Diagnosis of Discrete-Event Systems with Model-Based Prospection Knowledge

Roberto Garatti and **Gianfranco Lamperti** and **Marina Zanella** [1]

Abstract. Diagnosis of discrete-event systems is a complex and challenging task. Within the class of *active systems*, such a complexity is exacerbated by the possibility of queuing (possibly *uncertain*) events within connection links, thereby making essential the simulation of link behavior during the reconstruction of the system reaction. However, reconstructing such a reaction without any prospection in the search space is generally bound to detrimental backtracking. To cope with this 'short-sightedness', we present a technique which allows the automatic generation of prospection knowledge relevant to the mode in which events are produced and consumed in links. Such a 'far-sighted' diagnosis requires that a collection of *prospection graphs* be generated off-line, based on the system model, which are then exploited on-line to guide the search process. As a result, both time and space can be considerably reduced on-line. The approach is worthwhile whenever time constraints are far more severe on-line (when the diagnostic engine is running) than off-line (when no diagnostic process is ongoing), which is commonplace in a large variety of real systems.

1 INTRODUCTION

Discrete-event systems (DESs) are an important class of dynamic systems that has been receiving an increasing interest from both the model-based diagnosis and the FDI community. The current shared prospect about diagnosis of DESs is that, in the general case, the specific faults cannot be inferred without first finding out what has happened to the system to be diagnosed. Once the system evolution is available, the sets of candidate faults can be derived from it.

In this respect, in spite of slightly different terminologies, such as histories [1], situation histories or narratives [3], paths [4], and trajectories [5], all the distinct approaches describe the evolution of a DES as a sequence interleaving states and transitions, as the favorite behavioral models of DESs in the literature are automata.

Based on the method for tracking the evolutions of the system that explain a given observation, two broad categories of approaches to diagnosis of DESs can be basically singled out: (i) those that first generate (a model of) all possible evolutions and then retrieve only the evolutions that explain the observation, and (ii) those that generate in one shot the evolutions explaining the observation. The first category includes the automatic chronicle generation approach [7] and some relevant works in the automatic control area [14, 15, 6, 11]. In the second category there fall two approaches in the AI area [1, 12].

Since finding out the system evolutions is a computationally expensive and, therefore, inefficient process (see, for instance, [13] about the computational difficulties of the diagnoser approach [14, 15], or the worst case computational complexity analysis in [1]), most of the approaches exploit a trade-off between off-line and on-line computation.

Focusing on the second category outlined above, the decentralized diagnoser approach [12] draws off-line a local diagnoser for each component. Such a diagnoser is an automaton whose states and (observable) transitions are labeled with compiled knowledge about unobservable paths and interacting components, respectively. Each local diagnoser is employed on-line for both a more efficient reconstruction of all the possible evolutions of the relevant component that comply with the observation and a more efficient merging of the histories of distinct components into global system histories.

This paper applies knowledge compilation to the active system approach [1, 2], to which purpose it isolates a kind of knowledge, implicit in the models of the structure and behavior of the system at hand, that can be compiled off-line in order to speed up on-line execution. The framework is that of active systems, a class of DESs modeled as networks of nondeterministic automata communicating through asynchronous buffered directed links. The reaction of a system to an event coming from the external world is assumed to continue until there is no event left in the links. The component that sends events on a link is the event *producer* and that extracting them from the link is the *consumer*. The knowledge we compile is actually that inherent to the producer-consumer relationships between components. In particular, after surveying the modeling primitives and the basics of the evolution reconstruction method, we present, by means of an example, (1) a method for generating off-line, under the form of a deterministic automaton, called a *prospection graph*, the model of the way events are exchanged over one or more links, and (2) a method for exploiting prospection graphs on-line while reconstructing the evolutions of (sub)systems. Finally, the computational advantages of the proposal are discussed and some conclusions are hinted.

2 BACKGROUND

Topologically, an active system Σ is a network of *components* which are connected to one another through *links*. Each component is completely modeled by an automaton that reacts to events either coming from the external world or from neighboring components through links. Formally, the automaton is a 6-tuple $(\mathbb{S}, \mathbb{E}_{\text{in}}, \mathbb{I}, \mathbb{E}_{\text{out}}, \mathbb{O}, \mathbb{T})$, where $\mathbb{S}$ is the set of *states*, $\mathbb{E}_{\text{in}}$ the set of *input events*, $\mathbb{I}$ the set of *input terminals*, $\mathbb{E}_{\text{out}}$ the set of *output events*, $\mathbb{O}$ the set of *output terminals*, and $\mathbb{T}$ the (nondeterministic) *transition function*. In addition, components are implicitly equipped with three *virtual terminals*, the *standard input* (In) for events coming from the external world, the *standard output* (Out) for events directed toward the

[1] Dipartimento di Elettronica per l'Automazione, Università di Brescia, via Branze 38, 25123 Brescia, Italy, email: garatrob@tin.it, lamperti@ing.unibs.it, zanella@ing.unibs.it

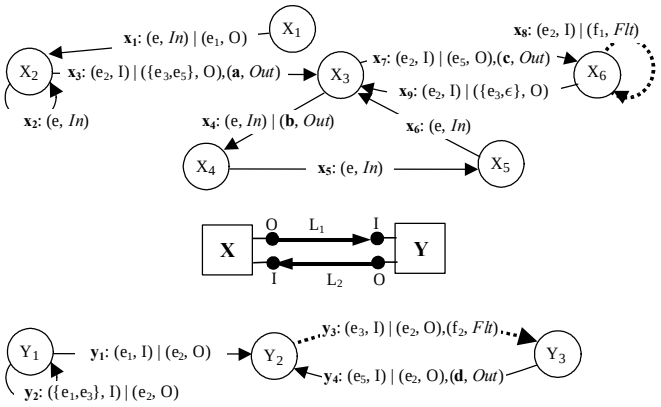

Figure 1. System Σ and models of components X (top) and Y (bottom).

external world (messages), and the *fault terminal* (*Flt*) for modeling faulty transitions. An event may be *uncertain* in nature, that is, represented by a disjunction of possible values, one of which is nondeterministically generated. Links are the means of storing the events exchanged between components. Each link L is characterized by a 4-tuple (I, O, χ, P), where I is the *input terminal*, O the *output terminal*, χ the *capacity*, that is, the maximum number of queued events, and P the *saturation policy*, which dictates the effect of the triggering of a transition T attempting to insert a new event E into L when L is *saturated* (full). Three case are possible: (1) *LOSE*: E is lost, (2) *OVERRIDE*: E overrides the last event in the queue of events of L, or (3) *WAIT*: T cannot be triggered until L becomes unsaturated, that is, until at least one event in L is consumed. The queue of events incorporated in L and the length of such a queue are denoted by $\|L\|$ and $|L|$, respectively. If $\alpha = (E, \theta)$ is an event relevant to a terminal θ, $Link(\alpha)$ denotes the link leaving θ. Initially, Σ is in a *quiescent* state Σ_0, wherein all links are empty. On the arrival of an event from the external world, Σ becomes *reacting*, thereby making a series of transitions until a final quiescent state is reached, wherein all links are empty anew. This *reaction* yields a sequence of observable events, the *messages*, which make up a system *observation* $OBS(\Sigma)$. Based on a *diagnostic problem* $\wp(\Sigma) = (OBS(\Sigma), \Sigma_0)$, a *reconstruction* of the system reaction is carried out, which yields an *active space*, that is, a graph representing the whole set of *candidate histories* that explain $OBS(\Sigma)$, each history being a path from Σ_0 to a final state, in other terms, a sequence of component transitions. Candidate *diagnoses* are eventually distilled from the active space, each diagnosis being a set of faulty components, that is, those components which performed at least one *faulty transition* during a candidate system history.

Example 1. In the center of Figure 1 a system Σ is displayed, where X and Y are components, while L_1 and L_2 are links. Both components are endowed with an input terminal I and an output terminal O. For both links we assume capacity = 1 and saturation policy = $WAIT$. The behavioral models of X and Y are displayed on the top and on the bottom, respectively. Accordingly, Y involves three states ($Y_1 \cdots Y_3$) and four transitions ($y_1 \cdots y_4$), one of which is faulty (y_3). For instance, transition y_4 is triggered by the input event (e_5, I) and generates the set of output events $\{(e_2, O), (d, Out)\}$, where the former is directed toward X on link L_2, while the latter is a message labeled d (y_4 is *observable*). Messages are bold in the figure. Transition y_2 involves the input uncertain event $(\{e_1, e_3\}, I)$, meaning that y_2 may either be triggered by e_1 or e_3. Considering the model of X, note that, when triggered, transition x_9 generates the uncertain event $(\{e_3, \epsilon\}, O)$, meaning that either e_3 or nothing is nondeterministically generated.

3 SHORT-SIGHTED DIAGNOSIS

The main task relevant to the resolution of a diagnostic problem $\wp(\Sigma) = (OBS(\Sigma), \Sigma_0)$ is the reconstruction of the system reaction to make up the relevant active space $Act(\wp(\Sigma))$. A node N in the search space is identified by three fields, $N = (\sigma, \Im, \mathcal{Q})$, where:

- $\sigma = (S_1, \ldots, S_n)$ is the record of states of the system components, where each S_i, $i \in [1\,..\,n]$, is a state relevant to a component C_i in Σ (n is the number of components in Σ);
- $\Im$ is the *index* of $OBS(\Sigma)$, that is, an integer ranging from 0 to the number of messages (length) of $OBS(\Sigma)$;
- $\mathcal{Q} = (Q_1, \ldots, Q_q)$ is the record of queues of the q links of Σ.

Node N is said to be *final* when $\Im$ equals the length of $OBS(\Sigma)$ and all links are empty. The search for the nodes of the active space is started at the initial node $N_0 = (\Sigma_0, 0, (\emptyset, \ldots, \emptyset))$. Each successor node of a given node is obtained by applying a component transition that is consistent with both the system topology and the observation. An applied transition is an edge of the search space. Nodes and edges are stored in variables $\aleph$ and $\mathcal{E}$, respectively. When the reconstruction process is carried out in one step (monolithically) without any prospection knowledge (*short-sightedly*), it can be described by Algorithm 1.

Algorithm 1. (*Short-sighted Reconstruction*)

1. $\aleph = \{N_0\}$; $\mathcal{E} = \emptyset$; (N_0 is unmarked)
2. *Repeat Steps 3 through 5 until all nodes in* $\aleph$ *are marked;*
3. *Get an unmarked node* $N = (\sigma, \Im, \mathcal{Q})$ *in* $\aleph$*;*
4. *For* i *in* $[1\,..\,n]$*, for each transition* T *within the model of component* C_i*, if* T *is triggerable, that is, if its triggering event is available and* T *is consistent with both* $OBS(\Sigma)$ *and the link policy (when* T *generates output events on non-virtual terminals), do the following steps:*
   (a) *Create a copy* $N' = (\sigma', \Im', \mathcal{Q}')$ *of* N*;*
   (b) *Set* $\sigma'[i]$ *to the state reached by* T*;*
   (c) *If* T *is observable, then increment* $\Im'$*;* (a message is generated)
   (d) *If the triggering event* E *of* T *is relevant to an internal link* L_j*, then remove* E *from* $\mathcal{Q}'[j]$*;*
   (e) *Insert the internal output events of* T *into the relevant queues in* $\mathcal{Q}'$*;*
   (f) *If* $N' \notin \aleph$ *then insert* N' *into* $\aleph$*;* (N' is unmarked)
   (g) *Insert edge* $N \xrightarrow{T} N'$ *into* $\mathcal{E}$*;*
5. *Mark* N*;*
6. *Remove from* $\aleph$ *all the nodes and from* $\mathcal{E}$ *all the edges that are not on a path from the initial state* N_0 *to a final state in* $\aleph$*.*

When uncertain events are involved, several new nodes N' are to be generated for the same transition T, specifically, one for each combination of possible values within each disjunction. For example, since transition x_3 in Figure 1 involves the uncertain output event $(\{e_3, e_5\}, O)$, two target nodes will be generated, one for e_3 and one for e_5. If the set of input and output events included several uncertain events, all possible combinations would be required to be enumerated. The triggering event of a transition, even if uncertain, cannot include the null event ϵ.

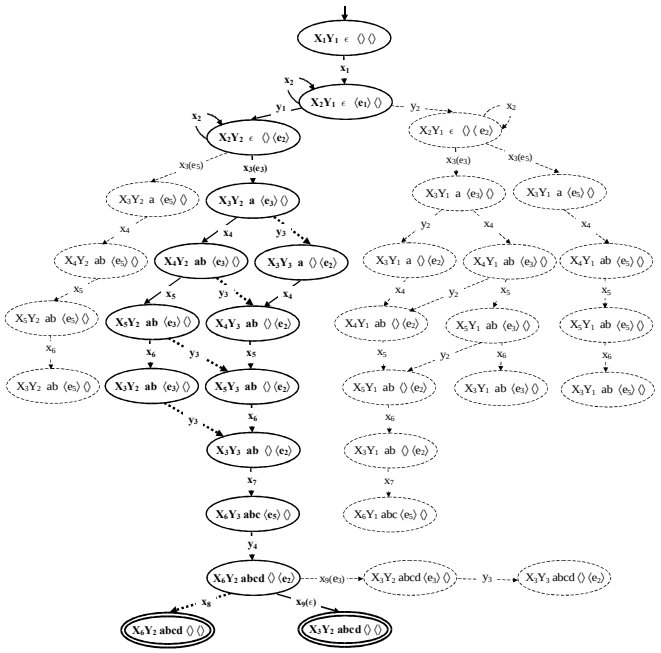

Figure 2. Short-sighted reconstruction space (see Example 2).

Example 2. Shown in Figure 2 is the reconstruction space generated short-sightedly for the diagnostic problem $\wp(\Sigma) = (OBS(\Sigma), \Sigma_0)$, where Σ is the system outlined in Figure 1, $OBS(\Sigma) = \langle a, b, c, d\rangle$, and $\Sigma_0 = (X_1, Y_1)$. Each node is depicted by an ellipse, wherein the fields σ, $\Im$, and $\mathcal{Q}$ are the pair of component states (X_i, Y_j), the prefix of the observation (instead of the index value), and the pair of link queues $(\|L_1\|, \|L_2\|)$, respectively. Edges are marked by the corresponding component transitions, possibly qualified by the relevant chosen label when the involved output event is uncertain. Dotted edges denote faulty transitions. Final nodes are depicted as double ellipses. The dashed part of the graph represents the inconsistent states, which are almost 60 percent of the search space. Owing to cycles in the graph (edges marked by x_2), the active space incorporates an unbound number of candidate histories. However, only two candidate diagnoses are possible, $\{Y\}$ and $\{X, Y\}$. That is, Y is certainly faulty, while X might be faulty or not.

4 FAR-SIGHTED DIAGNOSIS

The essential problem with short-sighted diagnosis lies in the lack of any prospection in the search space as to the consistency of the link queues. In other words, the inability to understand that a given configuration of $\mathcal{Q}$ is bound to a 'blind alley' forces the reconstruction algorithm to uselessly explore possibly large parts of the search space. In order to overcome this limitation, prospection knowledge can be automatically generated off-line based on the system model. Considering Figure 2, such a knowledge will allow us to avoid entering the inconsistent sub-space on the right through y_2.

The basic idea is to view a link L as a buffer in which a *producer* component C^{p} generates events that are consumed by a *consumer* component C^{c}. That is, L connects an output terminal of C^{p} to an input terminal of C^{c}. The way events are produced and consumed in L is both constrained by the characteristics of the link (capacity and saturation policy) and the models of C^{p} and C^{c}.

4.1 Prospection graphs

Let $L = (I, O, \chi, P)$ be a link from output terminal O^{p} of component C^{p} to input terminal I^{c} of component C^{c}. Let $M^{\mathrm{p}} = (\mathbb{S}^{\mathrm{p}}, \mathbb{E}^{\mathrm{p}}_{\mathrm{in}}, \mathbb{I}^{\mathrm{p}}, \mathbb{E}^{\mathrm{p}}_{\mathrm{out}}, \mathbb{O}^{\mathrm{p}}, \mathbb{T}^{\mathrm{p}})$ and $M^{\mathrm{c}} = (\mathbb{S}^{\mathrm{c}}, \mathbb{E}^{\mathrm{c}}_{\mathrm{in}}, \mathbb{I}^{\mathrm{c}}, \mathbb{E}^{\mathrm{c}}_{\mathrm{out}}, \mathbb{O}^{\mathrm{c}}, \mathbb{T}^{\mathrm{c}})$ be the models of C^{p} and C^{c}, respectively.

Let $\hat{M}^{\mathrm{p}} = (\hat{\mathbb{S}}^{\mathrm{p}}, \hat{\mathbb{E}}^{\mathrm{p}}_{\mathrm{in}}, \hat{\mathbb{I}}^{\mathrm{p}}, \hat{\mathbb{E}}^{\mathrm{p}}_{\mathrm{out}}, \hat{\mathbb{O}}^{\mathrm{p}}, \hat{\mathbb{T}}^{\mathrm{p}})$ be the automaton obtained from M^{p} by replacing each transition $T = S \xrightarrow{\alpha \mid \beta} S' \in \mathbb{T}^{\mathrm{p}}$ such that $L \notin \{L_\beta \mid L_\beta = Link(B), B \in \beta\}$ with an ϵ-transition and transforming the obtained nondeterministic automaton into an equivalent deterministic one. Likewise, let $\hat{M}^{\mathrm{c}} = (\hat{\mathbb{S}}^{\mathrm{c}}, \hat{\mathbb{E}}^{\mathrm{c}}_{\mathrm{in}}, \hat{\mathbb{I}}^{\mathrm{c}}, \hat{\mathbb{E}}^{\mathrm{c}}_{\mathrm{out}}, \hat{\mathbb{O}}^{\mathrm{c}}, \hat{\mathbb{T}}^{\mathrm{c}})$ be the automaton obtained from M^{c} by replacing each transition $T = S \xrightarrow{\alpha \mid \beta} S' \in \mathbb{T}^{\mathrm{c}}$ such that $L \neq Link(\alpha)$ with an ϵ-transition and transforming the obtained nondeterministic automaton into an equivalent deterministic one. $\hat{M}^{\mathrm{p}}$ and $\hat{M}^{\mathrm{c}}$ are called the *prospection models* of C^{p} and C^{c}, respectively.

A *prospection state* of L is a triple $\mathcal{L} = (\hat{S}^{\mathrm{p}}, \hat{S}^{\mathrm{c}}, \|L\|)$, where $\hat{S}^{\mathrm{p}} \in \hat{\mathbb{S}}^{\mathrm{p}}$, $\hat{S}^{\mathrm{c}} \in \hat{\mathbb{S}}^{\mathrm{c}}$. Let $\mathcal{L}$ be a prospection state and $\hat{S} \xrightarrow{T} \hat{S}' \in (\hat{\mathbb{T}}^{\mathrm{p}} \cup \hat{\mathbb{T}}^{\mathrm{c}})$, $\hat{S} \in \{\hat{S}^{\mathrm{p}}, \hat{S}^{\mathrm{c}}\}$, $T = S \xrightarrow{\alpha \mid \beta} S' \in (\mathbb{T}^{\mathrm{p}} \cup \mathbb{T}^{\mathrm{c}})$. Let $Head(L)$ denote the first consumable element in $\|L\|$, $Tail(L)$ the queue of events in $\|L\|$ following the first event, $App(L, e)$ the queue of event obtained by appending e to $\|L\|$, and $Repl(L, e)$ the queue of events obtained by replacing the last event in $\|L\|$ with e. The $Next$ function yields the set of next prospection states as follows:

$$\begin{aligned} Next(\mathcal{L}, T) \quad \stackrel{\mathrm{def}}{=} \quad & \{\mathcal{L}' \mid \mathcal{L}' \in Next^{\mathrm{p}}(\mathcal{L}, T), T \in \mathbb{T}^{\mathrm{p}}\} \cup \\ & \{\mathcal{L}' \mid \mathcal{L}' \in Next^{\mathrm{c}}(\mathcal{L}, T), T \in \mathbb{T}^{\mathrm{c}}\} \end{aligned}$$

where

$$\begin{aligned} Next^{\mathrm{p}}(\mathcal{L}, T) \stackrel{\mathrm{def}}{=} \{\mathcal{L}' \mid \mathcal{L}' = (\hat{S}', \hat{S}^{\mathrm{c}}, \|L\|'), B = (E, O^{\mathrm{p}}) \in \beta, \\ e \in E, \|L\|' = Ins(L, e), (|L| < \chi \text{ or} \\ (|L| = \chi, (e = \epsilon \text{ or } P \in \{LOSE, OVERRIDE\})))\}, \end{aligned}$$

$$Ins(L, e) \stackrel{\mathrm{def}}{=} \begin{cases} App(L, e) & \text{if } |L| < \chi \\ \|L\| & \text{if } |L| = \chi, (e = \epsilon \text{ or } P = LOSE) \\ Repl(L, e) & \text{if } |L| = \chi, P = OVERRIDE \end{cases}$$

and

$$\begin{aligned} Next^{\mathrm{c}}(\mathcal{L}, T) \stackrel{\mathrm{def}}{=} \{\mathcal{L}' \mid \mathcal{L}' = (\hat{S}^{\mathrm{p}}, \hat{S}', \|L\|'), \\ \alpha = (E, I^{\mathrm{c}}), e \in E, Head(L) = e, \|L\|' = Tail(L)\}. \end{aligned}$$

Let $\mathcal{C}_0 = (S^{\mathrm{p}}_0, S^{\mathrm{c}}_0)$ be the pair of initial states for C^{p} and C^{c}, respectively. The *spurious prospection graph* of L and $\mathcal{C}_0$ is the nondeterministic automaton $\tilde{\mathcal{P}}^{\mathrm{n}}(L, \mathcal{C}_0) = (\tilde{\mathbb{S}}^{\mathrm{n}}, \mathbb{E}^{\mathrm{n}}, \tilde{\mathbb{T}}^{\mathrm{n}}, S^{\mathrm{n}}_0, \mathbb{S}^{\mathrm{n}}_{\mathrm{f}})$, where

$\tilde{\mathbb{S}}^{\mathrm{n}} = \{\mathcal{L} \mid \mathcal{L}$ is a prospection state of $L\}$ is the set of states,
$\mathbb{E}^{\mathrm{n}} \subseteq \hat{\mathbb{E}}^{\mathrm{p}} \cup \hat{\mathbb{E}}^{\mathrm{c}} \subseteq \mathbb{T}^{\mathrm{p}} \cup \mathbb{T}^{\mathrm{c}}$ is the set of events,
$S^{\mathrm{n}}_0 = (S^{\mathrm{p}}_0, S^{\mathrm{c}}_0, \langle\rangle)$ is the initial state,
$\mathbb{S}^{\mathrm{n}}_{\mathrm{f}} = \{\mathcal{L} \mid \mathcal{L} \in \tilde{\mathbb{S}}^{\mathrm{n}}, \mathcal{L} = (S^{\mathrm{p}}, S^{\mathrm{c}}, \langle\rangle)\}$ is the set of final states,
$\tilde{\mathbb{T}}^{\mathrm{n}} : \tilde{\mathbb{S}}^{\mathrm{n}} \times \mathbb{E}^{\mathrm{n}} \mapsto 2^{\tilde{\mathbb{S}}^{\mathrm{n}}}$ is the transition function defined as follows:

$$\mathcal{L} \xrightarrow{T} \mathcal{L}' \in \tilde{\mathbb{T}}^{\mathrm{n}} \text{ iff } \mathcal{L}' \in Next(\mathcal{L}, T).$$

A state of $\tilde{\mathcal{P}}^{\mathrm{n}}(L, \mathcal{C}_0)$ which is not within a path from the initial state to a final state is an *inconsistent state*. Similarly, a transition either entering or leaving an inconsistent state is an *inconsistent transition*. The *nondeterministic prospection graph* of L and $\mathcal{C}_0$ is the nondeterministic automaton $\mathcal{P}^{\mathrm{n}}(L, \mathcal{C}_0) = (\mathbb{S}^{\mathrm{n}}, \mathbb{E}^{\mathrm{n}}, \mathbb{T}^{\mathrm{n}}, S^{\mathrm{n}}_0, \mathbb{S}^{\mathrm{n}}_{\mathrm{f}})$ obtained

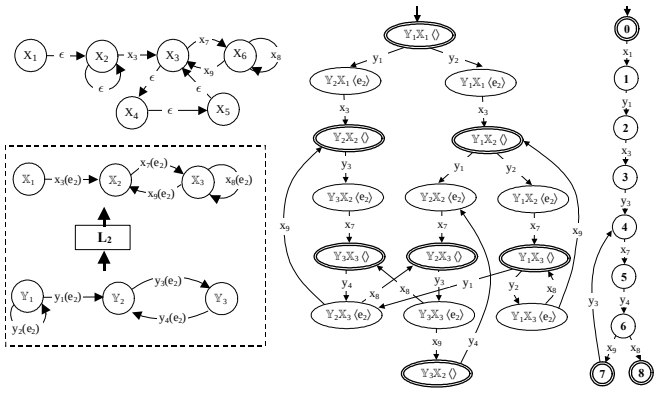

Figure 3. Generation of prospection graphs (see Example 3).

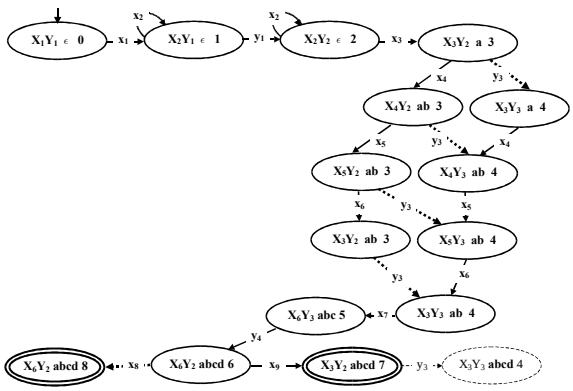

Figure 4. Far-sighted reconstruction space (see Example 4).

from $\tilde{P}^{\mathrm{n}}(L, \mathcal{C}_0)$ by removing inconsistent states and inconsistent transitions. The *prospection graph* $\mathcal{P}(L, \mathcal{C}_0)$ is the deterministic automaton $(\mathbb{S}, \mathbb{E}, \mathbb{T}, S_0, \mathbb{S}_{\mathrm{f}})$ equivalent to $\mathcal{P}^{\mathrm{n}}(L, \mathcal{C}_0)$.

The notion of a prospection graph can be naturally extended to that of a set of links $\mathbb{L}$ connecting a set $\mathbb{C}$ of components, namely $\mathcal{P}(\mathbb{L}, \mathbb{C}_0)$, where $\mathbb{L} = \{L_1, \ldots, L_{\mathrm{n}}\}$ and $\mathbb{C}_0$ is the set of initial states for components in $\mathbb{C}$.

Example 3. Depicted on the left of Figure 3 is the prospection graph $\mathcal{P}(L_2, (Y_1, X_1))$, where double ellipses denote final states. Essentially, the generation of such a graph is analogous to the generation of an active space, where (1) component models are substituted by prospection models, (2) only one link is considered, and (3) no observation index is considered. Displayed on the right-hand side of Figure 3 is the prospection graph $\mathcal{P}(\{L_1, L_2\}, \{X_1, Y_1\})$, which can be obtained either directly, based on the generalized definition of prospection graph, or by combining $\mathcal{P}(L_1, (X_1, Y_1))$ and $\mathcal{P}(L_2, (Y_1, X_1))$ appropriately.

Given a system Σ, in order to perform a far-sighted reconstruction, which is a variation of Algorithm 1, we need to create a set of prospection graphs $\mathbb{P}(\Sigma) = \{\mathcal{P}(\mathbb{L}_1, \mathbb{C}_{0_1}), \ldots, \mathcal{P}(\mathbb{L}_\ell, \mathbb{C}_{0_\ell})\}$ such that $\bigcup_{i=1}^{\ell} \mathbb{L}_i$ equals the whole set of links in Σ. $\mathbb{P}(\Sigma)$ is a *prospection coverage* of Σ.

Algorithm 2. (*Far-sighted Reconstruction*)

The first three statements are the same as in Algorithm 1. However, the $\mathcal{Q}$ field of each reconstruction space state here denotes a record of ℓ states relevant to the ℓ prospection graphs in the prospection coverage $\mathbb{P}(\Sigma)$, namely $\mathcal{Q} = (p_1, \ldots, p_\ell)$. Moreover, the third field of the initial node N_0 is the record of the initial states of the corresponding prospection graphs, namely $(p_{0_1}, \ldots, p_{0_\ell})$. Finally, Step 4 of Algorithm 1 is changed as follows:

For each i in $[1\,..\,n]$, for each transition T within the model of component C_i, if T is triggerable, that is, if the following two conditions hold

(i) T is consistent with $OBS(\Sigma)$;
Let $\Pi(T) = \{\bar{\mathcal{P}}_1, \ldots, \bar{\mathcal{P}}_r\}$ be the prospection graphs in $\mathbb{P}(\Sigma)$ that are relevant to links connected with terminals on which events are either consumed or generated by T; let $\bar{\mathcal{Q}}(N) = \{\bar{p}_1, \ldots, \bar{p}_r\}$ be the elements of $\mathcal{Q}(N)$ relevant to $\Pi(T)$:

(ii) $\forall j \in [1\,..\,r]$ ($\bar{p}_j \xrightarrow{T} \bar{p}'_j$ is an edge in $\bar{\mathcal{P}}_j$),

then do the following steps:

(a) Create a copy $N' = (\sigma', \Im', \mathcal{Q}')$ of N;

(b) Set $\sigma'[i]$ to the state reached by T;

(c) If T is observable, then increment $\Im'$;

(d) Replace the elements of $\mathcal{Q}'$ relevant to $\bar{\mathcal{Q}}(N)$ with the new prospection states;

(e) If $N' \notin \aleph$ then insert N' into $\aleph$; (N' is unmarked)

(f) Insert edge $N \xrightarrow{T} N'$ into $\mathcal{E}$.

Essentially, Algorithm 2 exploits the knowledge about the consistency of link states by means of the prospection graphs generated off-line. Although such a prospection is finite, thereby not eliminating completely the backtracking, it prevents the search from entering (possibly large) inconsistent parts of the space. Besides, it allows for an efficient treatment of nondeterminism caused by uncertain events. Recall that, in short-sighted reconstruction, such situations can only be dealt with by mere enumeration of all possible new link states generated by the collection of input and output events of the current transition. For example, if T generated 3 uncertain events (on three different links), each of which represented by a disjunction of 2 values, then we would have 8 new nodes. Instead, since the prospection graphs are deterministic, with far-sighted reconstruction only one new node is generated, as at most one edge marked by T can leave each current state of the prospection graphs.

Example 4. Shown in Figure 4 is the reconstruction space for the diagnostic problem $\wp(\Sigma) = (\langle a, b, c, d\rangle, (X_1, Y_1))$ based on the prospection graph outlined in Figure 3. It is striking comparing it with the short-sighted reconstruction (based on Algorithm 1) displayed in Figure 2. While the number of consistent states (15) is necessarily equal in both reconstructions, the far-sighted reconstruction space includes one inconsistent state only, against the 20 inconsistent states of the short-sighted reconstruction space. In fact, while the two states on top of both graphs are the same, there are branches stemming from such states in the short-sighted reconstruction which are missing in the far-sighted reconstruction. For example, the inconsistent right-hand branch in Figure 2 is actually disabled by prospection graph $\mathcal{P}(\{L_1, L_2\}, \{X_1, Y_1\})$, which constraints the occurrence of all the transitions involved in event exchange on the links of system Σ: according to this prospection graph, only transition y_1 is allowed to follow x_1, while y_2, the responsible for the blind alley in Figure 2, is not.

5 CONCLUSION

Referring to the active system approach [1, 2] to diagnosis of DESs, this paper has shown how the off-line compilation of knowledge about event exchange between components brings a computational advantage on-line in terms of reduction of the number of backtracking steps performed by the history reconstruction algorithm. This advantage is expecially tangible when relaxing a strong assumption of all the state-of-the-art approaches to diagnosis of DESs, namely, the preciseness of events.

In this work, all input and output events in behavioral models, and not only observable events, as instead in [9, 10], may have an imprecise value ranging over a set of labels, namely an *uncertain* value. In presence of uncertain events, the search performed by short-sighted diagnosis is nondeterministic, while that carried out with the support of prospection knowledge is deterministic.

Moreover, prospection graphs, once generated off-line, can be reused several times on-line for different diagnostic problems inherent to the same system, or even for the same diagnostic problem in case there are repetitive link patterns in the system structure.

A previous proposal [8], based itself on knowledge compilation, transforms the active system approach into a spectrum of approaches which, according to the classification in Section 1, range from a totally first category version, wherein an exhaustive simulation of the system evolution is performed off-line, while on-line activities are limited to rule-checking, to a totally second category version, i.e. the original approach wherein no computation is performed off-line. Each approach falling in between consists of both off-line and on-line processing. The contribution of this paper is orthogonal to this work, that is, it could be integrated within any version of the spectrum (with the exception of the exclusively on-line one) in order to reduce backtracking steps in any reconstruction.

The exchange of events among components dealt with in this paper, being both asynchronous and buffered, is peculiar only to the active system approach.

One might argue that providing for a specific modeling primitive, namely the link, for the structural objects that implement asynchronous buffered communication between components, along with specific methods for dealing with them, just increases the expressive power of the method but does not alter its computational power at all. In fact, each link could be replaced by a common component, whose behavioral model represents the link behavior, and, therefore, synchronous composition of automata would suffice.

This is correct in principle but scarcely feasible in practice, for many reasons. First, the size of the behavioral model of such a component depends not only on the capacity of the link buffer but also on the number of distinct kinds of events that can be transmitted on the link.

For instance, consider a link with capacity equal to three, on which four kinds of events, say a, b, c, and d, can be transmitted. As each state of the component representing the link is univocally identified by the sequence of events in the buffer, the behavioral model of such a component has $\sum_{k=0}^{3}(4^k) = 85$ states.

This model may include states that are physically impossible given the system structure, since corresponding to sequences of events that cannot be generated, and, therefore, it is a burden for history reconstruction.

Besides, as remarked above, such a model depends on the kinds of events that can be transmitted on the link, that is, it depends on the producer component of the link at hand. This is somewhat in contrast with the philosophy of compositional modeling, according to which individual component models are reciprocally independent.

Instead, in the active system approach and, consequently, in this paper, a link is just the instantiation of a model, encompassing only the terminals, capacity, and policy of the link, and such a model is independent of the structure of the system in which the link is instantiated.

Of course, notwithstanding the modeling simplicity, link states are bound to emerge in the computation, sooner or later: the methods introduced in this paper are actually aimed at minimizing the number of physically impossible link states (and, hence, since a link state is a part of any active system state, the number of active space states) visited by the history reconstruction search algorithm.

In short-sighted diagnosis, where a link state is represented as a sequence of events, not all sequences of events are considered but only those that can be generated given the system structure. In far-sighted diagnosis, where the link state becomes an index record, the number of visited link states is further reduced: only those states are generated that can evolve towards a state wherein the link is empty.

REFERENCES

[1] P. Baroni, G. Lamperti, P. Pogliano, and M. Zanella, 'Diagnosis of large active systems', *Artificial Intelligence*, **110**(1), 135–183, (1999).

[2] P. Baroni, G. Lamperti, P. Pogliano, and M. Zanella, 'Diagnosis of a class of distributed discrete-event systems', *IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans*, **30**(6), 731–752, (2000).

[3] C. Barral, S. McIlraith, and T.C. Son, 'Formulating diagnostic problem solving using an action language with narratives and sensing', in *Seventh International Conference on Knowledge Representation and Reasoning – KR'2000*, pp. 311–322, Breckenridge, Colorado, (2000).

[4] L. Console, C. Picardi, and M. Ribaudo, 'Diagnosis and diagnosability using PEPA', in *Fourteenth European Conference on Artificial Intelligence – ECAI'2000*, pp. 131–135, Berlin, D, (2000).

[5] M.O. Cordier and C. Largouët, 'Using model-checking techniques for diagnosing discrete-event systems', in *Twelfth International Workshop on Principles of Diagnosis – DX'01*, pp. 39–46, San Sicario, I, (2001).

[6] R. Debouk, S. Lafortune, and D. Teneketzis, 'Coordinated decentralized protocols for failure diagnosis of discrete-event systems', *Journal of Discrete Event Dynamical Systems: Theory and Application*, **10**, 33–86, (2000).

[7] P. Laborie and J.P. Krivine, 'Automatic generation of chronicles and its application to alarm processing in power distribution systems', in *Eighth International Workshop on Principles of Diagnosis – DX'97*, Mont St. Michel, F, (1997).

[8] G. Lamperti and M. Zanella, 'Generation of diagnostic knowledge by discrete-event model compilation', in *Seventh International Conference on Knowledge Representation and Reasoning – KR'2000*, pp. 333–344, Breckenridge, Colorado, (2000).

[9] G. Lamperti and M. Zanella, 'Uncertain temporal observations in diagnosis', in *Fourteenth European Conference on Artificial Intelligence – ECAI'2000*, pp. 151–155, Berlin, D, (2000).

[10] G. Lamperti and M. Zanella, 'Diagnosis of discrete-event systems from uncertain temporal observations', *Artificial Intelligence*, **137**(1–2), 91–163, (2002).

[11] J. Lunze, 'Diagnosis of quantized systems based on timed discrete-event model', *IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans*, **30**(3), 322–335, (2000).

[12] Y. Pencolé, 'Decentralized diagnoser approach: application to telecommunication networks', in *Eleventh International Workshop on Principles of Diagnosis – DX'00*, pp. 185–192, Morelia, MX, (2000).

[13] L. Rozé, 'Supervision of telecommunication network: a diagnoser approach', in *Eighth International Workshop on Principles of Diagnosis – DX'97*, Mont St. Michel, F, (1997).

[14] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D.C. Teneketzis, 'Diagnosability of discrete-event systems', *IEEE Transactions on Automatic Control*, **40**(9), 1555–1575, (1995).

[15] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D.C. Teneketzis, 'Failure diagnosis using discrete-event models', *IEEE Transactions on Control Systems Technology*, **4**(2), 105–124, (1996).