\mathcal{U}_\mathcal{P}$ by the post-set of their associated transition. Formally,

$$\text{we label event } e \in \mathcal{U}_\mathcal{P} \text{ by } \varphi(e)^\bullet \in Pow(P), \tag{9}$$

where Pow denotes the power set.

Running example, continued. Diagram (c) of Fig. 14 shows how labels of the form (9) can be inserted in our case. The reader is invited to reconsider Fig. 9 – Fig. 12 by making systematically the changes (a)↦(b)↦(c). ◇

The above discussion motivates the special kind of labeling we formally introduce now.

Labeling. For $\mathcal{E} = (E, \preceq, \#)$ an event structure, a *labeling* is a map

$$\lambda : E \mapsto Pow(P) \setminus \{\emptyset\} \tag{10}$$

where P is some finite alphabet; we extend (10) by convention by putting $\lambda(\star) = \emptyset$. Labeled event structures are denoted by $\mathcal{E} = (E, \preceq, \#, \lambda, P)$, and P is called the *label set,* by abuse of notation—the reader is kindly asked to remember that labels are subsets, not elements of label set P. We shall not distinguish labeled event structures that are identical up to a bijection that preserves labels, causalities, and conflicts; such event structures are considered equal, denoted by the equality symbol $=$. The notions of substructure and morphism need to be revisited to accommodate for labeling.

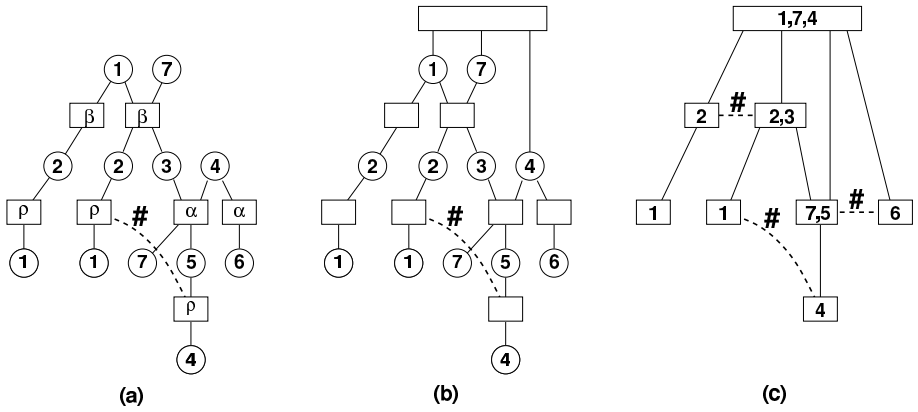

Figure 14: Adding labels to event structures. Following (9), the event structure of Fig. 10 has been enriched with the labels of the postset of each event.

Substructure. Let $\mathcal{E} = (E, \preceq, \#, \lambda, P)$ be a labeled event structure, and let $F \subseteq E$ and $Q \subseteq P$. Pair (F, Q) induces the substructure

$$\mathcal{E}_{|F,Q} \tag{11}$$

having $E_{F,Q} =_{\text{def}} \{e \in F \mid \lambda(e) \cap Q \neq \emptyset\}$ as set of events, and $\lambda_{F,Q}(e) = \lambda(e) \cap Q$ as labeling map. The causality and conflict relations are inherited by restriction.

Morphisms. For $\mathcal{E}_i = (E_i, \preceq_i, \#_i, \lambda_i, P_i)$, $i \in \{1, 2\}$ two labeled event structures such that $P_2 \subseteq P_1$, a *morphism* is a partial function $\psi : E_1 \mapsto_\star E_2$ satisfying conditions (5,6), plus the following monotonicity condition regarding labels:

$$\forall e_1 \in E_1 \cap dom(\psi) \quad : \quad \lambda_2(\psi(e_1)) = \lambda_1(e_1) \cap P_2. \tag{12}$$

By (12) and since events different from $\star$ must have a non empty label, we know that $dom(\psi) \subseteq \{e_1 \in E_1 \mid \lambda_1(e_1) \cap P_2 \neq \emptyset\}$. A morphism satisfying

$$dom(\psi) \quad = \quad \{e_1 \in E_1 \mid \lambda_1(e_1) \cap P_2 \neq \emptyset\} \tag{13}$$

is called a *strong morphism.* Strong morphisms compose. Thus we can consider two categories of labeled event structures, namely:

- The category $\underline{\underline{\mathcal{E}}}_s$ of labeled event structures equipped with *strong morphisms.*
- The category $\underline{\underline{\mathcal{E}}}_w$ of labeled event structures equipped with *weak morphisms,* i.e., morphisms satisfying (12) but not necessarily (13).

Most results we give below apply to both categories. To avoid mentioning systematically "strong" or "weak", we will simply refer to the category of labeled event structures $\underline{\underline{\mathcal{E}}}$ equipped with morphisms. This will refer either to $\underline{\underline{\mathcal{E}}}_s$ or to $\underline{\underline{\mathcal{E}}}_w$, in a consistent manner. A few results will hold for only one of these two categories; we will make this explicit in each case.

Trimming and reduction. When discussing our example, we have indicated that unnecessary replicas of parts of the diagnosis can occur. Here we discuss how to remove these. Fig. 15 shows in (a) a replica of 1st diagram of Fig. 10 with its suggestion for "trimming".

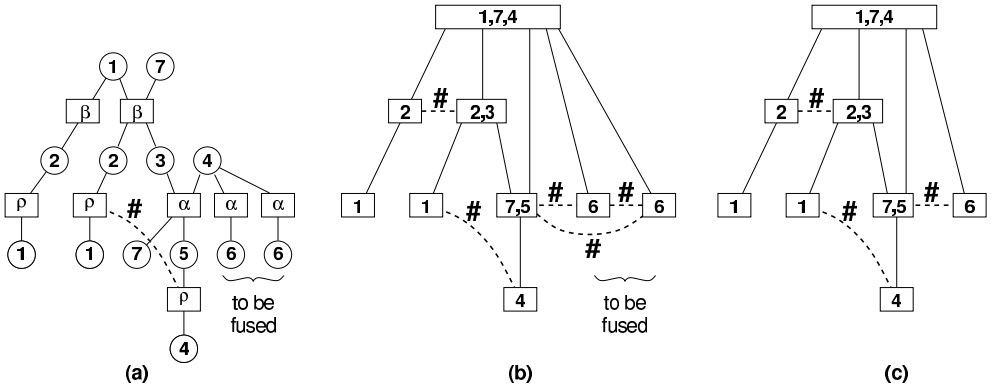

Figure 15: Illustrating trimming.

Diagram (b) shows the labeled event structure corresponding to (a). Finally, diagram (c) shows the result of applying, to (b), the *trimming* operator defined next. ◇

Let $\mathcal{E} = (E, \preceq, \#, \lambda, P)$ be a labeled event structure. Denote by $\rightarrow$ the successor relation, i.e., the transitive reduction of the relation $\preceq$. For $e \in E$, we denote by ${}^\bullet e$ the preset of e in $(E, \rightarrow)$. Then, $\mathcal{E}$ is called *trimmed* iff it satisfies the following condition:

$$\forall e, e' \in E : \left.\begin{array}{l} {}^\bullet e = {}^\bullet e' \\ \text{and} \quad \lambda(e) = \lambda(e') \end{array}\right\} \Rightarrow \quad e = e'. \tag{14}$$

Informally, $\mathcal{E}$ is trimmed iff any two configurations that have produced identical label histories are identical. Any labeled event structure $\mathcal{E} = (E, \preceq, \#, \lambda, P)$ can be made trimmed as explained next. Consider the following equivalence relation on configurations:

$$\kappa \sim \kappa' \quad \text{iff} \quad \left\{\begin{array}{l} \kappa \text{ and } \kappa' \text{ are isomorphic,} \\ \text{when seen as labeled partial orders.} \end{array}\right. \tag{15}$$

The equivalence class of κ modulo $\sim$ is denoted by $\kappa_\sim$; it represents the label history of the configuration κ. Define the function *trim* by:

$$trim : E \ni e \quad \mapsto \quad \lceil e \rceil_\sim$$

Informally, $trim(e)$ is the label history causing event e to occur. Define:

$$trim(\mathcal{E}) \quad = \quad (E_{\sqcap}, \preceq_{\sqcap}, \#_{\sqcap}, \lambda_{\sqcap}, P), \tag{16}$$

where

$$\begin{array}{rcl} E_{\sqcap} & = & trim(E) \\ \preceq_{\sqcap} & = & \subseteq \\ f_1 \#_{\sqcap} f_2 & \text{iff} & e_1 \# e_2 \text{ holds } \forall (e_1, e_2) \text{ such that} \\ & & f_i = trim(e_i) \text{ holds, for } i \in \{1, 2\} \\ \lambda_{\sqcap}(f) = \lambda(e) & \text{iff} & f = trim(e). \end{array} \tag{17}$$

Informally, $trim(\mathcal{E})$ is obtained by inductively superimposing events that satisfy the conditions listed on the left hand side of the bracket in (14); $trim(\mathcal{E})$ is a trimmed event structure, and $trim$ is a (total) morphism from $\mathcal{E}$ onto $trim(\mathcal{E})$. The map $trim$ satisfies the following self-reproducing property on labels:

$$\lceil f \rceil \quad \sim \quad \lceil e \rceil \text{ if } f = trim(e), \tag{18}$$

meaning that configurations $\lceil f \rceil$ and $\lceil e \rceil$ possess identical label histories.

For $\mathcal{E} = (E, \preceq, \#, \lambda, P)$ a labeled event structure and $Q \subseteq P$, we write by abuse of notation (cf. (11))

$$\mathcal{E}_{|Q} \quad =_{\text{def}} \quad \mathcal{E}_{|E_Q, Q} \tag{19}$$

where $E_Q = \{e \in E \mid \lambda(e) \cap Q \neq \emptyset\}$. Define the *reduction of $\mathcal{E}$ over Q* by:

$$\mathbf{R}_Q(\mathcal{E}) \quad =_{\text{def}} \quad trim\left(\mathcal{E}_{|Q}\right). \tag{20}$$

4.3 Event structures obtained from unfoldings

Let $\mathcal{P} = (P, T, \rightarrow, M_0)$ be a Petri net, $\mathcal{U}_{\mathcal{P}}$ its unfolding, and φ the associated net homomorphism. Denote by

$$\mathcal{E}_{\mathcal{P}} \quad = \quad (E, \preceq, \#, \lambda, P) \tag{21}$$

the trimmed event structure obtained by

1. labeling the events e of $\mathcal{U}_{\mathcal{P}}$ by $\lambda(e) =_{\text{def}} \varphi(e^{\bullet})$;
2. erasing the conditions in $\mathcal{U}_{\mathcal{P}}$ and restricting relations $\preceq$ and $\#$ accordingly;
3. adding an extra event e_0 such that $e_0 \preceq e$ for each event e of $\mathcal{U}_{\mathcal{P}}$ and labeling e_0 by $\lambda(e_0) = M_0$;
4. trimming the so obtained labeled event structure.

5 Distributed diagnosis: formal problem setting

We are now ready to formally state the problem of distributed diagnosis. We are given the following labeled Petri nets:

$\mathcal{P} = (P, T, \rightarrow, M_0, \lambda)$**:** the underlying "true" system. $\mathcal{P}$ is subject to faults, thus places from $\mathcal{P}$ are labeled by *faults,* taken from some finite alphabet (the non-faulty status is just one particular "fault"). The labeling map λ associates, to each transition of $\mathcal{P}$, a label belonging to some finite alphabet A of *alarm labels.* For its supervision, $\mathcal{P}$ produces so-called *alarm patterns,* i.e., sets of causally related alarms.

$\mathcal{Q} = (P^{\mathcal{Q}}, T^{\mathcal{Q}}, \rightarrow, M_0^{\mathcal{Q}}, \lambda^{\mathcal{Q}})$**:** $\mathcal{Q}$ represents the behavior of $\mathcal{P}$, *as observed via the sensor system.* Thus we require that: (i) The labeling maps of $\mathcal{Q}$ and $\mathcal{P}$ take their values in the same alphabet A of alarm labels, and (ii) $\mathcal{L}_{\mathcal{Q}} \supseteq \mathcal{L}_{\mathcal{P}}$, i.e., the language of $\mathcal{Q}$ contains the language of $\mathcal{P}$. In general, however, $\mathcal{Q} \neq \mathcal{P}$. For example, if a single sensor is assumed, which collects alarms in sequence by preserving causalities, then $\mathcal{Q}$ is the net which produces all linear extensions of runs of $\mathcal{P}$. In contrast, if several independent sensors are used, then the causalities between events collected by different sensors are lost. Configurations of $\mathcal{Q}$ are called *alarm patterns.*

5.1 Global diagnosis

Consider the map: $\mathcal{A} \mapsto \mathcal{U}_{\mathcal{A}\times\mathcal{P}}$, where $\mathcal{A}$ ranges over the set of all finite alarm patterns. This map filters out, during the construction of the unfolding $\mathcal{U}_{\mathcal{P}}$, those configurations which are not compatible with the observed alarm pattern $\mathcal{A}$. We can replace the unfolding $\mathcal{U}_{\mathcal{A}\times\mathcal{P}}$ by the corresponding event structure $\mathcal{E}_{\mathcal{A}\times\mathcal{P}}$. Then, we can project away, from $\mathcal{E}_{\mathcal{A}\times\mathcal{P}}$, the events labeled by places from $\mathcal{A}$ (see [6]–Theorem 1 for details). Thus we can state:

Definition 1 *Global diagnosis is represented by the following map:*

$$\mathcal{A} \longmapsto \mathbf{R}_P(\mathcal{E}_{\mathcal{A}\times\mathcal{P}}), \tag{22}$$

where $\mathcal{A}$ ranges over the set of all finite configurations of $\mathcal{Q}$.

5.2 Distributed diagnosis

Assume that Petri net $\mathcal{P}$ decomposes as $\mathcal{P} = \|_{i\in I}\mathcal{P}_i$. The different subsystems $\mathcal{P}_i$ interact via some shared places, and their sets of transitions are pairwise disjoint. In particular, the alphabet A of alarm labels decomposes as $A = \bigcup_{i\in I} A_i$, where the A_i are pairwise disjoint. Next, we assume that each subsystem $\mathcal{P}_i$ possesses its own local sets of sensors, and the local sensor subsystems are independent, i.e., do not interact. Thus $\mathcal{Q}$ also decomposes as $\mathcal{Q} = \|_{i\in I}\mathcal{Q}_i$, and the $\mathcal{Q}_i$ possess pairwise disjoint sets of places. Consequently, in (22), $\mathcal{A}$ decomposes as $\mathcal{A} = \|_{i\in I}\mathcal{A}_i$, where the $\mathcal{A}_i$, the locally recorded alarm patterns, possess pairwise disjoint sets of places too.

As stated in the introduction, distributed diagnosis consists in computing the local view, by each supervisor, of global diagnosis. This is formalized next.

Definition 2 *Distributed diagnosis is represented by the following map:*

$$\mathcal{A} \longmapsto [\mathbf{R}_{P_i}(\mathcal{E}_{\mathcal{A}\times\mathcal{P}})]_{i\in I}\,, \tag{23}$$

where $\mathcal{A}$ ranges over the set of all finite prefixes of runs of $\mathcal{Q}$. Our objective is therefore to compute $[\mathbf{R}_{P_i}(\mathcal{E}_{\mathcal{A}\times\mathcal{P}})]_{i\in I}$ without performing global diagnosis, i.e., without computing $\mathcal{E}_{\mathcal{A}\times\mathcal{P}}$.

As advocated in the introduction, in order to scale up to large distributed systems, it is requested that computing the local view, by each supervisor, of the global diagnosis, is performed *without* computing the global diagnosis. In other words, we want to compute $\mathbf{R}_{P_i}(\mathcal{E}_{\mathcal{A}\times\mathcal{P}})$ without computing $\mathcal{E}_{\mathcal{A}\times\mathcal{P}}$. The reader should notice that, in general, $\mathbf{R}_{P_i}(\mathcal{E}_{\mathcal{A}\times\mathcal{P}}) \neq \mathcal{E}_{\mathcal{A}_i\times\mathcal{P}_i}$, expressing the fact that the different supervisors must cooperate at establishing a coherent distributed diagnosis.

5.3 The need for a higher-level "orchestration"

The distributed diagnosis algorithm illustrated in Fig. 12 is easy to understand, for our running example. But this running example is very simple, for the following reasons: firstly, it involves only two components, and, second, interaction occurs through the two alternating places 3 and 7 and the interaction pattern $7 \to \beta \to 3 \to \alpha \to 7 \to \beta \to 3\ldots$ involves no concurrency and no conflict.

Now, distributed diagnosis with several supervisors and more complex interaction than in our toy example, results in a really messy algorithm. To scale up, we need to better structure our algorithm. In Section 7 we provide a high-level orchestration of distributed diagnosis. In this orchestration, details are hidden in the form of a set of primitive operations on certain event structures. The orchestration will be formally analyzed and proved correct. Before this, in section 6 we formally introduce our set of primitive operations.

6 Event structures and their use in distributed diagnosis

Running example, continued. Fig. 16 shows three prefixes of the mid diagram of Fig. 12. Diagram (a) illustrates local diagnosis performed by supervisor 1 and 2, independently, based on the observation of $\mathcal{A}_1 = \{\beta; \rho_1\}$ and $\mathcal{A}_2 = \{\alpha\}$; it consists in computing $\mathcal{E}_{1,1} =_{\text{def}} \mathcal{E}_{\mathcal{A}_1\times\mathcal{P}_1}$, at supervisor 1, and $\mathcal{E}_{2,1} =_{\text{def}} \mathcal{E}_{\mathcal{A}_2\times\mathcal{P}_2}$ at supervisor 2. In (b) a message $\mathcal{M}_{1,2} = \mathbf{R}_{P_1\cap P_2}(\mathcal{E}_{1,1})$ is sent by supervisor 1 to supervisor 2; it consists of the graph $(7) \to [] \to (3)$ sitting at the extremity of the thick right going arrow; this graph is "composed" with $\mathcal{E}_{2,1}$, this yields the result $\mathcal{E}_{2,2}$ shown in (b). Using $\mathcal{E}_{2,2}$, supervisor 2 can now reuse its alarm pattern $\mathcal{A}_2$ and further extend $\mathcal{E}_{2,2}$; the result is shown in (c), call it $\mathcal{E}_{2,3}$. Finally, (d) is the mirror of (b): a message $\mathcal{M}_{2,1} = \mathbf{R}_{P_1\cap P_2}(\mathcal{E}_{2,3})$ is sent by supervisor 2 to

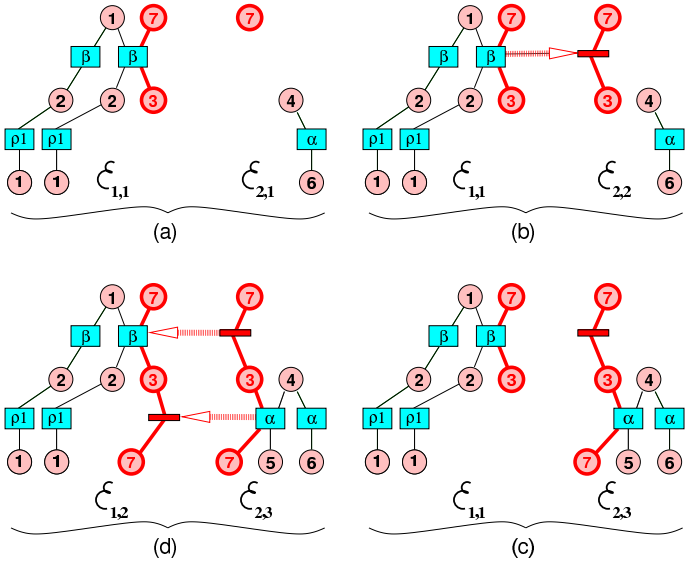

Figure 16: The detailed mechanism of off-line diagnosis—compare with Fig. 12.

supervisor 1; it consists of the longer graph (7) → [] → (3) → [] → (7) sitting at the extremity of the double thick left going arrow; this message is "composed" with $\mathcal{E}_{1,1}$ by supervisor 1, by glueing the common prefix (7) → [] → (3); this yields $\mathcal{E}_{1,2}$ shown in (d). ◇

Throughout this discussion, we have used two operations: 1/ the "composition" of a received message $\mathcal{M}$ with the diagnosis currently available at the receiver, and 2/ the "extension" of a prefix of local diagnosis by re-unfolding the alarms, e.g., from $\mathcal{E}_{2,1}$ to $\mathcal{E}_{2,2}$. The first operation will be formalized by considering the trimmed composition of labeled event structures, studied in Section 6.1. The second operation, we call it the "extended unfolding", will be studied in Section 6.2. As we shall see, they are sufficient to express and formally study distributed diagnosis in all cases.

6.1 Composition of labeled event structures

Focus again on Fig. 16, diagrams (c) and (d). The update, from (c) to (d), shows the kind of composition operator we need to formally specify our algorithm. This operator performs two things. First, it glues the two isomorphic graphs (7) → [] → (3) occurring in thick in the left and right parts of diagram (c): this is a parallel composition in which isomorphic parts are glued together by synchronizing events with same label and isomorphic causes. This parallel composition will be formally introduced below under the name of "strong parallel composition", denoted by $\times^s$. Next, concentrate on diagram (d). Besides glueing together the two isomorphic graphs (7) → [] → (3), it extends it by appending the thick path

(3) $\rightarrow$ [] $\rightarrow$ (7) to the condition (3). This is a different kind of parallel composition that performs a "continuation" of the strong parallel composition by configurations that exist only in one component. Such continuations will be formally defined below. Finally, combining these elementary operations will yield a primitive, called the "trimmed composition" and denoted by the symbol $\|$.

6.1.1 Parallel composition of event structures without labels

This is a classical notion, first introduced by Winskel [31]. We follow closely [30] with minimal changes in the notations. Let $\mathcal{E}_i = (E_i, \preceq_i, \#_i)$, $i \in \{1,2\}$, be two labeled event structures. Set

$$\begin{array}{rcl} E_1 \times_\star E_2 & =_{\text{def}} & \{(e_1,\star) \mid e_1 \in E_1\} \\ & \cup & \{(\star,e_2) \mid e_2 \in E_2\} \\ & \cup & \{(e_1,e_2) \mid e_1 \in E_1 \text{ and } e_2 \in E_2\} \end{array}$$

where $\star$ denotes a special event *undefined*. Denote by π_1 and π_2 the projections given by $\pi_i(e_1,e_2) = e_i$ for $i \in \{1,2\}$, respectively. Call a subset κ of $E_1 \times_\star E_2$ a *pre-configuration* iff:

(i) For $i \in \{1,2\}$, $\pi_i(\kappa)$ is a configuration of $\mathcal{E}_i$;

(ii) $\preceq_\kappa$, the transitive closure of relation $\leq \cap(\kappa\times\kappa)$, is a partial order, where $\leq \subseteq (E_1\times_\star E_2)$ is defined by:

$$f \leq f' \quad \Leftrightarrow \quad \pi_1(f) \preceq_1 \pi_1(f') \text{ or } \pi_2(f) \preceq_2 \pi_2(f'). \tag{24}$$

If κ moreover has a unique maximal element w.r.t. $\preceq_\kappa$, then κ is called a *complete prime*. Then, the *parallel composition* of $\mathcal{E}_1$ and $\mathcal{E}_2$, denoted by $\mathcal{E}_1 \times \mathcal{E}_2$, is the structure $(E, \preceq, \#)$ with:

$$\begin{array}{rcl} E & = & \{\kappa \mid \kappa \text{ is a complete prime}\}, \\ \kappa \preceq \kappa' & \Leftrightarrow & \kappa \subseteq \kappa', \\ \kappa \# \kappa' & \Leftrightarrow & \kappa \cup \kappa' \text{ is not a pre-configuration.} \end{array} \tag{25}$$

It is proved in [30] that the so defined $\mathcal{E}_1 \times \mathcal{E}_2$ is also a prime event structure. To conform with the usual notation for events, we shall denote by e the events of $\mathcal{E}_1 \times \mathcal{E}_2$ (instead of κ as in (25)). With this updated notation, two *canonical projections* are associated with the parallel composition: the first projection

$$\Pi_1 : E \mapsto_\star E_1 \quad \text{is defined by} \quad \forall e \in E : \Pi_1(e) =_{\text{def}} \pi_1(\max(e)), \tag{26}$$

and the second projection Π_2 is defined similarly. Note that this definition is consistent since κ is a complete prime.

Comments. The intuition behind (25) is that the product event structure is defined indirectly through its configurations. If $\mathcal{E}_1$ and $\mathcal{E}_2$ execute in parallel, then events of $\mathcal{E}_1$ and $\mathcal{E}_2$ can either occur in isolation (this corresponds to pre-events of the form $(e_1, \star)$ or $(\star, e_2)$), or an event of $\mathcal{E}_1$ can synchronize with an event of $\mathcal{E}_2$ (in which case we have a pre-event of the form (e_1, e_2)). Now, at any stage of the execution of $\mathcal{E}_1 \times \mathcal{E}_2$, a set of pre-events has occurred; the notion of pre-configuration gives a characterization of these sets. Condition (i) says that if we project a pre-configuration onto one of the two components, the result must be a configuration of this component. Condition (ii) says that the events of the component may occur only once and that both components must agree on the causal relations between events in the parallel composition. Once the finite configurations of the parallel composition have been defined, then a standard procedure can be used to turn this into a prime event structure, namely by identifying events of the composition with configurations having a unique maximal element. ◇

The following results are borrowed from [30]. They express that the parallel composition of event structures is the proper notion of composition:

1. The two projections $\Pi_i, i \in \{1,2\}$ associated with the parallel composition of event structures are morphisms.
2. The parallel composition $(\mathcal{E}_1, \mathcal{E}_2) \mapsto \mathcal{E}_1 \times \mathcal{E}_2$ with projections Π_1 and Π_2 is a product in the category $\underline{\underline{\mathcal{E}}}$ of event structures. This product is associative and commutative.

Statement 2 means that the parallel composition satisfies the following universal property:

$$\forall \mathcal{E}, \psi_1, \psi_2 : \quad \begin{array}{ccc} & \mathcal{E} & \\ {\scriptstyle \psi_1}\swarrow & & \searrow{\scriptstyle \psi_2} \\ \mathcal{E}_1 & & \mathcal{E}_2 \end{array} \quad \Rightarrow \quad \exists \psi : \quad \begin{array}{ccccc} & & \mathcal{E} & & \\ & {\scriptstyle \psi_1}\swarrow & \downarrow \psi & \searrow{\scriptstyle \psi_2} & \\ \mathcal{E}_1 & \xleftarrow[\Pi_1]{} & \mathcal{E}_1 \times \mathcal{E}_2 & \xrightarrow[\Pi_2]{} & \mathcal{E}_2 \end{array} \tag{27}$$

In (27), symbols ψ_i, Π_i, for $i \in \{1,2\}$, and ψ, denote morphisms, Π_1 and Π_2 are the two projections associated with the composition $\mathcal{E}_1 \times \mathcal{E}_2$, and the second diagram commutes.

6.1.2 Parallel composition of event structures with labels

As explained in Section 4.2, formulas (12,13), two categories $\underline{\underline{\mathcal{E}}}_s$ and $\underline{\underline{\mathcal{E}}}_w$ can be considered, depending on the classes of morphisms. Each category has its associated product that we defined next.

Define the *strong* and the *weak parallel composition* of two labeled event structures $\mathcal{E}_i = (E_i, \preceq_i, \#_i, \lambda_i, P_i)$, $i \in \{1,2\}$, denoted by $\mathcal{E}_1 \times^s \mathcal{E}_2$ and $\mathcal{E}_1 \times^w \mathcal{E}_2$, respectively. Both are variations of the case without labels. Two events $e_i \in E_i, i \in \{1,2\}$ are called *strongly*, resp.

weakly compatible, respectively written

$$\left.\begin{array}{r} e_1 \bowtie_s e_2 \\ \text{resp.} \quad e_1 \bowtie_w e_2 \end{array}\right\} \quad \text{iff:} \quad \left\{\begin{array}{ll} & \left\{\begin{array}{ll} & \lambda_1(e_1) \subseteq P_1 \backslash P_2 \text{ and } e_2 = \star \\ \text{resp.} & \lambda_1(e_1) \subseteq P_1 \quad \text{and } e_2 = \star \end{array}\right. \\ \text{or} & \\ & \left\{\begin{array}{ll} & \lambda_2(e_2) \subseteq P_2 \backslash P_1 \text{ and } e_1 = \star \\ \text{resp.} & \lambda_2(e_2) \subseteq P_2 \quad \text{and } e_1 = \star \end{array}\right. \\ \text{or} & \\ & \lambda_1(e_1) \cap P_1 \cap P_2 = \lambda_2(e_2) \cap P_1 \cap P_2 \neq \emptyset. \end{array}\right. \tag{28}$$

The first two cases correspond to an event that involves a single component, whereas the third case corresponds to two non silent events synchronizing (their labels agree on the shared places). Note the difference in the rules for $\bowtie_s$ and $\bowtie_w$: for $\bowtie_s$, a component can progress alone by means of a private event only, whereas, for $\bowtie_w$, the event does not need to be private. Define

$$\begin{array}{rcl} E_1 \times^s E_2 & =_{\text{def}} & \{(e_1, e_2) \in E_1 \times_\star E_2 \,|\, e_1 \bowtie_s e_2\}, \\ E_1 \times^w E_2 & =_{\text{def}} & \{(e_1, e_2) \in E_1 \times_\star E_2 \,|\, e_1 \bowtie_w e_2\}, \end{array} \tag{29}$$

with the convention $\lambda(\star) = \emptyset$. Then the two parallel compositions $\mathcal{E}_1 \times^s \mathcal{E}_2$ and $\mathcal{E}_1 \times^w \mathcal{E}_2$ are defined via (25), but with $E_1 \times^s E_2$ and $E_1 \times^w E_2$ replacing $E_1 \times_\star E_2$, respectively, and, for both cases:

$$\lambda(e) \quad = \quad \lambda_1(\Pi_1(e)) \cup \lambda_2(\Pi_2(e)), \tag{30}$$

where the projections Π_i are defined in (26). The parallel composition is illustrated on Fig. 17. By construction,

$$\mathcal{E}_1 \times^s \mathcal{E}_2 \sqsubseteq \mathcal{E}_1 \times^w \mathcal{E}_2, \quad \text{and} \quad \Pi_i\,(\mathcal{E}_1 \times^w \mathcal{E}_2) = \mathcal{E}_i, \text{ for } i \in \{1, 2\}. \tag{31}$$

Universal property (27) adapts for $\times^s$ with strong morphisms, and for $\times^w$ with weak morphisms.

6.1.3 Continuations

Consider an event structure $\mathcal{E} = (E, \preceq, \#, \lambda, P)$, a prefix $\mathcal{F} \sqsubseteq \mathcal{E}$ having F as its set of events, and $Q \subseteq P$. The *continuation of $\mathcal{F}$ by $\mathcal{E}$ through Q*, written

$$\mathcal{F} \bullet_Q \mathcal{E}, \tag{32}$$

is the prefix of $\mathcal{E}$ consisting of the following set of events: 1/ the events of $\mathcal{F}$, 2/ the events e of $\mathcal{E}$ such that the restriction $\lceil e \rceil_{|F,Q}$ is a maximal configuration of $\mathcal{F}_{|Q}$ (see (11)). By definition $\mathcal{F} \sqsubseteq (\mathcal{F} \bullet_Q \mathcal{E}) \sqsubseteq \mathcal{E}$.

The continuation is illustrated in Fig. 17. For this discussion, call $\mathcal{F}$ the bottom left diagram, $\mathcal{E}$ the bottom right one, and take $Q = \{3, 7\}$. Then, $\mathcal{F}_{|Q}$ is depicted in thick in the two bottom diagrams (except that we did not adjust the labels). The configuration of $\mathcal{E}$ in light gray is a continuation of the configuration $\{7\}$ of $\mathcal{F}_{|Q}$; since configuration $\{7\}$ is not maximal in $\mathcal{F}_{|Q}$, the configuration of $\mathcal{E}$ in light gray is discarded in constructing $\mathcal{F} \bullet_Q \mathcal{E}$, shown on the top right diagram.

6.1.4 Trimmed composition

Our final primitive operation for use in orchestrations will be the *trimmed composition* of an indexed family $\mathcal{E}_i, i \in I$, of labeled event structures, defined by:

$$\|_{i\in I}\mathcal{E}_i \quad =_{\text{def}} \quad trim\,(\,[\textstyle\prod^s_{i\in I}\,\mathcal{E}_i]\,\bullet_Q\,[\prod^w_{i\in I}\,\mathcal{E}_i]\,)\,, \tag{33}$$

where $\prod^s$ and $\prod^w$ refer to the $\times^s$ and $\times^w$ compositions, respectively, and

$$Q \quad = \quad interact(P_i)_{i\in I} \;=_{\text{def}}\; \textstyle\bigcup_{(i,j)\in I\times I\,:\,i\neq j}\,(P_i \cap P_j). \tag{34}$$

Note that $\|$, regarded as a binary operator, is not associative. This is why we define the trimmed composition as an n-ary operator directly.

Running example, continued. Construction $\mathcal{E}_1 \,\|\, \mathcal{E}_2$ is illustrated in Fig. 17. Recall that 3 and 7 are the shared places. The third diagram of this figure shows the $\times^s$-composition; the branch $[3] \to [7,5]$ that is offered by component 2 finds no counterpart in component 1, hence it does not appear in the $\times^s$-composition. For the $\|$ trimmed composition, it is allowed to continue maximal configurations of the $\times^s$-composition with configurations that exist in the more permissive $\times^w$-product. Now, the branch $[3] \to [7,5] \to [4]$ that is offered by component 2 contributes to the $\times^w$-product. It continues the configuration $[7] \to [3]$ that is maximal in the restriction $(\mathcal{E}_1 \times^s \mathcal{E}_2)_{|P_1\cap P_2}$. Therefore it gives raise, in the $\|$-composition, to the extension $[2,3] \to [7,5] \to [4]$ of the $\times^s$-product.

Note that, projecting away, from this $\|$-composition, the labels $5, 6, 4$ that are private to the 2nd component yields exactly the left diagram of Fig. 16-(d).

Star closure. Consider again Fig. 16. In (d), diagram $\mathcal{E}_{1,1}$ is receiving $\kappa =_{\text{def}} (7) \to [] \to (3)$ as an "echo" of its own message sent in (b). This means that, in (d), a composition $\kappa \,\|\, \kappa$ occurs. And this scheme is repeated throughtout the different steps of the informal algorithms shown in Fig. 12. Therefore, we need to pay attention to how $\mathcal{E} \,\|\, \mathcal{E}$ relates to $\mathcal{E}$, for $\mathcal{E}$ an arbitrary event structure.

Despite the notation that seems to refer to the conjunction in logic, the operator $\|$ is not idempotent: we do have $\mathcal{E} \sqsubseteq \mathcal{E} \,\|\, \mathcal{E}$, but equality does not hold in general. The reason is the following: assume that $\mathcal{E}$ contains two different configurations κ_1 and κ_2 such that there exists a bijective map ι, from the set of events of κ_1 to the set of events of κ_2, which is label preserving and such that $[(\iota,\iota)(\preceq_{|\kappa_1})] \cup \preceq_{|\kappa_2}$ generates a partial order—said differently, the two orders on each configuration do not contradict each other. Then, $\kappa =_{\text{def}} \{(e_1, \iota(e_1)) \mid e_1 \in \kappa_1\}$ is a preconfiguration of $E \times^s E$ that has no counterpart in $\mathcal{E}$. Thus, as soon as such pathological pairs of non-contradictory configurations exist in $\mathcal{E}$, we have $\mathcal{E} \neq \mathcal{E} \times^s \mathcal{E}$ and thus also $\mathcal{E} \neq \mathcal{E} \,\|\, \mathcal{E}$. The above discussion also reveals that the gap between $\mathcal{E}$ and $\mathcal{E} \,\|\, \mathcal{E}$ is indeed small, since it consist only in "reshuffling" pairs of non-contradictory configurations to form new ones. This leads to considering the star closure of event structures we introduce next.

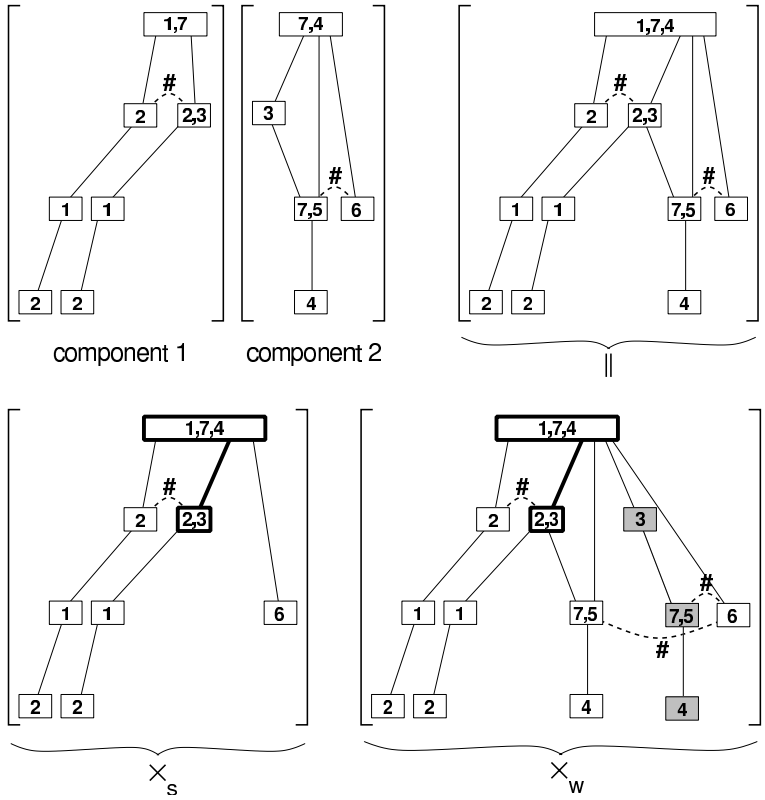

Figure 17: Parallel composition of labeled event structures. The first two diagrams show two components that are prefixes of the ones of Fig. 12. The two diagrams sitting on the bottom show the $\times^s$- and $\times^w$- compositions. The resulting $||$-composition is shown on the top right diagram.

Write $\mathcal{E}^n =_{\rm def} \mathcal{E} \,||\, \ldots \,||\, \mathcal{E}$ (n times). The sequence $(\mathcal{E}^n)_{n\geq 1}$ is increasing for the prefix order, and converges to a unique event structure,

$$\text{we denote it by } \mathcal{E}^* \text{ and call it the } \textit{star closure} \text{ of } \mathcal{E}. \tag{35}$$

The star closure $\mathcal{E}^*$ is the minimal (for prefix order) solution of the fixpoint equation $\mathcal{X} = \mathcal{E} \,||\, \mathcal{X}$. It satisfies $(\mathcal{E}^*)^n = \mathcal{E}^*$ for each $n \geq 1$, and, for each event structure $\mathcal{F}$, we have $\mathcal{F} \sqsubseteq \mathcal{E} \,||\, \mathcal{F}$ iff $\mathcal{F} \sqsubseteq \mathcal{E}^*$. ◇

Important properties of event structures and their composition are collected in Appendices A.1 and A.2.

6.2 Extended unfoldings

In section Section 4.3 we have introduced the event structure $\mathcal{E}_{\mathcal{P}}$ associated to the unfolding of a Petri net $\mathcal{P}$. In this section we generalize $\mathcal{E}_{\mathcal{P}}$ to situations in which the considered Petri net $\mathcal{P}$ is unfolded, starting from a given initial labeled event structure $\mathcal{I}$. This construction was used in step (c) of Fig. 16.

Let $\mathcal{P} = (P, T, \rightarrow, M_0)$ be a Petri net. For $M \subseteq P$, call $\mathcal{P}_M$ the Petri net $\mathcal{P}$ in which M has been substituted for the initial marking M_0 of $\mathcal{P}$ (note that we do not require that M shall be reachable from M_0). Write for short $\mathcal{E}_M =_{\rm def} \mathcal{E}_{\mathcal{P}_M}$. Each event $e \in \mathcal{E}_M$ represents some set T_e of transitions of $\mathcal{P}$ (T_e may not be a singleton, due to the trimming performed when mapping $\mathcal{U}_{\mathcal{P}}$ to $\mathcal{E}_{\mathcal{P}}$). For each $t \in T_e$, $t^\bullet = \lambda(e)$.

Let $\mathcal{I}$ be a labeled event structure having $Q \supseteq P$ as label set. Denote by λ the labeling map of $\mathcal{I}$. For I a co-set of $\mathcal{I}$ such that $\lambda(i) \cap P \neq \emptyset$ holds for each event $i \in I$, set $M =_{\rm def} \bigcup_{i \in I} (\lambda(i) \cap P)$. A total map $\rho : M \mapsto I$ such that $p \in \lambda(\rho(p))$ is called a *representation of M by I*. Denote by $M \hookrightarrow_\lambda I$ the set of all such representations. We shall "append" $\mathcal{E}_M$ to I via ρ as follows.

Denote by $e_{0,M}$ the minimal event of $\mathcal{E}_M$. Denote by $\mathcal{E}_{I,\rho,M}$ the event structure obtained by taking the disjoint union of I and $\mathcal{E}_M \setminus \{e_{0,M}\}$, and adding the following causalities: For each event i of I and each event e of $\mathcal{E}_M$ such that $e \in e^\bullet_{0,M}$, set

$$e \in i^\bullet \text{ in } \mathcal{E}_{I,\rho,M} \quad \text{iff} \quad \exists t \in T_e, \exists p \in {}^\bullet t, \text{ such that } i = \rho(p).$$

Let $\mathbf{I}$ be the set of all co-sets I of $\mathcal{I}$ such that $\lambda(i) \cap P \neq \emptyset$ holds for each event $i \in I$. The *unfolding of $\mathcal{P}$ from $\mathcal{I}$*, written $\mathcal{E}^{\mathcal{I}}_{\mathcal{P}}$, is defined by:

$$\mathcal{E}^{\mathcal{I}}_{\mathcal{P}} =_{\rm def} \mathit{trim}\left(\mathcal{I} \cup \left[\bigcup_{I \in \mathbf{I},\, \rho \in M \hookrightarrow_\lambda I} \mathcal{E}_{I,\rho,M}\right]\right), \text{ where } M =_{\rm def} \bigcup_{i \in I} \lambda(i) \tag{36}$$

Note that the trimming is essential here, since the expression in parentheses in formula (36) exhibits lots of redundancies. The extended unfolding satisfies the following properties, where S denotes an arbitrary label set:

$$\mathcal{I} \sqsubseteq \mathcal{I}' \text{ and } \exists \mathcal{P}'' : \mathcal{P}' = \mathcal{P} \,||\, \mathcal{P}'' \quad \Rightarrow \quad \mathcal{I} \sqsubseteq \mathcal{E}^{\mathcal{I}}_{\mathcal{P}} \sqsubseteq \mathcal{E}^{\mathcal{I}'}_{\mathcal{P}'} \tag{37}$$

$$\mathbf{R}_S(\mathcal{E}^{\mathcal{I}}_{\mathcal{P}}) \quad = \quad \mathbf{R}_S(\mathcal{E}^{\mathbf{R}_S(\mathcal{I})}_{\mathcal{P}}) \tag{38}$$

Running example, continued. The right hand side of diagram (c) of Fig. 16 shows $\mathcal{E}_{2,3} = \mathcal{E}^{\mathcal{I}}_{\mathcal{P}}$ for $\mathcal{I} \leftarrow \mathcal{E}_{2,2}$ and $\mathcal{P} \leftarrow \mathcal{A}_{2,1} \times \mathcal{P}_2$, where symbol $\leftarrow$ denotes substitution. $\diamond$

Important properties of extended unfoldings are found in Appendix A.3.

6.3 Detailed implementation of the primitives

In this section, we provide effective implementations of our primitives by means of pattern matching rules.

Parallel compositions $\times^s$ and $\times^w$. Recall that the parallel compositions $\times^s$ and $\times^w$ of labeled event structures is defined via formulas (25,29,30). Write $X \vdash e$ if $X \supseteq {}^\bullet e$, and say that X *enables* e. The parallel composition $\mathcal{E}_1 \times^s \mathcal{E}_2$ is constructed by inductively applying the following rule, in which X denotes a (possibly empty) co-set of $\mathcal{E}_1 \times^s \mathcal{E}_2$:

$$\left.\begin{array}{r} \Pi_1(X) \vdash e_1 \text{ and } \Pi_2(X) \vdash e_2 \\ e_1 \bowtie_s e_2 \\ X \text{ is minimal having the above properties} \end{array}\right\} \quad \Rightarrow \quad X = {}^\bullet(e_1, e_2), \tag{39}$$

meaning that event (e_1, e_2) is a new extension of $\mathcal{E}_1 \times^s \mathcal{E}_2$ beyond co-set X. The rule for $\times^w$ is identical, except that $\bowtie_s$ is replaced by $\bowtie_w$.

Trimming. The event structure $trim(\mathcal{E})$ is constructed by inductively applying the following rule, in which $X_i, i \in \{1,2\}$ denote (possibly empty) co-sets of $trim(\mathcal{E})$:

$$\left.\begin{array}{rcl} trim(X_1) & = & trim(X_2) \\ \forall i \in \{1,2\} \ : \ X_i & = & {}^\bullet e_i \\ \lambda(e_1) & = & \lambda(e_2) \end{array}\right\} \quad \Rightarrow \quad trim(e_1) = trim(e_2) \tag{40}$$

The following *canonical form* can be considered for a labeled trimmed event structure. Its events have the special inductive form (X, ℓ), where X is a co-set of $\mathcal{E}$ and $\ell \in Pow(P)$. The causality relation $\preceq$ is simply encoded by the preset function ${}^\bullet(X, \ell) = X$, and the labeling map is $\varphi(X, \ell) = \ell$. Events with empty preset have the form (nil, ℓ). The conflict relation is specified separately.

Trimmed composition $\|$. The trimmed composition $\mathcal{E}_1 \| \mathcal{E}_2$ is constructed by inductively applying the following two rules, in which X denotes a (possibly empty) co-set of $\mathcal{E}_1 \| \mathcal{E}_2$:

$$\left.\begin{array}{r} \Pi_1(X) \vdash e_1 \text{ and } \Pi_2(X) \vdash e_2 \\ e_1 \bowtie_s e_2 \\ X \text{ is minimal having the above properties} \end{array}\right\} \quad \Rightarrow \quad X = {}^\bullet(e_1, e_2), \tag{41}$$

$$\left.\begin{array}{r} \Pi_1(X) \vdash e_1 \text{ and } \Pi_2(X) \vdash e_2 \\ e_1 \bowtie_w e_2 \\ X \text{ is minimal having the above properties} \end{array}\right\} \quad \Rightarrow \quad X = {}^\bullet(e_1, e_2), \tag{42}$$

As the reader can easily check, rules (41) and (42) overlap. To ensure that only maximal configurations obtained by using $\times^s$ are further extended by means of $\times^w$, *we give higher priority to rule* (41), thus making the choice between the two rules deterministic.

Extended unfolding. The extended unfolding $\mathcal{E}_{\mathcal{P}}^{\mathcal{I}}$ is constructed by inductively applying the following rule, where $\mathcal{I}$ is the initial condition and X denotes a co-set of $\mathcal{E}_{\mathcal{P}}^{\mathcal{I}}$:

$$\left.\begin{array}{r} \lambda(X) \supseteq m \\ m = {}^\bullet t \text{ in Petri net } \mathcal{P} \\ t^\bullet = \ell \text{ in Petri net } \mathcal{P} \\ X \text{ is minimal having the above properties} \end{array}\right\} \Rightarrow \quad X = {}^\bullet(X, \ell), \tag{43}$$

7 Orchestration of distributed diagnosis

We are now ready to state our orchestration. Throughout this section, we assume the setup of Section 5.2. Petri net $\mathcal{P} = (P, T, \rightarrow, M_0)$ decomposes as $\mathcal{P} = \|_{i \in I} \mathcal{P}_i$, where $\mathcal{P}_i = (P_i, T_i, \rightarrow, M_{i,0})$.

Completeness. Let $\mathcal{P} = (P, T, \rightarrow, M_0)$ be a safe Petri net, and let $Q \subseteq P$. $\mathcal{P}$ is called *Q-complete* if, for every place $p \in Q$ such that ${}^\bullet p \cup p^\bullet \neq \emptyset$, there exists a place $\overline{p} \in Q$ such that (i) $\overline{p}^\bullet = {}^\bullet p \setminus p^\bullet$, (ii) ${}^\bullet \overline{p} = p^\bullet \setminus {}^\bullet p$, and (iii) $M_0(p) + M_0(\overline{p}) = 1$, where M_0 denotes the initial marking. We say that $\mathcal{P}$ is *complete* if it is P-complete. If $\mathcal{P}$ is not Q-complete, we can make it Q-complete by adding the missing places and arcs. The so obtained completion $\overline{\mathcal{P}}$ produces an event structure $\mathcal{E}_{\overline{\mathcal{P}}}$ such that $(\mathcal{E}_{\overline{\mathcal{P}}})_{|P} = \mathcal{E}_{\mathcal{P}}$, i.e., erasing the place $\overline{p}$ from the event structure generated by $\overline{\mathcal{P}}$ yields the event structure generated by $\mathcal{P}$, see Section 4.3. Informally said, completion does not change the behaviour of the Petri net. Our example $\mathcal{P} = \mathcal{P}_1 \| \mathcal{P}_2$ of Fig. 7 satisfies this property with $Q = \{3, 7\}$, since places 3 and 7 are complementary.

Distributed conflict. We say that $\mathcal{P}_1 \| \mathcal{P}_2$ *has no distributed conflict* if:

$$\forall p \in P_1 \cap P_2, \quad \exists i \in \{1, 2\} : \quad p^\bullet \subseteq T_i. \tag{44}$$

Note that our example of Fig. 7 satisfies (44). This is a reasonable assumption in our context, since shared places aim at representing the propagation of faults between components; having distributed conflict would have little meaning in this case. The following assumption will be used in the sequel:

Assumption 1 *Decomposition $\mathcal{P} = \|_{i \in I} \mathcal{P}_i$ involves no distributed conflict and $\mathcal{P}$ is complete.*

For $\|_{i \in I} \mathcal{P}_i$ a parallel composition of safe Petri nets satisfying Assumption 1, then

$$\forall p \in P_i \cap P_j, \quad \text{then} \quad p^\bullet \subset T_i \Rightarrow {}^\bullet p \subset T_j. \tag{45}$$

7.1 Off-line orchestration of distributed diagnosis

In this section we study off-line diagnosis, meaning that some finite alarm pattern $\mathcal{A}$ is given for diagnosis. The structure of the interaction between the different components $\mathcal{P}_i$, for $i \in I$, will play an important role for our distributed diagnosis algorithm. This is captured by the notion of interaction graph we introduce next.

Equip $I \times I$ with the following undirected graph structure: draw a branch (i, j) iff $P_i \cap P_j \neq \emptyset$, i.e., the ith and jth subsystems interact directly via shared places. Denote by $\mathcal{G}_I$ the resulting *interaction graph*. For $i \in I$, denote by $\mathrm{N}(i)$ the neighborhood of i, composed of the set of j's such that $j \neq I$ and $(i, j) \in \mathcal{G}_I$. Note that $\mathrm{N}(i)$ does not contain i.

Algorithm 1 shown in Fig. 18 performs distributed diagnosis (see (23)). It consists of a chaotic, unsupervised, cooperation between the different supervisors acting as peers. It is expressed in terms of the primitives $\mathcal{E}^{\mathcal{I}}_{\mathcal{P}}$ (to continue local diagnosis), $\mathbf{R}_Q$ (to model the relevant information for the interfaces of a subsystem), and $\|$ (to compose messages from other supervisors with current local diagnosis). This algorithm is analysed in the following

Algorithm 1 *Each site i maintains and updates, for each neighbor j, a message $\mathcal{M}_{i,j}$ toward j. Thus there are two messages per edge (i, j) of $\mathcal{G}_I$, one in each direction. The algorithm consists of chaotic iterations as follows:*

1. Initialization:
   - (a) *for each $i \in I$, set $\mathcal{E}_i := \{e_{i,0}\}$, where $\lambda(e_{i,0}) = M_{i,0}$;*
   - (b) *for each edge $(i, j) \in \mathcal{G}_I$, set $\mathcal{M}_{i,j} := \mathbf{R}_{P_i \cap P_j}(\mathbf{1})$.*
2. Chaotic iteration: *until the $\mathcal{E}_i$'s become constant whatever choice (a), (b), or (c) is made, choose nondeterministically:*
   - (a) *select $i \in I$ update message $\mathcal{E}_i$ by extending local diagnosis:*

$$\mathcal{E}_i \;:=\; \mathcal{E}^{\mathcal{I}}_{\mathcal{P}} \text{ with the substitutions } \left\{ \begin{array}{lcl} \mathcal{I} & \leftarrow & \mathcal{E}_i \\ \mathcal{P} & \leftarrow & \mathcal{A}_i \times \mathcal{P}_i \end{array} \right.$$

   - (b) *select an edge $(i, j) \in \mathcal{G}_I$ and update message $\mathcal{M}_{i,j}$ sent by i to j:*

$$\mathcal{M}_{i,j} \;:=\; \mathbf{R}_{P_i \cap P_j}(\mathcal{E}_i \,\|\, \left[\|_{k \in \mathrm{N}(i) \setminus j} \mathcal{M}_{k,i} \right]). \tag{46}$$

   - (c) *select $i \in I$ update message $\mathcal{E}_i$ by using incoming messages:*

$$\mathcal{E}_i \;:=\; \mathcal{E}_i \,\|\, \left[\|_{k \in \mathrm{N}(i)} \mathcal{M}_{k,i} \right].$$

Figure 18: Algorithm 1. Symbol := denotes assignment.

two theorems.

Theorem 1 *Algorithm 1 is monotonic (i.e., the $\mathcal{E}_i$'s and $\mathcal{M}_{i,j}$'s are increasing w.r.t. the prefix order), confluent, and converges in finitely many iterations.*

Proof. The monotony of Algorithm 1 results from the following properties: 1/ $\mathcal{I} \sqsubseteq \mathcal{E}_{\mathcal{P}}^{\mathcal{I}}$, by (37), and 2/ monotony of $\|$, by (60) from Corollary 1 of Appendix A.2. For the proof of the confluence, it will be convenient to encode schedulings of the three possible choices (2b), (2c), and (2a) by words. To this end, consider the alphabet

$$\Sigma \quad = \quad \{a_i \mid i \in I\} \cup \{b_{i,j}, b_{j,i} \mid (i,j) \in \mathcal{G}_\kappa\} \cup \{c_i \mid i \in I\}, \tag{47}$$

where a, b, c refer to the three different steps of Algorithm 1, and the index refers to the edge (i,j) or the node i that is selected in the corresponding step. Σ^* denotes the Kleene closure of Σ. Denote by $\mathcal{E}(\sigma)$ the collection $(\mathcal{E}_i)_{i\in I}$ obtained after having applied scheduling $\sigma \in \Sigma^*$. Clearly, $\mathcal{E}(\sigma) \sqsubseteq \mathcal{E}(\sigma.\sigma')$ and $\mathcal{E}(\sigma') \sqsubseteq \mathcal{E}(\sigma.\sigma')$, where the prefix relation is taken componentwise and $\sigma.\sigma'$ is the concatenation of σ and σ'. From this, the confluence follows immediately. This proves the theorem. ◇

So far we did not use the structure of the interaction between the different components. The next theorem makes deeper use of this structure. For this result we use the notion of star closure of the $\|$ composition, introduced in (35).

Theorem 2 *Assume Assumption 1 is in force and the interaction graph $\mathcal{G}_I$ is a tree. Then, if $(\mathcal{E}_i)_{i\in I}$ denotes the limit of Algorithm 1, we have*

$$\forall i \in I \ : \ \mathbf{R}_{P_i}(\mathcal{E}_{\mathcal{A}\times\mathcal{P}}) \quad \sqsubseteq \ \mathcal{E}_i \ \sqsubseteq \quad \mathbf{R}_{P_i}(\mathcal{E}^*_{\mathcal{A}\times\mathcal{P}}). \tag{48}$$

Proof. See Appendix B.4. ◇

Theorem 2 expresses that Algorithm 1 computes all solutions (23) of distributed diagnosis, plus some extra configurations that are obtained by re-shuffling non contradictory solutions as explained in (35).

Note that hierarchical architectures satisfy the assumptions of Theorem 2, but this result covers more architectures than just hierarchical ones. When $\mathcal{G}_I$ is not a tree, further "echoes" result from messages conflueing through different routes the resulting case is studied, in a more abstract context, in [18].

7.2 On-line orchestration of distributed diagnosis

Now, instead of our local alarm patterns $(\mathcal{A}_i)_{i\in I}$ being given once and for all, we are given, for each $i \in I$, a *local set* $\mathbf{A}_i$ *of alarm patterns*. We assume that $\mathbf{A}_i$ is totally ordered by the prefix relation. In the sequel, the statement "update $\mathcal{A}_i$" means that we take the next alarm pattern of $\mathbf{A}_i$, for the prefix order.

Running example, continued. Referring to Fig. 11 or Fig. 12, the total ordering of alarm patterns is depicted by regarding the partially ordered alarm pattern as a labeled graph directed downward. Fig. 19 shows three prefixes of the mid diagram of Fig. 12.

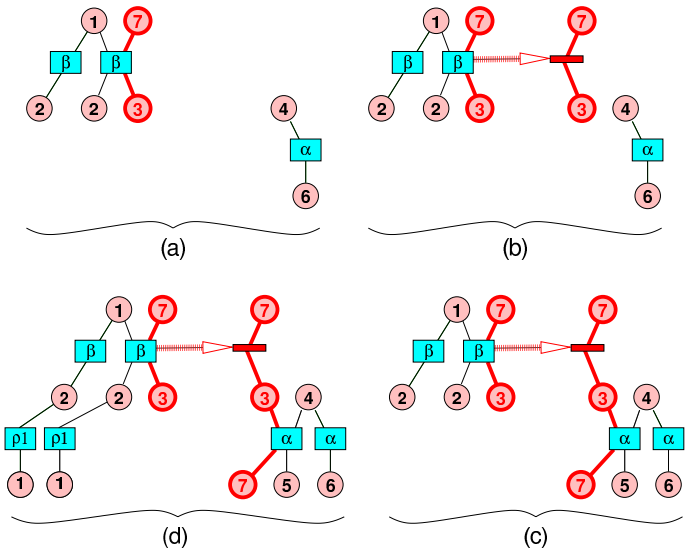

Figure 19: Illustrating on-line diagnosis. The four diagrams are prefixes of the mid diagram of Fig. 12; referring to Algorithm 2, (a) illustrates CASE 1, (b) illustrates the transmission of a message, and (c) illustrates CASE 2, and (d) illustrates once more CASE 1.

Diagram (a) illustrates local diagnosis performed by supervisor 1 and 2, independently, based on the observation of $\mathcal{A}_{1,1} = \{\beta\}$ and $\mathcal{A}_{2,1} = \{\alpha\}$; it consists in computing $\mathcal{E}_{\mathcal{A}_{1,1}\times\mathcal{P}_1}$, at supervisor 1, and $\mathcal{E}_{\mathcal{A}_{2,1}\times\mathcal{P}_2}$ at supervisor 2. In (b) a message $\mathcal{M}_{1,2}$ is sent by supervisor 1 to supervisor 2. Using this message, supervisor 2 can now reuse its $\mathcal{A}_{2,1}$ and extend $\mathcal{E}_{\mathcal{A}_{2,1}\times\mathcal{P}_2}$; the result is shown in (c), it is equal to $\mathbf{R}_{P_2}(\mathcal{E}_{\mathcal{A}_1\times\mathcal{P}})$, namely the local view, by supervisor 2, of the global diagnosis having observed $\mathcal{A}_1 =_{\text{def}} \mathcal{A}_{1,1} \,\|\, \mathcal{A}_{2,1}$. ◇

To adjust for this situation, Algorithm 1 is modified as Algorithm 2 of Fig. 20. Algorithm 2 is said to be *fairly* executed if it is applied in such a way that each of the three cases (a), (b), or (d), with every node i, and case (c) with every edge (i, j), are selected infinitely many times.

Theorem 3 *Assume Assumption 1 is in force, $\mathcal{G}_I$ is a tree, and Algorithm 2 is fairly executed. Then, for any given $\mathcal{A} = \|_{i\in I}\mathcal{A}_i$, where $\mathcal{A}_i \in \mathbf{A}_i$, after sufficiently many iterations, one has $\forall i \in I : \mathcal{E}_i \sqsupseteq \mathbf{R}_{P_i}(\mathcal{E}_{\mathcal{A}\times\mathcal{P}})$.*

Algorithm 2

1. Initialization:

(a) for each $i \in I$*, set* $\mathcal{A}_i := \mathbf{1}$*, and* $\mathcal{E}_i := \{e_{i,0}\}$*, where* $\lambda(e_{i,0}) = M_{i,0}$*;*

(b) for each edge $(i,j) \in \mathcal{G}_I$*, set* $\mathcal{M}_{i,j} := \mathbf{R}_{P_i \cap P_j}(\mathbf{1})$.

2. Chaotic non terminating iteration: *repeatedly choose nondeterministically:*

(a) select $i \in I$ *and update alarm pattern* $\mathcal{A}_i$*;*

(b) select $i \in I$ *and update message* $\mathcal{E}_i$ *by extending local diagnosis:*

$$\mathcal{E}_i \;:=\; \mathcal{E}_{\mathcal{P}}^{\mathcal{I}} \text{ with the substitutions } \left\{ \begin{array}{lcl} \mathcal{I} & \leftarrow & \mathcal{E}_i \\ \mathcal{P} & \leftarrow & \mathcal{A}_i \times \mathcal{P}_i \end{array} \right.$$

(c) select an edge $(i,j) \in \mathcal{G}_I$ *and update message* $\mathcal{M}_{i,j}$ *sent by* i *to* j*:*

$$\mathcal{M}_{i,j} \;:=\; \mathbf{R}_{P_i \cap P_j}(\mathcal{E}_i \,\|\, \left[\|_{k \in \mathrm{N}(i) \setminus j} \mathcal{M}_{k,i} \right]). \tag{49}$$

(d) select $i \in I$ *update message* $\mathcal{E}_i$ *by using incoming messages:*

$$\mathcal{E}_i \;:=\; \mathcal{E}_i \,\|\, \left[\|_{k \in \mathrm{N}(i)} \mathcal{M}_{k,i} \right].$$

Figure 20: Algorithm 2.

Theorem 3 expresses that, modulo a fairness assumption and with some delay, Algorithm 2 provides, as a prefix of the current event structure it computes, the diagnosis $\mathbf{R}_{P_i}(\mathcal{E}_{\mathcal{A} \times \mathcal{P}})$ of any given alarm pattern $\mathcal{A}$.

Proof. Monotony of Algorithm 2 is proved in the same way as for Algorithm 1. To study the confluence of Algorithm 2, we reuse the method and notations of the proof of Theorem 1. In particular, we adapt in an obvious way, to Algorithm 2, the coding (47) used for the schedulings of Algorithm 1.

Denote by $\mathcal{E}(\sigma)$ the collection $(\mathcal{E}_i)_{i \in I}$ obtained after having applied scheduling $\sigma \in \Sigma^*$. Clearly, $\mathcal{E}(\sigma) \sqsubseteq \mathcal{E}(\sigma.\sigma')$ and $\mathcal{E}(\sigma') \sqsubseteq \mathcal{E}(\sigma.\sigma')$, where the prefix relation is taken componentwise and $\sigma.\sigma'$ is the concatenation of σ and σ'. This is a kind of confluence property for our on-line algorithm.

Using this property, we can chose any particular scheduling. If we systematically apply (2a) until $\mathcal{A}_i$ has been read for each $i \in I$. Then, switch to the other cases. The resulting scheduling just yields the batch Algorithm 1. Hence, for this scheduling, Theorem 3 is just Theorem 2. ◇

8 Related work

This paper provides contributions to two different topics: distributed diagnosis, and event structures. For each topic we discuss related work.

8.1 Distributed diagnosis

Fault diagnosis in discrete event systems has attracted significant attention, see Lafortune *et al.* [11] and [26] for the *diagnoser* approach. Distributed diagnosis has been less investigated. A first class of studies consider synchronous communications. In [20] distributed diagnosis for Petri nets with synchronous communication is studied. Trading local computations for communications is analysed in [8]. In [28, 29], distributed diagnosis with synchronous product is studied in the linguistic framework of Wonham and Ramadge. Global and local consistency are introduced; global consistency means that local diagnoses are projected versions of global diagnosis, whereas local consistency only requires that local diagnoses agree on their mutual interfaces—the latter is an interesting concept, not considered in our paper.

In [24], Pencolé *et al.* study a more hybrid architecture consisting of synchronously communicating automata and a supervisor that communicates with the subsystems asynchronously. The solutions computed involve additional nondeterminism due to the asynchrony of the communications with the supervisor.

The book [21] proposes a different approach, more in the form of a simulation guided by the observed alarms, for a model of communicating automata, see also [5]. The considered systems are networks of automata interconnected by finitely buffered communications. The problem addressed is that of constructing all correlation scenarios that causally relate faults and observed alarms, for given finite alarm sequences. Monolithic and distributed diagnosis are both considered. This approach shares similarities with ours. In particular, the *active spaces* resemble our unfoldings, without, however, the handling of concurrency. A very interesting idea is proposed to handle unobservable transitions (which we did not consider here): cast in our framework, it consists in not unfolding unobservable loops in the executions of the systems. This allows to keep active spaces finite, even in the presence of loops of unobservable transitions. This is a nice idea that could be reused in our framework. Also, the modular construction of active spaces is studied. The associated architecture is not peer-to-peer but fully hierarchical: there is a single coordinator that communicates with the local diagnosers and performs the fusion of local diagnoses. Even though this is a simpler situation than ours, distributed algorithms are (and have to be) complicated. Still, they are described at a fine grain level, and not by using higher level primitives as we did.

Our results were announced in [2]. Asynchronous diagnosis using unfoldings was presented in [6]. Modeling and diagnosis of distributed systems was first introduced in [16, 15, 17]. Diagnosability for systems with concurrency is discussed in [19]. A systematic study of chaotic algorithms such as used in this paper is performed in [18], including the case of $\mathcal{G}_I$ not being a tree. A first version of this paper was given in [1].

8.2 Event structures

Event structures and unfoldings were first introduced by Glynn Winskel in his thesis, see also [23] and the seminal paper [31]. Unfoldings were subsequently studied in [22, 13, 14], with applications to model checking.

Equipping prime event structures with parallel composition has been recognized difficult. An inductive definition is presented in [10]. Indirect, non inductive, definitions have been proposed by G. Winskel in [33]. F. Vaandrager [30] has proposed a simple direct, non inductive, definition, in categorical style. This is the construction we used here.

Categorical properties of event structures and unfoldings were given in [32]. In particular, the mapping $\mathcal{P} \mapsto \mathcal{U}_\mathcal{P}$ is shown to satisfy $\mathcal{U}_{\mathcal{P}_1 \times \mathcal{P}_2} = \mathcal{U}_{\mathcal{P}_1} \times^s \mathcal{U}_{\mathcal{P}_2}$, expressing that $\mathcal{P} \mapsto \mathcal{U}_\mathcal{P}$ is a functor. Unfortunately, there exists no simple way to compute $\mathcal{U}_{\mathcal{P}_1 \parallel \mathcal{P}_2}$, as we have seen. That unfoldings are functors w.r.t. the synchronous product of nets and unfoldings suggests another approach to distributed diagnosis, in which subystems interact via shared transitions, not places. This gives raise to a different distributed algorithm, in which local diagnosers "over-estimate" solutions, and peer-to-peer cooperations between supervisors aim at keeping only local solutions that "synchronize well", *i.e.,* agree on their interfaces. In contrast, in our distributed algorithm, local diagnosers "under-estimate" solutions (see (66)), and peer-to-peer cooperations between supervisors aim at extending local solutions to reach global consistency. Such an approach by synchronous product is precisely studied in [17]. Despite the use of synchronous products, we insist that the approach of [17] assumes asynchronous communication, since partial order models are considered.

Regarding event structures per se, this paper has introduced the following new special constructs: the $\times^w$-composition (the $\times^s$-composition is classical and simply denoted $\times$), the continuation, the $\|$-composition, the generalized unfolding, and the reduction. These special operations were needed to overcome the fact that no simple "categorical" way of computing $\mathcal{U}_{\mathcal{P}_1 \parallel \mathcal{P}_2}$ exist.

9 Conclusion

For the context of fault management in SDH/SONET telecommunications networks, a prototype software implementing the method was developed in our laboratory, using Java threads to emulate concurrency. This software was subsequently deployed at Alcatel on a truly distributed experimental management platform. No modification was necessary to perform this deployment. A more detailed presentation of this industrial context is found in [3, 4].

To ensure that the deployed application be autonomous in terms of synchronization and control, we have relied on techniques from true concurrency. Regarding concurrency theory, we have adapted to our needs the existing compositional theory of event structures. Event structures form a category equipped with morphisms, projections, and parallel compositions. They provide the adequate mathematical framework and data structures to support distributed diagnosis. We believe that they can be also useful for other distributed problems of observation or control.

The application area which drives our research raises a number of additional issues for further investigation. Getting the model (the net $\mathcal{P}$) is the major one: building the model manually is simply not acceptable. A solution to generate the model automatically is presented in [3]. From the theoretical point of view, the biggest challenge is to extend our techniques to dynamically changing systems. This requires moving from safe Petri nets to the more powerful framework of graph grammars, and is the subject of ongoing research. Then, various robustness issues need to be considered: messages or alarms can be lost, the model can be approximate, etc. Probabilistic aspects are also of interest, to resolve nondeterminism by performing maximum likelihood diagnosis. Paper [7] proposes a mathematical framework for this.

ACKNOWLEDGMENTS. *The authors are deeply indebted to Philippe Darondeau for continuing discussions, fruitful suggestions, and gentle tutoring inside the categorical jungle of event structures.*

A Appendix: Collecting important properties of primitive operators

In this appendix we collect the properties needed to analyse the orchestration. These properties are of interest per se in the context of event structures.

A.1 Properties of the continuation

Lemma 1 *The continuation defined in (32) satisfies the following property:*

$$Q' \supseteq Q \quad \Rightarrow \quad \mathbf{R}_{Q'}(\mathcal{F} \bullet_Q \mathcal{E}) = [\mathbf{R}_{Q'}(\mathcal{F})] \bullet_Q [\mathbf{R}_{Q'}(\mathcal{E})]. \tag{50}$$

Proof. The inclusion $\sqsubseteq$ in (50) is immediate. To show the converse inclusion $\sqsupseteq$, pick an event e' of $[\mathbf{R}_{Q'}(\mathcal{F})] \bullet_Q [\mathbf{R}_{Q'}(\mathcal{E})]$ that is not an event of $\mathbf{R}_{Q'}(\mathcal{F})$. Denote by $F_{Q'}$ the set of events of $\mathbf{R}_{Q'}(\mathcal{F})$. By definition of the continuation,

$$\lceil e' \rceil_{|F_{Q'},Q} \text{ is a maximal configuration of } \mathbf{R}_{Q'}(\mathcal{F})_{|Q}. \tag{51}$$

On the other hand, there exists an event e of $\mathcal{E}$ such that $e' = \mathbf{R}_{Q'}(e)$. Using (51) and the assumption that $Q' \supseteq Q$, we deduce that $\lceil e \rceil_{|F,Q}$ is a maximal configuration of $\mathcal{F}_{|Q}$, which proves (50).

A.2 Properties of labeled event structures and their parallel composition

Proposition 1 *For the following statements, $\mathcal{E} = (E, \preceq, \#, \lambda, P)$ denotes a labeled event structure, and $\mathcal{E}_i = (E_i, \preceq_i, \#_i, \lambda_i, P_i)$, $i \in \{1, 2\}$ denote two labeled event structures. The following formulas hold:*

$$\forall Q \subseteq P \quad : \quad \mathbf{R}_Q(\mathit{trim}(\mathcal{E})) = \mathbf{R}_Q(\mathcal{E}) \tag{52}$$

$$\begin{cases} \mathit{trim}\,[\mathit{trim}(\mathcal{E}_1) \times^s \mathit{trim}(\mathcal{E}_2)] &= \mathit{trim}\,(\mathcal{E}_1 \times^s \mathcal{E}_2) \\ \mathit{trim}\,[\mathit{trim}(\mathcal{E}_1) \times^w \mathit{trim}(\mathcal{E}_2)] &= \mathit{trim}\,(\mathcal{E}_1 \times^w \mathcal{E}_2) \end{cases} \tag{53}$$

$$\forall Q \supseteq P_1 \cap P_2 \quad : \quad \begin{cases} \mathbf{R}_Q(\mathcal{E}_1 \times^s \mathcal{E}_2) &= \mathit{trim}\,[\mathbf{R}_Q(\mathcal{E}_1) \times^s \mathbf{R}_Q(\mathcal{E}_2)] \\ \mathbf{R}_Q(\mathcal{E}_1 \times^w \mathcal{E}_2) &= \mathit{trim}\,[\mathbf{R}_Q(\mathcal{E}_1) \times^w \mathbf{R}_Q(\mathcal{E}_2)] \end{cases} \tag{54}$$

Proof. See Appendix B.1. ◇

Corollary 1 *In the following statements, symbol $\mathbf{1} =_{\text{def}} \emptyset$ denotes the empty event structure. $\mathcal{E}, \mathcal{E}', \mathcal{E}'', \mathcal{E}_i$ denote arbitrary trimmed event structures with respective label sets P, P', P'', P_i.*

Label set Q is arbitrary unless otherwise specified. The following properties hold:

$$\mathcal{E} \,||\, \mathbf{1} = \mathcal{E} \tag{55}$$

$$\mathbf{R}_P(\mathcal{E}) = \mathcal{E} \tag{56}$$

$$\mathbf{R}_{P_1}(\mathbf{R}_{P_2}(\mathcal{E})) = \mathbf{R}_{P_1 \cap P_2}(\mathcal{E}) \tag{57}$$

$$Q \supseteq \mathit{interact}(P_i)_{i\in I} \;\Rightarrow\; \mathbf{R}_Q(\|_{i\in I}\mathcal{E}_i) = \|_{i\in I}\mathbf{R}_Q(\mathcal{E}_i) \tag{58}$$

$$\mathbf{1} \sqsubseteq \mathcal{E} \tag{59}$$

$$P' = P'' \text{ and } \mathcal{E}' \sqsubseteq \mathcal{E}'' \;\Rightarrow\; \mathcal{E}' \,||\, \mathcal{E} \sqsubseteq \mathcal{E}'' \,||\, \mathcal{E} \tag{60}$$

$$\mathcal{E} \sqsubseteq \mathcal{E}' \;\Rightarrow\; \forall Q : \mathbf{R}_Q(\mathcal{E}) \sqsubseteq \mathbf{R}_Q(\mathcal{E}') \tag{61}$$

Proof. To prove (55), note that $\mathcal{E} \times^s \mathbf{1} = \mathcal{E} \times^w \mathbf{1} = \mathcal{E}$. Property (56) is immediate. For (57), note that $\mathbf{R}_{P_1\cap P_2}(\mathcal{E}) =_{\rm def} \mathit{trim}(\mathcal{E}_{P_1\cap P_2}) = \mathit{trim}((\mathcal{E}_{P_2})_{P_1}) =_{\rm def} \mathbf{R}_{P_1}(\mathcal{E}_{P_2}) = \mathbf{R}_{P_1}(\mathbf{R}_{P_2}(\mathcal{E}))$, by Proposition 1, formula (52). For (58), note that $\mathbf{R}_Q(\|_{i\in I}\mathcal{E}_i) =_{\rm def} \mathbf{R}_Q([\prod^s_{i\in I}\ \mathcal{E}_i] \bullet_{Q'} [\prod^w_{i\in I}\ \mathcal{E}_i])$, where $Q' = \mathit{interact}(P_i)_{i\in I}$. Using (50) and the assumption that $Q \supseteq Q'$, we deduce that $\mathbf{R}_Q([\prod^s_{i\in I}\ \mathcal{E}_i] \bullet_{Q'} [\prod^w_{i\in I}\ \mathcal{E}_i]) = \mathbf{R}_Q([\prod^s_{i\in I}\ \mathcal{E}_i]) \bullet_{Q'} \mathbf{R}_Q([\prod^w_{i\in I}\ \mathcal{E}_i])$. Now, by formula (54) of Proposition 1, $\mathbf{R}_Q(\prod^s_{i\in I}\ \mathcal{E}_i) = \mathit{trim}\,[\prod^s_{i\in I}\ \mathbf{R}_Q(\mathcal{E}_i)]$, and similarly with $\prod^w$ instead of $\prod^s$. This proves (58). Finally, monotonicity properties (59)–(61) are immediate. ◇

Lemma 2 *The following properties hold:*

$$\forall j \in I \;:\; \Pi_j\,(\|_{i\in I}\mathcal{E}_i) = \mathcal{E}_j \quad \text{and} \quad \mathbf{R}_{P_j}(\|_{i\in I}\mathcal{E}_i) \sqsupseteq \mathcal{E}_j. \tag{62}$$

Proof. Let κ_j be a maximal configuration of $\mathcal{E}_j$. Let $\mathbf{C}^s_{\kappa_j}$ be the set of maximal configurations of $\prod^s_{i\in I}\ \mathcal{E}_i$ such that $\Pi_j(\kappa) \sqsubseteq \kappa_j$. The following two exclusive cases can occur.

1. There exists a maximal configuration κ of $\prod^s_{i\in I}\ \mathcal{E}_i$ that satisfies $\Pi_j(\kappa) = \kappa_j$ and is also a maximal configuration of $\prod^w_{i\in I}\ \mathcal{E}_i$. Then, κ is also a configuration of $\|_{i\in I}\mathcal{E}_i$ and $\Pi_j(\kappa) = \mathbf{R}_{P_j}(\kappa) = \kappa_j$ holds.
2. For each $\kappa \in \mathbf{C}^s_{\kappa_j}$, $\Pi_j(\kappa) \neq \kappa_j$. Then $\kappa^+_j =_{\rm def} \kappa_j \setminus \Pi_j(\kappa)$ is non empty, and co-set $\min(\kappa^+_j)$ satisfies: $\forall e \in \min(\kappa^+_j) \Rightarrow \lambda(e) \cap Q \neq \emptyset$. Thus, configuration κ cannot be extended at all in the $\times^s$-product. Hence, $\kappa_{|Q}$ is maximal in $\prod^s_{i\in I}\ \mathcal{E}_{i_{|Q}}$. Thus, while performing the $||$-composition, $\kappa_{|Q}$ can be further extended by any maximal configuration $\bar\kappa$ of $\prod^w_{i\in I}\ \mathcal{E}_i$ having κ as its prefix. It remains to chose $\bar\kappa$ such that $\Pi_j(\bar\kappa) = \kappa_j$, and $\bar\kappa$ also satisfies $\mathbf{R}_{P_j}(\bar\kappa) = \kappa_j$. This finishes the proof. ◇

A.3 Properties of event structures related to unfoldings

In this paragraph we collect important properties involving the parallel composition of event structures obtained from unfoldings.

Proposition 2 *Let $\mathcal{P} = \mathcal{P}_1 \,||\, \mathcal{P}_2$. We have*

$$\mathcal{E}_{\mathcal{P}} \sqsubseteq \mathit{trim}\,(\mathbf{R}_{P_1}(\mathcal{E}_{\mathcal{P}}) \times^s \mathbf{R}_{P_2}(\mathcal{E}_{\mathcal{P}}))\,. \tag{63}$$

Proof. See Appendix B.2. ◇

Corollary 2 *Let $\mathcal{P} = ||_{i\in I}\mathcal{P}_i$ be a Petri net, and let P_i be the set of places of $\mathcal{P}_i$. The following property holds:*

$$\mathcal{E}_{\mathcal{P}} \quad \sqsubseteq \quad \Big|\Big|_{i\in I}\mathbf{R}_{P_i}(\mathcal{E}_{\mathcal{P}}) \tag{64}$$

Proof. To show (64), note that Proposition 2 generalizes to more than two components, whence $\mathcal{E}_{\mathcal{P}} \sqsubseteq \mathit{trim}(\prod^s_{i\in I}\ \mathbf{R}_{P_i}(\mathcal{E}_{\mathcal{P}}))$. Then, Proposition 3 used with substitution $\mathcal{I} \leftarrow \mathcal{E}_{\mathcal{P}}$, together with the fact that $\mathit{trim}(\prod^s_{i\in I}\ \mathcal{E}_i) \sqsubseteq \Big|\Big|_{i\in I}\mathcal{E}_i$, proves (64). ◇

Proposition 3 *Assume that decomposition $\mathcal{P} = ||_{i\in I}\mathcal{P}_i$ satisfies assumption 1. Let $\mathcal{I}$ be a prefix of $\mathcal{E}_{\mathcal{P}}$, and assume that $\mathcal{I}_i =_{\rm def} \mathbf{R}_{P_i}(\mathcal{I})$ satisfies $\Big|\Big|_{i\in I}\mathcal{I}_i \sqsubseteq \mathcal{E}_{\mathcal{P}}$. For $i \in I$ consider $\mathcal{E}_i = \mathcal{E}^{\mathcal{I}_i}_{\mathcal{P}_i}$, the unfolding of $\mathcal{P}_i$ from $\mathcal{I}_i$. Then, we have:*

$$\Big|\Big|_{i\in I}\mathcal{E}_i \quad \sqsubseteq \quad \mathcal{E}_{\mathcal{P}}. \tag{65}$$

Note that, for $\mathcal{I} = \emptyset$, (65) boils down to

$$\Big|\Big|_{i\in I}\mathcal{E}_{\mathcal{P}_i} \quad \sqsubseteq \quad \mathcal{E}_{\mathcal{P}}. \tag{66}$$

Proof. See Appendix B.3. ◇

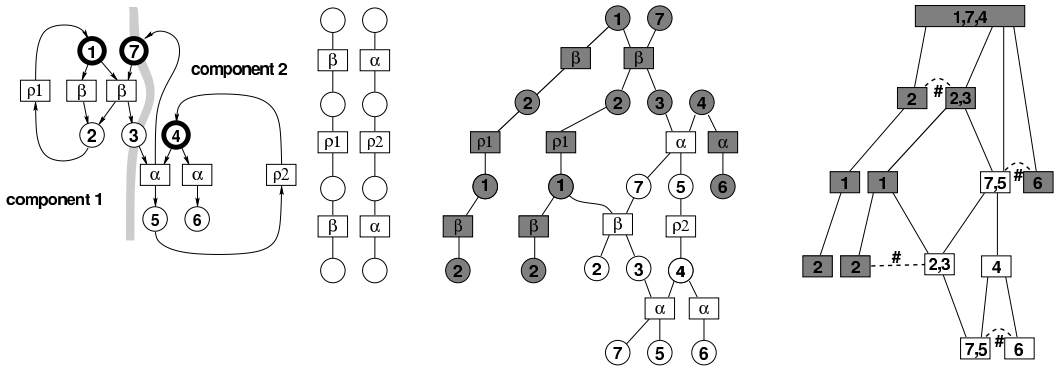

Figure 21: Illustrating Proposition 3.

Running example, continued. Proposition 3 is illustrated in Fig. 21. The first two diagrams are taken from Fig. 11; they show the two components together with the alarm pattern recorded with two independent sensors. The 3rd diagram is a copy of the last diagram of Fig. 11 in which we discarded the long configuration κ_3, for readibility purposes. The prefix filled in grey in this diagram shows the part of the diagnosis that can be inferred

by each supervisor locally, i.e., by observing its own alarms, knowing its local model only, and without cooperating with the other supervisor. In contrast, the white suffix requires the two supervisors to cooperate. The last diagram shows an event structure translation of the third one, by showing $\mathbf{R}_P(\mathcal{E}_{\mathcal{A}\times\mathcal{P}})$, and its prefix $\mathbf{R}_{P_1}(\mathcal{E}_{\mathcal{A}_1\times\mathcal{P}_1}) \,||\, \mathbf{R}_{P_2}(\mathcal{E}_{\mathcal{A}_2\times\mathcal{P}_2})$ (in grey). $\diamond$

B Appendix: Proofs

B.1 Proof of Proposition 1

Proof of formula (52). By expanding the definition of the reduction, this formula is rewritten

$$trim\left(\mathcal{F}_{|Q}\right) \quad = \quad trim\left(\mathcal{E}_{|Q}\right), \text{ where } \mathcal{F} =_{\text{def}} trim(\mathcal{E}) \tag{67}$$

For this proof we need to make precise the context in which we consider trimming and configurations: $trim_{\mathcal{E}}$, $\lceil e \rceil_{\mathcal{E}}$, and $\kappa_{\sim_{\mathcal{E}}}$, shall respectively denote the trimming function, the configuration generated by event e, and the equivalence class of configuration κ modulo $\sim$, in the context of $\mathcal{E}$. The reader is referred to (19) for the definition of $\mathcal{E}_{|Q}$.

Set $E_Q = \{e \in E \mid \lambda(e) \cap Q \neq \emptyset\}$. Then $\mathcal{E}_{|Q}$ possesses E_Q as event set, $\lambda_Q = \lambda \cap Q$ as labeling map, and inherits the causality and conflict relations from $\mathcal{E}$, by restriction. For $e \in E_Q$, write $\mathbf{e} =_{\text{def}} trim_{\mathcal{E}_{|Q}}(e)$. Write $\mathbf{E}_Q =_{\text{def}} trim(\mathcal{E}_{|Q})$. Using the self-reproducing property (18), we get

$$\lceil \mathbf{e} \rceil_{\mathbf{E}_Q} \quad \sim \quad \lceil e \rceil_{\mathcal{E}_Q}. \tag{68}$$

On the other hand, for $e \in E$, we have $\mathbf{f} =_{\text{def}} trim_{\mathcal{E}}(e) = (\lceil e \rceil_{\mathcal{E}})_{\sim_{\mathcal{E}}}$, $\mathbf{f}$ has label $\lambda(e)$, and $\mathcal{F}$ has its causality and conflict relations defined by (17). Thus $\mathcal{F}_{|Q}$ has $F_Q =_{\text{def}} \{trim_{\mathcal{E}}(e) \mid e \in E_Q\}$ as set of events, and inherits the causality and conflict relations from $\mathcal{F}$, by restriction. Then, by (17), the label of $trim_{\mathcal{E}}(e)$ in the context of $\mathcal{F}_{|Q}$ is equal to $\lambda(e) \cap Q = \lambda_Q(e)$. For $\mathbf{f} \in F_Q$, set $\mathbf{g} =_{\text{def}} trim_{\mathcal{F}_{|Q}}(\mathbf{f})$ and $\mathbf{G} =_{\text{def}} trim(\mathcal{F}_{|Q})$. Using twice the self-reproducing property (18)—once for each trimming operation—we get

$$\lceil \mathbf{g} \rceil_{\mathbf{G}} \sim \lceil \mathbf{f} \rceil_{\mathcal{F}_{|Q}} \quad , \quad \lceil \mathbf{f} \rceil_{\mathcal{F}} \sim \lceil e \rceil_{\mathcal{E}} \tag{69}$$

Restrict to Q the 2nd equivalence in (69), this yields $\lceil \mathbf{f} \rceil_{\mathcal{F}_Q} \sim \lceil e \rceil_{\mathcal{E}_Q}$. Combining the latter formula with (68) and (69) yields $\lceil \mathbf{g} \rceil_{\mathbf{G}} \sim \lceil \mathbf{e} \rceil_{\mathbf{E}_Q}$, which proves (67).

Proof of formula (53). We only prove the $\times^s$ case, the $\times^w$ case is proved in exactly the same way.

Events e of $\mathcal{E} =_{\text{def}} \mathcal{E}_1 \times^s \mathcal{E}_2$ are defined by (25,28), and (29). Write $\mathbf{e} =_{\text{def}} trim_{\mathcal{E}}(e)$ and $\mathbf{E} =_{\text{def}} trim(\mathcal{E})$. Using (25,28), and (29), and the self-reproducing property (18), yields

$$\lceil \mathbf{e} \rceil_{\mathbf{E}} \sim \lceil e \rceil_{\mathcal{E}} \quad , \quad \Pi_i(\lceil e \rceil_{\mathcal{E}}) = \lceil e_i \rceil_{\mathcal{E}_i}, \text{ for } i \in \{1,2\}. \tag{70}$$

On the other hand, write $\mathbf{E}_i =_{\rm def} trim(\mathcal{E}_i)$ and $\mathbf{e}_i =_{\rm def} trim_{\mathcal{E}_i}(e_i)$ for $i \in \{1,2\}$. Next, write $\mathcal{F} =_{\rm def} \mathbf{E}_1 \times \mathbf{E}_2$ and $\mathbf{F} =_{\rm def} trim(\mathcal{F})$; for f an event of $\mathcal{F}$, write $\mathbf{f} =_{\rm def} trim_{\mathcal{F}}(f)$. We have

$$\lceil \mathbf{f} \rceil_{\mathbf{F}} \sim \lceil f \rceil_{\mathcal{F}} \quad , \quad \Pi_i(\lceil f \rceil_{\mathcal{F}}) = \lceil \mathbf{e}_i \rceil_{\mathbf{E}_i} \sim \lceil e_i \rceil_{\mathcal{E}_i} \, , \text{ for } i \in \{1,2\}. \tag{71}$$

Formulas (70) and (71) together yield $\lceil \mathbf{f} \rceil_{\mathbf{F}} \sim \lceil \mathbf{e} \rceil_{\mathbf{E}}$, which proves formula 53.

Proof of formula 54. We only prove the $\times^s$ case, the $\times^w$ case is proved in exactly the same way.

Assume the following formula is proved:

$$(\mathcal{E}_1 \times^s \mathcal{E}_2)_{|Q} \quad = \quad (\mathcal{E}_1)_{|Q} \times^s (\mathcal{E}_2)_{|Q} \, . \tag{72}$$

Then, using formula (53), we get

$$\begin{aligned} \mathbf{R}_Q(\mathcal{E}_1 \times^s \mathcal{E}_2) &= trim\left((\mathcal{E}_1 \times^s \mathcal{E}_2)_{|Q} \right) \\ &= trim\left((\mathcal{E}_1)_{|Q} \times^s (\mathcal{E}_2)_{|Q} \right) \\ &= trim\left(trim((\mathcal{E}_1)_{|Q}) \times^s trim((\mathcal{E}_2)_{|Q}) \right) \\ &= trim\left(\mathbf{R}_Q(\mathcal{E}_1) \times^s \mathbf{R}_Q(\mathcal{E}_2) \right) \end{aligned}$$

which proves (54). Thus it remains to prove (72).

The reader is referred to the definition (25,28,29,30) for the parallel composition of labeled event structures. For $i \in \{1,2\}$, denote by $E_{i,Q}$ the restriction of E_i to the subset of events e_i such that $\lambda_i(e_i) \cap Q \neq \emptyset$, and define, for $e_i \in E_i$: $\lambda_{i,Q}(e_i) =_{\rm def} \lambda_i(e_i) \cap Q$. Similarly, we denote by E_Q the restriction of E to the subset of events e such that $\lambda(e) \cap Q \neq \emptyset$, and define, for $e \in E$: $\lambda_Q(e) =_{\rm def} \lambda(e) \cap Q$.

We first focus on labels. We can equip $E_{i,Q}$ with labeling map $\lambda_{i,Q}$, or alternatively with labeling map λ_i, for $i \in \{1,2\}$. By the definition (28) of $\bowtie_s$, and using the special condition that $Q \supseteq P_1 \cap P_2$, we get, for every pair $(e_1, e_2) \in E_{1,Q} \times_\star E_{2,Q}$:

$$\begin{array}{c} e_1 \bowtie_s e_2 \text{ w.r.t. labeling maps } (\lambda_{1,Q}, \lambda_{2,Q}), \text{ written } e_1 \bowtie_{s,Q} e_2 \\ \Updownarrow \\ e_1 \bowtie_s e_2 \text{ w.r.t. labeling maps } (\lambda_1, \lambda_2), \text{ simply written } e_1 \bowtie_s e_2 \end{array}$$

Consequently,

$$E_{1,Q} \times_{s,Q} E_{2,Q} \quad = \quad [E_1 \times^s E_2]_Q \tag{73}$$

where $\times_{s,Q}$ and $\times^s$ are defined in (29) by using $\bowtie_{s,Q}$ and $\bowtie_s$, respectively, and $[.]_Q$ denotes the restriction to events having a label that intersects Q.

We then focus on (24,25). Pre-configurations associated to the right hand side of (72) are the subsets κ of $E_{1,Q} \times_{s,Q} E_{2,Q}$ satisfying the two conditions (i,ii) stated before (25), where π_i^Q denotes the projection from $E_{1,Q} \times_{s,Q} E_{2,Q}$ onto $E_{i,Q}$. We explicit these two conditions next:

(i)^Q For $i \in \{1,2\}$, $\pi_i^Q(\kappa)$ is a configuration of $(\mathcal{E}_i)_{|Q}$;

(ii)^Q $\preceq_\kappa^Q$, the transitive closure of relation $\leq^Q \cap(\kappa\times\kappa)$, is a partial order, where the relation $\leq^Q \subseteq E_{1,Q} \times_{s,Q} E_{2,Q}$ is defined by:

$$e \leq^Q e' \quad \Leftrightarrow \quad \pi_1^Q(e) \preceq_1^Q \pi_1^Q(e') \text{ or } \pi_2^Q(e) \preceq_2^Q \pi_2^Q(e'),$$

where $\preceq_i^Q$ denotes the causality relation on $(\mathcal{E}_i)_{|Q}$.

On the other hand, pre-configurations associated to $\mathcal{E}_1 \times^s \mathcal{E}_2$ are the subsets κ of $E_1 \times^s E_2$ satisfying the two conditions (i,ii) stated before (25), where π_i denotes the projection from $E_1 \times^s E_2$ onto E_i. We repeat these two conditions next:

(i) For $i \in \{1,2\}$, $\pi_i(\kappa)$ is a configuration of $\mathcal{E}_i$;

(ii) $\preceq_\kappa$, the transitive closure of relation $\leq \cap(\kappa \times \kappa)$, is a partial order, where the relation $\leq \subseteq E_1 \times^s E_2$ is defined by:

$$e \leq e' \quad \Leftrightarrow \quad \pi_1(e) \preceq_1 \pi_1(e') \text{ or } \pi_2(e) \preceq_2 \pi_2(e').$$

Thus, the configurations of $(\mathcal{E}_1 \times^s \mathcal{E}_2)_{|Q}$ have the form $\kappa_Q =_{\text{def}} \kappa \cap E_Q$, where κ satisfies the above two conditions (i,ii). Now, with κ as above, we have $\pi_i^Q(\kappa \cap E_Q) = \pi_i(\kappa) \cap E_{i,Q}$, and thus:

$$\begin{array}{rl} & \pi_i^Q(\kappa_Q) \text{ is a configuration of } (\mathcal{E}_i)_{|Q} \\ \text{iff} & \pi_i(\kappa) \text{ is a configuration of } \mathcal{E}_i. \end{array} \tag{74}$$

Similarly, the two relations $\leq^Q$ and $\leq$ coincide on $E_{1,Q} \times_{s,Q} E_{2,Q} = [E_1 \times^s E_2]_Q$ (cf. (73)). Thus:

$$\begin{array}{rl} & \kappa_Q \text{ is a pre-configuration of } (\mathcal{E}_1 \times^s \mathcal{E}_2)_{|Q} \\ \text{iff} & \text{it is a pre-configuration of } (\mathcal{E}_1)_{|Q} \times^s (\mathcal{E}_2)_{|Q}. \end{array} \tag{75}$$

Finally, (73), (74), and (75) together with conditions (i)^Q, (ii)^Q, (i), and (ii), prove (72). This in turn proves (54).

B.2 Proof of Proposition 2

Equivalently, it is enough to show that

$$\begin{array}{r} \forall \kappa \text{ configuration of } \mathcal{E}_\mathcal{P}, \\ \exists \kappa' \text{ configuration of } \mathbf{R}_{P_1}(\mathcal{E}_\mathcal{P}) \times^s \mathbf{R}_{P_2}(\mathcal{E}_\mathcal{P}) \text{ such that } \kappa' \sim \kappa. \end{array} \tag{76}$$

This is shown by structural induction. Assume that (76) holds for each configuration κ contained in some finite prefix $\mathcal{F}$ such that $\mathcal{F} \sqsubseteq \mathcal{E}_\mathcal{P}$. Select $e \in \mathcal{E}_\mathcal{P}$ such that $e \notin \mathcal{F}$ and

$e \in {}^\bullet\mathcal{F}$. Since e is an event of the unfolding of $\mathcal{P}$, it represents a unique transition t of this Petri net, and $\lambda(e) = t^\bullet$. Furthermore, if $X =_{\rm def} {}^\bullet e$ is the preset of e in $\mathcal{E}_\mathcal{P}$, then $\lambda(X) \supseteq {}^\bullet t$. Since transitions are private, we can assume, say, that t is a transition of $\mathcal{P}_1$ only.

Set $\kappa =_{\rm def} \lceil e \rceil_{|\mathcal{F}}$, we have $X = \max(\kappa)$. Let κ' satisfy (76), such a κ' exists by induction hypothesis. Set $X' =_{\rm def} \max(\kappa')$, we have $X' \sim X$.

Denote by Π_i the canonical projections $\mathbf{R}_{P_1}(\mathcal{E}_\mathcal{P}) \times^s \mathbf{R}_{P_2}(\mathcal{E}_\mathcal{P}) \mapsto \mathbf{R}_{P_i}(\mathcal{E}_\mathcal{P})$, for $i \in \{1,2\}$, and set $X_i =_{\rm def} \Pi_i(X)$. The following cases can occur (recall that t is a transition of $\mathcal{P}_1$ only):

(a) $t^\bullet \cap P_2 = \emptyset$.

Equivalently, $\lambda(e) = t^\bullet \cap (P_1 \cap P_2) = \emptyset$. Then, $e' =_{\rm def} (\mathbf{R}_{P_1}(e), \star)$ is an event of $\mathbf{R}_{P_1}(\mathcal{E}_\mathcal{P}) \times^s \mathbf{R}_{P_2}(\mathcal{E}_\mathcal{P})$ such that $\kappa' \cup \{e'\} \sim \kappa \cup \{e\}$.

(b) $t^\bullet \cap P_2 \neq \emptyset$.

Note that $\lambda(\lceil {}^\bullet e \rceil) \cap (P_1 \cap P_2) \neq \emptyset$ holds, since $\mathcal{E}_\mathcal{P}$ has a unique minimal event labeled by the initial marking. Therefore, there exists some maximal event $f \in \lceil {}^\bullet e \rceil$ such that $\lambda(f) \cap (P_1 \cap P_2) \neq \emptyset$. Let Y be the co-set consisting of those f's, set $\kappa =_{\rm def} \lceil Y \rceil$, and let κ' satisfy (76), such a κ' exists by induction hypothesis. Set $Y' =_{\rm def} \max(\kappa')$, we have $Y' \sim Y$. Denote by $[Y, e]$ the postfix of $\lceil e \rceil$ consisting of the events posterior to or included in Y, and set $]Y, e[=_{\rm def} [Y, e] \setminus (Y \cup \{e\})$. By construction, $[Y, e]_{|P_2} = Y_{|P_2} \cup \{e\}_{|P_2}$, and $[Y, e]_{|P_1} =]Y, e[\cup Y_{|P_1} \cup \{e\}_{|P_1}$, see (19) for the definition of the restriction $\mathcal{E}_{|Q}$. Thus

$$\begin{array}{rcl} \kappa'' & =_{\rm def} & \kappa' \\ & & \cup \quad \{(\mathbf{R}_{P_1}(e_1), \star) \mid e_1 \in]Y, e[\} \\ & & \cup \quad \{e\} \end{array}$$

is a preconfiguration of $E_1 \times^s E_2$, where E_i is the set of events of $\mathbf{R}_{P_i}(\mathcal{E}_\mathcal{P})$, for $i \in \{1,2\}$. This pre-configuration κ'' is a complete prime that is, by construction, $\sim$-equivalent to $\lceil e \rceil$.

This shows (76).

B.3 Proof of Proposition 3

Let us prove first the following property, where symbol $\times$ denotes equally $\times^s$ or $\times^w$:

$$\begin{array}{c} \forall \kappa \text{ configuration of } \mathcal{E}_1 \times \mathcal{E}_2, \\ \exists\ \kappa' \text{ configuration of } \mathcal{E}_{\mathcal{P}_1 \parallel \mathcal{P}_2} \text{ such that } \kappa' \sim \kappa. \end{array} \tag{77}$$

This is shown by structural induction. Assume that (77) holds for each configuration κ contained in some finite prefix $\mathcal{F}$ such that $\mathcal{F} \sqsubseteq \mathcal{E}_1 \times \mathcal{E}_2$. With no loss of generality we can also assume $\mathcal{I} \sqsubseteq \mathcal{F}$. Select $e \notin \mathcal{F}$ such that $X =_{\rm def} {}^\bullet e \subseteq \mathcal{F}$, and denote by κ the smallest

configuration that contains X. Using the induction hypothesis, let κ' be a configuration of $\mathcal{E}_{\mathcal{P}_1 \,\|\, \mathcal{P}_2}$ such that $\kappa' \sim \kappa$.

Decompose e as $e = (e_1, e_2)$, where $e_i = \Pi_i(e)$ for $i \in \{1, 2\}$ (see (26) for the definition of Π_i). Apply rule (39). We must have, say, $\Pi_1(X) = {}^\bullet e_1$ and $\Pi_2(X) \supseteq {}^\bullet e_2$. There exists a transition t_1 of net $\mathcal{P}_1$ such that ${}^\bullet t_1 = \lambda({}^\bullet e_1)$ and $t_1^\bullet = \lambda(e_1)$. Then, the following cases can occur:

(a) $e_2 = \star$ and thus ${}^\bullet e_2 = \emptyset$. Hence firing t_1 alone yields a valid transition of $\mathcal{P}_1 \,||\, \mathcal{P}_2$, and t_1 can be fired from ${}^\bullet e$ and yields $e = (e_1, \star)$.

(b) $e_2 \neq \star$. Then there exists a transition t_2 of net $\mathcal{P}_2$ such that ${}^\bullet t_2 = \lambda({}^\bullet e_2)$ and $t_2^\bullet = \lambda(e_2)$. Since events e_1 and e_2 are compatible (see (28)), we must have $\lambda(e_1) \cap P_1 \cap P_2 = \lambda(e_2) \cap P_1 \cap P_2 \neq \emptyset$. Pick $p \in \lambda(e_2) \cap P_1 \cap P_2$. Since there is no distributed conflict (see (44)), we must have either $p^\bullet \subset T_1$ or $p^\bullet \subset T_2$. If the former occurs, then, by (45), ${}^\bullet p \subset T_2$ must occur. But this contradicts the assumption that $p \in \lambda(e_2) \cap P_1 \cap P_2$. Hence case (b) cannot occur.

Therefore, $\kappa \cup \{e\}$ is $\sim$-equivalent to a configuration of $\mathcal{E}_{\mathcal{P}_1 \,\|\, \mathcal{P}_2}$, which shows (77) by induction.

Now, by definition of the $||$ operator, each configuration of $\mathcal{E}_1 \,||\, \mathcal{E}_2$ is $\sim$-equivalent to some configuration of $\mathcal{E}_1 \times^w \mathcal{E}_2$. Since both $\times^s$ and $\times^w$ are associative, property (77) is valid for more than two components. This proves Proposition 3.

B.4 Proof of Theorem 2

Since we already know that Algorithm 1 is confluent, we can choose any particular scheduling of the steps (2b), (2c), or (2a), of Algorithm 1. To specify the particular scheduling we use for our analysis, we use the following notational conventions: (a_i) shall denote the application of step (2a) where node i is selected. Same convention holds regarding notation (c_i); finally, $(b_{i,j})$ shall denote the application of step (2b) where directed edge (i, j) is selected.

Select one node i_0 and regard it as the origin of the tree $\mathcal{G}_I$. Since $\mathcal{G}_I$ is a tree, we can label its nodes by the length ℓ_i of the unique path linking node $i \in I$ to the particular node i_0. Using ℓ and the alphabet Σ defined in (47), define the special scheduling $\sigma_\mathcal{G}$ shown in Fig. 22, and illustrated in Fig. 23. The expert reader will recognize the scheduling used for on-line smoothing algorithms in control.

The proof proceeds according to four successive steps. Our analysis of Algorithm 1 applied with scheduling $\sigma_\mathcal{G}$ proceeds by induction over the successive rounds (1;2;3). For each considered round, call $\mathcal{E}_i^-$ and $\mathcal{E}_i$ the ith event structures before and after the considered round, respectively. Also, $\mathcal{E}_i^\infty$ denotes the limit of Algorithm 1.

Step 1 *The following invariant holds at the end of each round (1;2;3) of Algorithm 1, applied with scheduling $\sigma_\mathcal{G}$: for all $i \in I$,*

$$\mathbf{R}_{P_i}(\|_{k \in I} \mathcal{E}^{\mathcal{I}_k}_{\mathcal{A}_k \times \mathcal{P}_k}) \sqsubseteq \mathcal{E}_i \sqsubseteq \mathbf{R}_{P_i}(\|_{k \in I} [\mathcal{E}^{\mathcal{I}_k}_{\mathcal{A}_k \times \mathcal{P}_k}]^2), \text{ where } \mathcal{I}_k =_{\text{def}} \mathcal{E}_k^- . \tag{78}$$

Repeat round (1;2;3) below, until convergence:

1. Local diagnosis*: perform* (a_i) *once for each* $i \in I$ *in some arbitrary order;*
2. Inward iteration*: repeat for* $\ell = \ell_{\max} - 1, \ell_{\max} - 2, \ldots, 0$
   - *For every* j *such that* $\ell_j = \ell$*, repeat*
     (i) $\forall i : (i,j) \in \mathcal{G}_I$ *and* $\ell_i = \ell_j + 1 \Rightarrow$ *perform* $(b_{i,j})$*;*
     (ii) *perform* (c_j)*;*
3. Outward iteration*: repeat for* $\ell = 0, 1, \ldots, \ell_{\max} - 1$
   - *For every* j *such that* $\ell_j = \ell$*, repeat*
     – $\forall i : (j,i) \in \mathcal{G}_I$ *and* $\ell_i = \ell_j + 1 \Rightarrow$ *perform* $(b_{j,i})$*; perform* (c_i).

Figure 22: Special scheduling $\sigma_{\mathcal{G}}$. Symbol ";" indicates sequencing.

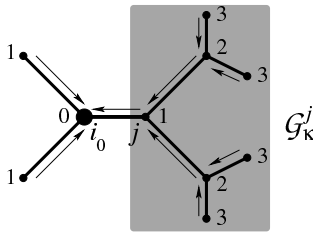

Figure 23: Illustrating the "inward iteration" of the scheduling of Fig. 22. The numbers shown give the ℓ distance.

Proof. The proof of (78) is by induction. Applying round 1 of $\sigma_{\mathcal{G}}$ yields

$$\mathcal{E}_i^+ \quad =_{\mathrm{def}} \quad \mathcal{E}_{\mathcal{P}}^{\mathcal{I}} \text{ with the substitutions } \left\{ \begin{array}{lcl} \mathcal{I} & \leftarrow & \mathcal{E}_i^- \\ \mathcal{P} & \leftarrow & \mathcal{A}_i \times \mathcal{P}_i \end{array} \right.$$

Next, apply round 2 of $\sigma_{\mathcal{G}}$. While performing this, mark $\mathcal{M}_{i,j}$ in formula (46) with a running multiset $J_{i,j} \in (I \mapsto \mathbf{N})$, initialized with $\forall k \in I : J_{i,j}(k) = 0$, and updated as follows:

$$\begin{array}{rcl} \mathcal{M}_{i,j} & := & \mathbf{R}_{P_i \cap P_j}(\mathcal{E}_i^+ \parallel \left[\|_{k \in \mathrm{N}(i) \setminus j} \mathcal{M}_{k,i} \right]) \\ J_{i,j} & := & J_{i,j} + \delta_i + \sum_{k \in \mathrm{N}(i) \setminus j} J_{k,i}, \\ \text{where } \delta_i(k) & =_{\mathrm{def}} & 1 \text{ iff } k = i, \text{ and } = 0 \text{ otherwise.} \end{array} \tag{79}$$

After having completed round (2.i) of scheduling $\sigma_{\mathcal{G}}$, we have:

$$J_{i,j}(k) \quad = \quad 1 \text{ if } k \text{ is a vertex of } \mathcal{G}_I^i, \; = \; 0 \text{ otherwise,} \tag{80}$$

where $\mathcal{G}_I^i$ denotes the subtree of $\mathcal{G}_I$ comprising the nodes $k \in I$ such that $\ell_k \geq \ell_i$ and i belongs to the path linking k to the origin i_0, see Fig. 23. We claim that the following invariant holds throughout round 2 of $\sigma_{\mathcal{G}}$:

$$\mathcal{M}_{i,j} = \mathbf{R}_{P_j}(\|_{k\in\mathrm{supp}(J_{i,j})}\, \mathcal{E}_k^+), \text{ where } \mathrm{supp}(J_{i,j}) =_{\mathrm{def}} \{k \mid J_{i,j}(k) > 0\}. \tag{81}$$

Property (81) holds if i is a leaf of the tree $\mathcal{G}_I$, since $J_{i,j} = \delta_i$ in this case. Assume that (81) holds for all nodes i sitting at a distance $\ell_i \geq n$, we shall prove that it also holds for all nodes i sitting at a distance $\ell_i \geq n-1$; this will prove the invariance of (81) by inward induction. Select such a node i, and denote by k^- the unique node of $\mathcal{G}_I$ that sits next to any node k in the path linking node k to the origin i_0. We have

$$\begin{aligned}
& \mathcal{M}_{i,j} \\
= \;& \mathbf{R}_{P_i\cap P_j}(\mathcal{E}_i^+ \,\|\, \left[\|_{k\in\mathrm{N}(i)\setminus j}\mathcal{M}_{k,i}\right]) && \text{(by (46))} \\
= \;& \mathbf{R}_{P_i\cap P_j}(\mathcal{E}_i^+ \,\|\, \left[\|_{k\in\mathrm{N}(i)\setminus j}\mathbf{R}_{P_i}(\|_{k\in\mathrm{supp}(J_{k,k^-})}\, \mathcal{E}_h^+)\right]) && \text{(b)} \\
= \;& \mathbf{R}_{P_i\cap P_j}(\mathcal{E}_i^+ \,\|\, \left[\mathbf{R}_{P_i}(\|_{k\in\mathrm{N}(i)\setminus j}(\,\|_{k\in\mathrm{supp}(J_{k,k^-})}\, \mathcal{E}_h^+\,))\right]) && \text{(by (58))} \\
= \;& \mathbf{R}_{P_i\cap P_j}(\mathbf{R}_{P_i}(\mathcal{E}_i^+) \,\|\, \left[\mathbf{R}_{P_i}(\|_{k\in\mathrm{N}(i)\setminus j}(\,\|_{k\in\mathrm{supp}(J_{k,k^-})}\, \mathcal{E}_h^+\,))\right]) && \text{(by (56))} \\
= \;& \mathbf{R}_{P_i\cap P_j}(\mathbf{R}_{P_i}(\|_{h\in\mathrm{supp}(J_{i,j})}\, \mathcal{E}_h^+)) && \text{(e)} \\
= \;& \mathbf{R}_{P_i\cap P_j}(\|_{h\in\mathrm{supp}(J_{i,j})}\, \mathcal{E}_h^+) && \text{(by (57))} \\
= \;& \mathbf{R}_{P_j}(\|_{h\in\mathrm{supp}(J_{i,j})}\, \mathcal{E}_h^+) && \text{(g)}
\end{aligned}$$

where (b) follows from the induction hypothesis, (e) follows from the identity $J_{i,j} = \delta_i + \sum_{k\in\mathrm{N}(i)\setminus j} J_{k,k^-}$, and (g) follows from the fact that, for each h such that $J_{i,j}(h) > 0$, the label set of $\mathcal{E}_h^+$ has empty intersection with $P_j \setminus P_i$. Thus (81) is proved by inward induction over the set of nodes of the tree $\mathcal{G}_I$.

The reader should have noticed that the assumption that $\mathcal{G}_I$ is a tree has been used in applying (58) for proving the third equality: the label set of $\|_{k\in\mathrm{supp}(J_{k,k^-})}\, \mathcal{E}_h^+$ is equal to $\overline{P}_k =_{\mathrm{def}} \bigcup_{k\in\mathrm{supp}(J_{k,k^-})} P_h$, and $\overline{P}_k \cap \overline{P}_{k'} \subseteq P_i$ for $k \neq k'$ belonging both to $\mathrm{N}(i) \setminus j$.

Using (81), we deduce that performing step (2.ii) after round (2.i) of scheduling $\sigma_{\mathcal{G}}$ yields at site j an updated event structure such that

$$\begin{array}{rcll}
\mathcal{E}_j & = & \mathcal{E}_j^+ \,||\, \left[||_{i\in \mathrm{N}(j)} \mathcal{M}_{i,j}\right] & \\
& = & \mathcal{E}_j^+ \,||\, \left[||_{i\in \mathrm{N}(j)\setminus j^-} \mathbf{R}_{P_j}(||_{k\in \mathrm{supp}(J_{i,j})} \mathcal{E}_k^+)\right] & \\
& = & \mathcal{E}_j^+ \,||\, \left[\mathbf{R}_{P_j}(||_{i\in \mathrm{N}(j)\setminus j^-} (\,||_{k\in \mathrm{supp}(J_{i,j})} \mathcal{E}_k^+))\right] & \text{(by (58))} \\
& = & \mathbf{R}_{P_j}(\mathcal{E}_j^+) \,||\, \left[\mathbf{R}_{P_j}(||_{i\in \mathrm{N}(j)\setminus j^-} (\,||_{k\in \mathrm{supp}(J_{i,j})} \mathcal{E}_k^+))\right] & \text{(by (56))} \\
& = & \mathbf{R}_{P_j}(\mathcal{E}_j^+ \,||\, \left[||_{i\in \mathrm{N}(j)\setminus j^-} (\,||_{k\in \mathrm{supp}(J_{i,j})} \mathcal{E}_k^+)\right]) & \text{(by (58))} \\
& = & \mathbf{R}_{P_j}(\,||_{k\in \mathrm{supp}(J_{j,j^-})} \mathcal{E}_k^+), &
\end{array} \tag{82}$$

since $J_{j,j^-} = \delta_j + \sum_{k\in \mathrm{N}(j)\setminus j^-} J_{k,k^-}$. Again, in applying (58) we have used the assumption that $\mathcal{G}_I$ is a tree. Taking $j = i_0$ in (82) yields in particular

$$\mathcal{E}_{i_0} \;=\; \mathbf{R}_{P_{i_0}}(||_{k\in I}\mathcal{E}_k^+).$$

So far each vertex of $\mathcal{G}_I$ was visited only once, thus all encountered multisets had multiplicity 0 or 1. Performing the "outward" round 3 of scheduling $\sigma_{\mathcal{G}}$ yields multisets $J_{i,j}$ with multiplicity 1 or 2. Using the convention $\mathcal{E}^0 =_{\mathrm{def}} \mathbf{1}$, rewrite (82) as follows:

$$\mathcal{E}_j \;=\; \mathbf{R}_{P_j}(\,||_{k\in I}[\mathcal{E}_k^+]^{J_{j,j^-}(k)}). \tag{83}$$

This new form (83) for the invariant also holds for the "outward" round 3 of scheduling $\sigma_{\mathcal{G}}$ (the proof is identical to that performed for the inward round). Since $1 \le J_{j,j^-}(k) \le 2$ holds at the end of round 3, (78) follows. ◇

Step 2 *For each $i \in I$,*

$$\mathbf{R}_{P_i}(\mathcal{E}_{\mathcal{A}\times\mathcal{P}}) \;\sqsubseteq\; \mathcal{E}_i^\infty \tag{84}$$

(Recall that $\mathcal{E}_i^\infty$ denotes the limit of Algorithm 1.)

Proof. By (78), and since $(\mathcal{E}_i^\infty)_{i\in I}$ is a fixpoint of Algorithm 1, we get

$$\forall i \in I \;:\; \mathcal{E}_i^\infty \;=\; \mathbf{R}_{P_i}(||_{k\in\mathcal{G}_I} \mathcal{E}^{\mathcal{I}_k}_{\mathcal{A}_k\times\mathcal{P}_k}),\ \text{where } \mathcal{I}_k =_{\mathrm{def}} \mathcal{E}_k^\infty \tag{85}$$

Let $\mathcal{F}$ be the maximal prefix of $\mathcal{E}^\infty_{\mathcal{A}\times\mathcal{P}}$ such that $\mathbf{R}_{P_i}(\mathcal{F}) \sqsubseteq \mathcal{E}_i^\infty$ holds for each $i \in I$. $\mathcal{F}$ is not trivial since it contains at least the minimal co-set of $\mathcal{E}_{\mathcal{A}\times\mathcal{P}}$. Assume that $\mathcal{F}$ is a strict prefix of $\mathcal{E}_{\mathcal{A}\times\mathcal{P}}$. Then, there exists an event $e \in \mathcal{E}_{\mathcal{A}\times\mathcal{P}}$ such that $e \notin \mathcal{F}$ but ${}^\bullet e \subseteq \mathcal{F}$. Event e represents some transition t_i of component $\mathcal{P}_i$, for some $i \in I$. Consequently, $\lambda(e) \subseteq P_i$. Therefore:

- $e \in \mathcal{E}^{\mathcal{F}}_{\mathcal{A}_i \times \mathcal{P}_i}$ holds, whence $\mathbf{R}_{P_i}(e) \in \mathcal{E}^{\infty}_i$ follows, by construction of $\mathcal{E}^{\infty}_i$.
- For $k \in I$ such that $\lambda(e) \cap P_k = \emptyset$, we have $\mathbf{R}_{P_k}(e) = \star$, whence $\mathbf{R}_{P_k}(e) \in \mathcal{E}^{\infty}_k$ trivially holds in this case.
- Pick $j \in I$ such that $\lambda(e) \cap P_j \neq \emptyset$. Since $e \in \mathcal{E}^{\mathcal{F}}_{\mathcal{A}_i \times \mathcal{P}_i}$ holds, then $\mathbf{R}_{P_j}(e) \in \mathbf{R}_{P_j}(\mathcal{E}^{\mathcal{F}}_{\mathcal{A}_i \times \mathcal{P}_i})$. But (38) implies $\mathbf{R}_{P_j}(\mathcal{E}^{\mathcal{F}}_{\mathcal{A}_i \times \mathcal{P}_i}) = \mathbf{R}_{P_j}(\mathcal{E}^{\mathbf{R}_{P_j}(\mathcal{F})}_{\mathcal{A}_i \times \mathcal{P}_i}) \sqsubseteq \mathbf{R}_{P_j}(\mathcal{E}^{\mathcal{E}^{\infty}_j}_{\mathcal{A}_i \times \mathcal{P}_i})$, which implies $\mathbf{R}_{P_j}(e) \in \mathbf{R}_{P_j}(\|_{k \in I} \mathcal{E}^{\mathcal{E}^{\infty}_k}_{\mathcal{A}_k \times \mathcal{P}_k})$, by (62).

Thus, $e \in \mathcal{F}$ holds, a contradiction. This proves (84). ◇

Step 3 *The following invariant holds at the end of each round (1;2;3) of Algorithm 1, applied with scheduling $\sigma_{\mathcal{G}}$: for each $i \in I$:*

$$\mathcal{E}_i \quad \sqsubseteq \quad \mathbf{R}_{P_i}(\mathcal{E}^*_{\mathcal{A} \times \mathcal{P}}). \tag{86}$$

Proof. By (66), (86) holds after the first round (without the need for taking the star closure). Assume it holds after some given round, we shall prove that it also holds after the next round. To this end, we reuse the notations of step 1. By step 1, we know that, $\forall i \in I$,

$$\mathcal{E}_i \quad \sqsubseteq \quad \mathbf{R}_{P_i}(\|_{k \in I} [\mathcal{E}^{\mathcal{I}_k}_{\mathcal{A}_k \times \mathcal{P}_k}]^2), \text{ where } \mathcal{I}_k =_{\text{def}} \mathcal{E}^{-}_k.$$

By induction hypothesis, $\mathcal{E}^{-}_i \sqsubseteq \mathbf{R}_{P_i}(\mathcal{E}^*_{\mathcal{A} \times \mathcal{P}})$. Denoting by $|I|$ the cardinal of I, we have

$$\begin{aligned}
\forall i \in I \ : \ \mathcal{E}_i \quad &\sqsubseteq \quad \mathbf{R}_{P_i}(\|_{k \in I} \left[\mathcal{E}^{\mathbf{R}_{P_i}(\mathcal{E}^*_{\mathcal{A} \times \mathcal{P}})}_{\mathcal{A}_k \times \mathcal{P}_k}\right]^2) && \text{(by (37))} \\
&= \quad \mathbf{R}_{P_i}(\|_{k \in I} \left[\mathcal{E}^{\mathcal{E}^*_{\mathcal{A} \times \mathcal{P}}}_{\mathcal{A}_k \times \mathcal{P}_k}\right]^2) && \text{(by (38))} \\
&= \quad \mathbf{R}_{P_i}([\mathcal{E}^*_{\mathcal{A} \times \mathcal{P}}]^{2 \times |I|}),
\end{aligned}$$

since $\mathcal{E}^{\mathcal{E}^*_{\mathcal{A} \times \mathcal{P}}}_{\mathcal{A}_k \times \mathcal{P}_k} = \mathcal{E}^*_{\mathcal{A} \times \mathcal{P}}$ holds, for each k. Finally, since the trimmed composition is idempotent when applied to star closures, $[\mathcal{E}^*_{\mathcal{A} \times \mathcal{P}}]^{2 \times |I|} = \mathcal{E}^*_{\mathcal{A} \times \mathcal{P}}$ holds, which proves (86). ◇

Step 4 *Setting, for all $i \in I$, $\mathcal{E}^{\max}_i =_{\text{def}} \mathbf{R}_{P_i}(\mathcal{E}^*_{\mathcal{A} \times \mathcal{P}})$ yields a stationary point of Algorithm 1.*

Proof. We only provide an outline, since the detailed arguments are similar to some arguments used in previous steps. First, as seen just before, note that round 1 will not modify $\mathcal{E}^{\max}_i$. Then, just repeat the calculations performed in step 1 and note that powers of $\mathcal{E}^*_{\mathcal{A} \times \mathcal{P}}$ appear, which are equal to $\mathcal{E}^*_{\mathcal{A} \times \mathcal{P}}$. ◇

Steps 1 – 4 together prove Theorem 2.

References

[1] E. Fabre, A. Benveniste, S. Haar, C. Jard. Distributed monitoring of concurrent and asynchronous systems. In *Proc. of 14th Int. Conf. on Concurrency Theory, CONCUR'2003,* R. Amadio and D. Lugiez, Eds., LNCS 2761, 1–26, 2003.

[2] A. Aghasaryan, C. Dousson, E. Fabre, Y. Pencolé, A. Osmani. Modeling Fault Propagation in Telecommunications Networks for Diagnosis Purposes. XVIII World Telecommunications Congress 22-27 September 2002 - Paris, France. Available: http://www.irisa.fr/sigma2/benveniste/pub/topic_distribdiag.html

[3] A. Aghasaryan, C. Jard and J. Thomas, UML Specification of a Generic Model for Fault Diagnosis of Telecommunication Networks, 2004 International Conference on Telecommunications, August 2-7, 2004, Fortalezza, Brasil.

[4] E. Fabre, A. Benveniste, S. Haar, C. Jard and A. Aghasaryan, Algorithms for Distributed Fault Management in Telecommunications, 2004 International Conference on Telecommunications, August 2-7, 2004, Fortalezza, Brasil.

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

[6] A. Benveniste, E. Fabre, C. Jard, and S. Haar. Diagnosis of asynchronous discrete event systems, a net unfolding approach. *IEEE Trans. on Automatic Control,* 48(5), May 2003. Preliminary version available from http://www.irisa.fr/sigma2/benveniste/pub/IEEE_TAC_AsDiag_2003.html

[7] A. Benveniste, S. Haar, and E. Fabre. Markov Nets: probabilistic Models for Distributed and Concurrent Systems. *IEEE Trans. on Automatic Control,* November 2003. Extended version available as IRISA *Report* **1538,** may 2003; available electronically at ftp://ftp.irisa.fr/techreports/2003/PI-1538.ps.gz

[8] R.K. Boel and J.H. van Schuppen. Decentralized Failure Diagnosis for Discrete-Event Systems with Costly Communication between Diagnosers. In *Proc. of 6th Int. Workshop on Discrete Event Systems, WODES'2002.* 175–181, 2002.

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

[10] P. Degano, R. De Nicola, and U. Montanari. On the Consistency of "Truly Concurrent" Operational and Denotational Semantics. *Proc. 3rd Symp. on Logics in Computer Science,* IEEE 1988, pp.133-141.

[11] 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.

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

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

[14] J. Esparza, S. Römer. An unfolding algorithm for synchronous products of transition systems. in proc. of CONCUR'99, LNCS Vol. 1664, Springer Verlag, 1999.

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

[16] E. Fabre. Compositional models of distributed and a synchronous dynamical systems. In *Proc of the 2002 IEEE Conf. on Decision and Control,* 1–6, Dec. 2002, Las Vegas, 2002.

[17] E. Fabre. Monitoring distributed systems with distributed algorithms. In *Proc of the 2002 IEEE Conf. on Decision and Control,* 411–416, Dec. 2002, Las Vegas, 2002.

[18] E. Fabre. Convergence of the turbo algorithm for systems defined by local constraints. IRISA Res. Rep. 1510, 2003.

[19] S. Haar, A. Benveniste, E. Fabre, C. Jard. Partial order diagnosability of discrete event systems using Petri net unfoldings. In *Proceedings of the 42nd Int. IEEE Conference on Decision and Control,* Maui, Dec. 9-12, 2003.

[20] S. Genc and S. Lafortune. Distributed diagnosis of discrete-event systems using Petri nets. In *Proc. of ICATPN 2003,* W.M.P. van der Aalst and E. Best Eds., LNCS 2679, 316–336, 2003.

[21] G. Lamperti and M. Zanella. *Diagnosis of active systems.* Kluwer Academic Publishers, 2003.

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

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

[24] Y. Pencolé, M-O. Cordier, and L. Rozé: A decentralized model-based diagnostic tool for complex systems. *International Journal on Artificial Intelligence Tools,* World Scientific Publishing Company, 11(3), 327–346, 2002.

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

[26] 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.

[27] M. Sampath, R. Sengupta, K. Sinnamohideen, S. Lafortune, and D. Teneketzis. Failure diagnosis using discrete event models. *IEEE Trans. on Systems Technology,* 4(2),105–124, March 1996.

[28] Su, R., Wonham, W.M., Kurien, J., and Koutsoukos, X., "Distributed Diagnosis for Qualitative Systems", 6th International Workshop on Discrete Event Systems (WODES'02), Zaragoza, Spain, pp. 169-174, October 2-4, 2002.

[29] Su, R., "Distributed Diagnosis for Discrete-Event Systems", Ph.D. Thesis, Dept. of Electl. & Cmptr. Engrg., Univ. of Toronto, June 2004.

[30] F. Vaandrager. A simple definition for parallel composition of prime event structures. CWI report CS-R8903, march 1989.

[31] G. Winskel. Event Structure Semantics for CCS and Related Languages. In: Proceedings of ICALP 82, M. Nielsen and M. Schmidt eds., LNCS 140, 561–576, Springer-Verlag, 1982. Extended version as DAIMI Research Report, University of Aarhus, 67 pp., April 1983.

[32] G. Winskel. Categories of Models for Concurrency. Seminar on Concurrency, Carnegie-Mellon Univ. July 1984. Also in *LNCS* Vol. 197, 246–267, 1985.

[33] G. Winskel. Event Structures. In Petri nets: applications and relationships to other models of concurrency, Advances in Petri nets 1986, part II, W. Brauer, W. Reisig and G. Rozenbegr eds., LNCS 255, 325–392, Springer-Verlag, 1987.

Unité de recherche INRIA Rennes
IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)

Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38330 Montbonnot-St-Martin (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399