Chapter 15
Diagnosis with Petri Net Unfoldings

Stefan Haar and Eric Fabre

15.1 Motivation

Large systems or softwares are generally obtained by designing independent modules or functions, and by assembling them through appropriate interfaces to obtain more elaborate functions and modules. The latter can in turn be assembled, up to forming huge systems providing sophisticated services. Consider for instance the various components of a computer, telecommunication networks, plane ticket reservation sofwares for a company, etc. Such systems are not only modular in their design, but often multithreaded, in the sense that many events may occur in parallel.

From a discrete event system perspective, such modular or distributed systems can be modeled in a similar manner, by first designing component models and then assembling them through an adequate composition operation. A first approach to this design principle has been presented in Chapter 5 (see Section 5.5): composition can be defined as the synchronous product of automata. The transitions of each component carry labels, and the product proceeds by synchronizing transitions with identical labels, while all the other transitions remain private. This construction is recalled in Fig. 15.1 on the simple case of three tiny automata. The size of the resulting system is rather surprising, given the simplicity of the three components! And this deserves a detailed study.

One first notices the classical state space explosion phenomenon: the number of states in the global system is the product of the number of states of their components (here $2 \times 2 \times 3 = 12$). So, the number of states augments exponentially fast with the

Stefan Haar
INRIA/LSV, CNRS & ENS de Cachan, 61, avenue du Président Wilson,
94235 CACHAN Cedex, France
e-mail: Stefan.Haar@inria.fr

Eric Fabre
INRIA/IRISA, Campus de Beaulieu, F-35042 Rennes Cedex, France
e-mail: fabre@irisa.fr

C. Seatzu et al. (Eds.): Control of Discrete-Event Systems, LNCIS 433, pp. 301–318.
springerlink.com

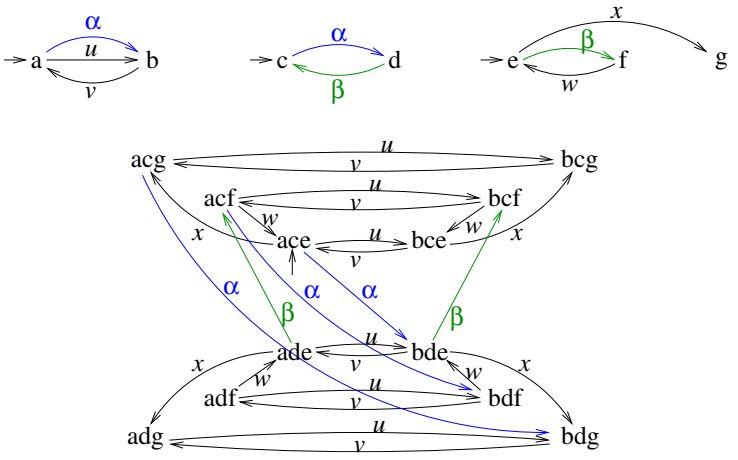

Fig. 15.1 Three components (top) as labeled automata, and their synchronous product (bottom)

number of components. Second, the number of transitions explodes as well: private transitions of a component are cloned many times (see transition (a,u,b) of the first component for example), and creates the so-called concurrency diamonds, representing different possible orderings of transitions (from state ade, one can reach bcf by firing either $u\beta$ or βu).

These phenomena motivate alternate methods of assembling components in order to make explicit the concurrency of transitions. Petri nets are a natural tool toward this objective. Reconsidering the above example under the form of Petri nets, one gets Fig. 15.2. Components are recast into simple PNs with a single token, and their assembling amounts to gluing transitions with identical labels. The explosion both in states and in transitions is now kept under control, and the token semantics of PN makes explicit the fact that several transitions are simultaneously firable. Observe

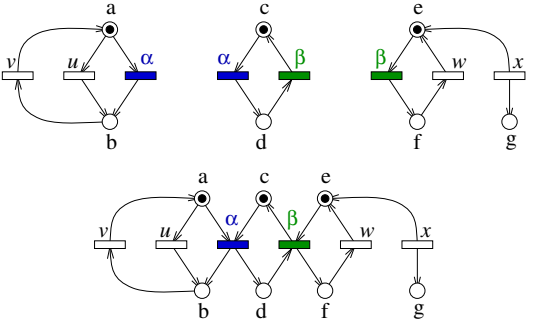

Fig. 15.2 Three components (top) as (safe) Petri nets, and their product (bottom)

in passing that the PNs obtained in this manner are *safe*: each place contains at most one token. Moreover, here the number of tokens is constant and characterizes the number of automata that were assembled.

Did all difficulties of large systems vanish with this simple modeling trick? Not really. When considering runs of (safe) Petri nets, one may still face explosive phenomena. In the usual sequential semantics, trajectories are modeled as sequences of events. So, one recovers the difficulty that different interleaving of concurrent events correspond to different trajectories. For example, trajectories ux and xu both lead from the inital state ace to state bcf, but correspond to two distinct trajectories. While it is clear that the exact ordering in which the private events u and x of the first and third component respectively does not really matter.

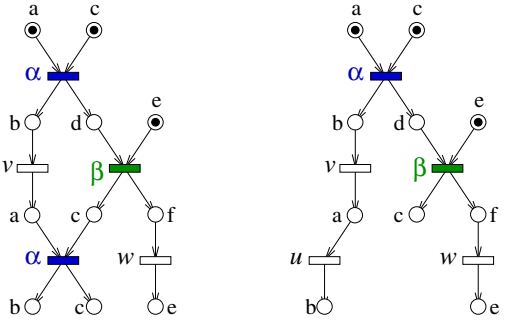

Fig. 15.3 Two trajectories of the PN in Fig. 15.2, as partial orders of events. 'Time' (or precedence, or causality) is oriented from top to bottom

To avoid the explosion in the number of possible runs of a large concurrent system, people soon realized the the parallelism, or the concurrency, also had to be handled in the description of trajectories. The first ideas in this direction came from Mazurkiewicz traces (not treated here), which consist in establishing an equivalence relation between sequences of events that only differ by the ordering of their concurrent events. A related idea is to directly represent runs as *partial orders* of events, rather than sequences. Fig. 15.3 illustrates this idea for the PN of Fig. 15.2. This representation encodes the causality of events, as derived by the use of resources (tokens), but discards any unnecessary timing information. The run on the left, for example, encodes that events v and β occur after the first α, but their order is unspecified. These simple five events partial order stands for five possible sequences, obtained as different interleaving of concurrent events (exercise).

Such *true concurrency semantics,* which handle time as partially ordered, offer many advantages. The first one being to keep under control the explosion due to the intrinsic parallelism of events in large distributed systems. But it also allows one to model that some global knowledge on the system may be unaccessible. It is a common place that the knowledge of global time, or of global state, may be unreachable in distributed asynchronous systems. This idea was already illustrated in

Chapter 5 in the case of distributed observations: if a sensor is placed on each component, the events observed locally by each sensor may be totally ordered, but the exact interleaving of observations collected on different sensors can not (always) be recovered. Therefore, one should also be able to represent distributed observations as partial orders of observations, rather than sequences.

This chapter aims at introducing the main concepts that make possible working with partial orders of event, or true concurrency semantics. It first recalls the notion of (safe) Petri net, and then introduces occurrence nets, a compact data structure to handle sets of runs, where runs are partial orders of events. It then examines how diagnosis can be performed with such semantics, by relating partially ordered observations to possible runs of a Petri net. As for automata, the approach extends to distributed systems. The chapter closes on the notion of diagnosability, which takes a new meaning in this context.

15.2 Asynchronous Diagnosis with Petri Net Unfoldings

Nets and homomorphisms. A *net* is a triple $N = (P,T,F)$, where P and T are disjoint sets of *places* and *transitions*, respectively, and $F \subset (P \times T) \cup (T \times P)$ is the *flow relation.* [1] In figures, places are represented by circles, rectangular boxes represent transitions, and arrows represent F; Fig. 15.4 shows two nets. For node $x \in P \cup T$, call ${}^\bullet x \triangleq \{x' \mid F(x',x)\}$ the *preset*, and $x^\bullet \triangleq \{x' \mid F(x,x')\}$ the *postset* of x. Let $<$ be the transitive closure of F and $\leqslant$ the reflexive closure of $<$; further, let $[x] \triangleq \{x' \mid x' \leqslant x\}$ be the *prime configuration* or *cone* of x, and $[x] \triangleq [x]\backslash\{x\}$ the *pre-cone* of x.

A *net homomorphism*[2] from N to N' is a map $\pi : P \cup T \longmapsto P' \cup T'$ such that *(i)* $\pi(P) \subseteq P'$, $\pi(T) \subseteq T'$, and *(ii)* $\pi_{|{}^\bullet t} : {}^\bullet t \to {}^\bullet\pi(t)$ and $\pi_{|t^\bullet} : t^\bullet \to \pi(t)^\bullet$ induce bijections, for every $t \in T$.

Petri Nets. Let $N = (P,T,F)$ be a finite net. A *marking* of net N is a multi-set $m : P \to \mathbb{N}$. A *Petri net* (PN) is a pair $\mathcal{N} = (N,m_0)$, where $m_0 : P \to \mathbb{N}$ is an *initial* marking. Transition $t \in T$ is *enabled* at marking m, written $m \xrightarrow{t}$, iff ${}^\bullet t \leqslant m$, where we interpret ${}^\bullet t$ as the multi set whose value is 1 on all preplaces of t, and 0 otherwise. If $m \xrightarrow{t}$, then t can *fire*, leading to $m' = (m - {}^\bullet t) + t^\bullet$ (in the multi-set interpretation); write in that case $m \xrightarrow{t} m'$. The set $\mathbf{R}(m_0)$ contains m_0 and the markings of $\mathcal{N}$ *reachable* through the transitive closure $\xrightarrow{+}$ of $\longrightarrow$.

Only safe nets are considered in this article; the net on the left-hand side of Fig. 15.4 is safe. If $m(p) > 0$, we will draw $m(p)$ black *tokens* in the circle representing p. A Petri net $\mathcal{N} = (N,m_0)$ is *safe* if for all $m \in \mathbf{R}(m_0)$ and $p \in P$, $m(p) \in \{0,1\}$.

[1] Only *ordinary* nets are considered here, i.e. with arc weights 0 or 1.

[2] There exist several notions of morphisms for nets and for Petri nets, which are needed e.g. to formalize composition of nets; see [27, 19, 18, 6, 17] and the references therein.

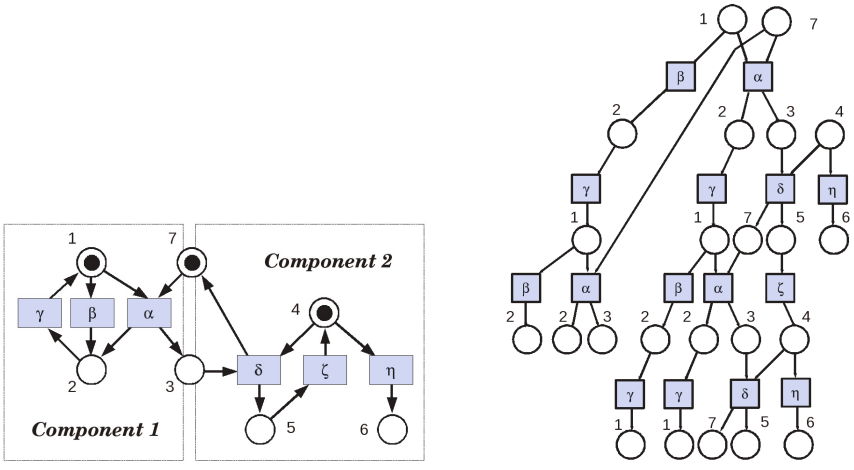

Fig. 15.4 A Petri net (left) and a prefix of its unfolding (right)

Semantics. The behavior of Petri nets can be recorded in either an *interleaved* or a *concurrent* fashion. To formalize this, we introduce *Occurrence Nets* (due to [28]) and *Branching Processes*. Occurrence nets are characterized by a particular structure. In a net $N = (P, T, F)$, let $<_N$ the transitive closure of F, and $\leqslant_N$ the reflexive closure of $<_N$. Further, set $t_1 \#_{im} t_2$ for transitions t_1 and t_2 if and only if $t_1 \neq t_2$ and ${}^\bullet t_1 \cap {}^\bullet t_2 \neq \emptyset$, and define $\# = \#_N$ by

$$a \# b \Leftrightarrow \exists t_a, t_b \in T : [(t_a \#_{im} t_b) \wedge (t_a \leqslant_N a) \wedge (t_b \leqslant_N b)] .$$

Finally, define concurrency relation $\mathbf{co} = \mathbf{co}_{\mathcal{N}}$ by setting, for any nodes $a, b \in P \cup T$,

$$a \,\mathbf{co}\, b \iff \neg(a \leqslant b) \wedge \neg(a \# b) \wedge \neg(b < a) .$$

Definition 15.1. *A net $ON = (B, E, G)$ is an **occurrence net** if and only if it satisfies*

1. *$\leqslant_{ON}$ is a partial order;*
2. *for all $b \in B$, $|{}^\bullet b| \in \{0, 1\}$;*
3. *for all $x \in B \cup E$, the set $[x] = \{y \in B \cup E \mid y \leqslant_{ON} x\}$ is finite;*
4. *no self-conflict,i.e. there is no $x \in B \cup E$ such that $x \#_{ON} x$;*
5. *the set cut_0 of $\leqslant_{ON}$-minimal nodes is contained in B and finite.*

In occurrence nets, the nodes of E are called *events*, and the elements of B are denoted *conditions*. The right-hand side of Fig. 15.4 shows an occurrence net.

Occurrence nets are the mathematical form of the *partial order unfolding semantics* for Petri nets [30]; although more general applications are possible, we will focus here on unfoldings of *safe* Petri nets only.

A *branching process* of safe Petri net $\mathscr{N} = (N, m_0)$ is a pair $\beta = (ON, \pi)$, where $ON = (B, E, G)$ is an occurrence net, and π is a homomorphism from ON to N such that:

1. The restriction of π to cut_0 is a bijection from cut_0 to the set $m_0 \triangleq \{p \in P : m_0(p) = 1\}$, and
2. for every $e_1, e_2 \in E$, if ${}^\bullet e_1 = {}^\bullet e_2$ and $h(e_1) = h(e_2)$ then $e_1 = e_2$.

Branching processes $\beta_1 = (ON_1, \pi_1)$ and $\beta_2 = (ON_2, \pi_2)$ for $\mathscr{N}$ are isomorphic iff there exists a bijective homomorphism $h : ON_1 \to ON_2$ such that $\pi_1 = \pi_2 \circ h$. The unique (up to isomorphism) maximal branching process $\beta_\mathscr{U} = (ON_\mathscr{U}, \pi_\mathscr{U})$ of $\mathscr{N}$ is called the *unfolding* of $\mathscr{N}$. A canonical algorithm (see [30] for details) for constructing the unfolding of $\mathscr{N} = (P, T, F, m_0)$ proceeds as follows: For any branching process $\beta = (ON_\beta, \pi_\beta)$ of $\mathscr{N} = (P, T, F)$, with $ON_\beta = (B_\beta, E_\beta, G_\beta)$, denote as $\mathbf{PE}(\beta) \subseteq T \times \mathrm{Pwrset}(B)$ the set of *possible extensions* of β, i.e. the set of the pairs (t, W) such that

- W is a co-set of ON_β , i.e. for $a, b \in W$, either $a = b$ or a **co** b,
- ${}^\bullet t = \pi_\beta(W)$,
- E_β contains no event e such that $\pi_\beta(e) = t$ and ${}^\bullet e = W$.

Now, let $cut_0 \triangleq m_0 \times \{\emptyset\}$ and initialize $\beta = (cut_0, \emptyset, \emptyset)$; recursively, for given $\beta = (ON_\beta, \pi_\beta)$ with $ON_\beta = (B_\beta, E_\beta, G_\beta)$, compute $\mathbf{PE}(ON_\beta)$ and replace:

$$\begin{aligned} &E_\beta \text{ by } E_\beta \cup \mathbf{PE}(ON_\beta), \\ &B_\beta \text{ by } B_\beta \cup \{(P, e) \mid e \in \mathbf{PE}(ON_\beta), p \in \pi_\beta(e)^\bullet\}, \text{ and} \\ &G_\beta \text{ by } G_\beta \cup \big\{(b, (t, W)) \mid (t, W) \in \mathbf{PE}(ON_\beta), b \in W\big\} \\ &\qquad \cup \big\{(e, (P, e)) \mid e \in \mathbf{PE}(ON_\beta), p \in \pi_\beta(e)^\bullet\big\}. \end{aligned}$$

We will assume that all transitions $t \in T$ have at least one output place, i.e. $t^\bullet$ is not empty.

Occurrence nets give rise to a specific kind of partially ordered set with *conflict* relation that is known in computer science as *event structure*.

Definition 15.2 (compare [28]). A *prime event structure is a tuple* $\mathscr{E} = (E, \leqslant, \#, \lambda)$, *where* $E = \| \mathscr{E} \|$ *is the* support*, or* set of events *of* $\mathscr{E}$, and such that

1. $\leqslant \subseteq E \times E$ is a partial order satisfying the property of *finite causes* i.e. setting $[e] \triangleq \{e' \in E \mid e' \leqslant e\}$, one has for all $e \in E$, $|[e]| < \infty$;
2. $\# \subseteq E \times E$ an irreflexive symmetric *conflict* relation satisfying the property of *conflict heredity*, i.e.

$$\forall\, e, e', e'' \in E : \; e \# e' \;\wedge e' \leqslant e'' \Rightarrow e \# e''. \tag{15.1}$$

Events $e, e' \in E$ are *concurrent*, written e **co** e', iff neither $e \leqslant e'$ nor $e' < e$ nor $e \# e'$ hold. If **co** is the empty relation, we call $\mathscr{E}$ *sequential*.

One notices quickly that occurrence nets form particular cases of event structures. The canonical association of an event structure to an occurrence net ON is by restricting $\leqslant$ and # to the event set E, "forgetting" conditions.

Sequential and Nonsequential behavior. In the net on the left-hand side of Fig. 15.4, the transition sequence $\alpha\delta\gamma\zeta$ is enabled; so is the sequence $\alpha\gamma\delta\zeta$, and it is immaterial to us which of the two sequences actually occurs; both lead to the same final marking (which is identical with the initial marking), and the same actions are performed, only in different order.

We therefore would like to use a unifying way to reason about such collections of firing sequences without having to examine each individual one. One way of capturing the equivalence up to permutation of independent events is developed in the theory of *Mazurkiewicz traces*, see [15, 26]. We will use another relation, which includes also the marking equivalence and which is provided by the concept of *configuration*: a unique partially ordered set that represents in a unique and compact way all enabled *interleavings* of a set of events. Let us formalize this.

Prefixes and Configurations. The *set of causes* or *prime configuration* of $e \in E$ is $[e] \triangleq \{e' \mid e' \leqslant e\}$, as defined above. A *prefix* of $\mathscr{E}$ is any downward closed subset $D \subseteq E$, i.e. such that for every $e \in D$, $[e] \subseteq D$. Prefixes of $\mathscr{E}$ induce, in the obvious way, subevent structures of $\mathscr{E}$ in the sense of the above definition. Denote the set of $\mathscr{E}$'s prefixes as $\mathscr{D}(\mathscr{E})$. Prefix $\mathbf{c} \in \mathscr{D}(\mathscr{E})$ is a *configuration* if and only if it is conflict-free, i.e. if $e \in \mathbf{c}$ and $e\#e'$ imply $e' \notin \mathbf{c}$. Denote as $\mathscr{C}(\mathscr{E})$ the set of $\mathscr{E}$'s configurations. Call any $\subseteq$-maximal element of $\mathscr{C}(\mathscr{E})$ a *run* of $\mathscr{E}$; denote the set of $\mathscr{E}$'s runs as $\Omega(\mathscr{E})$, or simply Ω if no confusion can arise.

In Fig. 15.4, the leftmost branch, with events labeled β, γ, β, is an example of a configuration.

Every *finite* configuration $\mathbf{c}$ terminates at a cut, i.e. a $\subseteq$-maximal co-set, which we denote $cut_{\mathbf{c}}$. The mapping $\mathbf{c} \mapsto cut_{\mathbf{c}}$ is bijective; for each cut cut, the union of the cones of all conditions in cut yield the unique configuration $\mathbf{c}$ such that $cut = \mathbf{c}_{cut}$. Moreover, one has the following two correspondences:

If $\mathbf{c}$ is a configuration of $\mathscr{U}_{\mathscr{N}}$ with $\mathscr{N} = (N, m_0)$, then every occurrence sequence σ obtained as a linear order extension, i.e. an *interleaving*, of the partial order $\leqslant_{\mathbf{c}}$ yields a firable transition sequence of $\mathscr{N}$. Conversely, every firable transition sequence of $\mathscr{N}$ corresponds to a linear order extension of some configuration of $\mathscr{U}_{\mathscr{N}}$. To sum up: the nonsequential executions of $\mathscr{N}$ are in one-to-one correspondence with the configurations of $\mathscr{U}(\mathscr{N})$. We will therefore speak of *$\mathscr{N}$'s configurations* and write $\mathscr{C}(\mathscr{N}) \triangleq \mathscr{C}(\mathscr{U}_{\mathscr{N}})$ and $\Omega(\mathscr{N}) \triangleq \Omega(\mathscr{U}_{\mathscr{N}})$.

• For every reachable marking m of $\mathscr{N}$, there exists at least one cut cut of $\mathscr{U}(\mathscr{N})$ such that $\|\pi(cut)(p) = m(p)\|$ for all $p \in P$, and for the unique configuration $\mathbf{c}$ such that $cut_{\mathbf{c}} = cut$, execution of $\mathbf{c}$ takes m_0 to m; write $m_0 \xrightarrow{\mathbf{c}} m$ for this. Conversely, every finite configuration $\mathbf{c}$ corresponds to a unique reachable marking $m(\mathbf{c})$ given by $m(\mathbf{c}) \triangleq \pi(cut_{\mathbf{c}})$. We call configurations such that $m(\mathbf{c}) = m(\mathbf{c}')$ *marking equivalent*, and denote this by $\mathbf{c} \equiv_m \mathbf{c}'$.

15.3 Asynchronous Diagnosis

The fundamental challenge is the same as in the case of finite state machines: correlation of the observation $\lambda \in \mathbb{A}1^*$ with the system model, and thus extract those runs that are compatible with the observation, i.e. whose image under the observation mask λ agrees with λ. To this end, one lets the observation steer the evolution of the system model, by synchronizing with observable transitions. Formally, this is ensured via a *synchronous* or **Labeled Product:** Let $\mathscr{N}_1 = (P_1, T_1, F_1, m_0^1)$ and $\mathscr{N}_2 = (P_2, T_2, F_2, m_0^2)$ be two Petri nets, with associated labellings $\lambda_1 : T_1 \to \mathbb{A}1$ and $\lambda_2 : T_2 \to \mathbb{A}1$ into the same label alphabet $\mathbb{A}1$. The *λ-synchronized product of $\mathscr{N}_1$ and $\mathscr{N}_2$* is the Petri net $\mathscr{N}_1 \times \mathscr{N}_2 \triangleq (P, T, F, m_0)$, where

1. $P_{\mathscr{Y}} = P_1 \uplus P_2$,
2. for $i \in \{1,2\}$, $T_i^{\varepsilon} \triangleq \{t \in T_i \mid \lambda(t) = \varepsilon\}$,
3. $T_{12} \triangleq t\{t \in T_1 \mid \lambda(t) \neq \varepsilon\}$,
4. $F_{\varepsilon} \triangleq \bigcup_{i=1}^{2}(F_i \cap P_i \times T_i^{\varepsilon}) \cup \bigcup_{i=1}^{2}(F_i \cap T_i^{\varepsilon} \times P_i)$,
5. $F_{12} \triangleq \bigcup_{i=1}^{2}(F_i \cap P_i \times T_{12}) \cup \bigcup_{i=1}^{2}(F_i \cap T_{12} \times P_i)$,
6. $T_{\mathscr{Y}} \triangleq T_1^{\varepsilon} \uplus T_2^{\varepsilon} \uplus T_{12}$ and $F_{\mathscr{Y}} \triangleq F_{\varepsilon} \uplus F_{12}$,
7. $m_0 \triangleq m_0^1 + m_0^2$

Figure 15.5 shows the product of a system model $\mathscr{N}$ and a Petri net model of a partially ordered alarm pattern $\mathscr{A}$. The unfolding of this product is shown on the right-hand side; it exhibits the behavior of $\mathscr{N}$ *steered* by the observation $\mathscr{A}$. Unfolding $\mathscr{U}_{\mathscr{N}\times\mathscr{A}}$ thus contains exactly those behaviors that explain at least a prefix of $\mathscr{A}$; the *full* explanations are highlighted as κ_1 and κ_2 in the figure.

Note that, the product of two 1-safe Petri nets is a 1-safe Petri net. Moreover, in perfect analogy with the synchronous product of finite automata, the semantics of the product projects into the semantics of the two factors; i.e., if $\Pi_{\mathscr{N}_i}$ denotes the operation of erasing, from any prefix of $\mathscr{U}_{\mathscr{N}}$, all arcs and conditions that are not mapped to parts of $\mathscr{N}_i$, one has (see [10]):

$$\forall\, \mathbf{c} \in \mathscr{C}(\mathscr{N}) : \begin{cases} \Pi_{\mathscr{N}_1} \in \mathscr{C}(\mathscr{N}_1) \\ \Pi_{\mathscr{N}_1} \in \mathscr{C}(\mathscr{N}_1) \end{cases} \tag{15.2}$$

Remark: The above construction can be formalized as a *pullback* in appropriate categories.

The advantage is that results such as 15.2 can be derived from much stronger results which imply that the unfolding of the pullback of two safe nets is isomorphic to the pullback of the two unfoldings. The theory necessary to detail these algebraic tools is beyond the scope of this chapter; see [6, 7, 18] . In the asynchronous diagnosis setting, observations are *partially ordered*. The representation of alarm patterns thus generalizes with respect to the linear automaton model above. Figure 15.5 shows , in the center, an alarm pattern represented as an occurrence net $\mathfrak{A}$ (without conflict), with concurrently observed alarm labels; for instance, the β-labeled event on top is concurrent with the α-labeled one to its right. Time flows top-down; we

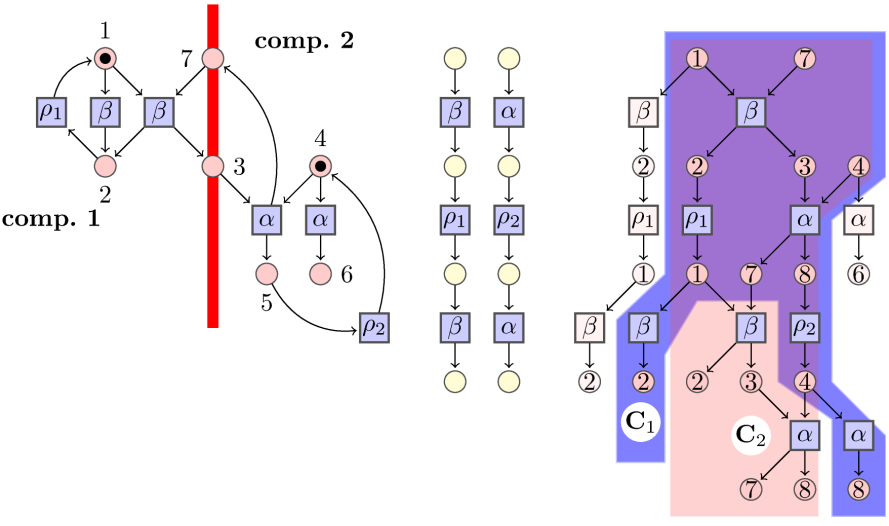

Fig. 15.5 The methodology for unfolding-based Diagnosis. From left to right: A Petri net model $\mathcal{N}$ of a system, taken from Fig. 15.4; the marked places are represented by thick-lined circles. Then, a partially ordered observation (alarm pattern) $\mathcal{A}$ consisting of two disjoint totally ordered chains of event labels $\beta \to \rho 1 \to \beta$ and $\alpha \to \rho 2 \to \alpha$, is represented as a small Petri net (arrows point downward) with added places between successive events. Then, one forms the product $\mathcal{N} \times \mathcal{A}$ by synchronizing transitions that bear the same label. Finally (right), the unfolding $\mathcal{U}_{\mathcal{N}\times\mathcal{A}}$ contains exactly two explanations for $\mathcal{A}$, namely the configurations κ_1 and κ_2. The events not belonging to either of these explanations are shown in gray; they cannot be extended into any explanation of $\mathcal{A}$ and can be *pruned* away

have sometimes omitted arrows. The central step in the diagnosis procedure now consists in computing computing $\mathcal{U}_{\mathcal{N}\times\mathfrak{A}}$, shown on the right hand side.

Remark: In the example, this unfolding is finite, as opposed to the unfolding of the system net $\mathcal{N}$. This is an important feature, which requires a strong *observability* property (compare Section 15.5):

- There must not exist any *cyclic* firing sequence $m_1 \xrightarrow{t_1} m_2 \xrightarrow{t_2} \ldots m_n \xrightarrow{t_n} m_1$ between reachable markings in $\mathcal{N}$ such that $\lambda(t_i) = \varepsilon$ for all $i \in \{1, \ldots, n\}$.

In fact, otherwise $\mathcal{U}_{\mathcal{N}\times\mathfrak{A}}$ will contain at least one infinite branch, since the transitions of any such loop can fire indefinitely, unrestrained by $\mathfrak{A}$. Conversely, if any loop in the behavior of $\mathcal{N}$ must contain at least one observable transition, then it can be performed only a finite number of times since $\mathfrak{A}$ is finite. If this requirement cannot be met, another remedy consists in truncating branches that produce two nested marking-equivalent configuration that are observation-equivalent; such a pair $\kappa \subseteq \kappa'$ with $\kappa' \setminus \kappa \neq \emptyset$ need not be explored further. *Cutoff* criteria like this, have been exploited by [29] and others to ensure all analysis can be carried out on a *complete finite prefix*; 1-safeness of the system model ensures that for any fixed

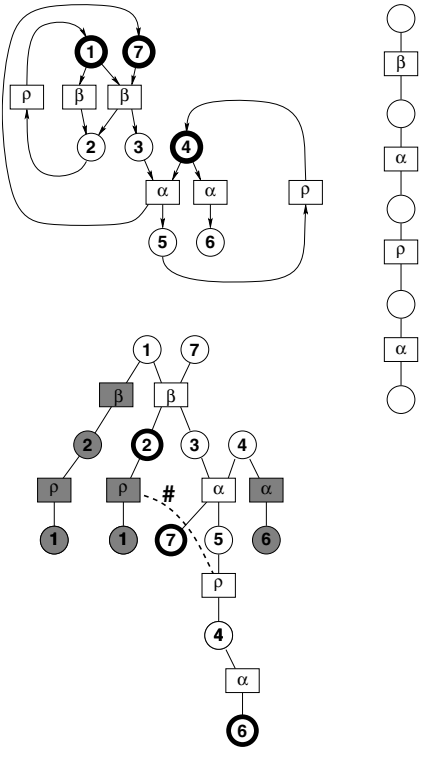

Fig. 15.6 Illustrating the correlation of an alarm pattern $\mathscr{A}$ on the right with a *linearly ordered* alarm pattern (right-hand side)

observation $\mathscr{A}$, a prefix of finite size of $\mathscr{U}_{\mathscr{N}\times\mathscr{A}}$ is sufficient to find essentially all explanations — up to the above surgery to remove unobservable loops — for $\mathscr{A}$.

By relation (15.2), we deduce that the set of configurations that *explain* $\mathfrak{A}$ is obtained as the following prefix of $\mathscr{U}_{\mathfrak{A}\times\mathscr{N}}$:

$$D_{\mathfrak{A}} \triangleq \Pi_{\mathscr{N}}\left(\Pi_{\mathfrak{A}}^{-1}(\mathfrak{A})\right), \tag{15.3}$$

where as above $\Pi_{\mathfrak{A}}$ is the operation of removing all non-$\mathfrak{A}$ parts from $\mathscr{U}_{\mathfrak{A}\times\mathscr{N}}$. Therefore, we have as diagnosis set

$$\mathbf{diag}(\mathfrak{A}) = \{\mathbf{c}\in\mathscr{C}(\mathscr{N}):\ \exists\,\bar{\mathbf{c}}\in\mathscr{C}(\mathscr{U}_{\mathfrak{A}\times\mathscr{N}}):\ \mathbf{c}\subseteq\Pi_{\mathscr{N}}(\bar{\mathbf{c}})\}. \tag{15.4}$$

Notice that $\mathbf{diag}(\mathfrak{A})$ is in general a proper *superset* of $\Omega(\mathscr{U}_{\mathfrak{A}\times\mathscr{N}})$. For the final *diagnosis* task it remains, once $\mathbf{diag}(\mathfrak{A})$ has been computed, to inspect all its configurations for the presence of an occurrence of ϕ.

It should be noted that the computation of (a sufficient prefix of) $\mathscr{U}_{\mathscr{N}\times\mathscr{A}}$ can be "sequentialized" and considerably simplified if $\mathscr{A}$ is *linearly ordered*, i.e. an

observation sequence $\mathscr{A} = a_1 a_2 \ldots a_n \in \mathbb{A}1^*$ (see [10, 29] for more details). In fact, letting $\mathscr{A}_i$ be the ith prefix of $\mathscr{A}$, one obtains $\mathbf{diag}(\mathfrak{A}_{i+1})$ from $\mathbf{diag}(\mathfrak{A}_i)$ in the following way:

1. Compute the extension with new events following the unfolding algorithm,
2. Stop each branch either after after the first occurrence e of an observable transition.
3. Then, remove all such e whose label is *not* a_{i+1}, and ..
4. ... prune away all those events that do not allow to explain $\mathfrak{A}_{i+1}$, i.e. that are in conflict with *all* occurrences of a_{i+1} computed in the previous steps.

In Fig. 15.6, we see that only the configuration shown in white in the unfolding prefix at the bottom is capable of explaining *entirely* the alarm pattern on the right-hand side. In fact, the ρ-labeled event shown in gray is part of an explanation for an observation sequence $\beta\alpha\rho$ (formed by its own prime configuration plus the α-event on the right hand side), but this configuration cannot be extended to explain the subsequent occurrence of α in the pattern. As a result, it is pruned away in the fourth round, since it is in conflict with the second α in white, a conflict inherited from the one indicated in the figure. Similarly, the other events shown in gray are pruned away since they cannot provide explanations for the present alarm observation, nor — a fortiori — for any of its extensions.

15.4 Taking the Methodology Further

The use of unfoldings brings conceptual and technical gains since it allows to abstract away from interleavings of concurrent events. Still, the *computation* of the diagnosis sets can still be hampered by the size of the necessary unfolding prefixes. One notices that the main factor that leads to high *widths* of branching processeses is the number of conflicting branches. Two approaches have launched for improving the data structures used, and both tackle the impact of branching:

1. First, suppose that the supervised system $\mathscr{N}$ is decomposed into subcomponents $\mathscr{N}_1, \ldots, \mathscr{N}_n$ that are supervised separately and locally; the observation $\mathfrak{A}$ is therefor also fragmented into local portions $\mathfrak{A}_1, \ldots, \mathfrak{A}_n$. The *global* diagnosis prefix $D_{\mathfrak{A}}$ is in general too big to be computed directly; by contrast, *local* prefixes $D_{\mathfrak{A}_i}$ obtained by unfolding $\mathscr{N}_i \times \mathfrak{A}_i$ are of more manageable size, and can be computed locally. The number $\mathbf{B}(D_{\mathfrak{A}})$ of branches in $D_{\mathfrak{A}}$ is bounded by $K \triangleq \prod_{i=1}^{n} \mathbf{B}(D_{\mathfrak{A}_i})$, so we can expect an exponential gain in the storage space required. However, care must be taken to compute the right local diagnosis: since not all combinations of local branches match into a global run, $D_{\mathfrak{A}_i}$ is an *over-approximation* of the local diagnosis obtained as the projection $\Pi_i(D_{\mathfrak{A}})$ of the global diagnosis $D_{\mathfrak{A}_i}$ to the ith component. The nontrivial task is thus to orchestrate correctly the distributed computation of the unfolding of an n-component net; see Fig. 15.7 for an illustration of the communication between two "unfolders" in the context of the running example. The work [17] carries out this task

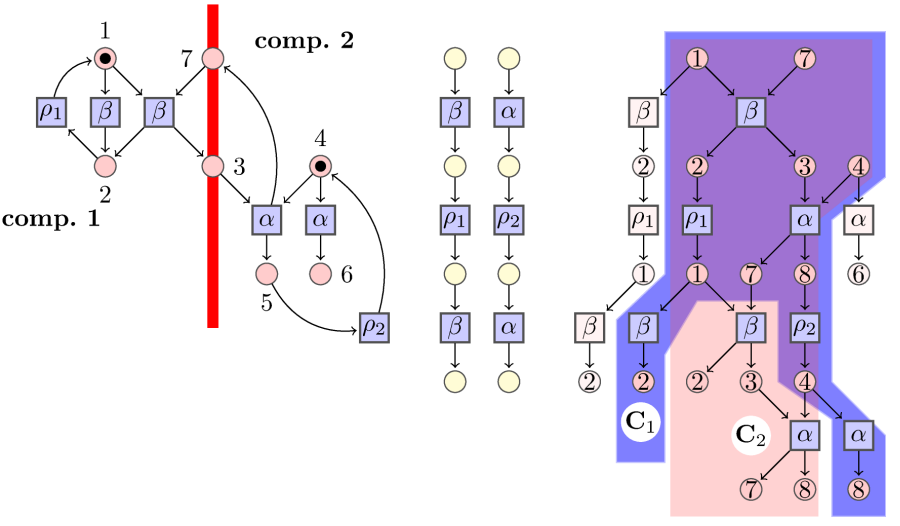

Fig. 15.7 The example from Fig. 15.5, with the distributed computation of the diagnosis net on the right, in two components

in the context of composition *via shared places*, as in Fig. 15.7. Another lead was followed in [6] where the composition was required to form a *pullback* in a suitable Petri net category (see also [18]), which allows to use powerful algebraic tools to characterize the data exchange between two components. While the details of this research are beyond the scope of this presentation, one should note that all results point to the fact that best theoretical (and practical ?) results can be obtained if the interfaces — i.e. the net parts that are shared between two components — should be *concurrency*-free. This is the assumption made, e.g. in [27].

2. In [16], the distributed approach is combined with a methodology for reducing the width of unfoldings by using *trellis* structures: when a state is reached on two or more different branches, the branches are fused on that state, and share the different future extensions. This avoids the width explosion of the stored data structures for long observations, by quotienting the occurrence net structure that forces the inheritance of conflict and thus the separation of branches even if they differ only on an initial segment. The technical challenges of this approach are tackled and solved in [16, 19].

Note also that the approach presented here for the case of *static* system topologies has been extended, in [21] and [7] to graph transformation systems (GTS) for modeling dynamically evolving system topologies; GTS are a proper generalization of Petri nets, yet share with them many properties and techniques, such as partial order unfoldings.

15.5 Asynchronous Diagnosability: Weak versus Strong

Let us turn now to analyzing the power of the unfolding-based diagnosis approach, and ask under which circumstances a fault ϕ is diagnosable. Recall the classical definition of diagnosability given by [31], which we give in the equivalent presentation of [14]. Write $s \sim_\eta s'$ iff $s, s' \in T^*$ are mapped to the same observable word in o^*, and call any sequence s such that ϕ occurs in s a *faulty* sequence, and all other sequences *healthy*. Then:

Definition 15.3 (Strong Diagnosability). *Language $\mathscr{L}$ is* not (strongly) diagnosable *iff there exist sequences $s_N, s_Y \in \mathscr{L}$ such that:*

1. *s_Y is faulty, s_N is healthy, and $s_N \sim_\eta s_Y$;*
2. *moreover, s_Y with the above is arbitrarily long after the first fault, i. e. for every $k \in \mathbb{N}$ there exists a choice of $s_N, s_Y \in \mathscr{L}$ with the above properties and such that the suffix $s_{Y/\phi}$ of s_Y after the first occurrence of fault ϕ in s_Y satisfies $|s_Y| \geq k$.*

Note that, verification of this property is possible *without* using unfoldings, see [12, 13] and this book. The verification of strong diagnosability *with* the use of unfoldings is studied in [29], via the construction of a *verifier net*: the verifier $\mathscr{V}$ is obtained as the product of two isomorphic copies $\mathscr{N}_1$ and $\mathscr{N}_2$ of the diagnosed system $\mathscr{N}$, with synchronization only on *observable* transitions. Therefore, two unsynchronized copies ϕ_1 and ϕ_2 of the unobservable fault event exist; the verification then consists in checking (on a suitable *complete* finite prefix of the unfolding) whether $\mathscr{V}$ allows some infinite run ω on which ϕ_1 occurs and ϕ_2 does not.

Weak Diagnosability. However, it was shown in [24, 25] that for Petri nets, this property is not the only relevant one; a net may violate strong diagnosability and still be *weakly* diagnosable, in the following sense : on any faulty execution, bounded observation is sufficient to detect that on all *maximal concurrent runs* are compatible with the observation, ϕ must have occurred *or is inevitable*, possibly in the future. The presence of these weak and strong properties reflects the choice of *semantics* that produces the event structure model of behavior for the system that is investigated.

What Interleavings do and do not see. Figure 15.8 illustrates that choosing a partial order versus an interleaving semantics has important consequences. Assume that a is the only observable transition. In *sequential semantics*, the net is *not* observable: Consider the run $\omega_s \in \Omega(\mathscr{E}_{seq})$ which consists only of occurrences of v and u; it contains no observable event. Further, when choosing fault $\phi = v$, the net is not diagnosable, since all runs without an occurrence y are observationally indiscernible from the run ω' formed only by occurrences of b and a; this $\sim_\eta$-class therefore contains both faulty and healthy runs.

By contrast, when we consider the partial order semantics of the same net $\mathscr{N}$, the above ω_s is not a run. Its only extension $\bar{\omega}$ into a maximal configuration contains also an infinite number of occurrences of a and b; $\bar{\omega}$ is also the only run with this observation pattern. In fact, all runs $\omega \in \Omega(\mathscr{E}_{\mathscr{U}})$ are fault-definite, i.e. *every* run must

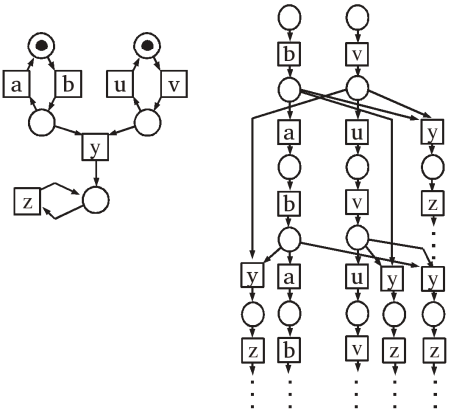

Fig. 15.8 A Petri Net (left) with a prefix of its unfolding (right) to illustrate the difference of strong and weak diagnosability

contain an occurrence of v. The example allows to observe several important phenomena. In fact, it illustrates that decentralized systems with weak synchronization between subsystems may elude diagnosis under the interleaved viewpoint, while being well captured under partial order semantics. In the example, consider now b the fault event, instead of v, and a observable. Then, the new system is neither classically observable nor classically diagnosable. However, removing the loop $u-v$ from the system leaves a classically diagnosable system. In other words, it is the presence of the second loop, running in parallel and without influence on the fault occurrence, that blocks diagnosis of the fault.[3] Thus, the partial order approach actually increases precision for partial observation of highly concurrent systems.

Following [25], we argue that systems like this which allow to derive from the observation that the fault inevitably occurs are *diagnosable* as well, albeit in a weaker sense: *weakly* diagnosable. The formalization given in [25] develops a topological description which we will not follow here.

Defining Weak Diagnosability. Let $\Phi \subseteq E$ be a set of *invisible fault events*; in particular, no event in Φ is observable, i.e. $\lambda(\Phi) \cup \mathrm{Dom}(\eta) = \emptyset$. A configuration $\mathbf{c} \in \mathscr{C}(\mathscr{E})$ is called *faulty* iff $\mathbf{c} \cap \Phi \neq \emptyset$, and *healthy* otherwise. Denote as Ω_F ($\mathscr{C}_F$) the set of faulty runs (configurations), and Ω_{NF} the set of healthy runs. Finally, set, for $\omega \in \Omega$:

$$[[\omega]]_\eta \triangleq \left\{ \omega' \in \Omega \mid \omega \sim_\eta \omega' \right\}.$$

Then *weak diagnosability* for a Petri net means that for all *maximal configurations*, observation equivalence implies fault equivalence:

[3] Thanks to A. Giua who made the first author discover this aspect by a remark in a DISC workshop discussion.

Definition 15.4. *Safe Petri net* $\mathcal{N} = (P, T, F, m_0)$ *is* weakly F-diagnosable *iff for every* $\omega \in \Omega(\mathcal{N})$,

$$\omega \in \Omega_{NF} \Rightarrow [[\omega]]_\eta \subseteq \Omega_{NF}, \tag{15.5}$$

and weakly N-diagnosable *iff for every* $\omega \in \Omega(\mathcal{N})$,

$$\omega \in \Omega_F \Rightarrow [[\omega]]_\eta \subseteq \Omega_F \tag{15.6}$$

It is interesting to note that both notions are equivalent, i.e. $\mathcal{N}$ is weakly F-diagnosable iff it is N-diagnosable. This property — obtained in [25] from the symmetry of pseudometrics — confirms a result in [32] for strong diagnosability, and shows that the symmetry is intrinsic to the concept of diagnosability, rather than a property of the semantic framework.

15.6 Conclusion and Outlook

The use of unfoldings in diagnosis constitutes an important tool in managing large and highly distributed systems, since it allows to avoid the explosion of state space size and the associated huge number of interleaved sequences that would otherwise have to be dealt with. The exploitation of the partial order semantics allows to exhibit concurrency and, dually, causal precedence exactly, thus permitting to focus on essential dependencies in the system. Techniques of correlation via event synchronizations, verifier construction, etc. that had been known in the sequential framework carry over in a natural way to the concurrent case. Also, one notices that systems that are highly distributed in space — both for the execution of their processes *and* their observation — may necessitate a *distributed* multisupervisor approach, to factorize the branching structure of the set of processes and curb the number of such processes to be handled by any *one* diagnoser. This field still leaves room for developments, both concerning diagnosis procedures and verification methods for diagnosability.

Effective verification of *weak* diagnosability is work in progress. The verification of strong diagnosability has been shown to **PSPACE**-complete for the sequential case in [8]. This theoretical bound is a fortiori true for the non-sequential case. It is therefore important now to develop efficient algorithms for verification of *weak* diagnosability.

Another approach to partial observation in concurrent systems, introduced in [22, 23, 24], consists in looking for *inevitable* occurrences that are revealed by observation, regardless of the possible time for occurrence (which may be concurrent with the observation, with no synchronization). Knowledge of such relations in the system allows to raise alarms and start countermeasures as soon as the threat becomes apparent, without waiting for evidence of its actual occurrence.

Finally, let us point out that probabilistic measures for concurrent runs of Petri net unfoldings have been studied in [1, 2, 3, 4, 5, 9, 20, 11]. It remains to develop, in the concurrency setting, probabilistic diagnosis methods on the one hand, and characterizations and verification methods of probabilistic diagnosability on the other, generalizing the existing works for the sequential case.

15.7 Further Reading

The central reference for asynchronous diagnosis with Petri nets is [10]; for the extension to graph grammar models of systems with evolving topology see [7]. Diagnosability for the unfolding-based approaches is treated in the references [22, 23, 24, 25, 29]. Readers that which to better understand occurrence nets and the partial order semantics in general may wish to read [28] and/or compare with Mazurkiewicz Traces [15, 26]. The practical computation of (complete prefixes of) unfoldings are very well described by [30].

References

1. Abbes, S.: The (True) Concurrent Markov Property and Some Applications to Markov Nets. In: Ciardo, G., Darondeau, P. (eds.) ICATPN 2005. LNCS, vol. 3536, pp. 70–89. Springer, Heidelberg (2005)
2. Abbes, S., Benveniste, A.: Branching Cells as Local States for Event Structures and Nets: Probabilistic Applications. In: Sassone, V. (ed.) FOSSACS 2005. LNCS, vol. 3441, pp. 95–109. Springer, Heidelberg (2005)
3. Abbes, S., Benveniste, A.: Probabilistic true-concurrency models: branching cells and distributed probabilities for event structures. Information & Computation 204(2), 231–274 (2006)
4. Abbes, S., Benveniste, A.: Probabilistic true-concurrency models: Markov nets and a law of large numbers. Theoretical Computer Science 390(2-3), 129–170 (2008)
5. Abbes, S., Benveniste, A.: Concurrency, σ-Algebras, and Probabilistic Fairness. In: de Alfaro, L. (ed.) FOSSACS 2009. LNCS, vol. 5504, pp. 380–394. Springer, Heidelberg (2009)
6. Baldan, P., Haar, S., König, B.: Distributed Unfolding of Petri Nets. In: Aceto, L., Ingólfsdóttir, A. (eds.) FOSSACS 2006. LNCS, vol. 3921, pp. 126–141. Springer, Heidelberg (2006)
7. Baldan, P., Chatain, T., Haar, S., König, B.: Unfolding-based diagnosis of systems with an evolving topology. Information and Computation 208(10), 1169–1192 (2010)
8. Bauer, A., Pinchinat, S.: A topological perspective on diagnosis. In: 9th International Workshop on Discrete Event Systems, Gothenburg, Sweden (2008)
9. Benveniste, A., Fabre, E., Haar, S.: Markov nets: Probabilistic models for distributed and concurrent systems. IEEE Transactions on Automatic Control 48(11), 1936–1950 (2003)
10. Benveniste, A., Fabre, E., Haar, S., Jard, C.: Diagnosis of asynchronous discrete event systems: A net unfolding approach. IEEE Transactions on Automatic Control 48(5), 714–727 (2003)

11. Bouillard, A., Haar, S., Rosario, S.: Critical Paths in the Partial Order Unfolding of a Stochastic Petri Net. In: Ouaknine, J., Vaandrager, F.W. (eds.) FORMATS 2009. LNCS, vol. 5813, pp. 43–57. Springer, Heidelberg (2009)
12. Cabasino, M.P., Giua, A., Seatzu, C.: Diagnosability of bounded Petri nets. In: Proc. 48th IEEE Conference on Decision and Control, Shangai, China (2009)
13. Cabasino, M.P., Giua, A., Lafortune, S., Seatzu, C.: Diagnosability analysis of unbounded Petri nets. In: Proc. 48th IEEE Conference on Decision and Control, Shangai, China (2009)
14. Cassandras, C.G., Lafortune, S.: Introduction to Discrete Event Systems, 2nd edn. Springer (2008)
15. Diekert, V., Rozenberg, G. (eds.): The Book of Traces. World Scientific (1995)
16. Fabre, E.: Distributed diagnosis based on trellis processes. In: 44th Conference on Decision and Control, Seville, Spain (2005)
17. Fabre, E., Benveniste, A., Haar, S., Jard, C.: Distributed monitoring of concurrent and asynchronous systems. Discrete Event Dynamic Systems: Theory and Applications 15(1), 33–84 (2005)
18. Fabre, E.: On the construction of pullbacks for safe Petri nets. In: Applications and Theory of Petri Nets and other Models of Concurrency, Turku, Finland (2006)
19. Fabre, E.: Trellis processes: A compact representation for runs of concurrent systems. Discrete Event Dynamic Systems 17, 267–306 (2007)
20. Haar, S.: Probabilistic cluster unfoldings. Fundamenta Informaticae 53(3-4), 281–314 (2002)
21. Haar, S., Benveniste, A., Fabre, A., Jard, C.: Fault diagnosis for distributed asynchronous dynamically reconfigured discrete event systems. In: Proc. 16th IFAC World Congress, Prague, Czech Republic (2005)
22. Haar, S.: Unfold and cover: Qualitative diagnosability for Petri nets. In: Proc. 46th IEEE Conference on Decision and Control, New Orleans, LA, USA (2007)
23. Haar, S.: Qualitative diagnosability of labeled Petri nets revisited. In: Proc. 48th IEEE Conference on Decision and Control and 28th Chinese Control Conference (CDC 2009), Shanghai, China (2009)
24. Haar, S.: Types of asynchronous diagnosability and the *reveals*-relation in occurrence nets. IEEE Transactions on Automatic Control 55(10), 2310–2320 (2010)
25. Haar, S.: What topology tells us about diagnosability in partial order semantics. In: Proc. 10th Workshop on Discrete Event Systems, Berlin (2010)
26. Kummetz, R., Kuske, D.: The topology of Mazurkiewicz Traces. Theoretical Computer Science 305, 237–258 (2003)
27. Madalinski, A., Fabre, E.: Modular Construction of Finite and Complete Prefixes of Petri Net Unfoldings. Fundamenta Informaticae 95(1), 219–244 (2009)
28. Nielsen, M., Plotkin, G., Winskel, G.: Petri nets, event structures and domains (I). Theoretical Computer Science 13, 85–108 (1981)
29. Nouioua, F., Madalinski, A., Dague, P.: Diagnosability verification with Petri net unfoldings. KES Journal 14(2), 49–55 (2010); Long version: Rapport de recherche No. 1516, UMR 8623, CNRS. Université Paris-Sud (March 2009)
30. Römer, S., Esparza, J., Vogler, W.: An improvement of Mcmillan's unfolding algorithm. Formal Methods in System Design 20(3), 285–310 (2002)
31. Sampath, M., Sengupta, R., Lafortune, S., Sinnamohideen, K., Teneketzis, D.: Diagnosability of discrete-event systems. IEEE Transactions on Automatic Control 40(9), 1555–1575 (1995)
32. Wang, Y., Lafortune, S., Yoo, T.-S.: Decentralized diagnosis of discrete event systems using unconditional and conditional decisions. In: Proc. 44th IEEE Conference on Decision and Control, Seville, Spain (2005)