Far-sighted Diagnosis of Active Systems

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

Abstract. Active systems are a class of discrete-event systems modeled as networks of nondeterministic automata communicating through either synchronous or asynchronous connection links. The model-based diagnosis of an active system is carried out by first reconstructing its behavior based on the observation, from which faults are later derived. The complexity of behavior reconstruction is exacerbated by the possibility of queuing events within links, thereby making essential the simulation of the order in which events are buffered within links. Unfortunately some sequences of events may lead to blind alleys in the search space. This is especially critical if events exchanged among components are assumed to be uncertain, as the number of alternative sequences of queued events is still larger. Therefore, behavior reconstruction without any prospection in the search space is generally bound to detrimental backtracking. To make diagnosis of active systems more efficient, we present an off-line technique for processing the models inherent to the system at hand so as to automatically generate prospection knowledge relevant to the mode in which events are produced and consumed over links. Such a knowledge is then exploited on-line, when the diagnostic engine is running, to guide the search process, thus reducing both time and space.

1 INTRODUCTION

Diagnosis of discrete-event systems (DESs) is a complex and challenging task that has been receiving an increasing interest from both the model-based diagnosis community [9], within the AI area, and the fault detection and isolation (FDI) community [16, 8, 10], within the automatic control area. The current shared prospect 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 [2], situation histories or narratives [4], paths [5], and trajectories [11, 6], 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:

- Those that first generate (a concise/partial model of) all possible evolutions and then retrieve only the evolutions that explain the observation;
- Those that generate in one shot the evolutions explaining the observation.

The first category includes some relevant works from both the automatic control area [19, 20, 7, 15] and the AI area [12, 6]. Embodied in the second category are some approaches of the AI area [2, 11, 17].

Since finding out the system evolutions is a computationally expensive and, therefore, inefficient process (see, for instance, [18] about the computational difficulties of the diagnoser approach [19, 20], or the worst case computational complexity analysis in [2], or the discussion in [11]), 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 [17] 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 [2, 3], 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 directed links. If an active system includes one or more asynchronous buffered links, its reaction 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, we present, by means of an example:

- An extension of both the modeling primitives and the on-line 'short-sighted' evolution reconstruction method so as to cope with uncertain events;
- 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;
- A 'far-sighted' method for exploiting prospection graphs on-line while reconstructing the evolutions of (sub)systems.

Finally, the computational advantages of far-sighted diagnosis are discussed and some conclusions are hinted.

2 ACTIVE SYSTEMS WITH UNCERTAIN EVENTS

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 which reacts to events either coming from the external world or from neighboring components through links. Formally, the automaton is a 6-tuple

$$(\mathbf{S}, \mathbf{E}_{\mathrm{in}}, \mathbf{I}, \mathbf{E}_{\mathrm{out}}, \mathbf{O}, \mathbf{T})$$

[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

where $\mathbf{S}$ is the set of *states*, $\mathbf{E}_{\text{in}}$ the set of *input events*, $\mathbf{I}$ the set of *input terminals*, $\mathbf{E}_{\text{out}}$ the set of *output events*, $\mathbf{O}$ the set of *output terminals*, and $\mathbf{T}$ the (nondeterministic) *transition function*:

$$\mathbf{T} : \mathbf{S} \times \mathbf{E}_{\text{in}} \times \mathbf{I} \times 2^{\mathbf{E}_{\text{out}} \times \mathbf{O}} \mapsto 2^{\mathbf{S}}.$$

A transition from state S to state S', which is triggered by the input event $\alpha = (E, I)$, $E \in \mathbf{E}_{\text{in}}$, $I \in \mathbf{I}$, and generates the set $\beta = \{(E_1, O_1), \ldots, (E_n, O_n)\}$ of output events, $E_k \in \mathbf{E}_{\text{out}}$, $O_k \in \mathbf{O}$, $k \in [1 \mathinner{..} n]$, is denoted by

$$S \xrightarrow{\alpha \mid \beta} S'.$$

Components are implicitly equipped with three *virtual terminals*, the *standard input* ($In \in \mathbf{I}$) for events coming from the external world, the *standard output* ($Out \in \mathbf{O}$) for events directed toward the external world (messages), and the *fault terminal* ($Flt \in \mathbf{O}$) for modeling faulty transitions.

An event (E, Flt) is a *fault event*. The approach assumes that both nominal and faulty behavior of each component are specified in the automaton. A fault event is not exchanged among components. Rather, it is a formal artifice to describe the faulty behavior of components uniformly. The name of fault events are supposed to be informative as to the specific fault affecting the component when the relevant transition is performed[2].

An event may be *uncertain* in nature, that is, represented by a disjunction of possible values. Links are the means of storing the events exchanged between components.

Each link L is characterized by a 4-tuple

$$(I, O, \chi, P)$$

where I is the *input terminal* (connected with a component output terminal), O the *output terminal* (connected with a component input 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*, that is, when the length of the queue equals χ. Three cases are possible:

- *LOSE*: E is lost;
- *OVERRIDE*: E replaces the last event in the queue of L;
- *WAIT*: T cannot be triggered until L becomes unsaturated, that is, until at least one event in L is consumed.

The *queue domain* $\mathbf{Q}$ of L is the set of possible sequences (queues) of events in L. The length of the queue Q of events incorporated in L is denoted by $|Q|$.

The polymorphic $Link$ function is defined as follows. Let

$$\alpha = (E, \theta)$$

represent an event relevant to a terminal θ. Then,

$$Link(\alpha) \stackrel{\text{def}}{=} L \mid L \text{ is the link connected with } \theta.$$

No more than one link can be connected with a component terminal. If θ is a virtual terminal, then $Link(\alpha) \stackrel{\text{def}}{=} null$. Let

$$\beta = \{(E_1, \theta_1), \ldots, (E_n, \theta_n)\}$$

be a set of events relevant to terminals θ_i, $i \in [1 \mathinner{..} n]$, respectively. Then,

$$Link(\beta) \stackrel{\text{def}}{=} \{L_\beta \mid L_\beta = Link(\mathcal{E}), \mathcal{E} \in \beta\}.$$

Initially, Σ is in a *quiescent* state Σ_0, wherein all links are empty. At 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* $OBS(\Sigma)$.

Let Σ_0 denote the initial state of system Σ. 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*, each history being a path from Σ_0 to a final state, in other terms, a sequence of component transitions which *explains* $OBS(\Sigma)$.

Candidate *diagnoses* are eventually distilled from the active space, each diagnosis being a set of faulty components, that is, those components which made at least one *faulty transition* during a candidate system history.

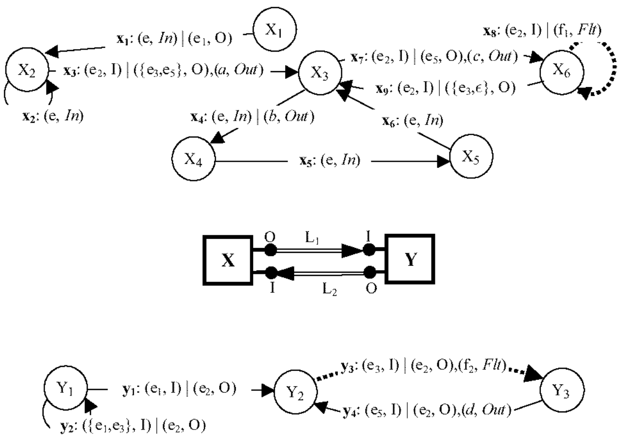
X1
X2
X3
X4
X5
X6
x1: (e, In) | (e1, O)
x2: (e, In)
x3: (e2, I) | ({e3,e5}, O),(a, Out)
x4: (e, In) | (b, Out)
x5: (e, In)
x6: (e, In)
x7: (e2, I) | (e5, O),(c, Out)
x8: (e2, I) | (f1, Flt)
x9: (e2, I) | ({e3,ϵ}, O)
X
Y
O
L1
I
I
L2
O
Y1
Y2
Y3
y1: (e1, I) | (e2, O)
y2: ({e1,e3}, I) | (e2, O)
y3: (e3, I) | (e2, O),(f2, Flt)
y4: (e5, I) | (e2, O),(d, Out)

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

Example 1. Displayed in the center of Figure 1 is a system Ψ, 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 $\chi = 1$ and $P = WAIT$. 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) (states and transitions are denoted by capital and small letters, respectively). 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 said to be *observable*). Transition y_2 involves the input 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_3 generates the uncertain event $(\{e_3, e_5\}, O)$, meaning that either e_3 or e_5 is randomly generated (no assumption is made about the likelihood of event generation). Likewise, x_9 generates the uncertain event $(\{e_3, \epsilon\}, O)$, meaning that either e_3 or nothing is generated (ϵ denotes the *null* event). $\square$

[2] For example, consider a breaker which is in the state *open* and is expected to change state to *close* when it receives a command (nominal behavior). The possible misbehavior of the breaker can be defined by inserting a faulty transition, from state *open* to *open*, that generates the fault event $(stuckToOpen, Flt)$.

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, each S_i, $i \in [1\,..\,n]$, being 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 $OBS(\Sigma)$, which implicitly denotes the prefix of the observation composed of the first $\Im$ messages;
- $\mathcal{Q} = (Q_1, \ldots, Q_\ell)$ is the record of queues of the ℓ links in 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 $N_0 = (\Sigma_0, 0, (\langle\rangle, \ldots, \langle\rangle))$, where all link queues are empty. 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. When the reconstruction process is carried out in one step (*monolithically*) without any prospection knowledge (*short-sightedly*), it can be described by Algorithm 1, where nodes and edges generated during the search are stored in variables $\aleph$ and $\mathcal{E}$, respectively.

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 each $i$ 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 within the link 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 node $(N' = (\sigma', \Im', \mathcal{Q}')) := N$; ($N'$ is created as a
       copy of $N$)
   (b) $\sigma'[i] :=$ the state reached by $T$;
   (c) If $T$ is observable, then $\Im := \Im + 1$; (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 $\aleph$.
```

The algorithm aims to make up all the nodes which are reachable from the initial node under the given observation. To this end, it considers, one at a time, all the nodes which have been reached already (those in $\aleph$) and have not yet been processed (the unmarked ones). For each of them, it attempts to find a transition that is triggerable by a component in the corresponding state. If so, it generates the target node N' with the appropriate values σ', $\Im'$, and $\mathcal{Q}'$. In the new node was not created already, it is inserted into $\aleph$ (note that two nodes which differ in the $\Im$ field only have to be considered different, as the mode in which messages have been generated differ). The corresponding edge $N \xrightarrow{T} N'$ is inserted into $\mathcal{E}$ too. Finally, when there are no more nodes to be processed (all nodes in $\aleph$ are marked), the search space is pruned by eliminating the inconsistent nodes, that is, those that are on a blind alley.

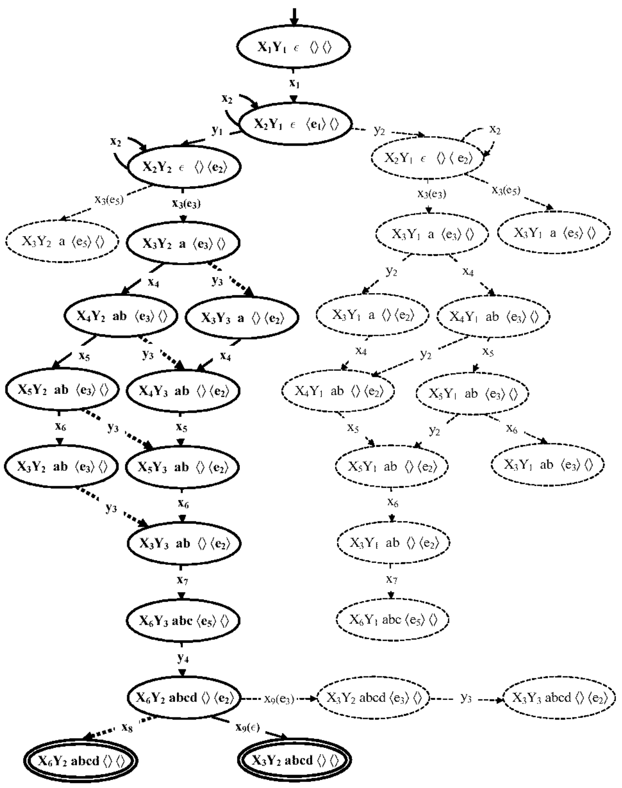

Figure 2. Short-sighted reconstruction space (see Example 2).

It is worthwhile highlighting that the search process does not terminate at a final node. In fact, the system might continue to react and loop on unobservable paths. In other words, when a final node is met in the search, it is inserted into $\aleph$ as an unmarked node like all other nodes, since in principle, unobservable paths might happen to leave it.

When uncertain output 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 output events included several uncertain events, all possible combinations would be required to be enumerated.

Example 2. Shown in Figure 2 is the reconstruction space generated short-sightedly for the diagnostic problem $\wp(\Psi) = (OBS(\Psi), \Psi_0)$, where Ψ is the system outlined in Figure 1, $OBS(\Psi) = \langle a, b, c, d \rangle$, and $\Psi_0 = (X_1, Y_1)$. Each node is depicted by an ellipse, wherein

- $\sigma = (X_i, Y_j)$ is the pair of component states;
- $\Im$ is the prefix of the observation generated so far;
- $\mathcal{Q} = (Q_1, Q_2)$ is the pair of link queues.

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 corresponds to inconsistent states, which are almost half the search space. Owing to cycles in the graph (edges marked by x_2), the active space includes an unbound number of candidate histories. However, only two candidate diagnoses are possible, namely $\{Y\}$ and $\{X, Y\}$. Note that, although not relevant to our example, the replication of the same faulty transition in a cycle does not change the diagnosis. A finer-grained diagnosis can be defined, as in [2], called *deep diagnosis*. The latter is a set of pairs (C, f), where C is a component and f a fault event. This way, even if not relevant to our example where each component model includes a single faulty transition, it is possible to know all the faulty transitions performed by each misbehaving component. $\square$

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 the reconstruction process to avoid entering the inconsistent sub-space 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}, with queue domain $\mathbf{Q}$. Let $M^{\mathrm{p}} = (\mathbf{S}^{\mathrm{p}}, \mathbf{E}^{\mathrm{p}}_{\mathrm{in}}, \mathbf{I}^{\mathrm{p}}, \mathbf{E}^{\mathrm{p}}_{\mathrm{out}}, \mathbf{O}^{\mathrm{p}}, \mathbf{T}^{\mathrm{p}})$ and $M^{\mathrm{c}} = (\mathbf{S}^{\mathrm{c}}, \mathbf{E}^{\mathrm{c}}_{\mathrm{in}}, \mathbf{I}^{\mathrm{c}}, \mathbf{E}^{\mathrm{c}}_{\mathrm{out}}, \mathbf{O}^{\mathrm{c}}, \mathbf{T}^{\mathrm{c}})$ be the models of C^{p} and C^{c}, respectively. Let

$$\hat{M}^{\mathrm{p}^{\mathrm{n}}} = (\hat{\mathbf{S}}^{\mathrm{p}^{\mathrm{n}}}, \hat{\mathbf{E}}^{\mathrm{p}^{\mathrm{n}}}, \hat{\mathbf{T}}^{\mathrm{p}^{\mathrm{n}}})$$

be the nondeterministic automaton obtained from M^{p} in such a way that

- $\hat{\mathbf{S}}^{\mathrm{p}^{\mathrm{n}}} = \mathbf{S}^{\mathrm{p}}$ is the set of states;
- $\hat{\mathbf{E}}^{\mathrm{p}^{\mathrm{n}}} \subseteq \mathbf{T}^{\mathrm{p}} \cup \{\epsilon\}$ is the set of events;
- $\hat{\mathbf{T}}^{\mathrm{p}^{\mathrm{n}}} : \hat{\mathbf{S}}^{\mathrm{p}^{\mathrm{n}}} \times \hat{\mathbf{E}}^{\mathrm{p}^{\mathrm{n}}} \mapsto 2^{\hat{\mathbf{S}}^{\mathrm{p}^{\mathrm{n}}}}$ is the transition function.

The transition function $\hat{\mathbf{T}}^{\mathrm{p}^{\mathrm{n}}}$ is obtained from $\mathbf{T}^{\mathrm{p}}$ as follows:

$$\forall T = S \xrightarrow{\alpha|\beta} S' \in \mathbf{T}^{\mathrm{p}} \begin{cases} S \xrightarrow{\epsilon} S' \in \hat{\mathbf{T}}^{\mathrm{p}^{\mathrm{n}}} & \text{if } L \notin \mathit{Link}(\beta) \\ S \xrightarrow{T} S' \in \hat{\mathbf{T}}^{\mathrm{p}^{\mathrm{n}}} & \text{otherwise.} \end{cases}$$

Similarly, let

$$\hat{M}^{\mathrm{c}^{\mathrm{n}}} = (\hat{\mathbf{S}}^{\mathrm{c}^{\mathrm{n}}}, \hat{\mathbf{E}}^{\mathrm{c}^{\mathrm{n}}}, \hat{\mathbf{T}}^{\mathrm{c}^{\mathrm{n}}})$$

be the nondeterministic automaton obtained from M^{c} in such a way that

- $\hat{\mathbf{S}}^{\mathrm{c}^{\mathrm{n}}} = \mathbf{S}^{\mathrm{c}}$ is the set of states;
- $\hat{\mathbf{E}}^{\mathrm{c}^{\mathrm{n}}} \subseteq \mathbf{T}^{\mathrm{c}} \cup \{\epsilon\}$ is the set of events;
- $\hat{\mathbf{T}}^{\mathrm{c}^{\mathrm{n}}} : \hat{\mathbf{S}}^{\mathrm{c}^{\mathrm{n}}} \times \hat{\mathbf{E}}^{\mathrm{c}^{\mathrm{n}}} \mapsto 2^{\hat{\mathbf{S}}^{\mathrm{c}^{\mathrm{n}}}}$ is the transition function.

The transition function $\hat{\mathbf{T}}^{\mathrm{c}^{\mathrm{n}}}$ is obtained from $\mathbf{T}^{\mathrm{c}}$ as follows:

$$\forall T = S \xrightarrow{\alpha|\beta} S' \in \mathbf{T}^{\mathrm{c}} \begin{cases} S \xrightarrow{\epsilon} S' \in \hat{\mathbf{T}}^{\mathrm{c}^{\mathrm{n}}} & \text{if } L \neq \mathit{Link}(\alpha) \\ S \xrightarrow{T} S' \in \hat{\mathbf{T}}^{\mathrm{c}^{\mathrm{n}}} & \text{otherwise.} \end{cases}$$

Let $\hat{M}^{\mathrm{p}} = (\hat{\mathbf{S}}^{\mathrm{p}}, \hat{\mathbf{E}}^{\mathrm{p}}, \hat{\mathbf{T}}^{\mathrm{p}})$ and $\hat{M}^{\mathrm{c}} = (\hat{\mathbf{S}}^{\mathrm{c}}, \hat{\mathbf{E}}^{\mathrm{c}}, \hat{\mathbf{T}}^{\mathrm{c}})$ be the deterministic automata equivalent to $\hat{M}^{\mathrm{p}^{\mathrm{n}}}$ and $\hat{M}^{\mathrm{c}^{\mathrm{n}}}$, respectively. A *prospection state* $\mathcal{L}$ of L is a triple

$$\mathcal{L} = (\hat{S}^{\mathrm{p}}, \hat{S}^{\mathrm{c}}, Q) \in \hat{\mathbf{S}}^{\mathrm{p}} \times \hat{\mathbf{S}}^{\mathrm{c}} \times \mathbf{Q}.$$

Let $\mathcal{L}$ be a prospection state and $\hat{S} \xrightarrow{T} \hat{S}' \in (\hat{\mathbf{T}}^{\mathrm{p}} \cup \hat{\mathbf{T}}^{\mathrm{c}})$, $\hat{S} \in \{\hat{S}^{\mathrm{p}}, \hat{S}^{\mathrm{c}}\}$, $T = S \xrightarrow{\alpha|\beta} S' \in (\mathbf{T}^{\mathrm{p}} \cup \mathbf{T}^{\mathrm{c}})$. Let Q be a queue of events in L and

- $\mathit{Head}(Q)$ denote the first consumable event in Q;
- $\mathit{Tail}(Q)$ denote the sequence of events in Q following the first event;
- $\mathit{App}(Q, e)$ denote the queue obtained by appending e to Q;
- $\mathit{Repl}(Q, e)$ denote the queue obtained by replacing the last event in Q with e.

The Next function yields the set of next prospection states as follows:

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

where

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

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

and

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

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

$$\widetilde{\Gamma}^{\mathrm{n}}(L, \mathcal{C}_0) = (\widetilde{\mathbf{S}}^{\mathrm{n}}, \mathbf{E}^{\mathrm{n}}, \widetilde{\mathbf{T}}^{\mathrm{n}}, S^{\mathrm{n}}_0, \mathbf{S}^{\mathrm{n}}_{\mathrm{f}})$$

where

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

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

A state of a spurious prospection graph which is not within a path from the initial state to a final state is an *inconsistent state*. Similarly, a transition entering or leaving an inconsistent state of a spurious prospection graph is an *inconsistent transition*.

The *nondeterministic prospection graph* is the nondeterministic automaton

$$\Gamma^{\mathrm{n}}(L, \mathcal{C}_0) = (\mathbf{S}^{\mathrm{n}}, \mathbf{E}^{\mathrm{n}}, \mathbf{T}^{\mathrm{n}}, S^{\mathrm{n}}_0, \mathbf{S}^{\mathrm{n}}_{\mathrm{f}})$$

obtained from $\widetilde{\Gamma}^{\mathrm{n}}(L, \mathcal{C}_0)$ by removing inconsistent states and inconsistent transitions.

The *prospection graph*

$$\Gamma(L, \mathcal{C}_0) = (\mathbf{S}, \mathbf{E}, \mathbf{T}, S_0, \mathbf{S}_{\mathrm{f}})$$

is the deterministic automaton equivalent to the nondeterministuic prospection graph $\Gamma^{\mathrm{n}}(L, \mathcal{C}_0)$.

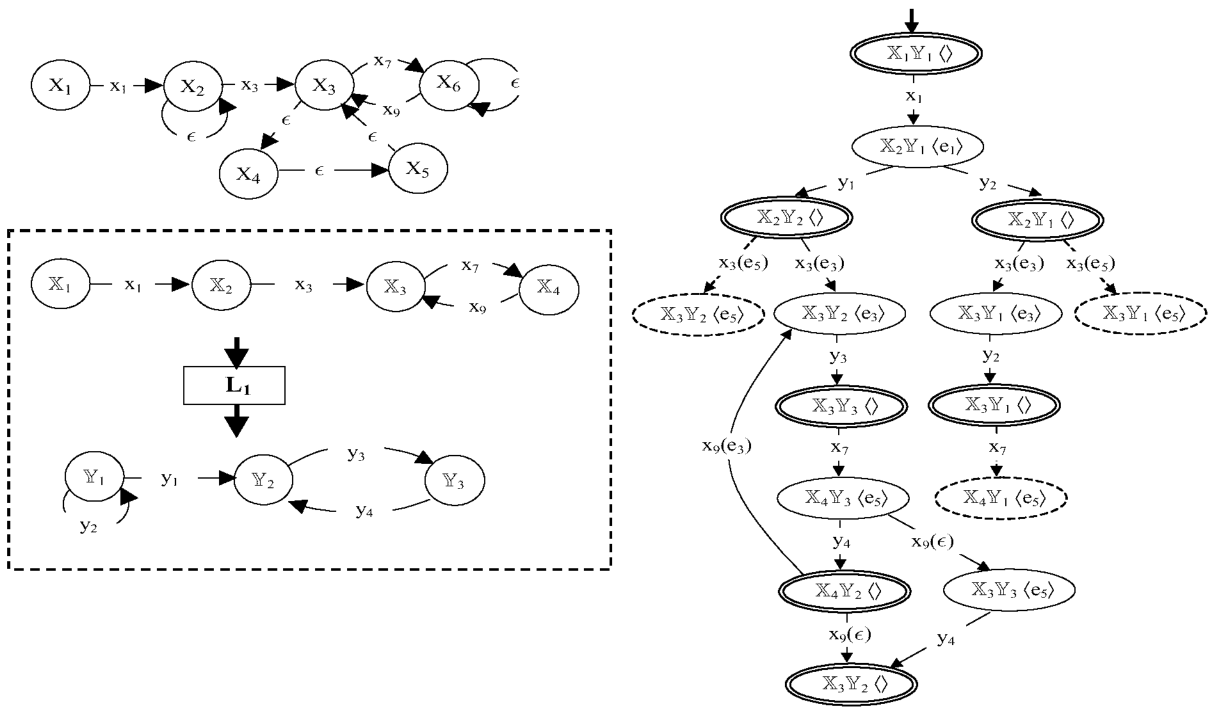

Figure 3. Generation of $\Gamma^{\mathrm{n}}(L_1, (X_1, Y_1))$ (see Example 3).

Example 3. Shown in the dashed box of Figure 3 are the prospection models $\hat{M}^{\mathrm{p}}(X)$ (top) and $\hat{M}^{\mathrm{c}}(Y)$ (bottom), inherent to link L_1, which are relevant to the components X and Y displayed in Figure 1. Depicted on the top of the box is the nondeterministic automaton $\hat{M}^{\mathrm{p}^{\mathrm{n}}}(X)$ equivalent to $\hat{M}^{\mathrm{p}}(X)$. The generation of the nondeterministic prospection graph $\Gamma^{\mathrm{n}}(L_1, (X_1, Y_1))$ is outlined on the right of Figure 4, where double ellipses denote final states, while dashed nodes and edges represent inconsistent states and transitions, respectively. Note that the latter includes a circular path involving four states. This situation is similar to that of active systems, where cycles may stem from (possibly) final states. Within the context of prospection graphs, cycles represent repetitive patterns of link state changes (in our example, events e_3 and e_5 are repeatedly produced and consumed, that is, inserted into and removed from link L_1). □

Note that, essentially, the generation of a prospection graph is analogous to the generation of an active space, where

- Component models are substituted by prospection models;
- Only one link is considered;
- No observation index is considered.

4.1.1 *Generalized prospection graphs*

The notion of the prospection graph of a single link can be naturally extended to that of a set of links. Let $\mathbb{L} = \{L_1, \dots, L_m\}$ be a set of links (with queue domains $\mathbf{Q}_1, \dots, \mathbf{Q}_m$, respectively) connecting a set $\mathbb{C} = \{C_1, \dots, C_t\}$ of components, where each component C_i, $i \in [1 \mathinner{..} t]$, is characterized by model

$$M_i = (\mathbf{S}_i, \mathbf{E}_{\mathrm{in}_i}, \mathbf{I}_i, \mathbf{E}_{\mathrm{out}_i}, \mathbf{O}_i, \mathbf{T}_i).$$

Let $\hat{M}_i^{\mathrm{n}} = (\hat{\mathbf{S}}_i^{\mathrm{n}}, \hat{\mathbf{E}}_i^{\mathrm{n}}, \hat{\mathbf{T}}_i^{\mathrm{n}})$ be the nondeterministic automaton obtained from M_i in such a way that

- $\hat{\mathbf{S}}_i^{\mathrm{n}} = \mathbf{S}_i$ is the set of states;
- $\hat{\mathbf{E}}_i^{\mathrm{n}} \subseteq \mathbf{T}_i \cup \{\epsilon\}$ is the set of events;
- $\hat{\mathbf{T}}_i^{\mathrm{n}} : \hat{\mathbf{S}}_i^{\mathrm{n}} \times \hat{\mathbf{E}}_i^{\mathrm{n}} \mapsto 2^{\hat{\mathbf{S}}_i^{\mathrm{n}}}$ is the transition function.

The transition function $\hat{\mathbf{T}}_i^{\mathrm{n}}$ is obtained from $\mathbf{T}_i$ as follows:

$$\forall T = S \xrightarrow{\alpha|\beta} S' \in \mathbf{T}_i \begin{cases} S \xrightarrow{T} S' \in \hat{\mathbf{T}}_i^{\mathrm{n}} & \text{if } \mathit{Relevant}(\alpha, \beta, \mathbb{L}) \\ S \xrightarrow{\epsilon} S' \in \hat{\mathbf{T}}_i^{\mathrm{n}} & \text{otherwise} \end{cases}$$

where

$$\mathit{Relevant}(\alpha, \beta, \mathbb{L}) \stackrel{\mathrm{def}}{=} (\{\mathit{Link}(\alpha)\} \cup \mathit{Link}(\beta)) \cap \mathbb{L} \neq \emptyset.$$

Let $\hat{M}_i = (\hat{\mathbf{S}}_i, \hat{\mathbf{E}}_i, \hat{\mathbf{T}}_i)$ be the deterministic automaton equivalent to $\hat{M}_i^{\mathrm{n}}$. A *generalized prospection state* $\mathfrak{L}$ of $\mathbb{L}$ is a pair

$$\mathfrak{L} = (\mathbb{S}, \mathbb{Q})$$

where

$$\mathbb{S} = (\hat{S}_1, \dots, \hat{S}_t) \in (\hat{\mathbf{S}}_1 \times \cdots \times \hat{\mathbf{S}}_t),$$
$$\mathbb{Q} = (Q_1, \dots, Q_m) \in (\mathbf{Q}_1 \times \cdots \times \mathbf{Q}_m).$$

Let $\mathfrak{L} = (\mathbb{S}, \mathbb{Q})$ be a generalized prospection state and $\hat{S}_i \xrightarrow{T} \hat{S}' \in \hat{\mathbf{T}}_i$, $i \in [1 \mathinner{..} t]$, $T = S \xrightarrow{\alpha|\beta} S'$.

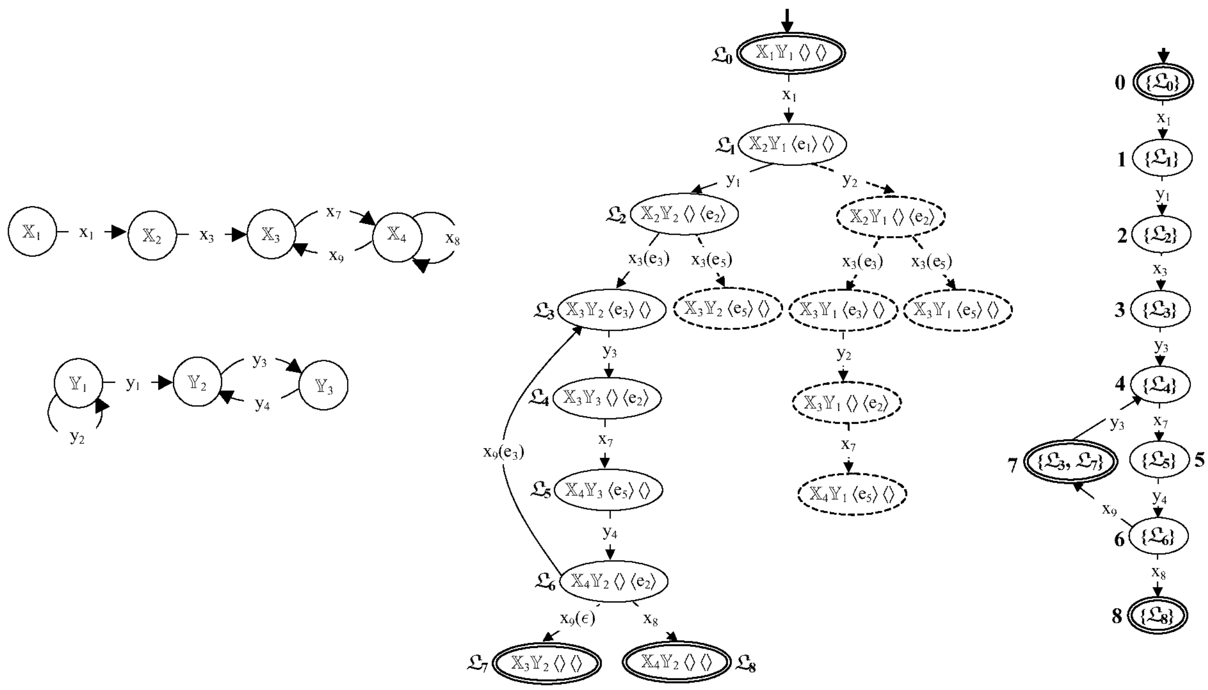

Figure 4. Generation of the generalized prospection graph $\Gamma(\mathbb{L}, \Psi_0)$ (see Example 4).

The *generalized Next* function yields the set of next generalized prospection states as follows:

$$\begin{aligned}
&\mathit{Next}(\mathscr{L}, T) \overset{\text{def}}{=} \{\mathscr{L}' \mid \mathscr{L}' = (\mathbb{S}', \mathbb{Q}'), \mathbb{S}' = (\hat{S}'_1, \ldots, \hat{S}'_t),\\
&\quad \mathbb{Q}' = (Q'_1, \ldots, Q'_m), \alpha = (E_\alpha, I_\alpha),\\
&\quad ((\mathit{Link}(I_\alpha) \notin \mathbb{L}) \text{ or}\\
&\quad (\mathit{Link}(I_\alpha) = L_j, L_j \in \mathbb{L}, e \in E_\alpha, \mathit{Head}(Q_j) = e,\\
&\qquad Q'_j = \mathit{Tail}(Q_j))),\\
&\quad \mathbb{L}_\beta = \{L_\beta \mid L_\beta = \mathit{Link}(O_\beta), (E_\beta, O_\beta) \in \beta, L_\beta \in \mathbb{L}\},\\
&\quad \forall L_h \in \mathbb{L}_\beta (e \in E_\beta, (E_\beta, O_\beta) \in \beta, L_h = \mathit{Link}(O_\beta),\\
&\qquad Q'_h = \mathit{Ins}(Q_h, e),\\
&\qquad (|Q_h| < \chi_h \text{ or}\\
&\qquad (|Q_h| = \chi_h, (e = \epsilon \text{ or } P_h \in \{\mathit{LOSE}, \mathit{OVERRIDE}\})))),\\
&\quad \forall L_k \in (\mathbb{L} - (\mathbb{L}_\beta \cup \{\mathit{Link}(I_\alpha)\}))\ (Q'_k = Q_k),\\
&\quad \hat{S}'_i = \hat{S}', \forall x \in [1 \,..\, t], x \neq i\ (\hat{S}'_x = \hat{S}_x)\}.
\end{aligned}$$

Let $\mathbb{C}_0 = (S_{0_1}, \ldots, S_{0_t})$ be the record of initial states for components in $\mathbb{C}$. The *generalized spurious prospection graph* of $\mathbb{L}$ and $\mathbb{C}_0$ is the nondeterministic automaton

$$\widetilde{\Gamma}^{\mathrm{n}}(\mathbb{L}, \mathbb{C}_0) = (\widetilde{\mathbf{S}}^{\mathrm{n}}, \mathbf{E}^{\mathrm{n}}, \widetilde{\mathbf{T}}^{\mathrm{n}}, S_0^{\mathrm{n}}, \mathbf{S}_{\mathrm{f}}^{\mathrm{n}})$$

where

$\widetilde{\mathbf{S}}^{\mathrm{n}} = \{\mathscr{L} \mid \mathscr{L}$ is a prospection state of $\mathbb{L}\}$ is the set of states,
$\mathbf{E}^{\mathrm{n}} \subseteq \bigcup_{i=1}^{t} \hat{\mathbf{E}}_i \subseteq \bigcup_{i=1}^{t} \mathbf{T}_i$ is the set of events,
$S_0^{\mathrm{n}} = (\mathbb{C}_0, (\langle\rangle \cdots \langle\rangle))$ is the initial state,
$\mathbf{S}_{\mathrm{f}}^{\mathrm{n}} = \{\mathscr{L} \mid \mathscr{L} \in \mathbf{S}^{\mathrm{n}}, \mathscr{L} = (\mathbb{S}, (\langle\rangle \cdots \langle\rangle))\}$ is the set of final states,
$\widetilde{\mathbf{T}}^{\mathrm{n}} : \widetilde{\mathbf{S}}^{\mathrm{n}} \times \mathbf{E}^{\mathrm{n}} \mapsto 2^{\widetilde{\mathbf{S}}^{\mathrm{n}}}$ is the transition function defined as follows:

$$\mathscr{L} \xrightarrow{T} \mathscr{L}' \in \widetilde{\mathbf{T}}^{\mathrm{n}} \text{ iff } \mathscr{L}' \in \mathit{Next}(\mathscr{L}, T).$$

The *generalized nondeterministic prospection graph* is the nondeterministic automaton

$$\Gamma^{\mathrm{n}}(\mathbb{L}, \mathbb{C}_0) = (\mathbf{S}^{\mathrm{n}}, \mathbf{E}^{\mathrm{n}}, \mathbf{T}^{\mathrm{n}}, S_0^{\mathrm{n}}, \mathbf{S}_{\mathrm{f}}^{\mathrm{n}})$$

obtained from $\widetilde{\Gamma}^{\mathrm{n}}(\mathbb{L}, \mathbb{C}_0)$ by removing inconsistent states and inconsistent transitions.

The *generalized prospection graph*

$$\Gamma(\mathbb{L}, \mathbb{C}_0) = (\mathbf{S}, \mathbf{E}, \mathbf{T}, S_0, \mathbf{S}_{\mathrm{f}})$$

is the deterministic automaton equivalent to the $\Gamma^{\mathrm{n}}(\mathbb{L}, \mathbb{C}_0)$.

Example 4. Shown in Figure 4 is the generation of the generalized prospection graph $\Gamma(\mathbb{L}, \Psi_0)$ relevant to the links in system Ψ (see Figure 1), where $\mathbb{L} = \{L_1, L_2\}$ and $\Psi_0 = (X_1, Y_1)$. Specifically, outlined on the left are the prospection models of components X and Y, namely $\hat{M}(X)$ and $\hat{M}(Y)$. Shown on the center is the generation of the generalized nondeterministic prospection graph $\Gamma^{\mathrm{n}}(\mathbb{L}, \Psi_0)$ (the dash part of the graph denotes the inconsistent search space), where consistent nodes are identified by labels $\mathscr{L}_0 \cdots \mathscr{L}_6$. Finally, displayed on the right is the corresponding deterministic prospection graph $\Gamma(\mathbb{L}, \Psi_0)$. The latter is determined based on the *subset construction algorithm* presented in [1], which identifies each node of the deterministic automaton by means of a subset of nodes of the nondeterministic one, specifically, those nodes that are reachable through the same marking transition. For example, since there are two edges, leaving the same state $\mathscr{L}_6$ in the nondeterministic automaton, that are marked by the same label x_9, the deterministic automaton will include the node identified by the subset $\{\mathscr{L}_3, \mathscr{L}_7\}$, which is reached from $\{\mathscr{L}_6\}$ by means of the (unique) edge marked by x_9. According to the algorithm, each node in the deterministic automaton that includes a final state of the nondeterministic one is final itself. Nodes of the deterministic automaton are identified by labels $0 \cdots 8$. □

Given a system Σ, in order to exploit the prospection knowledge in the reconstruction process, we need to create a set of g prospection graphs

$$\mathbf{\Gamma}(\Sigma) = \{\Gamma(\mathbb{L}_1, \mathbb{C}_{0_1}), \dots, \Gamma(\mathbb{L}_g, \mathbb{C}_{0_g})\}$$

such that $\bigcup_{i=1}^{g} \mathbb{L}_i$ equals the whole set of links in Σ. $\mathbf{\Gamma}(\Sigma)$ is a *prospection coverage* of Σ.

Algorithm 2. (*Far-sighted Reconstruction*)

The far-sighted reconstruction algorithm is a variation of Algorithm 1. First, the $\mathcal{Q}$ field of a node denotes a record of g states relevant to the g prospection graphs in the prospection coverage $\mathbf{\Gamma}(\Sigma)$, namely

$$\mathcal{Q} = (\gamma_1, \dots, \gamma_g).$$

Moreover, in the initial node $N_0 = (\sigma_0, \Im_0, \mathcal{Q}_0)$, $\mathcal{Q}_0$ is represented by the record of the initial states of the corresponding prospection graphs, namely $(\gamma_{0_1}, \dots, \gamma_{0_g})$. 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 $\Pi(T) = \{\bar{\Gamma}_1, \dots, \bar{\Gamma}_r\}$ be the prospection graphs in $\mathbf{\Gamma}(\Sigma)$ that are relevant to links connected with terminals on which events are either consumed or generated by T; let $\bar{\mathcal{Q}}(N) = \{\bar{\gamma}_1, \dots, \bar{\gamma}_r\}$ be the elements of $\mathcal{Q}(N)$ relevant to $\Pi(T)$:

(ii) $\forall i \in [1\,..\,r]$ ($\bar{\gamma}_i \xrightarrow{T} \bar{\gamma}'_i$ is an edge in $\bar{\Gamma}_i$),

then do the following steps:

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

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

(c) If T is observable, then $\Im := \Im + 1$;

(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 $\aleph$;

(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, thereby preventing the search from entering (possibly large) inconsistent parts of the space. Of course, such a prospection is finite, thereby not eliminating completely the backtracking. 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 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.

Proposition 1. *Let $\wp(\Sigma)$ be a diagnostic problem and $\|A\|$ denote the (possibly unbound) set of histories incorporated in an active space A. Let $Act^{\mathrm{s}}(\wp(\Sigma))$ and $Act^{\mathrm{f}}(\wp(\Sigma))$ denote the active spaces generated by Algorithm 1 and Algorithm 2, respectively. Then,*

$$\|Act^{\mathrm{s}}(\wp(\Sigma))\| = \|Act^{\mathrm{f}}(\wp(\Sigma))\|.$$

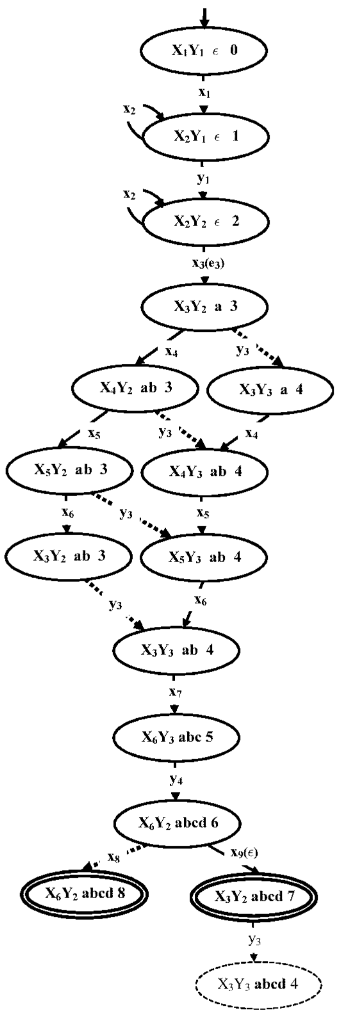

Figure 5. Far-sighted reconstruction space (see Example 4).

Example 5. Shown in Figure 5 is the reconstruction space for the diagnostic problem $\wp(\Psi) = (\langle a, b, c, d\rangle, (X_1, Y_1))$ based on the generalized prospection graph outlined on the right of Figure 4. 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 14 inconsistent states of the short-sighted reconstruction space. In fact, while the two states on top of both graphs are the same, there is a right branch stemming from the latter of such states in the short-sighted reconstruction which is missing in the far-sighted reconstruction. This branching is actually disabled by prospection graph $\Gamma(\{L_1, L_2\}, (X_1, Y_1))$, which constraints the occurrence of all the transitions involved in event exchange on the links of system Ψ: 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 [2, 3] 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 [14], 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 [13], 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 that 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, let us 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! So large a model is a burden for history reconstruction. In fact, the model may be unduly large as it includes even states that are physically impossible given the system structure, since corresponding to sequences of events that cannot be generated.

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 state of one or several links becomes a record of indexes, 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] A. Aho, R. Sethi, and J.D. Ullman, *Compilers – Principles, Techniques, and Tools*, Addison-Wesley, Reading, MA, 1986.

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

[3] 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).

[4] 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).

[5] 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).

[6] 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).

[7] 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).

[8] P.M. Frank, 'Analytical and qualitative model-based fault diagnosis – a survey and some new results', *European Journal of Control*, **2**, 6–28, (1996).

[9] *Readings in Model-Based Diagnosis*, eds., W. Hamscher, L. Console, and J. de Kleer, Morgan Kaufmann, San Mateo, CA, 1992.

[10] R. Isermann, 'Supervision, fault detection and fault-diagnosis methods – an introduction', *Control Engineering Practice*, **5**(5), 639–652, (1997).

[11] J. Kurien and P.P. Nayak, 'Back to the future for consistency-based trajectory tracking', in *Eleventh International Workshop on Principles of Diagnosis – DX'00*, pp. 92–100, Morelia, MX, (2000).

[12] 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).

[13] 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).

[14] 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).

[15] 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).

[16] R.J. Patton and J. Chen, 'A review of parity space appproaches to fault diagnosis', in *IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes – SAFEPROCESS'91*, Baden-Baden, D, (1991).

[17] 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).

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

[19] 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).

[20] 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).