A trellis notion for distributed system diagnosis with sequential semantics

Eric Fabre, Christoforos Hadjicostis

***Abstract*— We consider modular automata, obtained as a product of elementary components, and adopt the usual sequential semantics : runs of these systems are sequences of events (by contrast with partial orders). The set of all runs of an automaton can be described by a trellis, which forms the support of many recursive estimation or decision algorithms. In this paper, we revise this notion to adapt it to the case of modular automata, and obtain a factorization property on this revised trellis. This factorization opens the way to distributed estimation algorithms that were described up to now in the partial order semantics.**

I. Introduction

Large distributed systems, obtained by assembling components, very rapidly become intractable as their size augments. This is due to combinatorial explosions that take place both in the state space, and in the trajectory space. Nevertheless, such large systems are becoming common today (e.g. system on a chip, grid computing, telecommunication network, etc.) and require to develop appropriate analysis and monitoring tools. Because of the mentioned combinatorial explosions, global (or centralized) approaches developed for discrete event dynamic systems (DEDS) are no longer applicable. Therefore several authors have proposed to address this challenge by means of distributed (or modular) methods, for example distributed/modular supervisory control, distributed/modular diagnosis, etc. The idea is to solve the problem by parts, in such a way that combining local/partial solutions gives the global one. See [4], [5], [6], [8], [12], [15] for examples of this approach.

Following this principle, a collection of results has been obtained for the diagnosis problem (DP). In its simplest version, the DP amounts to computing all runs of a distributed system that could explain a set of (possibly distributed) observations produced by a hidden run of this system. This can obviously be refined to check the occurrence of failure events in the hidden run. Most solutions take the form of a collection of supervisors monitoring part of the system, and exchanging information with neighboring supervisors. The difficulty generally lies in the definition and processing of messages, but also in the "protocol" aspects, *i.e.* deciding when one should communicate.

E. Fabre is with IRISA/INRIA, Campus de Beaulieu, 35042 Rennes cedex, France, `Eric.Fabre@irisa.fr`

C. Hadjicostis is with the Univ. Illinois Urbana Champaign, 357 Coordinated Science Lab., 1308 West Main Street, Urbana, IL 61801-2307, `chadjic@control.csluiuc.edu`

In [12] and [15], the originality is to start from a global system defined as a combination (by product) of components. A local supervisor is then attached to each component, and exchanges information with neighboring supervisors to compute a set of local runs both explaining local observations and being compatible with local runs computed by all other supervisors. These approaches combine several advantages : first, messages are easy to define and to process, and don't require the common assumption that interactions between components are observable, and secondly communications are completely asynchronous (no protocol needed). Moreover, they are based on a formal analogy with estimation algorithms for Markov random fields (or Bayesian networks), for which sound results are available and can be completely recycled to process distributed systems. [12] adopts the usual sequential semantics (SS), encoding sets of runs by languages and automata, while [14], [15], [16] use a true concurrency semantics (TCS), considering runs as partial orders of events and encoding them into unfoldings. The keystone of both approaches is a *factorization property* : the set of runs of the global system is a product, in a specific sense, of sets of runs of its components.

The purpose of this paper is to clarify the unfolding-based approach [16] by recasting it in the simpler case of sequential semantics. Specifically, we propose a notion of trellis to represent runs of a modular system in the SS, and open the way to recursive algorithms. Let us stress that, in the case of a single component, the trellis is the structure one would implicitly use for recursive state (or run) estimation given some sequence of observations, *e.g.* in the Viterbi algorithm. We therefore extend this notion to distributed systems, in order to adapt it also to recursions in "space," *i.e.* component by component. This new notion of trellis enjoys a nice factorization property, that allows to establish an analogy between a chain of components and a standard markov chain. Based on this analogy, modular message passing estimation algorithms can be derived to explain distributed observations in a system.

II. A category of multi-clock automata

This section defines the systems we use along this paper, together with a notion of morphism relating them, in order to form a category. We also introduce a combination method in this category, to build large systems out of components.

A. Multi-clock automata and morphisms

An *automaton* is a 4-tuple $\mathcal{A} = (S, T, s^0, \rightarrow)$ composed of a state set S, a transition set T, an initial state $s^0 \in \mathcal{S}$ and a flow relation $\rightarrow \subseteq (S \times T) \cup (T \times S)$ satisfying $\forall t \in T, |{}^\bullet t| = |t^\bullet| = 1$, where ${}^\bullet t$ and $t^\bullet$ represent pre- and post-states, as usual. We use $\rightarrow$ instead of $T \subseteq S \times S$ to account for multiple transitions between two states. A *labeled automaton* (LA) $\mathcal{A} = (S, T, s^0, \rightarrow, \lambda, \Lambda)$ is an automaton enriched with a labeling function $\lambda : T \rightarrow \Lambda$. This is used below for synchronization purposes.

A *multi-clock automaton* (MCA) $\mathcal{A} = (S, T, s^0, \rightarrow, \chi, I)$ is an automaton enriched with an index set I and an indexing function $\chi : T \rightarrow 2^I$. These indexes are used to keep track of components when they are assembled to form larger systems. So $\chi(t)$ represents components of $\mathcal{A}$ where transition t has an influence, and conversely $\chi^{-1}(i) = \{t \in T : i \in \chi(t)\}$ identifies transitions of component i (see the "product" subsection).

Naturally, *labeled multi-clock automata* (LMCA) enjoy the two extra functions above. When dealing with several automata $\mathcal{A}_1, \mathcal{A}_2$, etc., we shall use subscripts to identify their elements, like S_i, T_i, etc.

Let $\mathcal{A}_1, \mathcal{A}_2$ be two MCAs, a *morphism* $\phi : \mathcal{A}_1 \rightarrow \mathcal{A}_2$ is a triple (ϕ_S, ϕ_T, ϕ_I) where

1) $\phi_S : S_1 \rightarrow S_2$ is a total function on states, satisfying $\phi(s_1^0) = s_2^0$,
2) $\phi_T : T_1 \rightarrow T_2$ is a partial function on transitions, where $\phi_T(t_1) = \star$ denotes that ϕ_T is undefined at t_1,
3) if $\phi_T(t_1) = \star$, then $\phi_S({}^\bullet t_1) = \phi_S(t_1^\bullet)$, *i.e.* one can erase a transition only if pre- and post-states are merged,
4) if $\phi_T(t_1) \neq \star$, then $\phi_S({}^\bullet t_1) = {}^\bullet \phi_T(t_1)$ and $\phi_S(t_1^\bullet) = \phi_T(t_1)^\bullet$, *i.e.* the flow relation is preserved,
5) $\phi_I : I_1 \rightarrow I_2$ is a relation such that its opposite relation $\phi_I^{op} : I_2 \rightarrow I_1$ is a partial function and $\forall t_1 \in T_1,\ \phi_I(\chi_1(t_1)) = \chi_2(\phi_T(t_1))$, with the convention $\chi_i(\star) = \emptyset$.

Condition 5 implies in particular $\phi_T(t_1) = \star \Leftrightarrow \phi_I(\chi_1(t_1)) = \emptyset$. So when $\chi_1(t_1) \cap Dom(\phi_I) = \emptyset$, one has $\phi_T(t_1) = \star$, and conversely. In summary, ϕ erases all transitions that have no influence on components indexed by $Dom(\phi_I)$, or in other words removes *all* transitions *local* to components $I \setminus Dom(\phi_I)$. A morphism ϕ is said to be a *folding* when ϕ_T is a total function and $\phi_I = Id$.

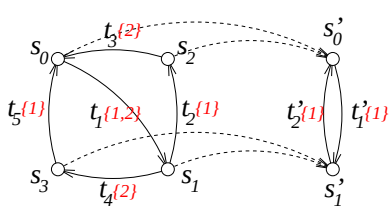

Fig. 1. *Two multi-clock automata, related by a morphism (dashed arrows).*

Fig. 1 gives an example of a morphism relating two MCA $\phi : \mathcal{A}_1 \rightarrow \mathcal{A}_2$. In $\mathcal{A}_1$ (left), one has $\chi_1^{-1}(1) = \{t_1, t_2, t_5\}$ and $\chi_1^{-1}(2) = \{t_1, t_3, t_4\}$, as indicated by the red numbers close to transition names. ϕ collapses s_0 and s_2 into s_0', and s_1, s_3 into s_1', and at the same time erases all transitions outside $\chi_1^{-1}(1)$.

For morphisms of labeled MCAs, we add requirements:

6) $\Lambda_2 \subseteq \Lambda_1$, *i.e.* ϕ reduces the label set,
7) $Dom(\phi_T) = \lambda_1^{-1}(\Lambda_2)$,
8) ϕ preserves labels on its domain $Dom(\phi_T)$, *i.e.* $\lambda_1(t_1) = \lambda_2(\phi_T(t_1))$ if $\phi_T(t_1) \neq \star$.

We denote by $\mathbb{A}$ the category having the LMCA as objects, and the above morphisms as arrows. By abuse of notations, we will write ϕ instead of ϕ_S, ϕ_T or ϕ_I.

B. Product

The product $\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2$ of two LMCA is defined by

1) $S = S_1 \times S_2$ and $s^0 = (s_1^0, s_2^0)$,
2) $T = T_s \cup T_p$ where

$$T_s = \{(t_1, t_2) \in T_1 \times T_2 : \lambda_1(t_1) = \lambda_2(t_2)\}$$
$$T_p = \{(t_1, \star_{s_2}) : t_1 \in T_1,\ s_2 \in S_2,\ \lambda_1(t_1) \in \Lambda_1 \setminus \Lambda_2\}$$
$$\cup \{(\star_{s_1}, t_2) : s_1 \in S_1,\ t_2 \in T_2,\ \lambda_2(t_2) \in \Lambda_2 \setminus \Lambda_1\}$$

The notation[1] $\star_{s_i}$ stands for a (fake) loop at state s_i.

3) $\rightarrow$ is defined by $(s_1, s_2) \rightarrow (t_1, t_2) \rightarrow (s_1', s_2')$ iff $s_i \rightarrow_i t_i \rightarrow_i s_i'$, $i = 1, 2$, where one of the t_i can be $\star_{s_i}$ and assuming $s_i \rightarrow_i \star_{s_i} \rightarrow_i s_i$ holds for every state $s_i \in S_i$,
4) $\Lambda = \Lambda_1 \cup \Lambda_2$ and λ follows accordingly,
5) $I = (I_1 \times \{\star\}) \cup (\{\star\} \times I_2)$ and χ is defined by $\chi(t_1, t_2) = (\chi_1(t_1) \times \{\star\}) \cup (\{\star\} \times \chi_2(t_2))$ where one of the t_i can be $\star_{s_i}$, with the convention $\chi_i(\star_{s_i}) = \emptyset$.

Observe that the disjoint union of indexes appearing in point 5 allows to keep track of components when a product is performed. Fig. 2 illustrates the product of labeled MCA.

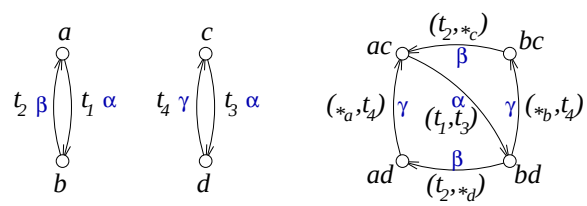

Fig. 2. *Two multi-clock automata (left) and their label-based product (right). Labels are indicated by Greek letters close to transitions ; we assume* $\Lambda_1 \cap \Lambda_2 = \{\alpha\}$.

Let us denote by $\pi_i : \mathcal{A}_1 \times \mathcal{A}_2 \rightarrow \mathcal{A}_i$ the canonical projections. It is straightforward to check that they are morphisms (with the convention $\pi_i(t_1, t_2) = \star_{s_i}$ means "undefined"). Moreover, one has the following property:

Lemma 1 (universal property of the product): Let $\mathcal{A}'$ be an LMCA and let the $\phi_i : \mathcal{A}' \rightarrow \mathcal{A}_i,\ i = 1, 2$ be two morphisms. There exists a unique morphism $\psi : \mathcal{A}' \rightarrow \mathcal{A}_1 \times \mathcal{A}_2$ such that $\phi_i = \pi_i \circ \psi,\ i = 1, 2$.

This result makes $\times$ the categorical product in $\mathbb{A}$, which immediately entails its associativity. Let us define the $\mathcal{A}_i' =$

[1]This notation avoids introducing stuttering transitions at every state, before computing the product.

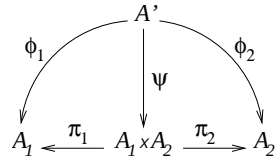

Fig. 3. *Universal property of the product in* $\mathbb{A}$.

$\pi_i(\mathcal{A}_1 \times \mathcal{A}_2)$, then $\mathcal{A}'_1 \times \mathcal{A}'_2$ is isomorphic to $\mathcal{A}_1 \times \mathcal{A}_2$, which we also write $\mathcal{A}_1 \times \mathcal{A}_2 = \mathcal{A}'_1 \times \mathcal{A}'_2$ (objects are defined up to isomorphism in category theory). More generally, let us write $\mathcal{A}' \subseteq \mathcal{A}$ when $\mathcal{A}'$ has the same states than $\mathcal{A}$ but less transtitions (so there is an injective morphism from $\mathcal{A}'$ to $\mathcal{A}$). Consider $\mathcal{A} \subseteq \mathcal{A}_1 \times \mathcal{A}_2$, and $\mathcal{A}'_i = \pi_i(\mathcal{A})$, then $\mathcal{A} \subseteq \mathcal{A}'_1 \times \mathcal{A}'_2$, and taking any $\mathcal{A}''_i \subset \mathcal{A}'_i$ instead of $\mathcal{A}'_i$ will break this inclusion. Therefore we call $\mathcal{A}'_1 \times \mathcal{A}'_2$ the *minimal product covering* of $\mathcal{A}$ in $\mathcal{A}_1 \times \mathcal{A}_2$.

III. Representation of trajectory spaces

In this section, we define runs of an LMCA in the sequential semantics. We then provide a compact way to represent sets of runs, by means of trellis automata. There exists a unique (up to isomorphism) maximal trellis automaton associated to an LMCA, that we call its sequential time-unfolding. It represents all possible runs of this LMCA. A universal property is stated on the time-unfolding of an LMCA, from which a factorization property is derived.

A. Trellis automaton

In the sequential semantics (SS), a *run* (or *trajectory*) of an LMCA $\mathcal{A}$ is modeled as a sequence $\sigma = (t_1, t_2, \ldots, t_N)$ of transitions such that ${}^\bullet t_1 = s^0$ and $t_n^\bullet = s_n = {}^\bullet t_{n+1},\ 1 \leq n \leq N-1$ (also denoted $\ldots s_{n-1}[t_n\rangle s_n[t_{n+1}\rangle s_{n+1} \ldots$, and $s^0[\sigma\rangle s_N$ for sequences). We denote by $\sigma_1|\sigma_2$ the concatenation of two sequences, when $s^0[\sigma_1\rangle s_1[\sigma_2\rangle s_2$.

Lemma 2: Let $\phi : \mathcal{A}_1 \to \mathcal{A}_2$ be an LMCA morphism, and let σ_1 be a run of $\mathcal{A}_1$, then $\sigma_2 = \phi(\sigma_1)$ is a run of $\mathcal{A}_2$.
Naturally, $\phi(\sigma_1)$ is defined by the recursion $\phi(\sigma_1|t_1) = \phi(\sigma_1)|\phi(t_1)$ if $\phi(t_1) \neq \star$, and $\phi(\sigma_1|t_1) = \phi(\sigma_1)$ if $\phi(t_1) = \star$. The proof of the lemma is straightforward and left to the reader. Let us stress that morphisms were precisely defined so as to guarantee this property.

We now focus on the representation of *sets* of trajectories for a given LMCA. To do so, we introduce a special family of LMCA: $\mathcal{T}$ is said to be a *trellis automaton* (TA) when

1) the graph of $\mathcal{T}$, determined by $\to$, is directed and acyclic, or equivalently $\to^*$ defines a partial order on nodes of $\mathcal{T}$ (*i.e.* states and transitions),
2) this partial order is well founded: $\forall x \in S \cup T,\ |\{x' \in S \cup T\ : x' \to^* x\}| < \infty$,
3) s^0 is the unique minimum of this partial order.

As a consequence, in any run $\sigma = (t_1, \ldots, t_n)$ of a TA, all transitions t_i are different. In the sequel, states of a TA are called *conditions* (denoted by $c \in C$ instead of $s \in S$), and transitions are called *events* (denoted by $e \in E$ instead of $t \in T$).

A trellis automaton $\kappa = (C', E', \to', c^0, \chi', I, \lambda', \Lambda)$ is called a *configuration* when it is composed of a single sequence of events: $\forall c \in C', |{}^\bullet c| \leq 1$ and $|c^\bullet| \leq 1$. Let $\mathcal{T} = (C, E, \to, c^0, \chi, I, \lambda, \Lambda)$ be another TA, κ is a "configuration of $\mathcal{T}$" if it is a sub-TA of $\mathcal{T}$. Naturally, λ' and χ' are then restrictions of λ and χ to E'. By a slight abuse of notations, we identify runs of $\mathcal{T}$ with its configurations.

To events of $\mathcal{T}$ we associate a multi-clock function:

$$\forall e \in E, \quad h_e : \begin{array}{l} I \to \mathbb{N} \\ i \mapsto h_e(i) = 1_{i \in \chi(e)} \end{array} \tag{1}$$

This definition extends to configurations κ of $\mathcal{T}$: assume κ corresponds to the sequence of events $(e_1, e_2, \ldots, e_N)$ in T, then $h_\kappa \triangleq \sum_{n=1}^{N} h_{e_n}$. For every component index $i \in I$, h_κ thus counts the number of events associated to component i in run κ, whence the name "multi-clock function."

A trellis automaton $\mathcal{T}$ is *correctly folded* iff $\forall c \in C$ and for any pair κ, κ' of configurations of $\mathcal{T}$ ending at c (which we abusively denote by $c^0[\kappa\rangle c$ and $c^0[\kappa'\rangle c$), one has $h_\kappa = h_{\kappa'}$. In other words, only sequences of events with the same multi-clock value can merge at the same condition c. Intuitively, this property will guarantee that by selecting transitions of a given component and erasing the others, which we will call a projection, one will get a valid trellis process of the selected component.

From now, we only consider correctly folded trellis automata; they form the subcategory $\mathbb{T}$ of $\mathbb{A}$. In a (correctly folded) TA $\mathcal{T}$, the multi-clock function naturally extends to conditions by $h_c \triangleq h_\kappa$ for any configuration κ of $\mathcal{T}$ such that $c^0[\kappa\rangle c$.

B. Sequential time-unfolding

Let $\mathcal{A} \in \mathbb{A}$, $\mathcal{T} \in \mathbb{T}$, and let $f : \mathcal{T} \to \mathcal{A}$ be a morphism. The pair $(\mathcal{T}, f)$ is a *trellis process* (TP) of $\mathcal{A}$ iff

1) f is a folding of $\mathcal{T}$ onto $\mathcal{A}$,
2) $\mathcal{T}$ satisfies the parsimony criterion:
$$\forall e, e' \in E, \qquad [\,{}^\bullet e = {}^\bullet e',\ \ f(e) = f(e')\,] \ \Rightarrow\ e = e'$$
3) $\mathcal{T}$ is maximally folded:
$$\forall c, c' \in C, \qquad [\,h_c = h_{c'},\ \ f(c) = f(c')\,] \ \Rightarrow\ c = c'$$

A TP $\mathcal{T}$ of $\mathcal{A}$ encodes a set of runs of $\mathcal{A}$. In effect, f can be viewed as a labeling of events of $\mathcal{T}$ by transitions of $\mathcal{A}$. Every configuration of $\mathcal{T}$ thus represents through f a run of $\mathcal{A}$ (point 1). And conversely a run of $\mathcal{A}$ is represented at most once in a TP of $\mathcal{A}$ (points 2,3).

Let $\mathcal{A} = (S, T, s^0, \to, \chi, I, \lambda, \Lambda)$ be an LMCA. A trellis process $(\mathcal{T}, f)$ of $\mathcal{A}$, with $\mathcal{T} = (C, E, c^0, \to, \chi', I, \lambda', \Lambda)$, can be obtained by the following recursion:

Procedure 1

1) Initialization: $C = \{c^0\},\ f(c^0) = s^0,\ E = \emptyset$
2) Recursion: for $c \in C$ and $t \in T$ such that $f(c) = {}^\bullet t$, if $\nexists e \in E$ such that ${}^\bullet e = c$ and $f(e) = t$, then
   a) create $e \in E$, set ${}^\bullet e = c,\ f(e) = t,\ \chi'(e) = \chi(t)$ and $\lambda'(e) = \lambda(t)$,
   b) create $c' \in C$, set ${}^\bullet c' = e$ and $f(c') = t^\bullet$,

c) if $\exists c'' \in C$ with $f(c'') = f(c')$ and $h_{c''} = h_{c'}$, then merge c' and c''.

The three conditions of the definition are satisfied by construction, and clearly any trellis process of $\mathcal{A}$ can be obtained in that way, with a suitable guiding of events to connect.

Let $(\mathcal{T}_1, f_1), (\mathcal{T}_2, f_2)$ be two trellis processes of $\mathcal{A}$, their union $(\mathcal{T}, f)$, with injective morphisms $\psi_i : \mathcal{T}_i \to \mathcal{T}$, can be defined by the following recursion. To simplify notations, we define properties P_i, $i \in \{1, 2\}$, by

$$P_i \quad \Leftrightarrow \quad f_i = f \circ \psi_i, \;\; \lambda_i = \lambda \circ \psi_i, \;\; \chi_i = \chi \circ \psi_i$$

that are preserved all along procedure 2.

Procedure 2

1) Initialization: $C = \{c^0\}$, $\psi_i(c_i^0) = c^0$, $f(c^0) = f_1(c_1^0) = f_2(c_2^0)$, $E = \emptyset$
2) Recursion:
   for $i \in \{1, 2\}$ and $e_i \in E_i$ such that ψ_i defined at $c_i = {}^\bullet e_i$ but not at e_i,
   a) if $\exists e \in E$ such that ${}^\bullet e = \psi_i(c_i)$ and $f(e) = f_i(e_i)$, then set $\psi_i(e_i) = e$ and $\psi_i(e_i^\bullet) = e^\bullet$ (which preserves P_i),
   b) otherwise create a new $e \in E$ with ${}^\bullet e = \psi_i(c_i)$, and extend f, λ, χ to e in order to preserve P_i,
   c) then, for $c_i' = e_i^\bullet$,
      - if $\exists c' \in C$ such that $h(c') = h(c_i')$ and $f(c') = f_i(c_i')$, set $e^\bullet = c'$ and $\psi_i(c_i') = c'$ (which preserves P_i),
      - otherwise create a new $c' \in C$, with ${}^\bullet c' = e$ and $\psi_i(c_i') = c'$, then extend f, λ, χ to c' in order to preserve P_i.

Notice that the union of two trellis processes of $\mathcal{A}$ may contain runs that are neither present in one nor in the other. The union defined by Procedure 2 extends without difficulty to any number of trellis processes of $\mathcal{A}$, and is associative.

Lemma 3: Let $(\mathcal{T}, f)$ be a trellis process of $\mathcal{A}$, then $\mathcal{T}$ is isomorphic to the union of its configurations (κ_i, f_i). As a consequence, two trellis processes of $\mathcal{A}$ that have isomorphic configurations are isomorphic.

Theorem 1: Given an LMCA $\mathcal{A}$, there exists a unique maximal trellis process $(\mathcal{U}_\mathcal{A}^{st}, f_\mathcal{A})$ of $\mathcal{A}$ (maximal for inclusion). We call it the *sequential time-unfolding* of $\mathcal{A}$ (ST-unfolding or STU for short).

Proof. Define $(\mathcal{U}_\mathcal{A}^{st}, f_\mathcal{A})$ as the union of all trajectories (κ, f) of $\mathcal{A}$. Unicity is obvious from lemma 3.

Let $(\mathcal{T}, f)$ be a TP of $\mathcal{A}$. $(\mathcal{T}, f)$ is isomorphic to the union $\cup_{j \in J}(\kappa_j, f_j)$ of its configurations by lemma 3, and since the union is associative, $(\mathcal{U}_\mathcal{A}^{st}, f_\mathcal{A})$ is obtained as the union of $\cup_{j \in J}(\kappa_j, f_j)$ with the remaining trajectories of $\mathcal{A}$. So from procedure 2 we know there exists an injective morphism from $\cup_{j \in J}(\kappa_j, f_j)$ to $(\mathcal{U}_\mathcal{A}^{st}, f_\mathcal{A})$, so from $(\mathcal{T}, f)$ to $(\mathcal{U}_\mathcal{A}^{st}, f_\mathcal{A})$. □

C. Universal property and its consequences

Theorem 2 (Universal property): Let $\mathcal{A} \in \mathbb{A}$ be an LMCA, let $\mathcal{T} \in \mathbb{T}$ be a trellis automaton and $\phi : \mathcal{T} \to \mathcal{A}$ a morphism, there exists a unique morphism $\psi : \mathcal{T} \to \mathcal{U}_\mathcal{A}^{st}$ such that $\phi = f_\mathcal{A} \circ \psi$.

$$\begin{tikzcd} T \arrow[rr, "\forall\phi"] \arrow[dr, bend right, "\exists!\,\psi"'] & & A \\ & U_A^{st} \arrow[ur, bend right, "f_A"'] & \end{tikzcd}$$

Fig. 4. *Universal property of the STU of $\mathcal{A}$.*

Proof. We adopt notations $\mathcal{T} = (C', E', c_0', \to', \chi', I')$, $\mathcal{U}_\mathcal{A}^{st} = (C, E, c_0, \to, \chi, I)$ and $\mathcal{A} = (S, T, s_0, \to_A, \xi, I)$, putting aside labels, which have no influence here. We denote by h' and h the multi-clock functions in $\mathcal{T}$ and $\mathcal{U}_\mathcal{A}^{st}$ respectively. Recall that ϕ erases events of $\mathcal{T}$ which are local to components indexed by $I' \setminus \phi^{op}(I)$.

We first have to build ψ from necessary conditions. Let $c' \in C'$, with multi-clock function $h_{c'}' : I' \to \mathbb{N}$, there exists a unique $c \in C$ such that $f_\mathcal{A}(c) = \phi(c')$ and $h_c = h_{c'}' \circ \phi^{op}$. In effect, let κ' be a configuration of $\mathcal{T}$ such that $[\kappa'\rangle c'$, then $\phi(\kappa)$ yields a run of $\mathcal{A}$, represented by a configuration κ in $\mathcal{U}_\mathcal{A}^{st}$, and c is given by $[\kappa\rangle c$. Uniqueness comes from the fact that $\mathcal{U}_\mathcal{A}^{st}$ is maximally folded. We define $\psi(c') = c$.

Let $e' \in E'$, if $\phi(e') = \star$, we take $\psi(e') = \star$. Otherwise, there exists a unique $e \in E$ such that $h_e = h_{e'}' \circ \phi^{op}$ and $f_\mathcal{A}(e) = \phi(e')$ (by the same arguments as for conditions). We define $\psi(e') = e$.

Finally, we take $\psi : I' \to I$ identical to ϕ.

ψ satisfies $\phi = f_\mathcal{A} \circ \psi$ by construction, and is clearly a morphism. Moreover, ψ preserves clock tics on events and conditions, in the following sense: $\forall x \in C' \cup E'$, $\psi(x) \neq \star \;\Rightarrow\; h_{\psi(x)} = h_x' \circ \psi_I^{op}$. □

Notice that given the universal property of $\mathcal{U}_\mathcal{A}^{st}$, the union of trellis processes $(\mathcal{T}_j, \phi_j)_{j \in J}$ of $\mathcal{A}$ can be redefined more easily than with the complex procedure 2: there exist unique $\psi_j : \mathcal{T}_i \to \mathcal{U}_\mathcal{A}^{st}$ such that $\phi_j = f \circ \psi_j$. The union of the $\mathcal{T}_j$ is then given by $\mathcal{T} \triangleq \cup_{j \in J} \psi_j(\mathcal{T}_j)$, where the latter is simply a union of sub-trellis automata of $\mathcal{U}_\mathcal{A}^{st}$ (in the sense of set union). This makes lemma 3 more intuitive.

Example: Consider the two LMCA of fig. 5, $\mathcal{A}$ (top left) and $\mathcal{A}'$ (bottom left), assuming labels are such that t_i synchronizes with t_i' except for t_4' which is local to $\mathcal{A}'$. The ST-unfolding of the product $\mathcal{A} \times \mathcal{A}'$ is represented by the top right TP, where condition and event names are replaced by their image through the folding $f_{\mathcal{A} \times \mathcal{A}'}$. Unreachable states are not represented. The bottom right TP represents $\mathcal{U}_\mathcal{A}^{st}$, with a similar labeling of nodes.

Observe that there exist two conditions labeled (c, c') in $\mathcal{U}_{\mathcal{A} \times \mathcal{A}'}^{st}$, because they have different height functions: the top one is reached after 2 firings in $\mathcal{A}$ and 2 firings in $\mathcal{A}'$, whereas the bottom one is reached after 2 firings in $\mathcal{A}'$ but a single one in $\mathcal{A}$. There obviously exists a morphism $\phi : \mathcal{U}_{\mathcal{A} \times \mathcal{A}'}^{st} \to \mathcal{A}$ (consider $\Pi_\mathcal{A} \circ f_{\mathcal{A} \times \mathcal{A}'}$ where $\Pi_\mathcal{A} : \mathcal{A} \times \mathcal{A}' \to \mathcal{A}$ is the canonical projection) and thus a unique $\psi : \mathcal{U}_{\mathcal{A} \times \mathcal{A}'}^{st} \to \mathcal{U}_\mathcal{A}$ such that $\phi = f_\mathcal{A} \circ \psi$. The two conditions labeled (c, c') would be merged if we were using the standard notion of trellis, based on a global clock, because both conditions are obtained after two firings in $\mathcal{A} \times \mathcal{A}'$. But in that case, the morphism ψ wouldn't exist anymore. This justifies the use of

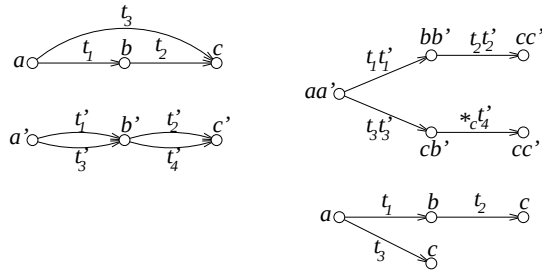

Fig. 5. *Two LMCA $\mathcal{A}$ and $\mathcal{A}'$ (left), the ST-unfolding $\mathcal{U}^{st}_{\mathcal{A}\times\mathcal{A}'}$ (top right) and the ST-unfolding $\mathcal{U}^{st}_{\mathcal{A}}$ (bottom right).*

a multi-clock functions to define merge points in sequential time-unfoldings. In the next paragraph, we examine more closely the relation between products and ST-unfoldings.

$\mathbb{T}$ being a sub-category of $\mathbb{A}$, there exists an inclusion functor $F = \subseteq : \mathbb{T} \to \mathbb{A}$. Conversely, the ST-unfolding operation on LMCA defines a functor $G = \mathcal{U}^{st} : \mathbb{A} \to \mathbb{T}$. By "functor," we mean that $\mathcal{U}^{st}$ applies also to morphisms and satisfies $\mathcal{U}^{st}(f \circ g) = \mathcal{U}^{st}(f) \circ \mathcal{U}^{st}(g)$. The universal property associated to any $\mathcal{U}^{st}_{\mathcal{A}}$ entails that (F, G) forms a pair of adjoint functors (see [1] chapter 4, in particular theorem 2.iv, p. 81). As the right adjoint of the pair, G preserves categorical limits, and so preserves products (see [1] chapter 5.5, in particular theorem 1, p. 114). In other words, one has

$$\mathcal{U}^{st}(\mathcal{A}_1 \times \mathcal{A}_2) \quad = \quad \mathcal{U}^{st}(\mathcal{A}_1) \times_T \mathcal{U}^{st}(\mathcal{A}_2) \qquad (2)$$

where $\times_T$ denotes the categorical product in $\mathbb{T}$. By contrast with $\times$, the product in $\mathbb{T}$ not only synchronizes events with shared labels, but also preserves the partial ordering of the resulting structure. It must be understood as an operation synchronizing trajectories instead of automata.

The adjoint pair (F, G), where F corresponds to an inclusion, is called a co-reflection. Since moreover one has $\forall \mathcal{T} \in \mathbb{T},\ \mathcal{T} \simeq \mathcal{U}^{st}_{\mathcal{T}}$ (read "isomorphic to"), one can actually define the product $\times_T$ by

$$\mathcal{T}_1 \times_T \mathcal{T}_2 \ \simeq\ \mathcal{U}^{st}(\mathcal{T}_1) \times_T \mathcal{U}^{st}(\mathcal{T}_2) \ =\ \mathcal{U}^{st}(\mathcal{T}_1 \times \mathcal{T}_2) \qquad (3)$$

This last relation has a practical consequence: by coupling Procedure 1, computing ST-unfoldings, to the definition of the product in $\mathbb{A}$, one gets a recursive procedure to compute the product in $\mathbb{T}$.

IV. Centralized diagnosis

Up to now we have defined a way to compose automata and keep track of components in the result. We have also defined a way to represent runs of these systems by means of a trellis structure adapted to the notion of compound system. These trellises are similar to the ordinary notion of trellis for an automaton, apart from the counting of time, which is local to each component rather than being global: two trajectories merge at their final state iff they reach the same state after the same number of ticks in all components. This augments the size of the resulting trellis, since there are less merge points compared to the ordinary case. But for that price one obtains the nice factorization property (2) that will be central in the derivation of distributed/modular diagnosis procedures.

A. Centralized diagnosis with a single sensor

Assume $\mathcal{A} = (S, T, s^0, \to, \xi, I, \lambda, \Lambda)$ produces a hidden run κ. One gets information about κ by means of a "sensor" $\mathcal{O}$ observing some of the transition labels produced by κ. Specifically, we assume $\mathcal{O}$ collects labels of $\Lambda' \subseteq \Lambda$ under the form of a sequence of observed labels. The objective is to recover all runs of $\mathcal{A}$ that could explain $\mathcal{O}$, among which will lie the true hidden run κ.

The traditional approach to the diagnosis problem consists in deciding whether some undesirable invisible event (the *fault*) occurred or not in the hidden κ. Classically, one can easily recast this problem into the previous one, by first introducing some memory in the system to keep track of the occurrence of a faulty transition (state augmentation, by one bit), and then applying a diagnosis decision on the set of possible final (augmented) states of $\mathcal{A}$ given $\mathcal{O}$, *i.e.* $\{c : c_0[\kappa\rangle c,\ \kappa \text{ explains } \mathcal{O}\}$.

We model observations as a configuration $\mathcal{O}$ with a single component $\mathcal{O} = (C', E', c'_0, \to', \chi' = 1, I' = \{1\}, \lambda', \Lambda')$. The "diagnosis" $\mathcal{D}$ is defined as the product:

$$\mathcal{D} \quad \triangleq \quad \mathcal{U}^{st}(\mathcal{A}) \times_T \mathcal{O} \ \simeq\ \mathcal{U}^{st}(\mathcal{A} \times \mathcal{O}) \qquad (4)$$

which synchronizes runs of $\mathcal{A}$ with observations through common labels Λ'. There exist two canonical projections $\pi_{\mathcal{A}} : \mathcal{D} \to \mathcal{U}^{st}(\mathcal{A})$ and $\pi_{\mathcal{O}} : \mathcal{D} \to \mathcal{O}$ associated to this product. The runs of $\mathcal{A}$ explaining observations are the $\Pi_{\mathcal{A}}(\bar{\kappa})$ where $\bar{\kappa}$ is a configuration of $\mathcal{D}$ such that $\Pi_{\mathcal{O}}(\bar{\kappa}) = \mathcal{O}$ (the full observed sequence must be explained by $\bar{\kappa}$). These $\bar{\kappa}$ are easy to isolate in $\mathcal{D}$ when $\mathcal{O}$ is finite: they contain a condition c which multi-clock function h_c satisfies $h_c((\star, 1)) = N$ where N is the number of events in $\mathcal{O}$.

Notice that unobservable transitions of $\mathcal{A}$, *i.e.* those labeled by $\Lambda \setminus \Lambda'$, are "free of use" in $\mathcal{D}$: they correspond to local events of the factor $\mathcal{U}^{st}_{\mathcal{A}}$, that do not have to synchronize with $\mathcal{O}$. There exist runs in $\mathcal{D}$ that only explain the first observations of $\mathcal{O}$ and then stop. All these useless runs can be easily discarded by recursively removing from $\mathcal{D}$ any maximal node x (for $\to^*$) which multi-clock value doesn't satisfy $h_x((\star, 1)) \geq N$, whether $\mathcal{O}$ is finite or not.

B. Centralized diagnosis with several sensors

1) Global system: We now assume that the events of the hidden run of $\mathcal{A}$ are observed by several sensors $\mathcal{O}_1, \ldots, \mathcal{O}_K$, where $\Lambda'_k \subseteq \Lambda$ is the label set of $\mathcal{O}_k$. Each $\mathcal{O}_k$ is a configuration, and we assume that the exact interleaving of the observations collected by the different sensors is *lost*. This is the essential difference with the previous case where a single sequence of observations was assumed. Since the interleaving of the $\mathcal{O}_k$ is unknown, all possibilities must be explored. In other words, our observation is

$$\mathcal{O} \quad \triangleq \quad \mathcal{O}_1 \times_T \ldots \times_T \mathcal{O}_K \qquad (5)$$

Each configuration in $\mathcal{O}$ is one possible interleaving of the K observed sequences. Notice that the above product

captures the fact that a given event of the hidden κ may be observed simultaneously by several sensors: this yields a synchronization point between sensors. See the example of fig. 6.

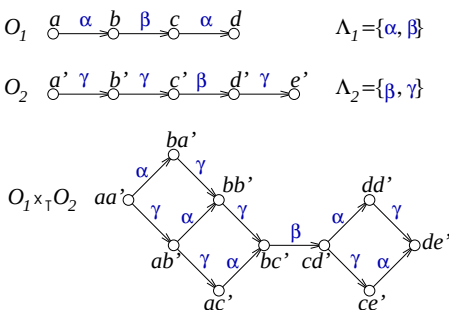

Fig. 6. *Two TA $\mathcal{O}_1, \mathcal{O}_2 \in \mathcal{T}$ (top) sharing a single label β. Only event labels are represented. The product $\mathcal{O}_1 \times_T \mathcal{O}_2$ (bottom) contains all possible interleavings of these two sequences, up to synchronized events. Unreachable states are not represented.*

The diagnosis $\mathcal{D}$ is still defined by (4), *i.e.*

$$\mathcal{D} \quad = \quad \mathcal{U}^{st}(\mathcal{A}) \times_T \mathcal{O}_1 \times_T \ldots \times_T \mathcal{O}_K \qquad (6)$$

and solutions are configurations $\kappa = \pi_{\mathcal{A}}(\bar{\kappa})$ such that $\forall 1 \leq k \leq K,\ \pi_{\mathcal{O}_k}(\bar{\kappa}) = \mathcal{O}_k$, *i.e.* runs of $\mathcal{A}$ explaining one possible interleaving of the observed sequences. Once again, $\mathcal{D}$ can be pruned so as to contain only maximal nodes x for which $\forall 1 \leq k \leq K,\ h_x((\star, k)) \geq N_k$, where N_k is the number of events in $\mathcal{O}_k$ (possibly ∞) and index $(*, k)$ represents component $\mathcal{O}_k$ in $\mathcal{D}$.

2) Modular system: Going further in the modularity, we now assume that $\mathcal{A}$ itself decomposes as a product of components $\mathcal{A} = \mathcal{A}_1 \times \ldots \times \mathcal{A}_K$, with respective label sets Λ_k, and that a specific sensor $\mathcal{O}_k$ is dedicated to each component $\mathcal{A}_k$ and observes labels of $\Lambda'_k \subseteq \Lambda_k$. The case where $\mathcal{A}_k$ produces no observation is captured by $\Lambda'_k = \emptyset$. Inserting (2) in definition (6), the diagnosis decomposes as

$$\mathcal{D} \quad = \quad (\mathcal{U}^{st}_{\mathcal{A}_1} \times_T \mathcal{O}_1) \times_T \ldots \times_T (\mathcal{U}^{st}_{\mathcal{A}_K} \times_T \mathcal{O}_K) \qquad (7)$$

Each $\mathcal{D}_k \triangleq \mathcal{U}^{st}_{\mathcal{A}_k} \times_T \mathcal{O}_k$ is a local diagnosis on component $\mathcal{A}_k$. It provides runs of $\mathcal{A}_k$ that match local observations but that do not necessarily comply with other observations nor with constraints due to the other components.

Let us denote by π_k the canonical projection of $\mathcal{D} = \mathcal{D}_1 \times_T \ldots \times_T \mathcal{D}_K$ on $\mathcal{D}_k$, and define $\mathcal{D}'_k = \pi_k(\mathcal{D})$. Each $\mathcal{D}'_k$ represents the *local view* on component $\mathcal{A}_k$ of the global diagnosis: it contains local runs of $\mathcal{A}_k$ that not only explain local observations, but are also compatible with local explanations on the other components. More precisely, one knows from section II-B, in particular the comments following lemma 1, that

$$\mathcal{D} = \mathcal{D}'_1 \times_T \ldots \times_T \mathcal{D}'_K \quad \text{where} \quad \mathcal{D}'_k = \pi_k(\mathcal{D}) \subseteq \mathcal{D}_k \qquad (8)$$

The diagnosis $\mathcal{D}$ admits several product forms with factors in the $\mathcal{U}^{st}(\mathcal{A}_k \times \mathcal{O}_k)$, for example (7) and (8). The latter is particular however: it is the *minimal product covering* of $\mathcal{D}$ in the sense that replacing a $\mathcal{D}'_k$ by any $\mathcal{D}''_k \subset \mathcal{D}'_k$ would break the equality.

V. Modular/distributed diagnosis

For a product system, it is equivalent to know the global diagnosis $\mathcal{D}$ or its local views $\mathcal{D}'_k$. The latter may be sufficient if one is only interested in what happened in some components, and in many cases the product form (8) of $\mathcal{D}$ is much more compact than the expanded form, considering that the product generally multiplies the number of events. It is therefore desirable to take the $\mathcal{D}'_k$ as our objective. Moreover, in some cases there exists an efficient way to obtain them directly, avoiding the expensive construction of $\mathcal{D}$. This direct derivation is based on local computations combining products and projections, and reproducing computation schemes borrowed to Bayesian networks, namely the so-called "belief propagation."

A. Notations

Let the $(\mathcal{T}_j, f_j)_{j \in J}$ be trellis processes of the same $\mathcal{A}$, we define their *intersection* $\mathcal{T} \triangleq \wedge_{j \in J} \mathcal{T}_j$ as follows: by the universal property of $\mathcal{U}^{st}_{\mathcal{A}}$, for every $\mathcal{T}_j$ there exists a unique $\psi_j : \mathcal{T}_j \to \mathcal{U}^{st}_{\mathcal{A}}$ such that $f_j = f_{\mathcal{A}} \circ \psi_j$. We then take $\mathcal{T} = \cap_{j \in J} \psi_j(\mathcal{T}_j)$: intersection in the sense of sub-trellis automata of $\mathcal{U}^{st}_{\mathcal{A}}$. Naturally there exists injective morphisms $\phi_j : \mathcal{T} \to \mathcal{T}_j$, and $\mathcal{T}$ is also the union of configurations κ of $\mathcal{U}^{st}_{\mathcal{A}}$ such that there exist injective morphisms $\phi_j : \kappa \to \mathcal{T}_j$ satisfying $f_j \circ \phi_j = f_{\mathcal{A}|\kappa}$ ($f_{\mathcal{A}}$ restricted to κ).

Let $\mathcal{T} \subseteq \mathcal{T}_1 \times_T \ldots \times_T \mathcal{T}_K$ where $\mathcal{T}_j$ is a TP of $\mathcal{A}_j$, and let $J \subseteq \{1, \ldots, K\}$. In the sequel, taking advantage of the associativity of $\times_T$, we denote by $\mathcal{T}_J$ the product $\times_{T, j \in J} \mathcal{T}_j$, and by $\pi_J(\mathcal{T})$ the projection of $\mathcal{T}$ on $\mathcal{T}_J$. For $L \subseteq J$, one naturally has $\pi_L \circ \pi_J = \pi_L$. We also define $\mathcal{A}_J \triangleq \times_{j \in J} \mathcal{A}_j$ and accordingly $\Lambda_J \triangleq \cup_{j \in J} \Lambda_j$.

B. Modular computations

Formally, the distributed diagnosis problem amounts to the following: Let $\mathcal{T}$ be defined as $\mathcal{T} = \mathcal{T}_1 \times_T \ldots \times_T \mathcal{T}_K$, where $\mathcal{T}_j$ is a TP of $\mathcal{A}_j$, one wishes to determine the minimal product covering of $\mathcal{T}$, *i.e.* the $\mathcal{T}'_j = \pi_j(\mathcal{T})$.

Let $L, M, N \subseteq \{1, \ldots, K\}$, with empty pairwise intersections, we say that $\mathcal{A}_M$ separates $\mathcal{A}_L$ from $\mathcal{A}_N$ iff $\Lambda_L \cap \Lambda_N \subseteq \Lambda_M$, and denote it graphically by $\mathcal{A}_L - \mathcal{A}_M - \mathcal{A}_N$. The separation has the following important consequences

Theorem 3: Let $\mathcal{T}_L, \mathcal{T}_M, \mathcal{T}_N$ be TP of $\mathcal{A}_L, \mathcal{A}_M, \mathcal{A}_N$ respectively, and assume $\mathcal{A}_L - \mathcal{A}_M - \mathcal{A}_N$. Then

$$\pi_M(\mathcal{T}_{L \cup M \cup N}) \quad = \quad \pi_M(\mathcal{T}_{L \cup M}) \wedge \pi_M(\mathcal{T}_{M \cup N}) \qquad (9)$$

$$\pi_L(\mathcal{T}_{L \cup M \cup N}) \quad = \quad \pi_L(\mathcal{T}_L \times_T \pi_M(\mathcal{T}_{M \cup N})) \qquad (10)$$

Proof (sketch of). For the merge relation (9), observe first that $\pi_{L \cup M}(\mathcal{T}_{L \cup M \cup N}) \subseteq \mathcal{T}_{L \cup M}$, so $\pi_M(\mathcal{T}_{L \cup M \cup N}) \subseteq \pi_M(\mathcal{T}_{L \cup M})$, and symmetrically $\pi_M(\mathcal{T}_{L \cup M \cup N}) \subseteq \pi_M(\mathcal{T}_{M \cup N})$. Together, these inequalities yield $\pi_M(\mathcal{T}_{L \cup M \cup N}) \subseteq \pi_M(\mathcal{T}_{L \cup M}) \wedge \pi_M(\mathcal{T}_{M \cup N})$.

For the converse inclusion, we rely on lemma 3. Let κ_M be a configuration of $\pi_M(\mathcal{T}_{L \cup M}) \wedge \pi_M(\mathcal{T}_{M \cup N})$. Since κ_M is a configuration of $\pi_M(\mathcal{T}_{L \cup M})$, there exists a configuration $\kappa' = \kappa_L \times_T \kappa_M$ in $\mathcal{T}_{L \cup M}$ such that $\pi_L(\kappa') = \kappa_L$ and $\pi_M(\kappa') = \kappa_M$. Similarly there exists a configuration

$\kappa'' = \kappa_M \times_T \kappa_N$ in $\mathcal{T}_{M\cup N}$ such that $\pi_M(\kappa'') = \kappa_M$ and $\pi_N(\kappa'') = \kappa_N$. Now consider $\kappa = \kappa_L \times_T \kappa_M \times_T \kappa_N$ in $\mathcal{T}_{L\cup M\cup N}$, we want to prove that $\pi_M(\kappa) = \kappa_M$. In general, $\pi_M(\kappa)$ is a strict prefix of κ_M, so we have to show that all events of κ_M are preserved by the product. The product $\kappa_L \times_T \kappa_M \times_T \kappa_N$ synchronizes these three sequences of events on their shared labels, in a unique manner, and computes all possible interleavings of events between synchronization points (this effect is illustrated in fig. 6). The synchronization points are the events labeled by $(\Lambda_L \cap \Lambda_M) \cup (\Lambda_M \cap \Lambda_N) \cup (\Lambda_L \cap \Lambda_N)$, which is included in Λ_M by assumption. So the order in which synchronization points appear in κ_M is identical to the order in which they appear in $\kappa_L \times_T \kappa_N$, whence the result $\pi_M(\kappa) = \kappa_M$ (see fig. 7).

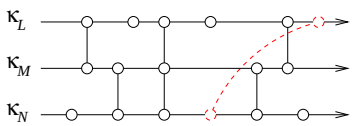

Fig. 7. *Three configurations (sequences of events) and their events (patches). Synchronous events are connected. A direct synchronization between κ_L and κ_N(red dashed line) is forbidden.*

The proof of the propagation relation (10) is based on the same ideas. □

We now show how these two properties can be combined into a message passing algorithm taking the $\mathcal{T}_j$ as input and providing the $\mathcal{T}'_j$ as output. We consider the example of a "Markov chain" system $\mathcal{A}_1 - \mathcal{A}_2 - \ldots - \mathcal{A}_K$, which means that every component $\mathcal{A}_j$ separates $\mathcal{A}_{\{1,\ldots,j-1\}}$ from $\mathcal{A}_{\{j+1,\ldots,K\}}$.

Let us define/compute recursively the following *messages* between neighboring components in the chain:

$$\mathcal{M}_1^+ = \mathcal{T}_1, \qquad \mathcal{M}_j^+ = \pi_j(\mathcal{T}_j \times_T \mathcal{M}_{j-1}^+), \quad {\scriptstyle 2\le j\le K}$$
$$\mathcal{M}_K^- = \mathcal{T}_K, \qquad \mathcal{M}_j^- = \pi_j(\mathcal{T}_j \times_T \mathcal{M}_{j+1}^-), \quad {\scriptstyle 1\le j\le K-1}$$

Using the propagation equation (10), one obviously has

$$\mathcal{M}_j^+ = \pi_j(\mathcal{T}_1 \times_T \ldots \times_T \mathcal{T}_j)$$
$$\mathcal{M}_j^- = \pi_j(\mathcal{T}_j \times_T \ldots \times_T \mathcal{T}_K)$$

Once these forward and backward recursions are completed, the merge equation (9) allows to conclude:

$$\mathcal{T}'_j = \pi_j(\mathcal{T}_1 \times_T \ldots \times_T \mathcal{T}_K) = \mathcal{M}_j^+ \wedge \mathcal{M}_j^-$$

It is remarkable that only local computations are involved in this procedure, which allows to deal with arbitrarily long chains of systems, with a linearly growing complexity. Observe that $\mathcal{T} = \mathcal{T}_1 \times_T \ldots \times_T \mathcal{T}_K$ is never computed. This approach generalizes to any type of interaction graph, including graphs with cycles. Convergence can be proved in any case, but the resulting $\mathcal{T}'_j$ obtained by merging messages can be proved to be the desired projection only in the case of trees. In general however, the $\mathcal{T}'_j$ obtained at convergence contain the true projections $\pi_i(\mathcal{T})$, and still satisfy $\mathcal{T} = \mathcal{T}'_1 \times_T \ldots \times_T \mathcal{T}'_K$ [17].

VI. Conclusion

We have proposed a strategy to deal with large DEDS, viewed as networks of automata. The central idea is an appropriate representation of their sets of runs. Several results were already available in the framework of true concurrency semantics, but they are rather involved technically, and were only published by parts. When moving to the more standard sequential semantics, many simplifications take place, that allow a more comprehensive study. We have shown here that the idea of time-unfolding (or trellis) [16] still applies, that factorization properties still hold and lead to distributed processings. Naturally, this approach is only relevant when the interactions between components remain sparse, in order to get a simple interaction graph (ideally a tree). A longer version of this work is in preparation, including distributed optimization procedures (in the case of stochastic systems), and describing as well recursive algorithms. The latter process observations on-line and thus take the form of cooperating Viterbi algorithms running for each component.

References

[1] S. Mac Lane, Categories for the Working Mathematician, Springer-Verlag, 1971.

[2] P. Baroni, G. Lamperti, P. Pogliano, M. Zanella, Diagnosis of Large Active Systems, Artificial Intell. 110, pp. 135-183, 1999.

[3] 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.

[4] R.K. Boel, G. Jiroveanu, Distributed Contextual Diagnosis for very Large Systems, in proc. of WODES'04, pp. 343-348, 2004.

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

[6] O. Contant, S. Lafortune, Diagnosis of Modular Discrete Event Systems, in proc. of WODES'04, pp. 337-342, 2004

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] R. Su, Distributed Diagnosis for Discrete-Event Systems, PhD thesis, Dept. of Elec. and Comp. Eng., Univ. of Toronto, June 2004.

[13] 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.

[14] 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.

[15] E. Fabre, A. Benveniste, S. Haar, C. Jard, Distributed Monitoring of Concurrent and Asynchronous Systems, Journal of Discrete Event Systems, special issue, vol. 15 no. 1, pp. 33-84, March 2005.

[16] E. Fabre, Distributed diagnosis based on trellis processes, in proc. Conf. on Decision and Control, Sevilla, Dec. 2005, pp. 6329-6334.

[17] E. Fabre, Convergence of the turbo algorithm for systems defined by local constraints, Irisa research report no. PI 1510, May 2003.