Proceedings of the
44th IEEE Conference on Decision and Control, and
the European Control Conference 2005
Seville, Spain, December 12-15, 2005

Distributed Diagnosis based on Trellis Processes

Eric Fabre

***Abstract*— Distributed or networked systems become rapidly intractable as their size augments. This is due to the combinatorial explosion taking place in the state space, but also in the trajectory space. The latter is reinforced by the amount of concurrency in the system (the fact that several non related events can occur simultaneously). We describe a triple strategy to extend standard monitoring approaches to such systems. We first use a true concurrency semantics to represent trajectories of the system. We then describe a very compact data structure to handle *sets* of trajectories, that we call the *trellis* of the system. This notion generalizes to distributed systems the standard notion of trellis of an automaton, where one dimension is time, and the other dimension encodes possible states at each time. Finally, we show that trellises have nice factorization properties, that allow the design of distributed monitoring approaches. In other words, a problem like state estimation for example, can be solved by parts, component per component, provided interactions between components are properly handled. We illustrate the approach on a distributed diagnosis problem.**

***Index Terms*— distributed system, concurrency, trellis, modular/distributed diagnosis, message passing algorithm.**

I. INTRODUCTION

Large distributed (discrete event) systems require new monitoring paradigms. The major issues are both related to the size of systems (combinatorial explosions on numbers of states and trajectories), and to the distributed nature of systems, as in web programming for example, which makes centralized monitoring approaches unpractical or impossible. "Divide and conquer" seems to be the most natural strategy to deal with such systems. The literature on this recent topic goes into this direction. Some approaches focus on issues related to distributed observations, and propose distributed monitoring methods based on several communicating supervisors [10], [12], [15]. Other approaches combine distributed observations with a modular characterization of system behaviors [11], [14], [18], [20], [24].

The present paper belongs to this second category. We model a distributed system as a set of interconnected components, the observations of which are collected by independent sensors. Two connected components interact through an interface (typically an intermediary automaton), and the global interaction structure can be represented as a graph, where an edge is drawn between two components as soon as they have a common interface. Our strategy to reduce combinatorial explosions is threefold. 1/ We take explicitly into account the parallelism in the system, through a true concurrency semantics on trajectories. This idea is borrowed to theoretical computer science and contrasts with traditional approaches for discrete event systems, that consider sequences of events (see [14], [19], etc.) 2/ We propose a compact data structure to encode sets of trajectories, that we call the trellis of the system. This notion is detailed in [26], and generalizes to distributed concurrent systems the usual notion of trellis of an automaton. And 3/ we use the fact that the trellis of a modular or distributed system factorizes as the product (in an appropriate sense) of trellises of its components. This last property allows to process systems by part, with "belief propagation"-like algorithms, exploiting the graphical structure of interactions in the system.

To show the efficiency of trellises for distributed computations, we consider the distributed diagnosis problem. It amounts to recovering all (hidden) system trajectories that explain observations collected on all sensors. We solve the problem by parts, with an architecture of local supervisors (one per component) that coordinate their work. All computations are based on trellis operations, specifically products, pullbacks and projections. Each supervisor ends up with a local view of the set of global trajectories that explain all observations. The latter can be recovered by gluing together compatible local solutions. We essentially describe supervisor interactions, and skip on the classical recursions in time inside each supervisor. These two recursions, in "space" and in "time" can be interleaved naturally and their result do not depend on the ordering of operations. Due to space limitations and to the technicality of the topic, we do not provide a full theoretical treatment but rather focus on the essential concepts, that we illustrate on a small example.

II. DISTRIBUTED SYSTEMS

There exist several related formalisms to describe concurrent systems. We adopt the formalism of Petri nets, that we specialize into multi-clock nets to make them suited to the notion of trellis. The latter take the form of a synchronous product of automata. We then describe distributed systems as a collection of components interacting through interfaces.

A. Concurrent system

We consider concurrent systems modeled as safe Petri nets (PN). We adopt the notation $\mathcal{N} = (P, T, \to, P^0)$ for nets, where P, T are respectively place and transition sets, and $\to$ is the flow relation relating places to transitions and transitions to places. $P^0 \subseteq P$ is the subset of initially marked places in the net (*i.e.* places containing a token). We adopt the standard notations ${}^\bullet t$ and $t^\bullet$ for pre- and post-sets of t,

This work was partly supported by the French Ministry of Research, through the RNRT project "SWAN" (Self-aWare network mANagement)

E. Fabre is with the Distribcom group, IRISA/INRIA, Campus de Beaulieu, 35042 Rennes cedex, France. eric.fabre@irisa.fr

as well as the standard dynamics for a PN (see for ex. [2] for details). A net $\mathcal{N}$ is *safe* when each place contains at most one token in any reachable *marking*. So markings can be identified with subsets of places.

We limit ourselves to a particular sub-class of safe nets, that we call *multi-clock nets* (MCN). A MCN $\mathcal{N} = (P, T, \rightarrow, P^0, \nu)$ is provided with a partition function on places $\nu : P \rightarrow P^0$, and satisfies $\forall t \in T$, ν is injective both on $^\bullet t$ and on $t^\bullet$, and $\nu(^\bullet t) = \nu(t^\bullet)$. In a MCN, the number of tokens is constant, and in any reachable marking $P' \subseteq P$, one has $\nu : P' \rightarrow P^0$ is bijective. In other words, $\forall p \in P$, the restriction $\mathcal{N}_{|\bar{p}}$ of net $\mathcal{N}$ to places $\bar{p} \triangleq \nu^{-1}(\nu(p))$ (the *class* of p) is a sequential machine, *i.e.* an ordinary automaton. Fig. 1 gives an example of a MCN with 3 classes. It is always possible to reshape a safe net into a MCN with essentially the same behavior, for example by introducing complementary places. So the limitation to MCNs is not a strong assumption.

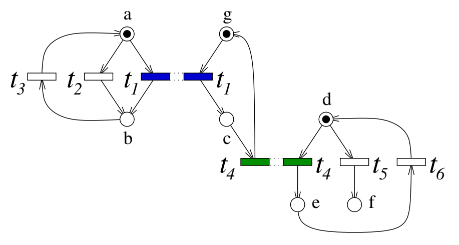

Fig. 1. *A multi-clock net, with place classes $\{a, b\}, \{g, c\}, \{d, e, f\}$. Sequential components $\mathcal{N}_{|\bar{a}}, \mathcal{N}_{|\bar{g}}, \mathcal{N}_{|\bar{d}}$ are evidenced on the picture. A transition like t_1 synchronizes the first two components.*

B. Building systems from components

A *labeled net* $\mathcal{N} = (P, T, \rightarrow, P^0, \nu, \lambda, \Lambda)$ is a net provided with a label set Λ and a labeling function $\lambda : T \rightarrow \Lambda$ on transitions. There exist several ways to combine two nets $\mathcal{N}_1, \mathcal{N}_2$ into a larger one. A well known one [3], [4] is the synchronous product of labeled nets. This product, denoted $\mathcal{N}_1 \times_N \mathcal{N}_2$, forms the disjoint union of places, then synchronizes (=glues) transitions t_1, t_2 in T_1, T_2 (resp.) that carry identical labels, and finally preserves all transitions $t_i \in T_i$ that carry a private label (*i.e.* $\lambda_i(t_i) \in \Lambda_i \setminus \Lambda_j$, $i \neq j$). Fig. 1 illustrates this idea: the left and central components synchronize on transitions labeled t_1 (we use transition names as label sets there). This generalizes: a MCN $\mathcal{N}$ can be expressed as the synchronous product of the automata $\mathcal{N}_{|\bar{p}}$, $p \in P^0$, still using transition names as labels.

We propose here another way of specifying interactions between components, by means of a shared sub-net. This definition suits better the computations we develop in the sequel, and suggests the important notion of interface.

We first briefly recall the notion of *morphism* between nets, as defined in [4]. A morphism $\phi : \mathcal{N}_1 \rightarrow \mathcal{N}_2$ is a pair (ϕ^P, ϕ^T), with ϕ^P a relation on places, and ϕ^T a partial function on transitions[1]. ϕ preserves the initial marking as follows: $P_2^0 = \phi(P_1^0)$ and $\forall p_2 \in P_2^0,\ \exists! p_1 \in P_1^0 : p_1 \phi\, p_2$. If ϕ is defined on $p_1 \in P_1$, then it is also defined on both $^\bullet p_1$ and $p_1^\bullet$. Finally, ϕ is required to preserve the neighborhood of each transition: $t_2 = \phi(t_1)$ implies that restrictions $\phi : {^\bullet t_1} \rightarrow {^\bullet t_2}$ and $\phi : t_1^\bullet \rightarrow t_2^\bullet$ are both bijective. In the special case of multi-clock nets, we further require that ϕ preserve partitions of places: $\forall (p_1, p_2) \in P_1 \times P_2,\ p_1 \phi\, p_2 \Rightarrow \nu_1(p_1)\, \phi\, \nu_2(p_2)$. MCNs provided with such morphisms define the category[2] $Nets$ [1].

[1]For simplicity, we write ϕ instead of ϕ^P or ϕ^T.

Let $\mathcal{N}_0, \mathcal{N}_1, \mathcal{N}_2$ be labeled nets, and $\phi_i : \mathcal{N}_i \rightarrow \mathcal{N}_0$, $i = 1, 2$, be net morphisms. We further assume that these ϕ_i are partial functions on places (we forbid place duplications). The *pullback* of this triple, that we denote by $\mathcal{N} = \mathcal{N}_1 \overset{\mathcal{N}_0}{\wedge}_N \mathcal{N}_2$ (or $\mathcal{N} = \mathcal{N}_1 \wedge_N \mathcal{N}_2$ for short), is defined on places by

$$\begin{aligned} P \;=\; & \{(p_1, \star) : p_1 \in P_1,\ p_1 \notin Dom(\phi_1)\} \\ \cup\; & \{(\star, p_2) : p_2 \in P_2,\ p_2 \notin Dom(\phi_2)\} \\ \cup\; & \{(p_1, p_2) \in P_1 \times P_2 : \phi_1(p_1) = \phi_2(p_2)\} \quad (1) \end{aligned}$$

and on transitions by

$$\begin{aligned} T \;=\; & \{(t_1, \star) : t_1 \in T_1,\ t_1 \notin Dom(\phi_1), \lambda_1(t_1) \in \Lambda_1 \setminus \Lambda_2\} \\ \cup\; & \{(\star, t_2) : t_2 \in T_2,\ t_2 \notin Dom(\phi_2), \lambda_2(t_2) \in \Lambda_2 \setminus \Lambda_1\} \\ \cup\; & \{(t_1, t_2) \in T_1 \times T_2 : t_i \notin Dom(\phi_i), \lambda_1(t_1) = \lambda_2(t_2)\} \\ \cup\; & \{(t_1, t_2) \in T_1 \times T_2 : \phi_1(t_1) = \phi_2(t_2)\} \quad (2) \end{aligned}$$

The flow relation follows accordingly, as well as the definition of initial places. We denote by $\psi_i : \mathcal{N} \rightarrow \mathcal{N}_i$ the canonical morphism that maps elements of $\mathcal{N}$ to the corresponding element in $\mathcal{N}_i$. By construction, one has $\phi_1 \circ \psi_1 = \phi_2 \circ \psi_2$.

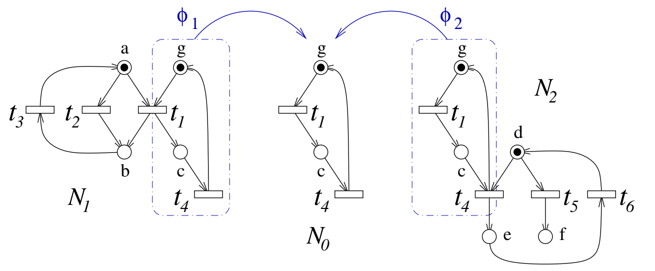

Fig. 2. *Nets $\mathcal{N}_1, \mathcal{N}_2$ and their interface $\mathcal{N}_0$, with associated morphisms $\phi_i : \mathcal{N}_i \rightarrow \mathcal{N}_0$. The net in Fig. 1 is the pullback of this triple, when transition names are used as synchronization labels.*

C. Notion of interface between systems

The pullback actually generalizes the product to the case of components sharing some places and transitions: when $\mathcal{N}_0$ is the empty net, $\mathcal{N}_1 \wedge_N^{\mathcal{N}_0} \mathcal{N}_2 = \mathcal{N}_2 \times_N \mathcal{N}_2$. Let us focus on a particular case: we say that $\mathcal{N}_0$ is an *interface* between $\mathcal{N}_1$ and $\mathcal{N}_2$ iff transitions of $\mathcal{N}_i$ carrying a shared label $(\Lambda_1 \cap \Lambda_2)$ are all in the definition domain of ϕ_i. In other words, there is no interaction between $\mathcal{N}_1$ and $\mathcal{N}_2$ "outside" the domains of the ϕ_i, which is characterized in (2) by a vanishing third line. In other words, the two components only interact by shared places and transitions.

[2]For the reader not familiar with these ideas, a category is a collection of objects, together with morphisms between these objects. This theory allows to introduce some algebra in object relations, combinations, transformations, etc.

Up to isomorphism, the pullback operation is both associative and commutative. We say that $\mathcal{N}$ is a *distributed system* when it can be expressed as the pullback of several components where the intermediary nets are interfaces. For simplicity in this paper, we stick to the simple case of two components and one interface, as depicted in Fig. 2.

III. Trajectory sets of distributed systems

A. True concurrency semantics (TCS)

In the TCS, runs of a concurrent system are represented as particular nets, called *configurations*. A configuration $\kappa = (C, E, \rightarrow, C^0, \nu)$ is a net such that the flow relation $\rightarrow$ induces a partial order on nodes $C \cup E$, which minimal nodes are exactly C^0, and such that each place c in C has at most one cause, *i.e.* $|{}^\bullet c| \leq 1$, and no place in C is branching, *i.e.* $|c^\bullet| \leq 1$. Places are usually called *conditions*, and transitions *events*, whence the change in notation. The pair (κ, ϕ) is a run of net $\mathcal{N}$ in the TCS iff κ is a configuration and morphism $\phi : \kappa \rightarrow \mathcal{N}$ a *folding*, *i.e.* a total function on κ such that $\phi : C^0 \rightarrow P^0$ is bijective. Fig. 3 shows two configurations that represent runs of the net $\mathcal{N}$ in Fig. 1. By definition of ϕ, a direct causality relation in a configuration κ indicates the presence of a shared place in $\mathcal{N}$. By contrast, two events (or nodes) are *concurrent* in κ iff they are not causally related (for ex. the second firing of t_1 and the firing of t_6 in the LHS configuration of Fig. 3). Consider any linear extension of the partial order $\rightarrow$ in κ, restricted to events E. Its image by ϕ yields a firable sequence of transitions of $\mathcal{N}$. Therefore κ corresponds to an equivalence class of runs of $\mathcal{N}$, considered as firable sequences.

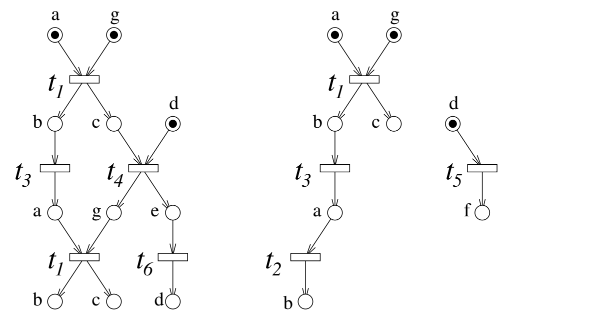

Fig. 3. *Two configurations κ_1 and κ_2 representing runs of the net in Fig. 1. The foldings $\phi_i : \kappa_i \rightarrow \mathcal{N}$ are represented by place and transition names of $\mathcal{N}$ next to conditions and events of κ_i.*

Given net $\mathcal{N}$, there exist several ways to represent trajectory *sets* of $\mathcal{N}$ in a compact manner. Branching processes and unfoldings have been used for long [8], [5], [6], [7], [21]. In this paper, we rather advocate the use of *trellis nets*, that offer the same algebraic properties than unfoldings, while being more compact. We refer the reader to [26] for a comparison of unfoldings and trellises.

We define a *pre-trellis net* $\mathcal{T} = (C, E, \rightarrow, C^0, \nu)$ as a MCN where C^0 is the set of places which have no predecessor, and where each restriction $\mathcal{T}_{|\bar{c}},\ c \in C^0$, is a partial order of nodes. Notice that $\mathcal{T}$ may not be itself a partial order of nodes, but no circuit of $\mathcal{T}$ can take part to a run. In other words, any run of $\mathcal{T}$ identifies a sub-net of $\mathcal{T}$ which is a partial order. To be more precise, let us define a *configuration* κ of $\mathcal{T}$ as a sub-net of $\mathcal{T}$ satisfying:

1) $C^0 \subseteq \kappa$,
2) $\forall e \in E \cap \kappa,\ {}^\bullet e \subseteq \kappa$ and $e^\bullet \subseteq \kappa$: each event has all its causes and consequences,
3) $\forall c \in C \cap \kappa,\ |{}^\bullet c \cap \kappa| = 1$ or $c \in C^0$: each condition is either minimal or has one of its possible causes,
4) $\forall c \in C \cap \kappa,\ |c^\bullet \cap \kappa| \leq 1$: each condition triggers at most one event,
5) the restriction of $\mathcal{T}$ to nodes of κ is a partial order.

Then runs of $\mathcal{T}$ in the TCS coincide with the configurations of $\mathcal{T}$. Given a configuration κ, let us define the *height* of condition c in κ as the number of conditions "below c" in $\kappa_{|\bar{c}}$:

$$H_\kappa(c) \quad = \quad |\{c' \in C \,:\, \nu(c') = \nu(c),\ c' \rightarrow^* c\}| \qquad (3)$$

A pre-trellis net $\mathcal{T}$ is a *trellis* iff every event e of $\mathcal{T}$ belongs at least to one configuration (*i.e.* is reachable), and the height of any condition c doesn't depend on the configuration κ containing c in $\mathcal{T}$ (we say that $\mathcal{T}$ is "correctly folded").

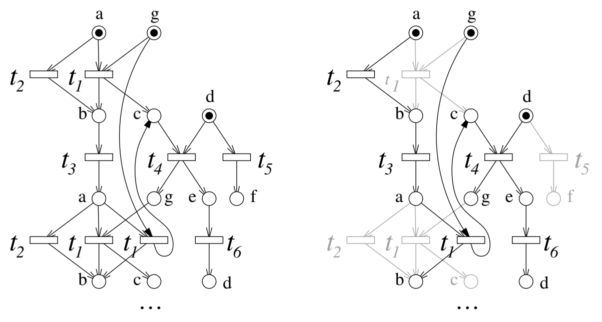

Fig. 4. *Left: beginning of the trellis (or time-unfolding) of the net in Fig. 1. Right: a configuration in this trellis (non selected nodes are dimmed).*

Trellis nets can be used to represent sets of runs of a given net $\mathcal{N}$ in the TCS. The trellis net $\mathcal{T}$ associated to folding $\phi : \mathcal{T} \rightarrow \mathcal{N}$ jointly form a *trellis process* (TP) of $\mathcal{N}$ iff they satisfy the double parsimony criterion:

1) $\forall e, e' \in E,\ [{}^\bullet e = {}^\bullet e',\ \phi(e) = \phi(e')] \ \Rightarrow\ e = e'$,
2) $\forall c, c' \in C,\ [H(c) = H(c'),\ \phi(c) = \phi(c')] \ \Rightarrow\ c = c'$.

In a TP, every configuration κ, associated to $\phi_{|\kappa}$, yields a run of $\mathcal{N}$. Conversely, a run of $\mathcal{N}$ is represented at most once in $\mathcal{T}$: there doesn't exist isomorphic runs of $\mathcal{N}$ in $\mathcal{T}$, thanks to the two parsimony criteria above. There exists a unique maximal trellis process of $\mathcal{N}$ containing all runs of $\mathcal{N}$. We call it *the trellis* of $\mathcal{N}$, or its *time unfolding*, and denote by $(\mathcal{U}^t_{\mathcal{N}}, f^t_{\mathcal{N}})$ (see Fig. 4). We refer the reader to [26] for proofs.

B. Factorization of trellises

The time unfolding operation $\mathcal{U}^t$ defines a *functor* between the category $Nets$ and the sub-category Tr of trellis nets. This essentially means that 1/ $\mathcal{U}^t$ can be extended to morphisms $\phi : \mathcal{N} \rightarrow \mathcal{N}'$, which yields $\mathcal{U}^t(\phi) : \mathcal{U}^t(\mathcal{N}) \rightarrow \mathcal{U}^t(\mathcal{N}')$, and 2/ $\mathcal{U}^t$ commutes with morphism composition:

$\mathcal{U}^t(\phi \circ \phi') = \mathcal{U}^t(\phi) \circ \mathcal{U}^t(\phi')$. Moreover, the functor $\mathcal{U}^t$ is the right adjoint of the inclusion functor from Tr to $Nets$. This abstract result has a nice consequence: we know that right adjoints preserve limits (in the categorical sense), and pullbacks, like products, are a special form of limit [1].

Translating this into practice, if $\mathcal{N} = \mathcal{N}_1 \overset{\mathcal{N}_0}{\wedge_N} \mathcal{N}_2$, then

$$\mathcal{U}^t(\mathcal{N}) \;=\; \mathcal{U}^t(\mathcal{N}_1) \overset{\mathcal{U}^t(\mathcal{N}_0)}{\wedge_T} \mathcal{U}^t(\mathcal{N}_2) \qquad (4)$$

where morphisms $\phi_i : \mathcal{N}_i \to \mathcal{N}_0$ and $\psi_i : \mathcal{N} \to \mathcal{N}_i$ are turned into $\mathcal{U}^t(\phi_i)$ and $\mathcal{U}^t(\psi_i)$. The index T in operator $\wedge_T$ indicates that the pullback must be performed in the sub-category Tr. Using again the co-reflection of Tr into $Nets$, $\wedge_T$ can defined by

$$\mathcal{T}_1 \overset{\mathcal{T}_0}{\wedge_T} \mathcal{T}_2 \;\simeq\; \mathcal{U}^t(\,\mathcal{T}_1 \overset{\mathcal{T}_0}{\wedge_N} \mathcal{T}_2\,) \qquad (5)$$

where $\simeq$ means "isomorphic to." There exists a recursive procedure to compute $\mathcal{U}^t(\mathcal{N})$ from a net $\mathcal{N}$. Coupled to the definition of $\wedge_N$ given section II-C, this yields by (5) an effective procedure to compute the pullback of trellis nets.

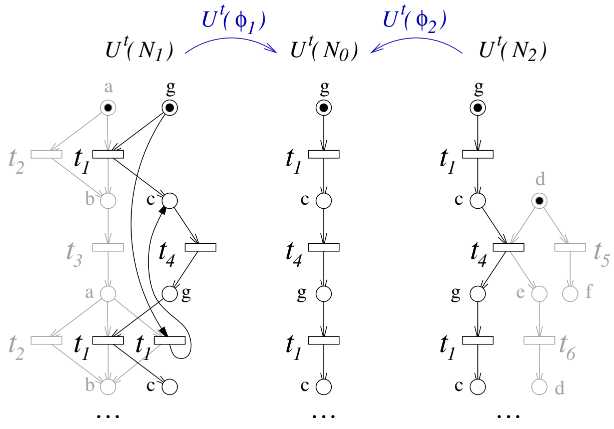

Fig. 5. *$\mathcal{U}^t$ applied to Fig. 2. The pullback of this triple, in the category of trellis nets, yields the trellis of Fig. 4. In each $\mathcal{U}^t(\mathcal{N}_i)$, the black sub-net corresponds to $Dom(\mathcal{U}^t(\phi_i))$.*

Important remark: a pullback operation, like a product, generally yields a net with a much larger structure than the original components, because *pairs* of transitions/events are formed. This would appear clearly on the example if places e and f were identical in Fig. 1 (we leave this instructive verification to the reader). Therefore, computations based on the trellis of a net $\mathcal{N}$ should rather be based on the factorized form given by (4). This is what we propose in section V.

IV. Projection of trellises

Let $\mathcal{N}, \mathcal{M}$ be two nets, related by morphism $\psi : \mathcal{N} \to \mathcal{M}$, where ψ is for example the restriction of $\mathcal{N}$ to a sub-net (notice that a restriction selects entire sequential components of $\mathcal{N}$, thanks to properties of MCN morphisms). Let $(\mathcal{T}, \phi)$ be a TP of $\mathcal{N}$, one has $\psi \circ \phi : \mathcal{T} \to \mathcal{M}$, and applying $\mathcal{U}^t$ to this morphism yields $\mathcal{U}^t(\psi \circ \phi) : \mathcal{T} \to \mathcal{U}^t(\mathcal{M})$ because $\mathcal{T} \simeq \mathcal{U}^t(\mathcal{T})$. We define the projection of $\mathcal{T}$ on $\mathcal{M}$ as

$$\Pi_{\mathcal{M}}(\mathcal{T}) \;=\; \mathcal{U}^t(\phi \circ \psi)(\mathcal{T}) \;=\; \mathcal{U}^t(\phi) \circ \mathcal{U}^t(\psi)(\mathcal{T}) \qquad (6)$$

$\mathcal{U}^t(\psi)$ is an injective map that makes $\mathcal{T}$ a prefix of $\mathcal{U}^t_{\mathcal{N}}$, and $\mathcal{U}^t(\phi)$ thus maps configurations of $\mathcal{T}$ to $\mathcal{U}^t_{\mathcal{M}}$. So $\Pi_{\mathcal{M}}(\mathcal{T})$ is a trellis process of $\mathcal{M}$. In the simple case where $\mathcal{M}$ is a sub-net of $\mathcal{N}$, $\Pi_{\mathcal{M}}(\mathcal{T})$ can be obtained by 1/ restricting $\mathcal{T}$ to conditions corresponding to places selected in $\mathcal{M}$, followed by 2/ a *trimming* operation in order to satisfy the parsimony requirements on the result (Fig. 6).

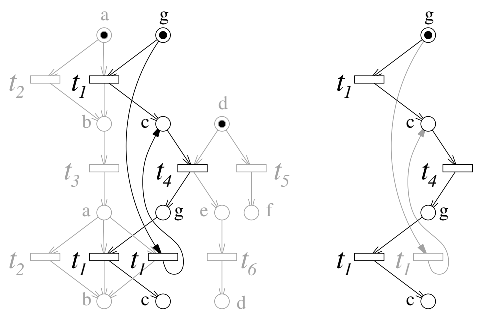

Fig. 6. *Projection of the TP of $\mathcal{N}$ in Fig. 4 on the net $\mathcal{N}_0$ of Fig. 2. First, nodes not related to $\mathcal{N}_0$ are removed (dimmed nodes on the left TP). Then the result is trimmed: redundant copies of the same transition firing are merged (right).*

A projection is said to be *non misleading* iff every configuration κ' in $\Pi_{\mathcal{M}}(\mathcal{T})$ is the image of a configuration κ and no causality relation is lost (on events of κ'). This is not always the case because a morphism generally erases causalities and conflicts, thus creating the appearance of concurrency on events that can't appear in the same configuration of $\mathcal{T}$. A special case deserves attention: If $\mathcal{M}$ is a sequential machine, all projections on $\mathcal{M}$ are non misleading (Fig. 6).

Consider system $\mathcal{N} = \mathcal{N}_1 \wedge_N \mathcal{N}_2$, and trellis processes $(\mathcal{T}_i, \phi_i)$ of the $\mathcal{N}_i$, $i = 1, 2$. It can be proved [26] that any trellis process $(\mathcal{T}, \phi)$ of a net $\mathcal{N}$ is the union, in the sense of trellis processes, of all its configurations (κ, ϕ_κ). In the case of $\mathcal{T} = \mathcal{T}_1 \wedge_T \mathcal{T}_2$, $\mathcal{T}$ is thus the union of configurations$\kappa_1 \wedge_T \kappa_2$, where (κ_i, ϕ_i) is a configuration of $\mathcal{T}_i$. Looking carefully at the definition of $\wedge_T$, one sees that not all nodes of a configuration κ_i will remain in the pullback $\kappa_1 \wedge_T \kappa_2$. The projection aims at identifying these "useful nodes." Choosing $\mathcal{M} = \mathcal{N}_i$ reveals that $\kappa_i' = \Pi_{\mathcal{N}_i}(\kappa_1 \wedge_T \kappa_2)$ is generally a strict prefix of κ_i. Moreover, one obviously has $\kappa_1 \wedge_T \kappa_2 = \kappa_1' \wedge_T \kappa_2'$. Extended to all pairs of configurations, this yields the *minimal factorization* of $\mathcal{T}$:

$$\mathcal{T}_1 \wedge_T \mathcal{T}_2 \;=\; \Pi_{\mathcal{N}_1}(\mathcal{T}_1 \wedge_T \mathcal{T}_2) \wedge_T \Pi_{\mathcal{N}_2}(\mathcal{T}_1 \wedge_T \mathcal{T}_2) \qquad (7)$$

where $\Pi_{\mathcal{N}_i}(\mathcal{T}_1 \wedge_T \mathcal{T}_2)$ selects nodes of $\mathcal{T}_i$ that actually take part to the pullback.

An interesting property arises in the case of a distributed system, *i.e.* when $\mathcal{N}_1$ and $\mathcal{N}_2$ are related by an interface (section II-C). The following result forms the basis of distributed/modular computations.

Theorem 1: Let $\mathcal{N} = \mathcal{N}_1 \overset{\mathcal{N}_0}{\wedge_N} \mathcal{N}_2$, where $\mathcal{N}_0$ is an interface between $\mathcal{N}_1$ and $\mathcal{N}_2$, and assume projections on $\mathcal{N}_0$ are non misleading (for example, $\mathcal{N}_0$ is a sequential machine). Let the $\mathcal{T}_i$ be TPs of the $\mathcal{N}_i$, then

$$\Pi_{\mathcal{N}_1}(\mathcal{T}_1 \wedge_T \mathcal{T}_2) \;=\; \mathcal{T}_1 \wedge_T \Pi_{\mathcal{N}_0}(\mathcal{T}_2) \qquad (8)$$

and similarly for the projection on $\mathcal{N}_2$.

We only sketch the proof. It is enough to show that $\Pi_{\mathcal{N}_1}(\kappa_1 \wedge_T \kappa_2) = \kappa_1 \wedge_T \Pi_{\mathcal{N}_0}(\kappa_2)$ because $\Pi_{\mathcal{N}_1}(\mathcal{T}_1 \wedge_T \mathcal{T}_2)$ is the union of its configurations. We then use the recursive procedure that computes pullbacks. We first show that $\Pi_{\mathcal{N}_0}(\kappa_1 \wedge_T \kappa_2) = \Pi_{\mathcal{N}_0}(\kappa_1) \wedge_T \Pi_{\mathcal{N}_0}(\kappa_2)$, and then obtain the result by noticing that κ_1 and κ_2 have no interaction/synchronization outside their projections on $\mathcal{N}_0$. Details can be found in [22] (theorem 2). □

Remark. This result generalizes to any type of interface: to deal with the issue of misleading projections, one has to introduce the notion of *augmented trellis* in order to keep track of causalities and conflicts due to neighboring components. These ideas will be developed in a forthcoming paper.

V. Distributed diagnosis

The diagnosis problem, as stated in [21], [24] can be summarized as follows: A concurrent system $\mathcal{N}$ produces a run κ that is partially observed through labels produced by its transitions, that we call *alarms* to avoid confusions. The objective is to recover all runs of $\mathcal{N}$ explaining the observed pattern of alarms.

Here, we significantly complexify the picture. We first assume that $\mathcal{N}$ is a distributed system $\mathcal{N} = \mathcal{N}_1 \wedge_N \mathcal{N}_2$, where the interface $\mathcal{N}_0$ between the two components $\mathcal{N}_1$ and $\mathcal{N}_2$ is a sequential machine[3]. Then, we assume that alarms produced by transitions of the two components are collected separately, by independent sensors (one per component). WLOG, we can assume that the two components have no common alarm. The alarm pattern $\mathcal{A}_i$ collected on $\mathcal{N}_i$ is supposed to be a partial order of alarms (to account for the fact that sensor i may be itself a collection of sensors), that we can represent as a configuration (see Fig. 7). The way alarms are collected by each sensor ensures the Causal Observation Assumption: if event e_1 precedes event e_2 in the run κ, *i.e.* $e_1 \rightarrow^* e_2$, then the alarms α_1, α_2 they produce can't be observed in the reverse order, *i.e.* $\neg(\alpha_2 \rightarrow^* \alpha_1)$.

The centralized approach to the diagnosis problem essentially amounts to computing $\mathcal{U}^t_{\mathcal{N}} \times_T \mathcal{A}$, where $\mathcal{A} = \mathcal{A}_1 \times_T \mathcal{A}_2$ is the global alarm pattern (see [21] or [**?**]). $\times_T$ corresponds to $\wedge_T$ with an empty intermediary net, and uses alarm labels for synchronization. Here, we assume that $\mathcal{U}^t_{\mathcal{N}}$ is too large to allow this approach, and rather want to solve the problem by parts, using the factorized form of $\mathcal{U}^t_{\mathcal{N}}$. Specifically, our objective is to recover all runs of $\mathcal{N}_i$ that both explain local observations $\mathcal{A}_i$, and are compatible with at least one local explanation in the other component. Obviously, putting together two such compatible local solutions κ_i provides a possible global trajectory $\kappa = \kappa_1 \wedge_T \kappa_2$ which is a valid explanation for $\mathcal{A}_1 \times_T \mathcal{A}_2$, and conversely.

[3]For the sake of simplicity, we present concepts on a toy distributed system made of only two components. But the approach extends naturally to larger structures with N components, provided the interaction graph is a tree. In the general case, extra phenomena appear, due to the shape of the graph, that require specifi c development [25].

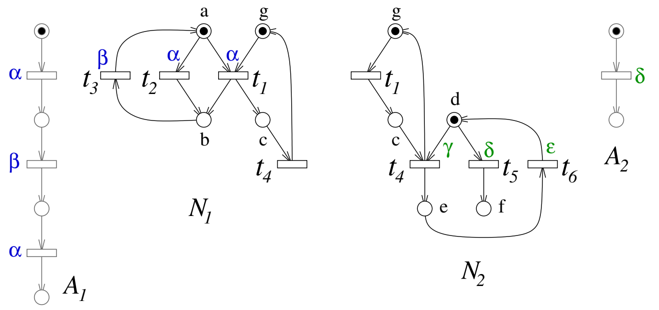

Fig. 7. *The two components of the net of Fig. 2, decorated with alarms produced by each transitions. On the sides, the observed patterns of alarms, represented as configurations (simple sequences here).*

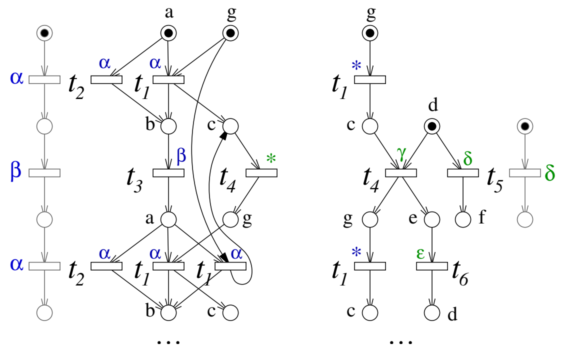

Fig. 8. *Time-unfolding applied to elements of Fig. 7.*

More formally, we want to compute

$$\mathcal{U}^t_{\mathcal{N}} \times_T \mathcal{A} = (\mathcal{U}^t_{\mathcal{N}_1} \times_T \mathcal{A}_1) \wedge_T (\mathcal{U}^t_{\mathcal{N}_2} \times_T \mathcal{A}_2) \qquad (9)$$

This "diagnosis" thus has a factorized form, where $\mathcal{U}^t_{\mathcal{N}_i} \times_T \mathcal{A}_i$ represents the *local diagnosis* for component $\mathcal{N}_i$. However, not all configurations in this local diagnosis remain after the $\wedge_T$ operation: only pairs of compatible local solutions are preserved, as in (7). The *minimal* factorized form of $\mathcal{U}^t_{\mathcal{N}} \times_T \mathcal{A}$ is given by:

$$\mathcal{U}^t_{\mathcal{N}} \times_T \mathcal{A} = \Pi_{\mathcal{N}_1}(\mathcal{U}^t_{\mathcal{N}} \times_T \mathcal{A}) \wedge_T \Pi_{\mathcal{N}_2}(\mathcal{U}^t_{\mathcal{N}} \times_T \mathcal{A}) \quad (10)$$

where each projection in the right hand side term yields a local view of the global diagnosis. From theorem 1, these local views are given by

$$\Pi_{\mathcal{N}_1}(\mathcal{U}^t_{\mathcal{N}} \times_T \mathcal{A}) = (\mathcal{U}^t_{\mathcal{N}_1} \times_T \mathcal{A}_1) \wedge_T \Pi_{\mathcal{N}_0}(\mathcal{U}^t_{\mathcal{N}_2} \times_T \mathcal{A}_2) \ (11)$$

This formula has the great advantage of being essentially based on local computations: Each $\mathcal{U}^t_{\mathcal{N}_i} \times_T \mathcal{A}_i$ requires a much weaker computational effort than $\mathcal{U}^t_{\mathcal{N}} \times_T \mathcal{A}$, and the term $\Pi_{\mathcal{N}_0}(\mathcal{U}^t_{\mathcal{N}_2} \times_T \mathcal{A}_2)$ represents a small size *message* from (the supervisor of) component $\mathcal{N}_2$ to (the supervisor of) component $\mathcal{N}_1$.

Fig. 9 illustrates these computations. Local diagnoses $\mathcal{U}^t_{\mathcal{N}_i} \times_T \mathcal{A}_i$ are a product of labeled nets where alarms are used for synchronization (transitions labeled with a $*$ represent silent transitions in a component, that synchronize with no observation). The projections of local diagnoses on the interface $\mathcal{N}_0$ are represented in the middle. A configuration κ_i explaining all local alarms $\mathcal{A}_i$ defines a *local solution*.

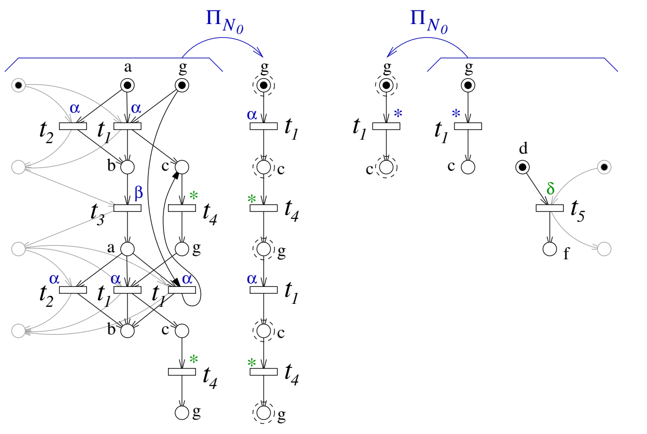

Fig. 9. *Local diagnoses $\mathcal{U}^t_{\mathcal{N}_i} \times_T \mathcal{A}_i$ and their projection on interface $\mathcal{N}_0$.*

Terminal nodes of a local solution indicate possible stop points on the interface, represented by dotted circles. The result of (11) for component $\mathcal{N}_1$ yields the configurations of Fig. 10. Observe that all local solutions using transition t_4 in Fig. 9 have been discarded because the firing of t_4 appears in no solution for component $\mathcal{N}_2$.

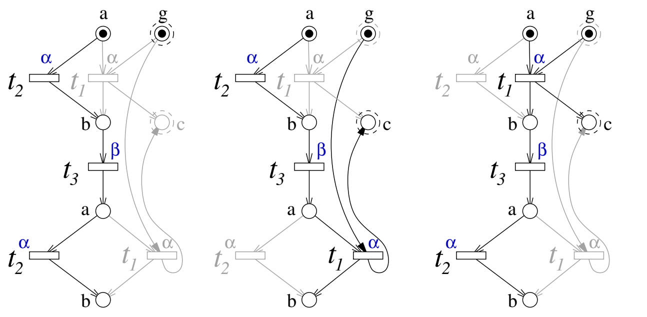

Fig. 10. *Gray+black lines on each net represent the local view of the global diagnosis in $\mathcal{N}_1$. Black lines emphasize the 3 maximal configurations explaining the full alarm pattern $\mathcal{A}_1$, and ending at a stop point of both components on the interface.*

VI. Conclusion

We have described a modular construction of a distributed system by synchronizing components through an interface. We have also presented a trellis structure to describe all possible trajectories of the resulting system. This trellis notion generalizes to concurrent systems the usual notion of trellis for an automaton, on which standard estimation and control techniques are based. Taking the time unfolding of a concurrent system defines a continuous functor, which means that it preserves synchronizations. In other words, the trellis of a compound system is the "product" of trellises of its components. This factorized representation for the trajectory set of a distributed system forms the basis of modular computations. Product, pullback and projection of trellises enjoy nice joint properties that allow to process systems by parts, by exchanging messages between local supervisors. This message passing strategy, that can be tightly related to estimation strategies for Bayesian networks, combines naturally with recursions "in time" inside each local supervisor, to explain observations on-line as they reach the supervisor. These aspects of computations will be developed in a forthcoming paper.

References

[1] S. Mac Lane, Categories for the Working Mathematician, Springer-Verlag, 1971.
[2] W. Reisig, Petri Nets, Springer Verlag, 1985.
[3] G. Winskel, Categories of models for concurrency, Seminar on Concurrency, Carnegie-Mellon Univ. (July 1984), LNCS 197, pp. 246-267, 1985.
[4] G. Winskel, Petri Nets, Algebras, Morphisms, and Compositionality, Information and Computation, no. 72, pp. 197-238, 1997.
[5] J. Esparza, Model checking using net unfoldings, Science of Computer Programming 23, pp. 151-195, 1994.
[6] J. Esparza, C. Schröter, Reachability Analysis Using Net Unfoldings, Workshop of Concurrency, Specification and Programming, volume II of Informatik-Bericht 140, pp. 255-270, Humboldt-Universität zu Berlin, 2000.
[7] S. Melzer, S. Römer, Deadlock checking using net unfoldings, CAV'97, LNCS 1254, pp. 352-363, 1997.
[8] K.L. McMillan, Using unfoldings to avoid the state explosion problem in the verification of asynchronous circuits, in Proc. 4th Workshop of Computer Aided Verification, Montreal, 1992, pp. 164-174.
[9] P. Baroni, G. Lamperti, P. Pogliano, M. Zanella, Diagnosis of Large Active Systems, Artificial Intell. 110, pp. 135-183, 1999.
[10] R.K. Boel, J.H. van Schuppen, Decentralized Failure Diagnosis for Discrete Event Systems with Costly Communication between Diagnosers, in proc. 6th Int. Workshop on Discrete Event Systems, WODES'02, pp. 175-181, 2002.
[11] R.K. Boel, G. Jiroveanu, Distributed Contextual Diagnosis for very Large Systems, in proc. of WODES'04, pp. 343-348, 2004.
[12] R. Debouk, S. Lafortune, D. Teneketzis, Coordinated decentralized protocols for failure diagnosis of discrete event systems, Discrete Event Dynamic Systems : theory and applications, vol. 10(1/2), pp. 33-86, 2000.
[13] O. Contant, S. Lafortune, Diagnosis of Modular Discrete Event Systems, in proc. of WODES'04, pp. 337-342, 2004
[14] S. Genc, S. Lafortune, Distributed Diagnosis Of Discrete-Event Systems Using Petri Nets, in proc. 24th Int. Conf. on Applications and Theory of Petri Nets, LNCS 2679, pp. 316-336, June, 2003.
[15] T. Yoo, S. Lafortune, A General Architecture for Decentralized Supervisory Control of Discrete-Event Systems, Discrete Event Dynamic Systems: Theory and Applications, vol. 12(3), pp. 335-377, July, 2002.
[16] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, D. Teneketzis, Diagnosability of Discrete-event systems, IEEE Trans. Autom. Control, vol. 40(9), pp. 1555-1575, 1995.
[17] M. Sampath, R. Sengupta, K. Sinnamohideen, S. Lafortune, D. Teneketzis, Failure diagnosis using discrete event models, IEEE Trans. on Systems Technology, vol. 4(2), pp. 105-124, March 1996.
[18] R. Su, W.M. Wonham, J. Kurien, X. Koutsoukos, Distributed Diagnosis for Qualitative Systems, in proc. 6th Int. Workshop on Discrete Event Systems, WODES'02, pp. 169-174, 2002.
[19] R. Su, Distributed Diagnosis for Discrete-Event Systems, PhD thesis, Dept. of Elec. and Comp. Eng., Univ. of Toronto, June 2004.
[20] Y. Pencole, M-O. Cordier, L. Roze, A decentralized model-based diagnostic tool for complex systems. Int. J. on Artif. Intel. Tools, World Scientific Publishing Comp., vol. 11(3), pp. 327-346, 2002.
[21] A. Benveniste, E. Fabre, S. Haar, C. Jard, Diagnosis of asynchronous discrete event systems, a net unfolding approach, IEEE Trans. on Automatic Control, vol. 48, no. 5, pp. 714-727, May 2003.
[22] E. Fabre, Factorization of Unfoldings for Distributed Tile Systems, Part 1 : Limited Interaction Case, INRIA research report no. RR-4829, April 2003.
[23] E. Fabre, Factorization of Unfoldings for Distributed Tile Systems, Part 2 : General Case, INRIA research report no. RR-5186, May 2004.
[24] E. Fabre, A. Benveniste, S. Haar, C. Jard, Distributed Monitoring of Concurrent and Asynchronous Systems, Journal of Discrete Event Systems, special issue, to appear in March 2005.
[25] E. Fabre, Convergence of the turbo algorithm for systems defined by local constraints, INRIA research report no. RR-4860, May 2003.
[26] E. Fabre, Trellis processes : a compact representation for runs of concurrent systems, INRIA research report, no. RR-5554, March 2005.