ELSEVIER

Artificial Intelligence 110 (1999) 135–183

Artificial Intelligence

www.elsevier.com/locate/artint

Diagnosis of large active systems

P. Baroni [a,1], G. Lamperti [a,*], P. Pogliano [b,2], M. Zanella [a,3]

[a] *Dipartimento di Elettronica per l'Automazione, Università di Brescia, Via Branze 38, I-25123 Brescia, Italy*
[b] *Enel Ricerca – Area Trasmissione e Dispacciamento, Via Volta 1, I-20093 Cologno Monzese (MI), Italy*

Received 28 July 1998

Abstract

This paper presents a modular technique, amenable to parallel implementation, for the diagnosis of large-scale, distributed, asynchronous event-driven (namely, active) systems. An active system is an abstraction of a physical system that can be modeled as a network of communicating automata. Due to the distributed nature of the class of systems considered, and unlike other approaches based on synchronous composition of automata, exchanged events are buffered within communication links and dealt with asynchronously. The main goal of the diagnostic technique is the reconstruction of the behavior of the active system starting from a set of observable events. The diagnostic process involves three steps: interpretation, merging, and diagnosis generation. Interpretation generates a representation of the behavior of a part of the active system based on observable events. Merging combines the result of several interpretations into a new, broader interpretation. The eventual diagnostic information is generated on the basis of fault events possibly incorporated within the reconstructed behavior. In contrast with other approaches, the proposed technique does not require the generation of the, possibly huge, model of the entire system, typically, in order to yield a global diagnoser, but rather, it allows a modular and parallel exploitation of the reconstruction process. This property, to a large extent, makes effective the diagnosis of real active systems, for which the reconstruction of the global behavior is often unnecessary, if not impossible.

Keywords: Model-based diagnosis; Behavior reconstruction; Asynchronous discrete-event systems

* Corresponding author. Email: lamperti@ing.unibs.it.
[1] Email: baroni@ing.unibs.it.
[2] Email: pogliano@pea.enel.it.
[3] Email: zanella@ing.unibs.it.

0004-3702/99/$ – see front matter
PII: S0004-3702(99)00019-3

1. Introduction

Informally, diagnosis can be defined as the task of explaining why a given physical system does not exhibit its nominal behavior. Every diagnostic problem is characterized by a set of observations to account for. Due to its generality, to its dramatic importance in many application domains, and to its intrinsic complexity, automated diagnosis has received constant and considerable attention in AI research.

The classical theory of model-based diagnosis [12,32] addresses the limited range of static systems whose nominal behavior can be specified as a time invariant mapping from input to output variables. The need for model-based diagnosis of dynamic (as opposed to "static") systems was recognized very early and several relevant contributions can be found in the literature [6,13,15,17–19,23,29,31,38].

This paper deals with the diagnosis of a class of asynchronous discrete-event systems, namely active systems, whose inputs are not observable, for they are typically unpredictable events occurring in the external world. Asynchronous (as opposed to "synchronous") behavior is a major point that distinguishes our approach from previous approaches in the literature.

The proposed diagnostic method stems from the experience of the authors in the domain of fault diagnosis of power protection systems [2,3,25]. However, the method is not limited to this domain only. Instead, it is applicable to every physical system that can be modeled as an active system. Thus, the keyword "active" refers to the model of the system, rather than to the system itself. A physical system is not active *per se*: it is the way in which the system is modeled that "makes" it active. Of course, there are some physical systems that can be modeled as active systems more naturally than others, and this is what makes the difference. In some cases, only a sub-part of the physical system can be conveniently modeled as an active system. In other cases, even the part of the physical system that is normally viewed as "continuous" can be viewed as "discrete-event" for the purpose of diagnosis.

In our view, diagnosing an active system requires the reconstruction of what happened to it, based on observation and (structural and behavioral) models. This task is called history reconstruction. A reconstructed system history includes the input value and the state transitions such that the collected system observation is explained and provides important diagnostic information, since each transition is either normal or faulty. Therefore, the proposed technique lends itself to the notion of explanatory diagnosis [27], according to which diagnosis is the explanation of the behavior of the considered system, rather than the mere identification of a set of faulty components.

One of the main challenges presented by the history reconstruction of real-size active systems is keeping the search space as small as possible. Particular attention has been paid to this aspect, both in system representation and in the history reconstruction algorithm, which is equally applicable to the whole system, to every sub-system (called cluster), and even to single components. The reconstruction method can be applied in parallel to small clusters rather than to the whole system, thus obtaining histories pertaining to the considered clusters. Then, clusters can be merged to form clusters at a higher aggregation level and the reconstruction procedure can be applied iteratively. In order to prune the search space, the reconstruction task exploits the results produced by reconstruction at a

lower aggregation level. This way, a significant efficiency gain is obtained with respect to the case where reconstruction is directly applied to the whole system. Whatever the topology of the system under consideration, this modular approach copes with the intrinsic computational difficulties involved in dealing with large systems and, as such, is more general than the approach in [34], which addresses only systems endowed with a rigid hierarchical structure and including several independent sub-systems.

Some approaches to the diagnosis of discrete event-driven systems [22,34,36,37] are very close to ours, both because they adopt the same modeling technique, namely communicating finite state machines (FSMs) [5], and because they do not consider system inputs in making a diagnosis. However, such approaches substantially differ from ours from the point of view of the diagnostic method, which requires, in their case, the generation of a (possibly huge) global diagnoser. Another unique feature of the proposed approach is the ability to manage unobservable behavioral cycles, which are not considered in [34,36,37].

Furthermore, our approach allows attention to be focused on any particular sub-system, producing a set of diagnoses each of which entails the observations of all the components in the considered sub-system. This is very important in many practical domains where the sub-system in which to look for the fault(s) can be isolated based on domain-dependent (heuristic) techniques. Focalizing attention on specific sub-systems which are connected with other sub-systems is impossible in [22,34,36,37].

The remainder of the paper is organized as follows. The diagnostic technique is informally described in Section 2, based on a simple example, and formally introduced in Sections 3 and 4. Specifically, Section 3 defines the basic concepts of the method and asserts a theorem, while Section 4 details the algorithms in a general-purpose abstract language. The complexity analysis of the algorithms is provided in Section 5. Section 6 applies the proposed modeling principles to a sample power sub-network. Section 7 discusses the relationship of the proposed approach with relevant works in the literature. Conclusions are drawn in Section 8. Appendix A contains the proof of the theorem asserted in Section 3.

2. Informal presentation of the approach

This section is devoted to the informal presentation of the diagnostic method, which is then formalized in Section 3. Specifically, we present how the active system is characterized and modeled, how histories can be reconstructed from the available observation, and how the diagnoses can eventually be extracted from histories. Furthermore, we show how the reconstruction technique can conveniently be modularized to cope with large active systems.

2.1. Characterization

An active system is a dynamic system that interacts with the external world and, while operating, is either quiescent or reacting. When the system is quiescent, it neither changes its state nor generates any event. If, at an unpredictable time instant, the quiescent system

receives an event (necessarily coming from the external world), it begins reacting to it, in other words, it begins evolving.

During the reaction, the active system may generate events directed to the external world, which, vice versa, may generate events directed to the active system, thereby affecting its subsequent evolution. When the reaction has finished, the active system becomes quiescent again. If the reaction is faulty, one or more of its state transitions are abnormal.

The diagnostic method takes into account the observation of the active system recorded during a reaction, i.e., the sequence of events sent by the active system to an external observer. Diagnosis is performed after the reaction has finished, assuming that

(i) every reaction takes a finite time interval,
(ii) it is possible to ascertain when any reaction has finished, and
(iii) the state of the active system at the beginning of any reaction is known.

2.2. Modeling

An active system is modeled as a set of interconnected components. Each component is endowed with input and output terminals, where each output terminal of a component may be connected with at most one input terminal of another one through a directed connection link. Interactions among components are represented by events produced and received on terminals. The behavioral model of a component involves a (generally nondeterministic) function of the input event and the internal state. The occurrence of an input event gives rise to a state transition and, possibly, to the production of some output events. As such, a component is an FSM that can communicate with neighboring components. Every behavioral model is assumed to be complete, i.e., it incorporates all the possible behaviors of the component, both nominal and faulty.

Fig. 1 represents (on the top) an active system, hereafter denoted by $\mathcal{A}$, incorporating four components, namely C_1, C_2, C_3, and C_4, and five (directed) links L_1, L_2, L_3, L_4, and L_5. C_1, for instance, has three terminals, namely the input terminals I_1 and I_2, and the output terminal O, which are connected with links L_1, L_5, and L_2, respectively. These links allow C_1 to interact with C_2 and C_4.

In the picture two disjoint sub-systems, called clusters, are highlighted, namely ξ_1 and ξ_2. A cluster is a connected sub-graph incorporating one or several components and all the links among them. Cluster ξ_1 contains components C_1 and C_2 and links L_1 and L_2, while ξ_2 consists of C_3, C_4, and L_4. The set of links L_3 and L_5, which are neither included in ξ_1 nor in ξ_2, but connect a terminal in ξ_1 with a terminal in ξ_2, is called the interface of the two clusters. The notion of interface can be extended to several disjoint clusters and represents the whole set of links that connect any possible pair of components in two different clusters.

The models of components C_1, C_2, C_3, and C_4 are represented at the bottom of Fig. 1, and called M_1, M_2, M_3, and M_4, respectively. For example, model M_1 is characterized by three states, namely S_{11}, S_{12}, and S_{13}, and four transitions, T_{11}, T_{12}, T_{13}, and T_{14}. Thus, if C_1 is in state S_{11} and event E_5 is ready at terminal I_2 on link L_5, then transition T_{11} can be triggered. This transition can be expressed as $S_{11} \xrightarrow{\alpha|\beta} S_{12}$, where $\alpha = (E_5, I_2)$ and $\beta = \{(E_2, O), (A_1, Msg)\}$. This means that T_{11} is triggered by the input event α and generates the set of output events in β. In particular, E_2 is buffered in link L_2, while

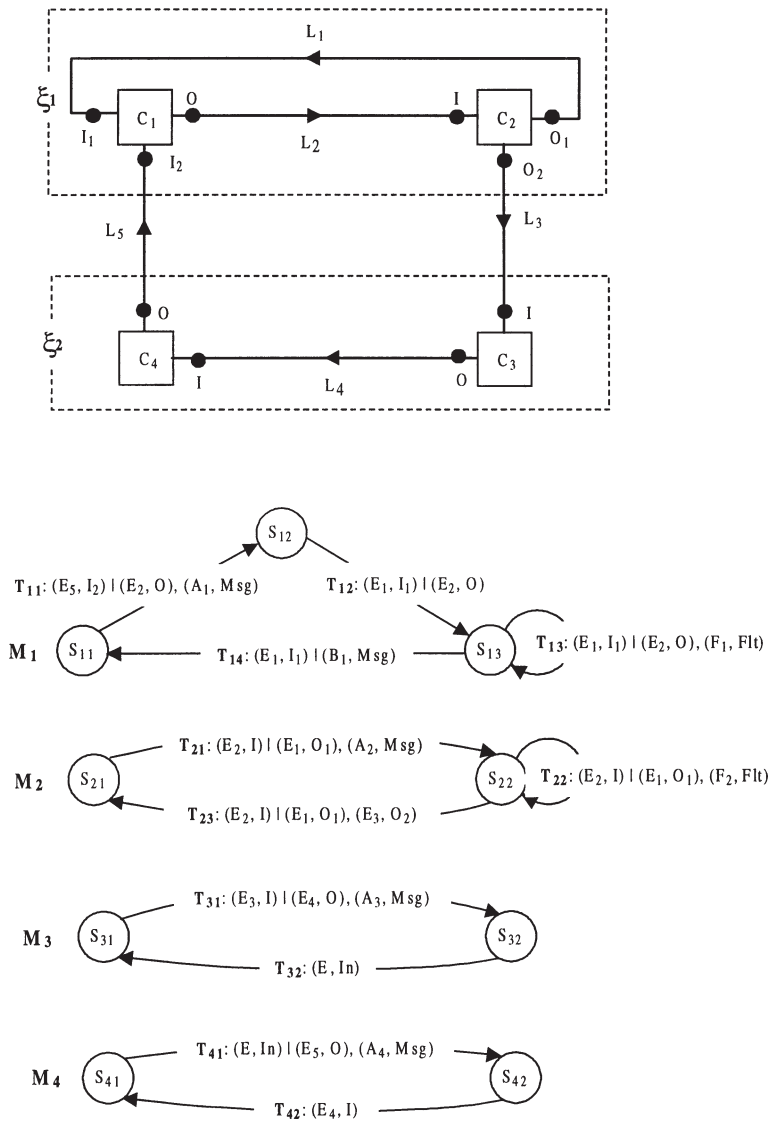

Fig. 1. An active system and the relevant component models.

event (A_1, Msg) is a message. Every link is modeled as a queue of finite capacity. Each component may be endowed with any of four virtual terminals, namely *In*, *Out*, *Msg*, and *Flt*. *In* and *Out* are called the standard input and the standard output, respectively, and are meant to connect the component with the external world. *Msg* is the message terminal, which is assumed to be linked with the external observer. Only events on *Msg* are observable. A transition incorporating a message is observable, otherwise it is silent. *Flt* is the fault terminal, which is a modeling artifice to specify faulty transitions. For

example, T_{13}, since generating event (F_1, Flt), is a faulty transition characterized by fault F_1 concerning component C_1.

2.3. Reaction

When the active system is quiescent, no events are buffered within its links. A quiescent system becomes reacting on the arrival of an event on the standard input of a component. During the reaction, the active system generates events directed to the external world and/or to the system itself. The reaction is characterized by a finite number of messages generated by the observable transitions, namely the observation of the system, which is the composition of a set of sequences of messages, each of which is relevant to a single component. For instance, $Obs(\mathcal{A}) = (obs(C_1), \ldots, obs(C_4))$, where $obs(C_i)$ is a totally temporally ordered sequence of messages relevant to C_i, such as $obs(C_1) = \langle A_1, B_1 \rangle$. No ordering constraints are assumed among messages of different components.

During a reaction, each component undergoes a sequence of transitions, called a history, which connect its initial state with a final state. For example, assuming that the initial state of C_1 is S_{11}, a history of C_1 might be $\langle T_{11}, T_{12}, T_{14} \rangle$, which in this case can be written without ambiguity as $S_{11} \rightarrow S_{12} \rightarrow S_{13} \rightarrow S_{14}$.

The projection of a history on relevant messages is the observation of the component according to that history. In our example, the projection of the history of C_1 equals the observation $obs(C_1) = \langle A_1, B_1 \rangle$ given above. Observations constrain the way in which histories can be hypothesized for each component involved in the reaction. Roughly, the richer the set of observable transitions in a model, the more constrained the set of the histories consistent with the observation. For example, since model M_1 includes a cycle corresponding to the silent transition T_{13}, there is an infinite number of histories of C_1 consistent with $obs(C_1) = \langle A_1, B_1 \rangle$. The same applies to model M_2, which involves transition T_{22}. Every observation of the relevant components of models M_3 and M_4, instead, always gives rise to a finite number of consistent histories, since no silent cycles are involved. For example, assuming that the initial state of C_3 is S_{31}, the histories consistent with $obs(C_3) = \langle A_3 \rangle$ are $S_{31} \rightarrow S_{32}$ and $S_{31} \rightarrow S_{32} \rightarrow S_{31}$.

Two remarks are worthwhile. First, even if there is an unlimited number of histories consistent with the observation of a component, they can always be represented by means of a finite graph. For example, the infinite number of histories of C_1 consistent with $obs(C_1) = \langle A_1, B_1 \rangle$ is a graph isomorphic to the regular expression $T_{11}T_{12}T_{13}^*T_{14}$, where the star operator denotes zero or more instances.

Second, the consistency of each component history with its own component observation is a necessary (but, in general, not sufficient) condition for the global consistency of histories with the system observation. In fact, it does not account for the additional constraints imposed by the topology of the active system, called interface constraints, which affect the interaction of components during the reaction. In particular, the composition of the component histories has to respect the constraint on the state at the end of the reaction, which is assumed to be quiescent, that is, a state in which all the events on the internal links have been consumed. Intuitively, the larger the number of links and exchanged events among components, the more stringent the interface constraints.

A history of an active system is a sequence of transitions, each one relevant to a component. Considering our example, a history of $\mathcal{A}$ is $H = \langle T_{41}, T_{11}, T_{21}, T_{12}, T_{23}, T_{14}, T_{31}, T_{42}\rangle$, where S_{11}, S_{21}, S_{31}, and S_{41} are assumed to be the initial states of components C_1, C_2, C_3, and C_4, respectively. The order among transitions in the system history is essential for the satisfaction of the interface constraints.

By projecting H on the transitions relevant to a specific component, one can extract the single component histories. For example, to find the history of C_1 within the reaction specified by H, the transitions relevant to M_1 have to be isolated, thereby obtaining $h_1 = \langle T_{11}, T_{12}, T_{14}\rangle$, which is a special instance of the regular expression given above (transition T_{13} is skipped). The same applies to C_2, C_3, and C_4 with the following results: $h_2 = \langle T_{21}, T_{23}\rangle$, $h_3 = \langle T_{31}\rangle$, and $h_4 = \langle T_{41}, T_{42}\rangle$. It is easy to show that all these histories are consistent with the observation of the corresponding component in $Obs(\mathcal{A}) = (\langle A_1, B_1\rangle, \langle A_2\rangle, \langle A_3\rangle, \langle A_4\rangle)$.

Table 1 shows that the system history H given above is globally consistent with $Obs(\mathcal{A})$, by following step by step its dynamics. Each row of the table represents a state of the active system, which is identified by a label (e.g., $Id = 3$), the composition of the states of the components, (e.g., $(S_{11}, S_{21}, S_{31}, S_{41})$), the composition of the partial observations of the components, (e.g., $(\langle A_1\rangle, \langle A_2\rangle, \emptyset, \langle A_4\rangle)$), and the composition of the status of the links, (e.g., $(\langle E_1\rangle, \emptyset, \langle E_3\rangle, \emptyset, \emptyset)$). The first triggered transition is T_{41}, which generates an event on link L_5 and message A_4 relevant to $obs(C_4)$. This information is represented in the second row of the table. Note that the state of C_4 is now S_{42}. The second transition is T_{11}, which is triggered by event E_5 stored in link L_5 by the previous transition. The triggering of T_{11} brings the system into a new global state where the state of component C_1 is S_{12}, a new message A_1 is generated for $obs(C_1)$, event E_5 is consumed, and a new internal event E_2 is buffered in link L_2. This mechanism continues until the system reaches state 9, in which all the messages relevant to the given observation have been produced and all the links are empty. This corresponds to the new quiescent state of the active system. As such, H is actually a system history consistent with both the observation and the interface constraints.

Table 1
The dynamics of the system history $H = \langle T_{41}, T_{11}, T_{21}, T_{12}, T_{23}, T_{14}, T_{31}, T_{42}\rangle$

Id	S_1	S_2	S_3	S_4	$obs(C_1)$	$obs(C_2)$	$obs(C_3)$	$obs(C_4)$	L_1	L_2	L_3	L_4	L_5
1	S_{11}	S_{21}	S_{31}	S_{41}	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
2	S_{11}	S_{21}	S_{31}	S_{42}	$\emptyset$	$\emptyset$	$\emptyset$	$\langle A_4\rangle$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\langle E_5\rangle$
3	S_{12}	S_{21}	S_{31}	S_{42}	$\langle A_1\rangle$	$\emptyset$	$\emptyset$	$\langle A_4\rangle$	$\emptyset$	$\langle E_2\rangle$	$\emptyset$	$\emptyset$	$\emptyset$
4	S_{12}	S_{22}	S_{31}	S_{42}	$\langle A_1\rangle$	$\langle A_2\rangle$	$\emptyset$	$\langle A_4\rangle$	$\langle E_1\rangle$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
5	S_{13}	S_{22}	S_{31}	S_{42}	$\langle A_1\rangle$	$\langle A_2\rangle$	$\emptyset$	$\langle A_4\rangle$	$\emptyset$	$\langle E_2\rangle$	$\emptyset$	$\emptyset$	$\emptyset$
6	S_{13}	S_{21}	S_{31}	S_{42}	$\langle A_1\rangle$	$\langle A_2\rangle$	$\emptyset$	$\langle A_4\rangle$	$\langle E_1\rangle$	$\emptyset$	$\langle E_3\rangle$	$\emptyset$	$\emptyset$
7	S_{11}	S_{21}	S_{31}	S_{42}	$\langle A_1, B_1\rangle$	$\langle A_2\rangle$	$\emptyset$	$\langle A_4\rangle$	$\emptyset$	$\emptyset$	$\langle E_3\rangle$	$\emptyset$	$\emptyset$
8	S_{11}	S_{21}	S_{32}	S_{42}	$\langle A_1, B_1\rangle$	$\langle A_2\rangle$	$\langle A_3\rangle$	$\langle A_4\rangle$	$\emptyset$	$\emptyset$	$\emptyset$	$\langle E_4\rangle$	$\emptyset$
9	S_{11}	S_{21}	S_{32}	S_{41}	$\langle A_1, B_1\rangle$	$\langle A_2\rangle$	$\langle A_3\rangle$	$\langle A_4\rangle$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$

2.4. History reconstruction

The central task of the proposed diagnostic technique, namely interpretation, is aimed to generate a representation of all the histories of an active system that are rooted in a known initial state and consistent with a given observation. To this end, first a naive, monolithic approach is considered, which is nevertheless useful for the sake of conceptual clearness and simplicity. This approach is then refined in Section 2.6, where a modular technique is presented.

As above, for $\mathcal{A}$ it is assumed that the observation is $Obs(\mathcal{A}) = (\langle A_1, B_1\rangle, \langle A_2\rangle, \langle A_3\rangle, \langle A_4\rangle)$ and the initial system state is $(S_{11}, S_{21}, S_{31}, S_{41})$. The interpretation task is expected to generate a graph (actually an FSM) in which each path connecting the root with a node corresponding to a quiescent state of the system represents a history consistent with $Obs(\mathcal{A})$.

Fig. 2 represents the search space of the *Interpreter* algorithm of Section 4. The dashed part of the graph corresponds to unsuccessful explorations. The remaining part constitutes the actual interpretation result. Bold nodes, namely N_1, which is the initial system state, N_{23} and N_{25}, represent quiescent system states. Thus, all the paths starting in N_1 and ending in N_{23} or N_{25} are system histories consistent with $Obs(\mathcal{A})$. Due to the cycle $N_5 \rightarrow N_6 \rightarrow N_5$, there exists an unlimited number of such histories.

Each node is explained in Table 2, which is structurally identical to Table 1, apart from the fact that the columns corresponding to component observations are replaced by the integer values $k_1, \ldots, k_4$. This is only a shorthand, such that k_i is the number of messages produced by C_i within the history up to the node. If a message does not match the corresponding message in the given observation $obs(C_i)$, the state is inconsistent.

Intuitively, the *Interpreter* algorithm, which takes as input the observation $Obs(\mathcal{A})$ and the initial (quiescent) system state, makes a depth-first search of the possible sequences of transitions from the initial node to a quiescent node. In doing so, it stores the visited nodes in a structure $\aleph$, where each node is either marked as consistent, inconsistent, or unknown. A consistent node is part of the interpretation, whereas inconsistent nodes are not. Unknown nodes correspond to temporary uncertainty that is eventually resolved before the end of the interpretation. When the interpretation process is over, each node is either qualified as consistent or inconsistent. Table 2 is the tabular representation of the nodes in $\aleph$.

Considering the example, the only transition that can be fired in the initial system state is T_{41}, which is triggered by the external event E. This transition brings the system into the new state N_2, where component C_4 has changed state from S_{41} to S_{42}, and an event E_5 is generated on link L_5. Generally speaking, there are several different transitions which can be triggered in a given node, but this is not the case for nodes $N_1, \ldots, N_4$. In N_5, two different transitions, that incidentally belong to the same model M_2, are ready to be triggered by input event E_2, namely T_{22} and T_{23}. The interpretation process has to consider all the alternatives. Assume that it first tries T_{22}, which leads the system to node N_6, where event E_2 has been consumed and E_1 generated on link L_1. In N_6, there are two alternatives still, T_{13} and T_{14}. The former moves the system to the already visited node N_5, while the latter brings the system to the inconsistent node N_7. A node N is inconsistent either when:

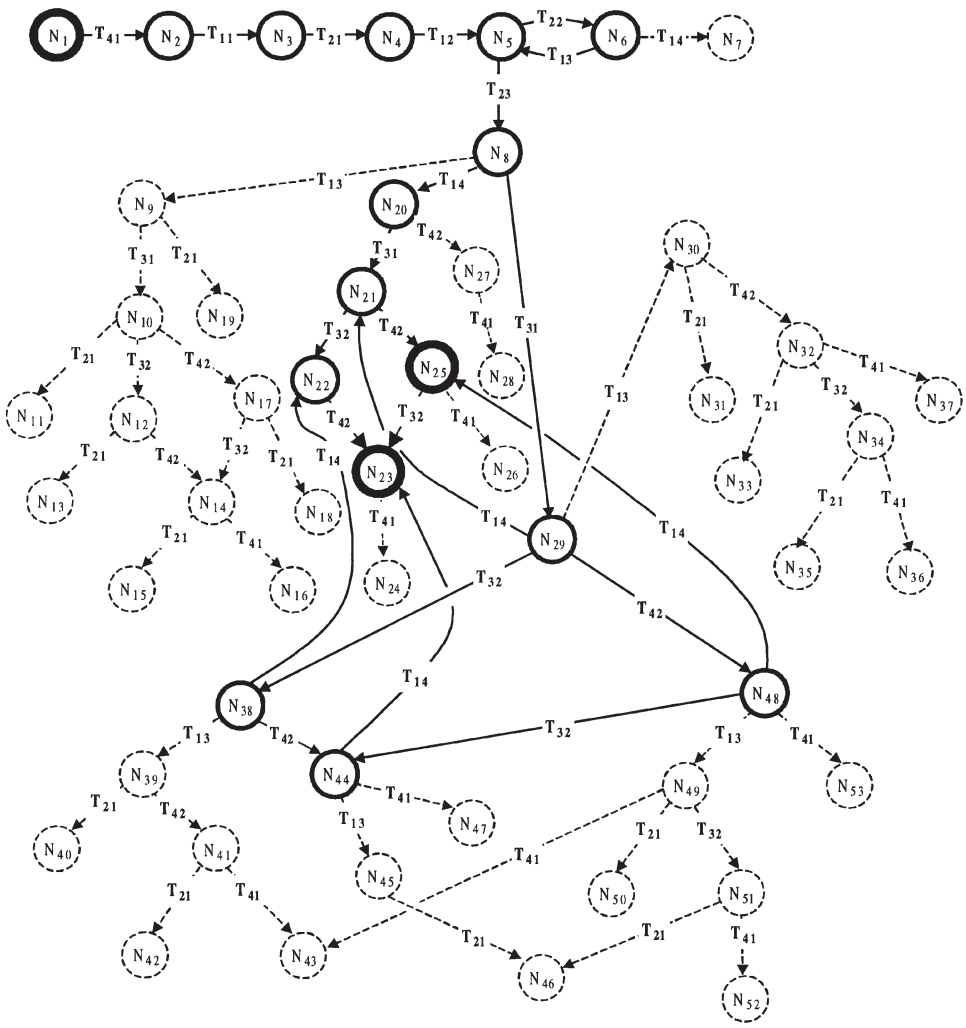

Fig. 2. The search space of the interpretation of the active system outlined in Fig. 1.

(1) N is stalling (no further transitions can be triggered) and there exists at least one component observation still incomplete (e.g., N_7);
(2) N is stalling, but the links are not empty, that is the node is not quiescent;
(3) the transition that led the system to N is observable, but the relevant message does not match the corresponding message in the given observation;
(4) the transition that led the system to N is observable, but the relevant component observation is complete already (e.g., N_{19}); or
(5) all the successive nodes of N are inconsistent (e.g., N_9).

The use of the "unknown" qualification for a node is typically useful as long as not all of its successors have been visited. This is the case, for example, for N_{30}, whose inconsistency

Table 2
Node details relevant to Fig. 2

N	S_1	S_2	S_3	S_4	k_1	k_2	k_3	k_4	L_1	L_2	L_3	L_4	L_5
N_1	S_{11}	S_{21}	S_{31}	S_{41}	0	0	0	0	∅	∅	∅	∅	∅
N_2	S_{11}	S_{21}	S_{31}	S_{42}	0	0	0	1	∅	∅	∅	∅	$\langle E_5\rangle$
N_3	S_{12}	S_{21}	S_{31}	S_{42}	1	0	0	1	∅	$\langle E_2\rangle$	∅	∅	∅
N_4	S_{12}	S_{22}	S_{31}	S_{42}	1	1	0	1	$\langle E_1\rangle$	∅	∅	∅	∅
N_5	S_{13}	S_{22}	S_{31}	S_{42}	1	1	0	1	∅	$\langle E_2\rangle$	∅	∅	∅
N_6	S_{13}	S_{22}	S_{31}	S_{42}	1	1	0	1	$\langle E_1\rangle$	∅	∅	∅	∅
N_7	S_{11}	S_{22}	S_{31}	S_{42}	2	1	0	1	∅	∅	∅	∅	∅
N_8	S_{13}	S_{21}	S_{31}	S_{42}	1	1	0	1	$\langle E_1\rangle$	∅	$\langle E_3\rangle$	∅	∅
N_9	S_{13}	S_{21}	S_{31}	S_{42}	1	1	0	1	∅	$\langle E_2\rangle$	$\langle E_3\rangle$	∅	∅
N_{10}	S_{13}	S_{21}	S_{32}	S_{42}	1	1	1	1	∅	$\langle E_2\rangle$	∅	$\langle E_4\rangle$	∅
N_{11}	S_{13}	S_{22}	S_{32}	S_{42}	1	2	1	1	$\langle E_1\rangle$	∅	∅	$\langle E_4\rangle$	∅
N_{12}	S_{13}	S_{21}	S_{31}	S_{42}	1	1	1	1	∅	$\langle E_2\rangle$	∅	$\langle E_4\rangle$	∅
N_{13}	S_{13}	S_{22}	S_{31}	S_{42}	1	2	1	1	$\langle E_1\rangle$	∅	∅	$\langle E_4\rangle$	∅
N_{14}	S_{13}	S_{21}	S_{31}	S_{41}	1	1	1	1	∅	$\langle E_2\rangle$	∅	∅	∅
N_{15}	S_{13}	S_{22}	S_{31}	S_{41}	1	2	1	1	$\langle E_1\rangle$	∅	∅	∅	∅
N_{16}	S_{13}	S_{21}	S_{31}	S_{42}	1	1	1	2	∅	∅	∅	∅	$\langle E_5\rangle$
N_{17}	S_{13}	S_{21}	S_{32}	S_{41}	1	1	1	1	∅	$\langle E_2\rangle$	∅	∅	∅
N_{18}	S_{13}	S_{22}	S_{32}	S_{41}	1	2	1	1	$\langle E_1\rangle$	∅	∅	∅	∅
N_{19}	S_{13}	S_{22}	S_{31}	S_{42}	1	2	0	1	$\langle E_1\rangle$	∅	$\langle E_3\rangle$	∅	∅
N_{20}	S_{11}	S_{21}	S_{31}	S_{42}	2	1	0	1	∅	∅	$\langle E_3\rangle$	∅	∅
N_{21}	S_{11}	S_{21}	S_{32}	S_{42}	2	1	1	1	∅	∅	∅	$\langle E_4\rangle$	∅
N_{22}	S_{11}	S_{21}	S_{31}	S_{42}	2	1	1	1	∅	∅	∅	$\langle E_4\rangle$	∅
N_{23}	S_{11}	S_{21}	S_{31}	S_{41}	2	1	1	1	∅	∅	∅	∅	∅
N_{24}	S_{11}	S_{21}	S_{31}	S_{42}	3	1	1	1	∅	∅	∅	∅	$\langle E_5\rangle$
N_{25}	S_{11}	S_{21}	S_{32}	S_{41}	2	1	1	1	∅	∅	∅	∅	∅
N_{26}	S_{11}	S_{21}	S_{32}	S_{42}	3	1	1	1	∅	∅	∅	∅	$\langle E_5\rangle$
N_{27}	S_{11}	S_{21}	S_{31}	S_{41}	2	1	0	1	∅	∅	∅	$\langle E_4\rangle$	∅
N_{28}	S_{11}	S_{21}	S_{31}	S_{42}	3	1	0	1	∅	∅	∅	$\langle E_4\rangle$	$\langle E_5\rangle$
N_{29}	S_{13}	S_{21}	S_{32}	S_{42}	1	1	1	1	$\langle E_1\rangle$	∅	∅	$\langle E_4\rangle$	∅
N_{30}	S_{13}	S_{21}	S_{32}	S_{42}	2	1	1	1	∅	$\langle E_2\rangle$	∅	$\langle E_4\rangle$	∅
N_{31}	S_{13}	S_{22}	S_{32}	S_{42}	2	2	1	1	$\langle E_1\rangle$	∅	∅	$\langle E_4\rangle$	∅
N_{32}	S_{13}	S_{21}	S_{32}	S_{41}	2	1	1	1	∅	$\langle E_2\rangle$	∅	∅	∅
N_{33}	S_{13}	S_{22}	S_{32}	S_{41}	2	2	1	1	$\langle E_1\rangle$	∅	∅	∅	∅
N_{34}	S_{13}	S_{21}	S_{31}	S_{41}	2	1	1	1	∅	$\langle E_2\rangle$	∅	∅	∅
N_{35}	S_{13}	S_{22}	S_{31}	S_{41}	2	2	1	1	$\langle E_1\rangle$	∅	∅	∅	∅
N_{36}	S_{13}	S_{21}	S_{31}	S_{42}	3	1	1	1	∅	$\langle E_2\rangle$	∅	∅	$\langle E_5\rangle$
N_{37}	S_{13}	S_{21}	S_{32}	S_{42}	3	1	1	1	$\langle E_1\rangle$	∅	∅	∅	$\langle E_5\rangle$
N_{38}	S_{13}	S_{21}	S_{31}	S_{42}	1	1	1	1	$\langle E_1\rangle$	∅	∅	$\langle E_4\rangle$	∅
N_{39}	S_{13}	S_{21}	S_{31}	S_{42}	1	1	1	1	∅	$\langle E_2\rangle$	∅	$\langle E_4\rangle$	∅
N_{40}	S_{13}	S_{22}	S_{31}	S_{42}	1	2	1	1	$\langle E_1\rangle$	∅	∅	$\langle E_4\rangle$	∅
N_{41}	S_{13}	S_{21}	S_{31}	S_{41}	1	1	1	1	∅	$\langle E_2\rangle$	∅	∅	∅
N_{42}	S_{13}	S_{22}	S_{31}	S_{41}	1	2	1	1	$\langle E_1\rangle$	∅	∅	∅	∅
N_{43}	S_{13}	S_{21}	S_{31}	S_{42}	1	1	1	2	∅	$\langle E_2\rangle$	∅	∅	$\langle E_5\rangle$
N_{44}	S_{13}	S_{21}	S_{31}	S_{41}	1	1	1	1	$\langle E_1\rangle$	∅	∅	∅	∅
N_{45}	S_{13}	S_{21}	S_{31}	S_{42}	1	1	1	1	∅	$\langle E_2\rangle$	∅	∅	∅
N_{46}	S_{13}	S_{22}	S_{31}	S_{42}	1	2	1	1	$\langle E_1\rangle$	∅	∅	∅	∅
N_{47}	S_{11}	S_{21}	S_{31}	S_{42}	1	1	1	2	$\langle E_1\rangle$	∅	∅	∅	$\langle E_5\rangle$
N_{48}	S_{13}	S_{21}	S_{32}	S_{41}	1	1	1	1	$\langle E_1\rangle$	∅	∅	∅	∅
N_{49}	S_{13}	S_{21}	S_{32}	S_{41}	1	1	1	1	∅	$\langle E_2\rangle$	∅	∅	∅
N_{50}	S_{13}	S_{22}	S_{32}	S_{41}	1	2	1	1	$\langle E_1\rangle$	∅	∅	∅	∅
N_{51}	S_{13}	S_{21}	S_{31}	S_{41}	1	1	1	1	∅	$\langle E_2\rangle$	∅	∅	∅
N_{52}	S_{13}	S_{21}	S_{31}	S_{42}	1	1	1	2	∅	$\langle E_2\rangle$	∅	∅	$\langle E_5\rangle$
N_{53}	S_{13}	S_{21}	S_{32}	S_{42}	1	1	1	2	$\langle E_1\rangle$	∅	∅	∅	$\langle E_5\rangle$

can be detected only after nodes $N_{31}, \ldots, N_{37}$ have been visited and thereby qualified as inconsistent. However, depending on the visiting order, there are situations in which a node remains qualified as unknown until the end of the search process. This typically arises in cycles, that make infeasible in depth-first searching the simple rule of deferring the consistency of a node until at least a neighboring node is qualified as consistent. For example, node N_6 is first marked as unknown and is still qualified as unknown when the sub-graph rooted in N_6, including N_7 and N_5, is completely built. [4] As a result, at the end of the exploration of the search space, N_6 is still qualified as unknown.

This is why the resulting graph has to be eventually analyzed and, possibly, pruned to remove those nodes whose consistency is unknown and which are not part of any path connecting the initial state with some quiescent node. On the basis of this condition, N_6 (the only node suffering from this pending qualification) is in the end qualified as consistent.

2.5. Diagnosis generation

Once the interpretation of $Obs(\mathcal{A})$ has been performed, the diagnostic information which is implicitly embodied in the interpretation graph has to be extracted so as to determine possible faulty behaviors in the reaction of the system, which is the final goal of the diagnostic process.

Each history within an interpretation graph is either qualified as faulty, if it includes at least one faulty transition, or nominal, otherwise. In the interpretation graph of Fig. 2, all the faulty histories incorporate both faulty transitions T_{13} and T_{22}.

When considering a system history, the faulty components are those for which there exists a faulty transition at least in the history. The set of faulty components is empty if the history is nominal. A shallow diagnosis, or simply a diagnosis, is the (possibly empty) set of faulty components relevant to a history of the interpretation graph. In the interpretation graph outlined in Fig. 2, only two different diagnoses can be extracted, namely $\{C_1, C_2\}$ and $\emptyset$ (the empty set).

Even though the number of histories represented in an interpretation graph may be infinite (as in our example), the number of distinct diagnoses is always finite as the number of components is finite. Actually, what is relevant to a diagnosis is not the history but rather the set of nodes and edges of the interpretation graph traversed by the history, which is called a route. There is a one-to-many relationship between a route and the relevant histories. Since all the histories associated with the same route are either faulty or nominal, a route is either classified as faulty or nominal. A faulty route is called an explanation. Several different routes may give rise to the same diagnosis. In our example, there is a single route corresponding to the diagnosis $\{C_1, C_2\}$, but several distinct routes lead to the same empty diagnosis. This shows that a one-to-many relationship also exists between a diagnosis and the relevant routes.

The notion of diagnosis introduced above is rather coarse-grained as the interpretation graph and the component models together incorporate further information about faulty components. A faulty component involves a set of distinct faulty transitions in the same

[4] Remember that N_5 has been created already when considering the possible transitions from N_6 and, thus, transition $N_5 \rightarrow N_8$ is created only after the sub-graph rooted in N_6 has been completed.

history and, thereby, a set of faulty events. This is why the additional concept of deep diagnosis is introduced, this being a sequence where each element is an association between a faulty component and the set of relevant faulty events. Considering our example, the only deep diagnosis is $\{(C_1, \{F_1\}), (C_2, \{F_2\})\}$.

There exists a one-to-many relationship between a deep diagnosis and the corresponding explanations. Furthermore, as a shallow nonempty diagnosis is obtained by projecting a deep diagnosis on the relevant components, there is also a one-to-many relationship between a shallow diagnosis and the corresponding deep diagnoses.

The notions of shallow diagnosis, deep diagnosis, and explanation, which represent, at different abstraction levels, the diagnostic information distilled from the interpretation graph and the component models, can be accommodated within a so-called diagnostic hierarchy. A tabular representation of the diagnostic hierarchy relevant to our example is given in Table 3, where Δ is the set of shallow diagnoses, Δ^d the set of deep diagnoses, Ψ the set of explanations, and φ the mapping function from explanations to deep diagnoses and from deep diagnoses to shallow diagnoses. The star operator denotes cycles in routes.

Table 3
The diagnostic hierarchy $(\Delta, \Delta^d, \Psi, \varphi)$ relevant to the interpretation graph outlined in Fig. 2

Item	Value
Δ	$\{\delta_1, \delta_2\}$
Δ^d	$\{\delta^d\}$
Ψ	$\{\psi_1, \ldots, \psi_{11}\}$
φ	$\{\delta^d \mapsto \delta_i$ such that $i \in \{1,2\}\} \cup \{\psi_j \mapsto \delta^d$ such that $j \in \{1,\ldots,11\}\}$
δ_1	$\emptyset$
δ_2	$\{C_1, C_2\}$
δ^d	$\{(C_1, \{F_1\}), (C_2, \{F_2\})\}$
ψ_1	$N_1 \xrightarrow{T_{41}} N_2 \xrightarrow{T_{11}} N_3 \xrightarrow{T_{21}} N_4 \xrightarrow{T_{12}} N_5(\xrightarrow{T_{22}} N_6 \xrightarrow{T_{13}} N_5)^* \xrightarrow{T_{23}} N_8 \xrightarrow{T_{14}} N_{20} \xrightarrow{T_{31}} N_{21} \xrightarrow{T_{42}} N_{25}$
ψ_2	$N_1 \xrightarrow{T_{41}} N_2 \xrightarrow{T_{11}} N_3 \xrightarrow{T_{21}} N_4 \xrightarrow{T_{12}} N_5(\xrightarrow{T_{22}} N_6 \xrightarrow{T_{13}} N_5)^* \xrightarrow{T_{23}} N_8 \xrightarrow{T_{14}} N_{20} \xrightarrow{T_{31}} N_{21} \xrightarrow{T_{42}} N_{25} \xrightarrow{T_{32}} N_{23}$
ψ_3	$N_1 \xrightarrow{T_{41}} N_2 \xrightarrow{T_{11}} N_3 \xrightarrow{T_{21}} N_{22} \xrightarrow{T_{12}} N_5(\xrightarrow{T_{22}} N_6 \xrightarrow{T_{13}} N_5)^* \xrightarrow{T_{23}} N_8 \xrightarrow{T_{14}} N_{20} \xrightarrow{T_{31}} N_{21} \xrightarrow{T_{32}} N_{22} \xrightarrow{T_{42}} N_{23}$
ψ_4	$N_1 \xrightarrow{T_{41}} N_2 \xrightarrow{T_{11}} N_3 \xrightarrow{T_{21}} N_4 \xrightarrow{T_{12}} N_5(\xrightarrow{T_{22}} N_6 \xrightarrow{T_{13}} N_5)^* \xrightarrow{T_{23}} N_8 \xrightarrow{T_{31}} N_{29} \xrightarrow{T_{14}} N_{21} \xrightarrow{T_{42}} N_{25}$
ψ_5	$N_1 \xrightarrow{T_{41}} N_2 \xrightarrow{T_{11}} N_3 \xrightarrow{T_{21}} N_4 \xrightarrow{T_{12}} N_5(\xrightarrow{T_{22}} N_6 \xrightarrow{T_{13}} N_5)^* \xrightarrow{T_{23}} N_8 \xrightarrow{T_{31}} N_{29} \xrightarrow{T_{14}} N_{21} \xrightarrow{T_{42}} N_{25} \xrightarrow{T_{32}} N_{23}$
ψ_6	$N_1 \xrightarrow{T_{41}} N_2 \xrightarrow{T_{11}} N_3 \xrightarrow{T_{21}} N_4 \xrightarrow{T_{12}} N_5(\xrightarrow{T_{22}} N_6 \xrightarrow{T_{13}} N_5)^* \xrightarrow{T_{23}} N_8 \xrightarrow{T_{31}} N_{29} \xrightarrow{T_{14}} N_{21} \xrightarrow{T_{32}} N_{22} \xrightarrow{T_{42}} N_{23}$
ψ_7	$N_1 \xrightarrow{T_{41}} N_2 \xrightarrow{T_{11}} N_3 \xrightarrow{T_{21}} N_4 \xrightarrow{T_{12}} N_5(\xrightarrow{T_{22}} N_6 \xrightarrow{T_{13}} N_5)^* \xrightarrow{T_{23}} N_8 \xrightarrow{T_{31}} N_{29} \xrightarrow{T_{32}} N_{38} \xrightarrow{T_{14}} N_{22} \xrightarrow{T_{42}} N_{23}$
ψ_8	$N_1 \xrightarrow{T_{41}} N_2 \xrightarrow{T_{11}} N_3 \xrightarrow{T_{21}} N_4 \xrightarrow{T_{12}} N_5(\xrightarrow{T_{22}} N_6 \xrightarrow{T_{13}} N_5)^* \xrightarrow{T_{23}} N_8 \xrightarrow{T_{31}} N_{29} \xrightarrow{T_{32}} N_{38} \xrightarrow{T_{42}} N_{44} \xrightarrow{T_{14}} N_{23}$
ψ_9	$N_1 \xrightarrow{T_{41}} N_2 \xrightarrow{T_{11}} N_3 \xrightarrow{T_{21}} N_4 \xrightarrow{T_{12}} N_5(\xrightarrow{T_{22}} N_6 \xrightarrow{T_{13}} N_5)^* \xrightarrow{T_{23}} N_8 \xrightarrow{T_{31}} N_{29} \xrightarrow{T_{42}} N_{48} \xrightarrow{T_{32}} N_{44} \xrightarrow{T_{14}} N_{23}$
ψ_{10}	$N_1 \xrightarrow{T_{41}} N_2 \xrightarrow{T_{11}} N_3 \xrightarrow{T_{21}} N_4 \xrightarrow{T_{12}} N_5(\xrightarrow{T_{22}} N_6 \xrightarrow{T_{13}} N_5)^* \xrightarrow{T_{23}} N_8 \xrightarrow{T_{31}} N_{29} \xrightarrow{T_{42}} N_{48} \xrightarrow{T_{14}} N_{25}$
ψ_{11}	$N_1 \xrightarrow{T_{41}} N_2 \xrightarrow{T_{11}} N_3 \xrightarrow{T_{21}} N_4 \xrightarrow{T_{12}} N_5(\xrightarrow{T_{22}} N_6 \xrightarrow{T_{13}} N_5)^* \xrightarrow{T_{23}} N_8 \xrightarrow{T_{31}} N_{29} \xrightarrow{T_{42}} N_{48} \xrightarrow{T_{14}} N_{25} \xrightarrow{T_{32}} N_{23}$

2.6. Modular reconstruction

The interpretation technique informally introduced in its naive form in Section 2.4 is not appropriate for real, large-scale active systems, as the search space might become too large. Although system $\mathcal{A}$ is very small, it is symptomatic that, out of 52 visited nodes, 36 nodes are inconsistent, which is a considerable percentage. On the basis of the experience of the authors in large-scale application domains [2,3,25], there is evidence that the generation of the interpretation graph can be carried out more efficiently using a modular approach, which is also appropriate for implementation on a parallel processing architecture.

To show how the modular interpretation works, let us consider again system $\mathcal{A}$ with the same observation and initial state. The central point is that now the active system is considered no longer as a whole, but rather as the composition of several clusters and relevant interfaces. The idea is to generate the interpretation graph for each cluster and then to merge such interpretations until the whole system is covered.

$\mathcal{A}$ is viewed as the union of the disjoint clusters ξ_1 and ξ_2, along with the relevant interface composed of links L_3 and L_5. The modular technique requires making two (possibly parallel) interpretations for clusters ξ_1 and ξ_2, respectively, and then merging the resulting interpretation graphs into a new graph on the basis of the interface constraints.

The final graph obtained by merging several interpretations is the same as the final graph generated by a monolithic (naive) interpretation of the cluster corresponding to the union of the clusters involved in the merging and the relevant interface (see Theorem 1 in Section 3).

In order to generate the interpretation graphs for clusters ξ_1 and ξ_2, the observation of the system is projected on the two clusters, respectively, thus obtaining $Obs(\xi_1) = (\langle A_1, B_1\rangle, \langle A_2\rangle)$ and $Obs(\xi_2) = (\langle A_3\rangle, \langle A_4\rangle)$, and each cluster is viewed as an active system on its own. Furthermore, each event relevant to a terminal connected with a link belonging to the interface $\{L_3, L_5\}$ is viewed as external with respect to the cluster and, thereby, dealt with the same way as an event on the standard input or output. Thus, input events relevant to $C_1.I_2$ and $C_3.I$ are treated as coming from the standard input, while output events relevant to $C_2.O_2$ and $C_4.O$ are treated as going to the standard output.

With this in mind, the interpretation algorithm for a cluster is identical to the interpretation algorithm for an active system informally introduced in Section 2.4.

Fig. 3 illustrates the interpretation search space relevant to ξ_1. Each node is identified by three fields, namely K, σ, and D, where

(i) K is the status of the observation (left),
(ii) σ the composition of the states of components C_1 and C_2, and
(iii) D the status of the internal links L_1 and L_2 (right).

Each consistent node is renamed within the circle on its left. The notation $\|L\|$ represents the queue of events in link L.

The transition triggered at the initial state is

$$T_{11} = S_{11} \xrightarrow{(E_5, I_2)|(E_2, O), (A_1, Msg)} S_{12},$$

whose input event is relevant to link L_5. T_{11} cannot be the first transition triggered in the history of $\mathcal{A}$, since it requires the presence of an internal event at the initial state of $\mathcal{A}$, which contrasts with the fact that $\mathcal{A}$ is initially quiescent. Thus, T_{11} is only the first

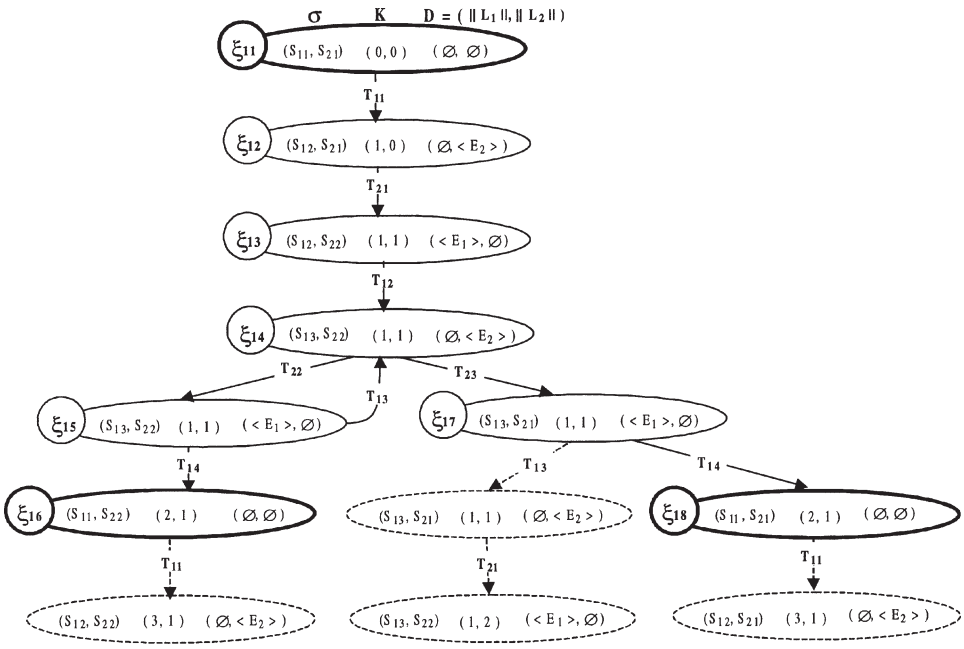

Fig. 3. The search space for the interpretation of cluster ξ_1.

transition relevant to cluster ξ_1 in the reaction of $\mathcal{A}$. The interpretation search space of ξ_2 is shown in Fig. 4.

The interpretation graph relevant to a cluster can be thought of as the model of that cluster, constrained by both the observation and the interface constraints between the components in the cluster. In other words, the cluster can be viewed as a (macro)component whose behavior is regulated by the automaton representing the relevant interpretation, where states are the nodes of the graph and transitions are the edges. The FSM includes also a set of final (quiescent) states. The label of the edges (e.g., T_{11}) are not transition identifiers in the new FSM, but rather the events of the new FSM.

The search space for the merging of the interpretation graphs of ξ_1 and ξ_2, called $\Im_1$ and $\Im_2$, respectively, is represented in Fig. 5. Each node is now identified by the composition of two nodes of $\Im_1$ and $\Im_2$, plus the status of the interface of ξ_1 and ξ_2, namely $\{L_3, L_5\}$.

The merging algorithm first creates the initial node as the composition of the initial nodes of $\Im_1$ and $\Im_2$ together with the empty interface. Then, it looks for a firable transition among those leaving ξ_{11} or ξ_{21}, namely $\{T_{11}, T_{31}, T_{41}\}$. The central point here is distinguishing between transitions triggered by internal events (i.e., events relevant to L_1, L_2, or L_4) and transitions triggered by interface events, these being events buffered either within L_3 or L_5.

In the merging task, the role of internal and external events is somehow inverted: if a transition is enabled by an internal event of either cluster, then it can be fired without any further consideration; if instead the transition is enabled by an interface event, then it can be fired only if its input event is actually the first event in the corresponding link. In our

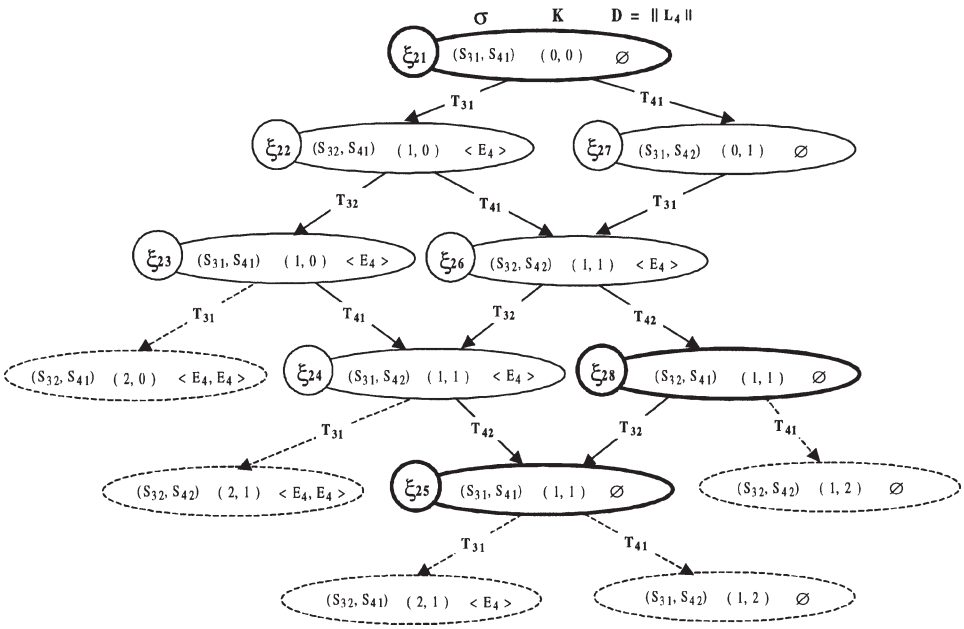

Fig. 4. The search space for the interpretation of cluster ξ_2.

example, given the initial node, the only transition that can be fired is T_{41}, which is relevant to the external input event E, while T_{11} and T_{41} cannot be triggered since they need events E_5 in L_5 and E_3 in L_3, respectively. The triggering of T_{41} causes the generation of event E_5 on link L_5, thereby making T_{11} the next firable transition.

Within the merging process, a node N is inconsistent either when:

(1) N is stalling (no further transitions can be triggered) but the interface links are not empty;
(2) N is stalling but not all the interpretation nodes it merges are final (e.g., node (ξ_{16}, ξ_{27}) in Fig. 6, where ξ_{27} is not final in $\Im_2$); or
(3) All the successive nodes of N are inconsistent.

In Fig. 5, faulty transitions (T_{13} and T_{22}) are denoted by dotted lines. For each consistent node of Fig. 5, Table 4 gives the details extracted from the corresponding nodes in $\Im_1$ and $\Im_2$. The last column of the table indicates for each node $\mathcal{A}_i$ the corresponding node N_j in the interpretation graph outlined in Fig. 2. It is easy to verify that Table 4 equals the selection of the successful nodes in Table 2 and that the graph of consistent nodes of Fig. 5 is isomorphic to the successful part of the interpretation graph depicted in Fig. 2. In other words, consistently with Theorem 1 in Section 3, the result of the merging of $\Im_1$ and $\Im_2$ equals the (monolithic) interpretation of the whole active system.

In the modular reconstruction process, the constraints imposed by the messages and the interconnections among components are first enforced locally, so that the size of the resulting graph is reduced before a merging takes place. The advantage of the modular approach over the monolithic interpretation depends on the clusterization policy.

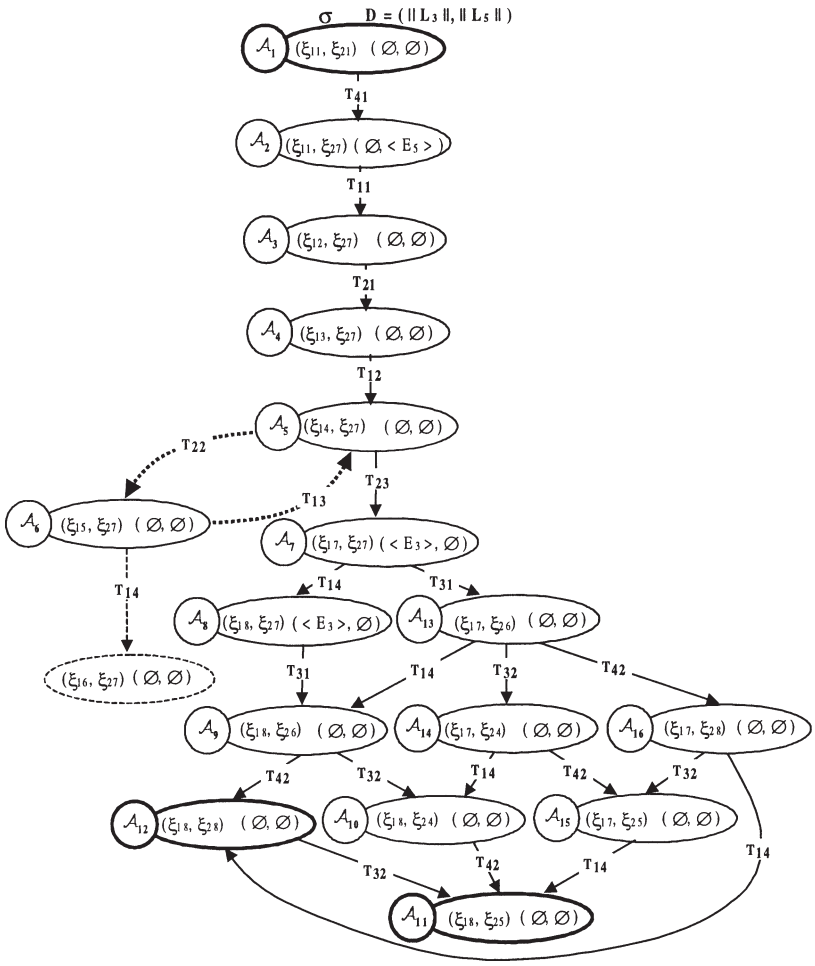

Fig. 5. The search space for the merging of the interpretations of clusters ξ_1 and ξ_2.

Intuitively, the tighter the interconnections among components within clusters, the greater the benefit of the modularity.

Further advantages in using the modular approach rather than the monolithic one can be envisaged:

(1) interpretation and merging algorithms can be run in parallel, thereby reducing the total computation time;
(2) under the hypothesis of a distributed diagnostic architecture, the possibly overwhelming set of messages generated during real large-scale reactions can be split into several sub-sets that are considered separately;
(3) the diagnosis may be shown in different steps: at each step the representation of component histories is refined as well as the diagnosis;

Table 4
Node details relevant to Fig. 5

$\mathcal{A}$	S_1	S_2	S_3	S_4	k_1	k_2	k_3	k_4	L_1	L_2	L_3	L_4	L_5	N
$\mathcal{A}_1$	S_{11}	S_{21}	S_{31}	S_{41}	0	0	0	0	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	N_1
$\mathcal{A}_2$	S_{11}	S_{21}	S_{31}	S_{42}	0	0	0	1	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\langle E_5\rangle$	N_2
$\mathcal{A}_3$	S_{12}	S_{21}	S_{31}	S_{42}	1	0	0	1	$\emptyset$	$\langle E_2\rangle$	$\emptyset$	$\emptyset$	$\emptyset$	N_3
$\mathcal{A}_4$	S_{12}	S_{22}	S_{31}	S_{42}	1	1	0	1	$\langle E_1\rangle$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	N_4
$\mathcal{A}_5$	S_{13}	S_{22}	S_{31}	S_{42}	1	1	0	1	$\emptyset$	$\langle E_2\rangle$	$\emptyset$	$\emptyset$	$\emptyset$	N_5
$\mathcal{A}_6$	S_{13}	S_{22}	S_{31}	S_{42}	1	1	0	1	$\langle E_1\rangle$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	N_6
$\mathcal{A}_7$	S_{13}	S_{21}	S_{31}	S_{42}	1	1	0	1	$\langle E_1\rangle$	$\emptyset$	$\langle E_3\rangle$	$\emptyset$	$\emptyset$	N_8
$\mathcal{A}_8$	S_{11}	S_{21}	S_{31}	S_{42}	2	1	0	1	$\emptyset$	$\emptyset$	$\langle E_3\rangle$	$\emptyset$	$\emptyset$	N_{20}
$\mathcal{A}_9$	S_{11}	S_{21}	S_{32}	S_{42}	2	1	1	1	$\emptyset$	$\emptyset$	$\emptyset$	$\langle E_4\rangle$	$\emptyset$	N_{21}
$\mathcal{A}_{10}$	S_{11}	S_{21}	S_{31}	S_{42}	2	1	1	1	$\emptyset$	$\emptyset$	$\emptyset$	$\langle E_4\rangle$	$\emptyset$	N_{22}
$\mathcal{A}_{11}$	S_{11}	S_{21}	S_{31}	S_{41}	2	1	1	1	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	N_{23}
$\mathcal{A}_{12}$	S_{11}	S_{21}	S_{32}	S_{41}	2	1	1	1	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	N_{25}
$\mathcal{A}_{13}$	S_{13}	S_{21}	S_{32}	S_{42}	1	1	1	1	$\langle E_1\rangle$	$\emptyset$	$\emptyset$	$\langle E_4\rangle$	$\emptyset$	N_{29}
$\mathcal{A}_{14}$	S_{13}	S_{21}	S_{31}	S_{42}	1	1	1	1	$\langle E_1\rangle$	$\emptyset$	$\emptyset$	$\langle E_4\rangle$	$\emptyset$	N_{38}
$\mathcal{A}_{15}$	S_{13}	S_{21}	S_{31}	S_{41}	1	1	1	1	$\langle E_1\rangle$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	N_{44}
$\mathcal{A}_{16}$	S_{13}	S_{21}	S_{32}	S_{41}	1	1	1	1	$\langle E_1\rangle$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	N_{48}

(4) the scalability of the approach to any sub-part of the active system is useful whenever the final diagnosis may be found without reconstructing the history of the entire system, but rather a small sub-set of it; and
(5) partial interpretation results can be recorded with a view to reuse, so as to avoid a new computation in both current and successive diagnostic sessions.

Considering the last point, it is not unlikely that distinct clusters having the same topological structure behave similarly in the same reaction or that a cluster behaves similarly across different reactions. When a new reaction occurs, component observations have to be considered so as to find possible matches with the current reaction or previous ones. If there are several clusters sharing the same topological structure, the same observation, and the same initial state in the current reaction, the cluster interpretation can be carried out only once. If a cluster exhibits an observation that was already analyzed in a previous session, its interpretation can be skipped, provided that the initial state of the cluster is the same.

3. Formal specification

In this section, the concepts informally introduced in Section 2 are formally defined. Each concept is introduced by a definition which is then possibly explained and/or extended

with remarks. A theorem is given, which establishes the functional equivalence between the monolithic and the modular approach.

Definition 1 (*Component*). A *component* is the abstraction of a physical device, which is characterized by a model and is possibly connected with other components (see Definitions 2 and 3).

Definition 2 (*Component model*). The *model* M of a component C is a finite state machine, $M = (\mathcal{S}, \mathcal{E}^{in}, \mathcal{I}, \mathcal{E}^{out}, \mathcal{O}, \mathcal{T})$, where $\mathcal{S}$ is the set of states, $\mathcal{E}^{in}$ the set of input events, $\mathcal{I}$ the set of input terminals, $\mathcal{E}^{out}$ the set of output events, $\mathcal{O}$ the set of output terminals, and $\mathcal{T}$ the transition function, $\mathcal{T} : \mathcal{S} \times \mathcal{E}^{in} \times \mathcal{I} \times 2^{\mathcal{E}^{out} \times \mathcal{O}} \mapsto 2^{\mathcal{S}}$.

Remark 1. The codomain of $\mathcal{T}$ is the power set of $\mathcal{S}$, $2^{\mathcal{S}}$. This means that, generally speaking, transitions allow for nondeterminism. A transition T from state S_1 to state S_2 is triggered by an input event $\alpha = (E, I)$ at an input terminal I and in general generates the set of output events $\beta = \{(E_1, O_1), \ldots, (E_n, O_n)\}$ at output terminals $O_1, \ldots, O_n$, respectively. This is denoted by $T = S_1 \xrightarrow{\alpha|\beta} S_2$. When α is available at the relevant input terminal, S_2 is reached instantaneously and the set of output events are generated at the relevant output terminals (α is *consumed*). An output terminal O may appear at most once in β. Four *virtual* terminals are implicitly defined in a model: the *standard input*, $In \in \mathcal{I}$, the *standard output*, $Out \in \mathcal{O}$, the *message terminal*, $Msg \in \mathcal{O}$, and the *fault terminal*, $Flt \in Out$. An output event (E, Msg) is called a *message*. If transition T is labeled with a message, T is *observable*, otherwise T is *silent*. Among events, only messages are visible from the outside of the system. An event (E, Flt) is called a *faulty event* and the relevant transition, a *faulty* transition. A model M is assumed to be complete, that is, it incorporates all possible behaviors.

A *path* $\wp \in M$ is a (possibly empty) sequence of contiguous transitions, $\wp = \langle T_1, \ldots, T_n \rangle$ in M. Alternatively, $\wp$ can be denoted also by

$$S_1 \xrightarrow{\alpha_1|\beta_1} S_2 \xrightarrow{\alpha_2|\beta_2} S_3 \to \cdots \to S_n \xrightarrow{\alpha_n|\beta_n} S_{n+1},$$

where $T_i = S_i \xrightarrow{\alpha_i|\beta_i} S_{i+1}$, $T_i \in \wp$. Furthermore, $S_1 \rightsquigarrow S_{n+1}$ indicates a generic path from S_1 to S_{n+1}. S_1 is called the *root* of the path.

Definition 3 (*Link*). Let C and C' be two components. Let O and I' be, respectively, an output terminal of C and an input terminal of C'. A *link* $L = \langle C.O, C'.I' \rangle$ is a connection through which each event E generated at O at time t is available at I' at time $t' > t$.

Remark 2. No assumptions are made about the duration of interval $[t, t']$. Within the link, E is said to be a *dangling event* in L. Only a limited number of dangling events may exist within L at any time instant. The maximum number of events that can be buffered in L is a characteristic of the link and is called the *capacity* of L. The set of dangling events of L at a given time is denoted by $\|L\|$. The cardinality of $\|L\|$ is denoted by $|L|$. If $|L|$ equals the capacity of L, L is *saturated*. When L is saturated, a new event generated at $C.O$ is lost. If two events (E_1, O) and (E_2, O) are generated at time t_1 and t_2, respectively, $t_1 < t_2$,

then they are available at $C'.I'$ at time t'_1 and t'_2, respectively, where $t'_1 < t'_2$, that is, E_1 cannot overtake E_2 within the same link; consequently, $\|L\|$ is a queue of dangling events in which the ith generated event is denoted by $L[i]$. The first event $L[1]$ is called the *candidate event*, as it is the first that is going to be consumed. When a candidate event α is consumed, it is dequeued from $\|L\|$. Each output terminal is connected with only one input terminal and vice versa. We assume that virtual terminals be connected with the external world through *virtual* links.

Definition 4 (*Active system*). An *active system* $\mathcal{A}$ is a set of components which are connected with each other by means of links among terminals.

Remark 3. Considering Definition 4 and Remark 2, it turns out that there is a functional dependency from a terminal ϑ of a component C in an active system $\mathcal{A}$ to the relevant link L. Therefore, it is possible to unambiguously write $Link(\vartheta)$ to denote the (possibly virtual) link relevant to terminal ϑ. By definition, if $\alpha = (E, \vartheta)$ is an event, $Link(\alpha) = Link(\vartheta)$.

Definition 5 (*Cluster*). A *cluster* of an active system $\mathcal{A}$ is a pair $\xi = (\mathcal{C}, \mathcal{L})$ where $\mathcal{C}$ is a set of components in $\mathcal{A}$ and $\mathcal{L}$ is a set of links in $\mathcal{A}$ such that:

(1) ξ is connected,
(2) if $L = \langle C.O, C'.I\rangle \in \mathcal{L}$, then $C \in \mathcal{C}$ and $C' \in \mathcal{C}$, and
(3) if $C \in \mathcal{C}$ and $C' \in \mathcal{C}$ then every link connecting C and C' is in $\mathcal{L}$.

Remark 4. Let (E, ϑ) be an event relevant to a cluster ξ, where ϑ is a terminal of a component $C \in \xi$ and $Link(\vartheta)$ the relevant link (remind Remark 3). Three cases are possible:

(1) ϑ is a virtual terminal: (E, ϑ) is called an *external* event of ξ;
(2) ϑ is not a virtual terminal and $Link(\vartheta) \in \xi$: (E, ϑ) is called an *internal* event of ξ;
(3) ϑ is not a virtual terminal and $Link(\vartheta) \notin \xi$: (E, ϑ) is called an *interface* event of ξ.

Definition 6 (*Decomposition*). A *decomposition* $\Xi = \{\xi_1, \ldots, \xi_n\}$ of a cluster $\xi = (\mathcal{C}, \mathcal{L})$ is a set of disjoint clusters $\xi_i = (\mathcal{C}_i, \mathcal{L}_i)$ such that $\bigcup_{i\in\{1,\ldots,n\}} \mathcal{C}_i = \mathcal{C}$. The *interface* of Ξ, $Interf(\Xi)$, is the set of links defined as follows: $Interf(\Xi) = \{L \in \mathcal{L}$ such that $L = \langle C_1.O, C_2.I\rangle, C_1 \in \xi_i, C_2 \in \xi_j, \xi_i \in \Xi, \xi_j \in \Xi,\ i \neq j\}$.

Definition 7 (*Dangling set*). Let ξ be a cluster incorporating the set of links $\mathcal{L} = \{L_1, \ldots, L_m\}$. The *dangling set* $Dang(\xi)$ of ξ at a certain instant is the composition of sequences of dangling events relevant to each $L \in \mathcal{L}$, that is, $Dang(\xi) = (\|L_1\|, \ldots, \|L_m\|) = \|\mathcal{L}\|$. If D is a dangling set, $D[L]$ denotes the set of dangling events in L, that is $D[L] = \|L\|$; $D[L, i]$ denotes the ith event in $\|L\| \in D$. The *candidate event set* of a dangling set D, $Cand(D)$, is the set of candidate events in D.

Remark 5. The notion of dangling set can be extended to the interface of a decomposition Ξ, $Interf(\Xi) = \{L_1, \ldots, L_m\}$, as follows: $Dang(Interf(\Xi)) = \|Interf(\Xi)\| = (\|L_1\|, \ldots, \|L_m\|)$.

Definition 8 (*Component observation*). An *observation* $obs(C)$ of a component C having model M, is a (possibly empty) sequence of messages in M, that is, $obs(C) = \langle m_1, \ldots, m_n \rangle$. The number of messages in $obs(C)$ is denoted by $|obs(C)|$. The ith message of $obs(C)$ is denoted by $obs(C)[i]$. A k-partial observation $obs_{\langle k \rangle}(C)$, $0 \leqslant k \leqslant |obs(C)|$, is a projection of $obs(C)$ on its first k messages, that is, $obs_{\langle k \rangle}(C) = \langle m_1, \ldots, m_k \rangle$.

Definition 9 (*Cluster observation*). An observation $Obs(\xi)$ of a cluster ξ incorporating components $C_1, \ldots, C_n$ is the composition of the observations of components in ξ, that is, $Obs(\xi) = (obs(C_1), \ldots, obs(C_n))$. A K-partial cluster observation $Obs_{\langle K \rangle}(\xi)$, $K = \langle k_1, \ldots, k_n \rangle$, is the composition of the k_i-partial component observations in ξ, that is, $Obs_{\langle K \rangle}(\xi) = (obs_{\langle k_1 \rangle}(C_1), \ldots, obs_{\langle k_n \rangle}(C_n))$.

Remark 6. When a cluster ξ incorporates a single component C, the cluster observation becomes the component observation, that is: $Obs(\xi) = (obs(C))$. Also, if $\forall i \in \{1, \ldots, n\}(obs_{\langle k_i \rangle}(C_i) = obs(C_i))$, then $Obs_{\langle K \rangle}(\xi) = Obs(\xi)$.

Definition 10 (*Interpretation*). Let $Obs(\xi)$ be a cluster observation, $\{C_1, \ldots, C_n\}$ the set of components in ξ, $M_i = (\mathcal{S}_i, \mathcal{E}_i^{in}, \mathcal{I}_i, \mathcal{E}_i^{out}, \mathcal{O}_i, \mathcal{T}_i)$ the model of component C_i, $\Sigma = \mathcal{S}_1 \times \cdots \times \mathcal{S}_n$ the Cartesian product of state domains of components, $\mathcal{K}$ the domain of K such that $Obs_{\langle K \rangle}(\xi)$ is a K-partial cluster observation, $\mathcal{D}$ the domain of $Dang(\xi)$, and $\xi_0 = (S_{01}, \ldots, S_{0n})$, where $\forall i \in \{1, \ldots, n\}$ $(S_{0i} \in \mathcal{S}_i)$. The *spurious interpretation* $\Im^s$ of $(Obs(\xi), \xi_0)$ is a finite state machine:

$$\Im^s(Obs(\xi), \xi_0) = \big(\mathcal{S}^s, \mathcal{E}, \mathcal{T}^s, S_0\big), \tag{1}$$

where $\mathcal{S}^s \subseteq \Sigma \times \mathcal{K} \times \mathcal{D}$ is the set of states, $\mathcal{E}$ the set of events, $\mathcal{T}^s$ the transition function, and S_0 the initial state. S_0, $\mathcal{E}$, and $\mathcal{S}^s$ are defined as follows:

$$S_0 = \big(\xi_0, (0 \cdots 0), (\emptyset \cdots \emptyset)\big), \tag{2}$$

$$\mathcal{E} = \bigcup_{i \in \{1, \ldots, n\}} \mathcal{T}_i, \tag{3}$$

$$\mathcal{S}^s = \{S_0\} \cup \big\{S' \text{ such that } S \xrightarrow{T_{ij}} S' \in \mathcal{T}^s\big\}. \tag{4}$$

The transition function $\mathcal{T}^s : \mathcal{S}^s \times \mathcal{E} \mapsto \mathcal{S}^s$ is defined as follows:

$$(\sigma, K, D) \xrightarrow{T_{ij}} \big(\sigma', K', D'\big) \in \mathcal{T}^s$$

if:

(1) $\sigma = (S_1, \ldots, S_{i-1}, S_i, S_{i+1}, \ldots, S_n)$, $\sigma' = (S_1, \ldots, S_{i-1}, S_i', S_{i+1}, \ldots, S_n)$, $T_{ij} = S_i \xrightarrow{\alpha|\beta} S_i' \in \mathcal{T}_i$;

(2) K' is such that:

(a) if T_{ij} is silent, then $K' = K$,

(b) if T_{ij} is observable and $obs(C_i)[K[i]+1] \in \beta$, then

$$K'[i] = K[i] + 1, \quad \forall x \neq i\big(K'[x] = K[x]\big);$$

(3) D' is such that, calling $L_\alpha = Link(\alpha)$ and $\mathcal{L}_\beta = \{L_\beta,$ where $L_\beta = Link(B), B \in \beta, L_\beta$ is not saturated$\}$:
   (a) if $\alpha = (E, \vartheta)$ is an internal event and $D[L_\alpha, 1] = E$, then $\forall x \in \{1 \cdots |D[L_\alpha]| - 1\}$ $(D'[L_\alpha, x] = D[L_\alpha, x+1])$, $|D'[L_\alpha]| = |D[L_\alpha]| - 1$;
   (b) $\forall$ internal event $(E, \vartheta) \in \beta$ $(L_\beta = Link(\vartheta), L_\beta \in \mathcal{L}_\beta, \forall x \in \{1 \cdots |D[L_\beta]|\}$ $(D'[L_\beta, x] = D[L_\beta, x])$, $D'[L_\beta, |D[L_\beta]| + 1] = E$, $|D'[L_\beta]| = |D[L_\beta]| + 1$;
   (c) $\forall L \in \xi$, $L \notin (\{L_\alpha\} \cup \mathcal{L}_\beta)$ $(D'[L] = D[L])$.

A state $S = (\sigma, K, D) \in \mathcal{S}^s$ is *quiescent* if $Obs_{\langle K \rangle}(\xi) = Obs(\xi)$ and $D = (\emptyset \cdots \emptyset)$. A state $S \in \mathcal{S}^s$ is *consistent* if $\exists$ a (possibly empty) path $\wp = S \rightsquigarrow S_q$ in $\Im^s(Obs(\xi), \xi_0)$ such that S_q is quiescent. A transition $S \xrightarrow{T} S' \in \mathcal{T}^s$ is *consistent* if both S and S' are consistent. The *interpretation* $\Im$ of $(Obs(\xi), \xi_0)$ is a finite state machine:

$$\Im(Obs(\xi), \xi_0) = (\mathcal{S}, \mathcal{E}, \mathcal{T}, S_0, \mathcal{S}_f), \tag{5}$$

where $\mathcal{S}$ is the set of states, $\mathcal{E}$ the set of events, $\mathcal{T}$ the transition function, S_0 the initial state, and $\mathcal{S}_f$ the set of final states. $\mathcal{S}$ equals the set of consistent states in $\mathcal{S}^s$, $\mathcal{T}$ the set of consistent transitions in $\mathcal{T}^s$, and $\mathcal{S}_f$ the set of quiescent states in $\mathcal{S}^s$.

Definition 11 (*Merging*). Let $\Xi = \{\xi_1, \ldots, \xi_n\}$ be a decomposition of an active system $\mathcal{A}$, $\Omega = \{\Im(Obs(\xi_1), \xi_{01}), \ldots, \Im(Obs(\xi_n), \xi_{0n})\}$ the set of relevant interpretations, where $\forall i \in \{1, \ldots, n\}$ $(\Im(Obs(\xi_i), \xi_{0i}) = (\mathcal{S}_i, \mathcal{E}_i, \mathcal{T}_i, S_{0i}, \mathcal{S}_{fi}))$, $\Sigma = \mathcal{S}_1 \times \cdots \times \mathcal{S}_n$ the Cartesian product of the state domains of interpretations in Ω, and $\mathcal{D}$ the domain of $Dang(Interf(\Xi))$. The *spurious merging* μ^s of Ω is a finite state machine:

$$\mu^s(\Omega) = \big(\mathcal{S}^s, \mathcal{E}, \mathcal{T}^s, S_0\big), \tag{6}$$

where $\mathcal{S}^s \subseteq \Sigma \times \mathcal{D}$ is the set of states, $\mathcal{E}$ the set of events, $\mathcal{T}^s$ the transition function, and S_0 the initial state. S_0, $\mathcal{E}$, and $\mathcal{S}^s$ are defined as follows:

$$S_0 = \big(S_{01}, \ldots, S_{0n}, (\emptyset \cdots \emptyset)\big), \tag{7}$$

$$\mathcal{E} = \bigcup_{i \in \{1,\ldots,n\}} \mathcal{E}_i, \tag{8}$$

$$\mathcal{S}^s = \{S_0\} \cup \big\{S' \text{ such that } S \xrightarrow{T_{ij}} S' \in \mathcal{T}^s\big\}. \tag{9}$$

Denoting

$$T_{ij} = S_{ij} \xrightarrow{\alpha|\beta} S'_{ij}, \quad T_{ij} \in \mathcal{T}_{ij},$$

where $\mathcal{T}_{ij}$ is the transition function of component $C_{ij} \in \xi_i$, the transition function $\mathcal{T}^s : \mathcal{S}^s \times \mathcal{E} \mapsto \mathcal{S}^s$ is defined as follows:

$$(\sigma, D) \xrightarrow{T_{ij}} \big(\sigma', D'\big) \in \mathcal{T}^s$$

if:

(1) $\sigma = (S_1, \ldots, S_{i-1}, S_i, S_{i+1}, \ldots, S_n)$, $\sigma' = (S_1, \ldots, S_{i-1}, S'_i, S_{i+1}, \ldots, S_n)$, $S_i \xrightarrow{T_{ij}} S'_i \in \mathcal{T}_i$;

(2) D' is such that, calling $L_\alpha = Link(\alpha)$ and $\mathcal{L}_\beta = \{L_\beta,$ where $L_\beta = Link(B), B \in \beta, L_\beta \in Interf(\Xi), L_\beta$ is not saturated$\}$:
   (a) if $\alpha = (E, \vartheta)$ is an interface event and $D[L_\alpha, 1] = E$, then $\forall x \in \{1 \cdots |D[L_\alpha]| - 1\}$ $(D'[L_\alpha, x] = D[L_\alpha, x+1])$, $|D'[L_\alpha]| = |D[L_\alpha]| - 1$;
   (b) $\forall$ interface event $(E, \vartheta) \in \beta$ $(L_\beta = Link(\vartheta), L_\beta \in \mathcal{L}_\beta, D'[L_\beta, |D[L_\beta]| + 1] = E)$, $|D'[L_\beta]| = |D[L_\beta]| + 1$;
   (c) $\forall L \in Interf(\Xi)$, $L \notin (\{L_\alpha\} \cup \mathcal{L}_\beta)$ $(D'[L] = D[L])$.

A state $S = (\sigma, D) \in \mathcal{S}^s$ is *quiescent* if $\forall S_i \in \sigma$ $(S_i \in \mathcal{S}_{f_i})$ and $D = (\emptyset \cdots \emptyset)$. A state $S \in \mathcal{S}^s$ is *consistent* if $\exists$ a (possibly empty) path $\wp = S \rightsquigarrow S_q$ in $\mu^s(\Omega)$ such that S_q is quiescent. A transition $S \xrightarrow{T} S' \in \mathcal{T}^s$ is *consistent* if both S and S' are consistent. The *merging* μ of Ω is a finite state machine:

$$\mu(\Omega) = (\mathcal{S}, \mathcal{E}, \mathcal{T}, S_0, \mathcal{S}_f), \tag{10}$$

where $\mathcal{S}$ is the set of states, $\mathcal{E}$ the set of events, $\mathcal{T}$ the transition function, S_0 the initial state, and $\mathcal{S}_f$ the set of final states. $\mathcal{S}$ equals the set of consistent states in $\mathcal{S}^s$, $\mathcal{T}$ the set of consistent transitions in $\mathcal{T}^s$, and $\mathcal{S}_f$ the set of quiescent states in $\mathcal{S}^s$.

Theorem 1. *Let $\Xi = \{\xi_1, \ldots, \xi_n\}$ be a decomposition of a cluster ξ and*

$$\Omega = \left\{\Im(Obs(\xi_1), \xi_{0_1}), \ldots, \Im(Obs(\xi_n), \xi_{0_n})\right\}$$

the relevant interpretations. Then:

$$\mu(\Omega) = \Im(Obs(\xi), \xi_0), \tag{11}$$

where

$$\xi_0 = \bigcup_{i \in \{1,\ldots,n\}, j \in \{1,\ldots,n_i\}} S_{0ij},$$

where n_i is the number of components in cluster ξ_i, S_{0ij} is the initial state of component $C_{ij} \in \xi_i$, and

$$Obs(\xi) = \bigcup_{i \in \{1,\ldots,n\}} Obs(\xi_i).$$

Definition 12 (*History*). Let $\Im = (\mathcal{S}, \mathcal{E}, \mathcal{T}, S_0, \mathcal{S}_f)$ be an interpretation of $(Obs(\xi), \xi_0)$. A path $\wp \in \Im$, $\wp = S_0 \rightsquigarrow S_f$, $S_f \in \mathcal{S}_f$, is called a *history* of ξ. A *route* in $\Im$ is the subgraph of $\Im$ traversed by a history of $\Im$. The *extent* of a route R, denoted by $|R|$, is the number of edges incorporated in R. A transition

$$S \xrightarrow{T_{ij}} S' \in \mathcal{T}$$

such that T_{ij} is a faulty transition in a component model, is called a faulty transition in $\Im$. A history incorporating a faulty transition is called a *faulty history*. A route incorporating a faulty transition is called a *faulty route*. A component relevant to a faulty transition in a faulty history is called a *faulty component*.

Remark 7. The notion of route differs from the notion of history in that, due to possible cycles, the former implicitly defines a (possibly infinite) class of histories in $\Im$. However, given a history, there is a single corresponding route. Thus, even though the number of histories in $\Im$ is possibly unlimited, the number of different routes in $\Im$ is always finite.

Definition 13 (*Diagnosis*). Let $\Im$ be an interpretation of $(Obs(\xi), \xi_0)$. A *shallow diagnosis* (or, simply, a *diagnosis*) δ of $(Obs(\xi), \xi_0)$ is the set of faulty components relevant to a route $R \in \Im$. A *deep diagnosis* δ^d of $(Obs(\xi), \xi_0)$ is a set of pairs $\{(C_1, \mathcal{F}_1), \ldots, (C_p, \mathcal{F}_p)\}$ where C_i, $i \in \{1, \ldots, p\}$, is a faulty component relevant to a route $R \in \Im$, and $\mathcal{F}_i$ is the set of faulty events in R relevant to C_i. An *explanation* ψ of $(Obs(\xi), \xi_0)$ is a faulty route in $\Im$.

Remark 8. From Definition 13 it follows that the notions of shallow diagnosis, deep diagnosis, explanation, and history can be accommodated within a hierarchical structure. In fact, the same shallow diagnosis δ can be obtained by the restriction of several deep diagnoses on relevant components. Similarly, the same deep diagnosis δ^d can be obtained by the restriction of several faulty routes on faulty transitions, thereby on relevant components and faulty events. This is formalized by the notion of *diagnostic hierarchy* given in Definition 14.

Definition 14 (*Diagnostic hierarchy*). The *diagnostic hierarchy* ∇ of $(Obs(\xi), \xi_0)$, is a 4-tuple:

$$\nabla(Obs(\xi), \xi_0)) = \big(\Delta, \Delta^d, \Psi, \varphi\big), \tag{12}$$

where $\Delta = \{\delta_1, \ldots, \delta_n\}$ is the set of shallow diagnoses of $(Obs(\xi), \xi_0)$, $\Delta^d = \{\delta_1^d, \ldots, \delta_m^d\}$ is the set of deep diagnoses of $(Obs(\xi), \xi_0)$, $\Psi = \{\psi_1, \ldots, \psi_r\}$ is the set of explanations of $(Obs(\xi), \xi_0)$, and φ is the mapping function defined as follows (where ψ, δ^d, and δ belong to Ψ, Δ^d, and Δ, respectively):

(1) $\psi \mapsto \delta^d \in \varphi$, if the restriction of ψ on the faulty events and relevant components equals δ^d, and
(2) $\delta^d \mapsto \delta \in \varphi$ if the restriction of δ^d on the components equals δ.

4. Algorithms

This section is devoted to the specification of the algorithms that implement the operators defined in Section 3. Each algorithm is briefly introduced and then specified in terms of pseudo-code written in an abstract, general-purpose language. Four algorithms are given: *Interpreter*, *Merger*, *Viewer*, and *Diagnoser*. *Interpreter* and *Merger*, which implement operators $\Im$ and μ, respectively, constitute the kernel of the diagnostic engine, as they are meant to generate the interpretation relevant to the whole active system. Based on this reconstruction, the *Viewer* algorithm makes up the corresponding diagnostic hierarchy, thereby implementing the ∇ operator. The "self-contained" diagnostic engine is represented by *Diagnoser*, which generates the diagnostic hierarchy starting from the observation and the initial state of the system.

4.1. Interpreter

The *Interpreter* function corresponding to Algorithm 1 produces as output a graph-based representation of the interpretation $\Im(Obs(\xi, \xi_0))$ by calling function *IStep* recursively. The definition of *IStep* is embedded within the definition of *Interpreter*, between the parameter specification and the body of *Interpreter*, the latter starting at Line 58.

The interpretation graph is generated depth-first. Considering, for example, the interpretation graph outlined in Fig. 3, each node in the graph corresponds to a recursive call of *IStep*. At Line 59, the parameters of the first call of *IStep* are instantiated, namely K, D, and $\aleph$ (ξ_0 is directly given as input to *Interpreter*). Thus, the input parameters of *IStep* are:

(1) σ, the composition of states of components within the cluster,
(2) K, the composition of integers such that $Obs_{\langle K \rangle}(\xi)$ is the K-partial observation of $Obs(\xi)$,
(3) D, the dangling set of ξ, and
(4) $\aleph$, the set of visited nodes ($\aleph$ is also the output parameter, as it is possibly updated at each call of *IStep*).

At the first call, we have $\sigma = \xi_0$, $K = (0 \cdots 0)$, $D = (\emptyset \cdots \emptyset)$, and $\aleph = \emptyset$. At Line 2, the node is inserted into $\aleph$ and marked as unknown.[5] Then, a matching transition is searched for, based on the candidate events, the interface events, and the external events (Lines 3–5). If there exists such a matching transition, σ' is set to denote the new configuration of states of components in ξ after the transition, while K' and D' are the copies of K and D, respectively (Line 8). These are meant to mirror the context of the new node after the transition: if the matching transition is triggered by an internal event, then this is removed from the relevant link (Lines 10–15). In any case, the unsaturated links relevant to the output events are updated appropriately (Lines 16–21).

At this point (Line 22), if the output events of the matching transition include a message, K is updated and a check on the consistency of this message is performed: if the message is inconsistent with the observation of the relevant component (Line 25), then the current iteration of the loop in Line 5 is broken, otherwise the new node $N' = (\sigma', K', D')$ is considered (Line 28). If N' has been visited already, two alternatives are possible: either N' is marked as inconsistent or not. In the first case, variable $N'_\Im$ is set to nil, otherwise $N'_\Im$ is assigned N'. If instead N' has not been visited yet, then *IStep* is called recursively on the new parameters σ', K', D', and the new configuration of $\aleph$ (Line 34).

Thus, at Line 35, the value of $N'_\Im$ can be possibly nil. $N'_\Im$ is set to nil either when:

(1) the new node N' has been visited already and marked as inconsistent (Line 30); or
(2) the recursive call of *IStep* at Line 34 returns an inconsistent node.

If $N'_\Im$ is not nil, N is marked as consistent (Line 38) provided that N is marked as unknown and N' as consistent and, anyhow, a new edge from N to N' is created in the interpretation graph, which is marked by the matching transition T_{ij} (Line 39).

The loop at Line 3 terminates when all the matching transitions have been considered. At this point (Line 42), the current call of *IStep* is expected to return the control to the calling function, namely either *IStep* or *Interpreter*. Before that, a check on the quiescence of node

[5] According to the informal discussion of Section 2.4, a node can be marked either as "consistent", "inconsistent", or "unknown".

N is performed and, if N is quiescent, it is marked as consistent and returned (Line 45). If N is not quiescent (Line 47), N is returned anyway (Line 50), provided that at least an edge has been created at Line 39, otherwise N is marked as inconsistent and a nil value is returned (Line 54).

Lines 62–65 in *Interpreter* remove from the interpretation graph $G_{\Im}$ the possible dangling nodes and edges which are not part of any path from the root of $G_{\Im}$ to a quiescent node. [6]

Lines 66–71 establish the isomorphism between the graph representation of the interpretation $\Im$ and its elements $\mathcal{S}$, $\mathcal{E}$, $\mathcal{T}$, S_0, and $\mathcal{S}_f$. The resulting interpretation is eventually returned at Line 72.

Algorithm 1.

function *Interpreter*($Obs(\xi), \xi_0$): the interpretation $\Im(Obs(\xi), \xi_0) = (\mathcal{S}, \mathcal{E}, \mathcal{T}, S_0, \mathcal{S}_f)$,
$\mathcal{S} \subseteq \Sigma \times \mathcal{K} \times \mathcal{D}$
input
$Obs(\xi) = (obs(C_1), \ldots, obs(C_n))$: the observation of cluster ξ,
$\xi_0 = (S_{01}, \ldots, S_{0n})$: the initial state of cluster ξ;

function *IStep*($\sigma, K, D, \aleph$): the (possibly nil) root of a sub-graph of the interpretation
input
$\sigma = (S_1, \ldots, S_i, \ldots, S_n) \in \Sigma$,
$K \in \mathcal{K}$,
$D \in \mathcal{D}$: the dangling set,
$\aleph \subseteq \Sigma \times \mathcal{K} \times \mathcal{D}$: the set of nodes visited up to the current call of *IStep*
output
$\aleph$: the updated set of visited nodes;

1. **begin**
2. insert $N = (\sigma, K, D)$ into $\aleph$ and mark it as *unknown*;
3. **for each** event (E, ϑ) such that $((E, \vartheta) \in Cand(D)$ **or**
(E, ϑ) is an interface event for ξ **or** (E, ϑ) is an external event for ξ) **do**
4. **for each** $S_i \in \sigma$ **do**
5. **for each** transition $T_{ij} = S_i \xrightarrow{\alpha|\beta} S_i'$, $T_{ij} \in M_i$, where M_i is the model of C_i,
such that $\alpha = (E_\alpha, \vartheta_\alpha) = (E, \vartheta)$ **do**
6. **begin**
7. $\sigma' := (S_1, \ldots, S_i', \ldots, S_n)$;
8. $K' := K$; $D' := D$;
9. $\mathcal{L}_\alpha := Link(\vartheta_\alpha)$;
10. **if** $(E_\alpha, \vartheta_\alpha) \in Cand(D')$ **then**
11. **begin**
12. **for each** $x \in \{1 \cdots |D[L_\alpha]| - 1\}$ **do**
13. $D'[L_\alpha, x] := D[L_\alpha, x + 1]$;
14. $D'[L_\alpha, |D[L_\alpha]|] :=$ **nil**
15. **end**;

[6] These anomalous sub-graphs of $G_{\Im}$ correspond to cycles in which all the nodes are marked as unknown. However, consistent silent cycles still remain in $G_{\Im}$ after the pruning.

```
16.                 for each (E_β, ϑ_β) ∈ β such that E_β is an internal event for ξ do
17.                   begin
18.                     L_β := Link(ϑ_β);
19.                     if L_β is not saturated then
20.                       D'[L_β, |D[L_β]| + 1] := E_β
21.                   end;
22.                 if ∃(m, Msg) ∈ β then
23.                   begin
24.                     K'[i] := K'[i] + 1;
25.                     if (K'[i] > |obs(C_i)|  or  obs(C_i)[K'[i]] ≠ m) then
26.                       goto 5
27.                   end;
28.                 if N' = (σ', K', D') ∈ ℵ then
29.                   if N' is marked as inconsistent then
30.                     N'_ℑ := nil
31.                   else
32.                     N'_ℑ := N'
33.                 else
34.                   N'_ℑ := IStep(σ', K', D', ℵ);
35.                 if N'_ℑ ≠ nil then
36.                   begin
37.                     if N is marked as unknown and N' as consistent then
38.                       mark N as consistent;
39.                     create an edge N --T_ij--> N'
40.                   end
41.               end;
42.       if N = (σ, K, (∅ ··· ∅)) and Obs_⟨K⟩(ξ) = Obs(ξ) then
43.         begin
44.           mark N as consistent;
45.           return N
46.         end
47.       else
48.         begin
49.           if at least one edge was created at Line 39 then
50.             return N
51.           else
52.             begin
53.               mark N as inconsistent;
54.               return nil
55.             end
56.         end
57.     end IStep;

58.   begin Interpreter
59.     K := (0 ··· 0); D := (∅ ··· ∅); ℵ := ∅;
60.     N_0 := IStep(ξ_0, K, D, ℵ);
61.     G_ℑ := the graph rooted in N_0;
62.     for each N ∈ ℵ such that N is marked as unknown do
```

63. if $\nexists$ a path $\wp = N \rightsquigarrow N_f$, $\wp \in G_{\Im}$, such that ($N_f = (\sigma_f, K_f, (\emptyset \cdots \emptyset))$ **and**
    $Obs_{\langle K_f \rangle}(\xi) = Obs(\xi)$) **then**
64. remove N from $G_{\Im}$;
65. remove from $G_{\Im}$ all the dangling edges;
66. $\mathcal{S} :=$ the set of nodes in $G_{\Im}$;
67. $\mathcal{E} :=$ the set of labels of edges in $G_{\Im}$;
68. $\mathcal{T} :=$ the set of edges in $G_{\Im}$;
69. $S_0 :=$ the root of $G_{\Im}$;
70. $\mathcal{S}_f \subseteq \mathcal{S} :=$ the set of nodes $N_f = (\sigma_f, K_f, (\emptyset \cdots \emptyset))$ such that $Obs_{\langle K_f \rangle}(\xi) = Obs(\xi)$;
71. $\Im := (\mathcal{S}, \mathcal{E}, \mathcal{T}, S_0, \mathcal{S}_f)$;
72. **return** $\Im$
73. **end** Interpreter.

Proposition 1. *Let $Obs(\xi)$ be an observation of cluster ξ and $\xi_0 = (S_{01}, \ldots, S_{0n})$ the initial states of components in ξ. Then, Algorithm* 1 *computes the interpretation of $(Obs(\xi), \xi_0)$, namely*

$$Interpreter(Obs(\xi), \xi_0) = \Im(Obs(\xi), \xi_0). \tag{13}$$

4.2. Merger

The *Merger* function corresponding to Algorithm 2 generates a graph-based representation of the merging $\mu(\Omega)$ of a set of interpretations Ω by calling function *MStep* recursively. Structurally, *Merger* is very similar to the *Interpreter* algorithm. In fact, as discussed in Section 2, interpretations in Ω can be viewed as constrained cluster models (see Fig. 5). In other words, clusters in *Merger* play the role of components in *Interpreter*, while the links within the interface of these clusters in *Merger* play the role of links among components in *Interpreter*.

Considering the code of *MStep*, note that the set of input parameters does not include K, which is instead included in *IStep*. This is due to the fact that in the merging process the constraints imposed by observations are implicitly guaranteed by the interpretation graphs. The aim of the *Merger* is to combine the cluster interpretations on the basis of the interface constraints among clusters.

Note also that the three nested loops in *Interpreter* (Lines 3–5) are replaced by two nested loops in *Merger*, the outer on interpretations in Ω (Line 3) and the inner on transitions within these interpretations (Line 4). If the transition is triggered by an interface event α (Line 8), one must check whether α is among the candidate events within the interface of the clusters (Line 9): if this is the case, α is consumed. Later, the output events in β are put on the relevant unsaturated links (Lines 17–18). Then, Lines 19–32 parallel Lines 28–41 of *Interpreter*. In particular, a recursive call of *Mstep* is possibly performed at Line 25.

Line 33 of *Merger* corresponds to Line 42 of *Interpreter*, where a check on the quiescence of node N is performed. Notice that in *Interpreter* (Line 42) the quiescence of node N entails, besides the emptiness of the dangling set, the completeness of the observations. In the case of *Merger* (Line 33), the condition on the observations is replaced by the condition on the final nodes, which implicitly entails the completeness of the

observations. This condition is checked also at Line 54, which corresponds to Line 63 of *Interpreter*.

Algorithm 2.

function *Merger*(Ω): the merging $\mu(\Omega) = (\mathcal{S}, \mathcal{E}, \mathcal{T}, S_0, \mathcal{S}_f)$
input
$\Omega = \{\Im(Obs(\xi_1), \xi_{01}), \ldots, \Im(Obs(\xi_n), \xi_{0n})\}$ where $\Im(Obs(\xi_i), \xi_{0i}) = (\mathcal{S}_i, \mathcal{E}_i, \mathcal{T}_i, S_{0i}, \mathcal{S}_{fi})$:
the interpretations of a set of clusters $\Xi = \{\xi_1, \ldots, \xi_n\}$;

function *MStep*(σ, D, $\aleph$): the (possibly nil) root of a sub-graph of $\mu(\Omega)$
input
$\sigma = (S_1, \ldots, S_i, \ldots, S_n)$, $S_i \in \mathcal{S}_i$,
$D = Dang(Interf(\Xi))$: the dangling set relevant to $Interf(\Xi)$,
$\aleph$: the set of nodes visited up to the current call of *Mstep*
output
$\aleph$: the updated set of visited nodes;
1. **begin**
2. insert $N = (\sigma, D)$ into $\aleph$ and mark it as *unknown*;
3. **for each** interpretation $\Im(Obs(\xi_i), \xi_{0i}) = (\mathcal{S}_i, \mathcal{E}_i, \mathcal{T}_i, S_{0i}, \mathcal{S}_{fi}) \in \Omega$ **do**
4. **for each** transition $T = S_i \xrightarrow{T_{ij}} S'_i \in \mathcal{T}_i$, $T_{ij} = S_{ij} \xrightarrow{\alpha|\beta} S'_{ij} \in \mathcal{T}_{ij}$, $\mathcal{T}_{ij} \in M_{ij}$ where M_{ij} is the model of $C_{ij} \in \xi_i$ **do**
5. **begin**
6. $\sigma' := (S_1, \ldots, S'_i, \ldots, S_n)$;
7. $D' := D$;
8. **if** $(L_\alpha = Link(\alpha)) \in Interf(\Xi)$ **then**
9. **if** $\alpha \in Cand(D)$ **then**
10. **begin**
11. **for each** $x \in \{1 \cdots |D[L_\alpha]| - 1\}$ **do**
12. $D'[L_\alpha, x] := D[L_\alpha, x + 1]$;
13. $D'[L_\alpha, |D[L_\alpha]|] :=$ **nil**
14. **end**
15. **else**
16. **goto** 4;
17. **for each** $(E_\beta, \vartheta_\beta) \in \beta$ such that $((L_\beta = Link(\vartheta_\beta)) \in Interf(\Xi)$ **and** L_β is not saturated) **do**
18. $D'[L_\beta, |D[L_\beta]| + 1] := E_\beta$;
19. **if** $N' = (\sigma', D') \in \aleph$ **then**
20. **if** N' is marked as *inconsistent* **then**
21. $N'_\mu :=$ **nil**
22. **else**
23. $N'_\mu := N'$
24. **else**
25. $N'_\mu := MStep(\sigma', D', \aleph)$;
26. **if** $N'_\mu \neq$ **nil then**
27. **begin**
28. **if** N is marked as *unknown* and N' as *consistent* **then**
29. mark N as *consistent*;

30. create an edge $N \xrightarrow{T_{ij}} N'$
31. **end**
32. **end**;
33. **if** $N = (\sigma, (\emptyset \cdots \emptyset))$ **and** $\forall S_i \in \sigma\ (S_i \in \mathcal{S}_{f_i})$ **then**
34. **begin**
35. mark N as *consistent*;
36. **return** N
37. **end**
38. **else**
39. **begin**
40. **if** at least one edge was created at Line 34 **then**
41. **return** N
42. **else**
43. **begin**
44. mark N as *inconsistent*;
45. **return nil**
46. **end**
47. **end**
48. **end** MStep;

49. **begin** Merger
50. $\sigma_0 := (S_{01}, \ldots, S_{0n})$; $D := (\emptyset \cdots \emptyset)$; $\aleph := \emptyset$;
51. $N_0 := MStep(\sigma_0, D, \aleph)$;
52. $G_\mu :=$ the graph rooted in N_0;
53. **for each** $N \in \aleph$ such that N is marked as *unknown* **do**
54. if $\nexists$ a path $\wp = N \rightsquigarrow N_f$, $\wp \in G_\mu$, such that $(N_f = (\sigma_f, (\emptyset \cdots \emptyset))$ **and** $\forall S_i \in \sigma_f\ (S_i \in \mathcal{S}_{f_i})$ **then**
55. remove N from G_μ;
56. remove from G_μ all the dangling edges;
57. $\mathcal{S} :=$ the set of nodes in G_μ;
58. $\mathcal{E} :=$ the set of labels of edges in G_μ;
59. $\mathcal{T} :=$ the set of edges in G_μ;
60. $S_0 :=$ the root of G_μ;
61. $\mathcal{S}_f \subseteq \mathcal{S} :=$ the set of nodes $N_f = (\sigma_f, (\emptyset \cdots \emptyset))$ such that $\forall S_i \in \sigma_f\ (S_i \in \mathcal{S}_{f_i})$;
62. $\mu := (\mathcal{S}, \mathcal{E}, \mathcal{T}, S_0, \mathcal{S}_f)$;
63. **return** μ
64. **end** Merger.

Proposition 2. *Let Ω be a set of interpretations relevant to a set Ξ of disjoint clusters. Then, Algorithm* 2 *computes the merging of Ω, namely*

$$Merger(\Omega) = \mu(\Omega). \tag{14}$$

4.3. Viewer

The *Viewer* function corresponding to Algorithm 3 generates the diagnostic hierarchy $\nabla(Obs(\xi), \xi_0)$ starting from the interpretation graph $\Im(Obs(\xi), \xi_0)$. Notice that ξ might be either a proper cluster of an active system $\mathcal{A}$ or $\mathcal{A}$ itself. In the first case, the

diagnostic hierarchy is not definitive for the cluster. In fact, the merging with other cluster interpretations is expected to refine this diagnostic hierarchy. If, instead, the cluster corresponds to the whole system, the diagnostic hierarchy is definitive (at least with respect of the system observation $Obs(\mathcal{A})$ and its initial state $\mathcal{A}_0$).

Considering the code of *Viewer*, at Line 2 the elements Δ, Δ^d, Ψ, and φ of the diagnostic hierarchy are initialized. Then, three loops are executed sequentially, corresponding to the bottom-up instantiation of Ψ (Lines 3–5), Δ^d (Lines 6–11), and Δ (Lines 12–17). The mapping function φ is instantiated in parallel with the instantiation of Δ^d and Δ. Lines 18 and 19 are meant to complete the set of diagnoses in Δ with the possibly empty diagnosis. [7]

Algorithm 3.

function *Viewer*$(\Im(Obs(\xi), \xi_0))$: the diagnostic hierarchy $\nabla(Obs(\xi), \xi_0) = (\Delta, \Delta^d, \Psi, \varphi)$
  input
    $\Im(Obs(\xi), \xi_0) = (\mathcal{S}, \mathcal{E}, \mathcal{T}, S_0, \mathcal{S}_f)$: an interpretation;
1. **begin**
2. $\Delta := \emptyset$; $\Delta^d := \emptyset$; $\Psi := \emptyset$; $\varphi := \emptyset$;
3. **for each** $S_f \in \mathcal{S}_f$ do
4. **for each** faulty route R connecting S_0 with S_f do
5. $\Psi := \Psi \cup \{R\}$;
6. **for each** $\psi \in \Psi$ **do**
7. **begin**
8. $\delta^d := \{(C_i, \mathcal{F}_i)$ such that $\mathcal{F}_i$ is the set of faulty events relevant to C_i in $\psi\}$;
9. $\Delta^d := \Delta^d \cup \{\delta^d\}$;
10. $\varphi := \varphi \cup \{\psi \mapsto \delta^d\}$
11. **end**;
12. **for each** $\delta^d \in \Delta^d$ do
13. **begin**
14. $\delta := \{C_i$ such that $(C_i, \mathcal{F}_i) \in \delta^d\}$;
15. $\Delta := \Delta \cup \{\delta\}$;
16. $\varphi := \varphi \cup \{\delta^d \mapsto \delta\}$
17. **end**;
18. **if** $\exists$ a nonfaulty route R connecting S_0 with a node $S_f \in \mathcal{S}_f$ **then**
19. $\Delta := \Delta \cup \{\emptyset\}$;
20. **return** $(\Delta, \Delta^d, \Psi, \varphi)$
21. **end** Viewer.

Proposition 3. *Let* $\Im(Obs(\xi), \xi_0)$ *be an interpretation relevant to a cluster* ξ. *Then, Algorithm* 3 *computes the diagnostic hierarchy of* $(Obs(\xi), \xi_0)$, *namely*

$$Viewer\big(\Im(Obs(\xi), \xi_0)\big) = \nabla(Obs(\xi), \xi_0). \tag{15}$$

[7] In fact, the set Δ can never be empty: if no faults are involved in the reconstructed behavior of the system, Δ includes the empty shallow diagnosis only. However, as shown in Section 2.5, the empty shallow diagnosis may come with other nonempty shallow diagnoses as well.

4.4. Diagnoser

The *Diagnoser* function corresponding to Algorithm 4 generates the diagnostic hierarchy $\nabla(Obs(\xi), \xi_0)$ starting from the observation $Obs(\mathcal{A})$ of an active system $\mathcal{A}$ and its initial state $\mathcal{A}_0$. To this end, *Diagnoser* makes use of algorithms *Interpreter*, *Merger*, and *Viewer*.

Thus, at Line 2, a decomposition Ξ of the active system $\mathcal{A}$ is obtained (see Definition 5). Then, at Line 3, the interpretation relevant to each cluster in Ξ is performed (virtually) in parallel. This gives rise to the set Ω of interpretations.

At this point (Line 4), a loop is started in order to merge several interpretations into a new larger interpretation until the whole system is covered. If Ω is not a singleton, a partition of Ω is carried out (Line 6) and a new merging based on this partition is performed (virtually) in parallel (Line 7). Each merging is relevant to a part $\Omega_i \subseteq \Omega$, so that, if the number of parts is m, so is the number of interpretation graphs resulting from this parallel merging. The loops continues until the partition made at Line 6 groups all the interpretation graphs in Ω into a single part,[8] in other words, until $m = 1$, so that the result of the merging in Line 7 is in fact the interpretation graph of the active system $\mathcal{A}$, namely $\Im_{\mathcal{A}}$ (Line 9). The diagnostic hierarchy relevant to $\Im_{\mathcal{A}}$ is then generated at Line 10 by means of the *Viewer* function.

Algorithm 4.

function *Diagnoser*$(Obs(\mathcal{A}), \mathcal{A}_0)$: the diagnostic hierarchy $\nabla(Obs(\mathcal{A}), \mathcal{A}_0) = (\Delta, \Delta^d, \Psi, \varphi)$
  input
    $Obs(\mathcal{A})$: an observation of system $\mathcal{A}$,
    $\mathcal{A}_0 = (S_{0_1}, \ldots, S_{0_n})$: the initial states of components in $\mathcal{A}$;
1. **begin**
2. make a decomposition $\Xi = \{\xi_1, \ldots, \xi_n\}$ of $\mathcal{A}$;
3. $\Omega := \{Interpreter(Obs(\xi_1), \xi_{0_1}), \ldots, Interpreter(Obs(\xi_n), \xi_{0_n})\}$;
4. **while** Ω is not a singleton **do**
5. **begin**
6. make a partition $\{\Omega_1, \ldots, \Omega_m\}$ of Ω;
7. $\Omega := \{Merger(\Omega_1), \ldots, Merger(\Omega_m)\}$
8. **end**;
9. let $\Omega = \{\Im_{\mathcal{A}}\}$ where $\Im_{\mathcal{A}} = \Im(Obs(\mathcal{A}), \mathcal{A}_0) = (\mathcal{S}_{\mathcal{A}}, \mathcal{E}_{\mathcal{A}}, \mathcal{T}_{\mathcal{A}}, S_{0\mathcal{A}}, S_{f\mathcal{A}})$ is the resulting interpretation for system $\mathcal{A}$;
10. $\nabla_{\mathcal{A}} := Viewer(\Im_{\mathcal{A}})$;
11. **return** $\nabla_{\mathcal{A}}$
12. **end** Diagnoser.

[8] Note that, in principle, Ω might be a singleton even at Line 3, that is when the decomposition Ξ generated at Line 2 corresponds to the trivial case in which $n = 1$. This amounts to making the interpretation of the system in a single step.

Proposition 4. *Let* $Obs(\mathcal{A})$ *be an observation of an active system* $\mathcal{A}$*, and* $\mathcal{A}_0 = (S_{01}, \ldots, S_{0n})$ *the initial states of components in* $\mathcal{A}$*. Then, Algorithm* 4 *computes the diagnostic hierarchy of* $(Obs(\mathcal{A}), \mathcal{A}_0)$*, namely*

$$Diagnoser(Obs(\mathcal{A}), \mathcal{A}_0) = \nabla(Obs(\mathcal{A}), \mathcal{A}_0). \tag{16}$$

5. Complexity analysis

This section is devoted to the time complexity analysis of the algorithms defined above, with the main purpose of finding out which are the most critical parameters to take into consideration for performance issues. The analysis assumes that all the sets in the algorithms which require insertion and retrieval (such as the set of visited nodes in the interpreter algorithm) are implemented through optimized data structures, such as binary trees, which require logarithmic time to perform these operations.

5.1. Complexity parameters

A *complexity parameter* is a parameter of a system $\mathcal{A}$ which is relevant to the complexity of the diagnostic algorithms. Complexity parameters refer either to *topological* or *modeling* complexity. Topological complexity is concerned with the number of components and links in $\mathcal{A}$. Modeling complexity relates to the structure of behavioral models in terms of number of states, transitions, messages, events, and so on. Complexity parameters are defined in Table 5. Some of them require an explanation.

The *event nondeterminism factor* φ is the number of transitions, in a component model, exiting the same state and sharing the same input event at the same input terminal. As such, φ represents the nondeterminism associated with input events in a given state.

The branching transitions β_M and $\beta_{\Im}$ correspond, respectively, to the maximum number of transitions exiting a node in a component model and in any interpretations considered in the algorithms.

5.2. Complexity analysis of the Interpreter algorithm

To evaluate the time required to execute the interpretation algorithm we start by analyzing the complexity of the recursive function *IStep*. To begin with, we observe that *IStep* performs a complete depth first search on a graph whose nodes are represented by states which can be possibly included in the interpretation. Thus, the time required by the algorithm is proportional to the time required to analyze a single node multiplied by the number of nodes to be analyzed.

The number of explored nodes essentially depends on the complexity of the models of the components and on the maximum extent (see Definition 12) of the routes for a given set of messages. In detail, the branching factor of the search tree can be evaluated by considering that the recursive call of *IStep* is enclosed in the three loops on Lines 3–5.

The first loop is executed $L + E$ times at most, where L is the number of links and E is the number of internal and external events; the loop at Line 4 is executed C times, where C is the number of components in the cluster, and the next loop at Line 5, φ times, where φ is

the event nondeterminism factor. So, we can bound the branching factor by the following formula:

$$B = \varphi C(L + E), \tag{17}$$

which is less than $\beta_M C$, β_M being the maximum number of transitions exiting a node. The depth of the search tree is given by the number $R_{\Im}$ of transitions included in the route having maximum extent, so that the number of explored nodes is

$$N = \mathrm{O}\big(B^{R_{\Im}}\big). \tag{18}$$

We consider now the evaluation time required by each node, which is dominated by the time required by the nested loops starting at Line 3: the complexity of the statement at Line 2 (O(log(N))) and the complexity of the statements following the loops (O(1)) turns out to be neglectable when added to the higher complexity of the loops.

Within the three loops, the assignments on Lines 7–9 take a constant time. The conditional statement starting at Line 10 requires O($\ell \log(L)$) time due to the included loop and the evaluation of the condition. The loop starting at Line 16 requires O($L \log(L)$). All the other statements require a constant time, with the exception of the statements at Line 28, which requires O(log(N)). So, the overall time required by the conditional statements at Lines 22, 28 and 35 is O(1), O(log($B^{R_{\Im}}$)) and O(1), respectively.

Therefore, considering that all the statements are included in the three nested loops starting at Line 3, the time required to analyze each node is

$$\mathrm{O}\big(B\big(\log\big(B^{R_{\Im}}\big) + \ell \log(L) + L \log(L)\big)\big)$$

Table 5
The complexity parameters

Parameter	Description
φ	Event nondeterminism factor
C	Number of components in a cluster
E	Number of internal and external events in a cluster
ℓ	Maximum capacity of the links in a cluster
L	Number of links in a cluster
$R_{\Im}$	Maximum extent of routes in the interpretation of a cluster
β_M	Branching transitions in a component model
ξ_μ	Number of merged clusters
ξ_D	Number of clusters considered in *Diagnoser*
ℓ_μ	Maximum capacity of links in the interface
L_μ	Number of links in the interface
R_μ	Maximum extent of routes in the merging
$\beta_{\Im}$	Branching transitions in an interpretation graph
F	Maximum number of final states in an interpretation

and the final complexity of a single call of *IStep* is given by:

$$T(\mathit{IStep}) = \mathrm{O}\big(B\big(\log\big(B^{R_\Im}\big) + \ell\log(L) + L\log(L)\big)B^{R_\Im}\big). \tag{19}$$

Now, we consider the statements of *Interpreter* body. Statements at Lines 59 and 61 require a constant time, while the complexity of the statement at Line 60 is given by Eq. (19) multiplied by the number of calls of the recursive function. The loop at Line 62 performs $B^{R_\Im}$ searches within the global transition graph and the required time can be estimated by adding up the contribution of each depth level of the search graph as follows:

$$\begin{aligned} T(\mathit{loop}) &= \mathrm{O}\big(B^{R_\Im}\big) + \mathrm{O}\big(BB^{R_\Im-1}\big) + \mathrm{O}\big(B^2B^{R_\Im-2}\big) + \cdots + \mathrm{O}\big(B^{R_\Im}\big) \\ &= \mathrm{O}\big(R_\Im B^{R_\Im}\big), \end{aligned} \tag{20}$$

where $R_\Im$ is the maximum depth of the search graph and B the branching factor. The statement in Line 65 requires the exploration of all the transitions in the graph, whose number is again $\mathrm{O}(B^{R_\Im})$, while the remaining statements require a constant time to be executed, assuming that the relevant information is collected during the execution of *IStep*.

We are now able to evaluate the overall time required by the *Interpreter* function as follows:

$$T(\mathit{Interpreter}) = \mathrm{O}\big(B(R_\Im\log(B) + \ell\log(L) + L\log(L))B^{R_\Im}\big) + \mathrm{O}\big(R_\Im B^{R_\Im}\big), \tag{21}$$

which can be simplified to:

$$T(\mathit{Interpreter}) = \mathrm{O}\big((R_\Im\log(B) + \ell\log(L) + L\log(L))B^{R_\Im}\big) \tag{22}$$

whose dominant term is $B^{R_\Im}$.

5.3. Complexity analysis of the Merger algorithm

The *Merger* algorithm is very similar in nature to the *Interpreter*, in that a depth first search is performed on a search graph built from a set of automata. The difference is that, here, the starting automata are the interpretations of a set of clusters instead of the basic models of the components.

As expected, the complexity analysis (which, for the sake of brevity, is not fully reported here) gives a set of formulas which are analogous to the ones obtained for the *Interpreter* function, with the appropriate substitution of parameters. Here, the branching factor is limited by:

$$B_\mu = \beta_\Im\xi_\mu, \tag{23}$$

where $\beta_\Im$ is the maximum number of transitions exiting a node of the interpretation graph and ξ_μ the number of merged clusters.

The complexity of *Merger* is given by:

$$T(\mathit{Merger}) = \mathrm{O}\big((R_\mu\log(B_\mu) + \ell_\mu\log(L_\mu) + L_\mu\log(L_\mu) + \xi_\mu\log(F))B_\mu^{R_\mu}\big), \tag{24}$$

which is dominated by $B_\mu^{R_\mu}$, R_μ being the maximum extent of the routes resulting from the merging.

5.4. Complexity analysis of the Diagnoser algorithm

The complexity of the *Diagnoser* algorithm is dominated by the time required to perform interpretations (Line 3) and merging actions (Lines 4–8), as the call to the *Viewer* function and the other statements take a negligible time. In fact, the *Viewer* only performs simple computations on the resulting interpretation, whose complexity is proportional to that of the interpretation graph. This complexity is neglectable with respect to the complexity required to obtain the final interpretation through the interpretation and merging steps of the *Diagnoser* function.

Considering the *Diagnoser* function, the number of interpretations to be performed is given by the number of clusters considered, namely by ξ_D. The number of merging actions, on the other hand, is bounded by $\log(\xi_D)$, as the maximum number of merging actions is given by including, each time, two elements for each sub-set of the partition built at Line 6. Denoting with T_μ the time required to perform the most complex merging and with $T_\Im$ the time required to perform the most complex interpretation, the complexity of the diagnostic algorithm is given by:

$$T(\textit{Diagnoser}) = \mathrm{O}(\log(\xi_D)T_\mu + \xi_D T_\Im). \tag{25}$$

Thus, we can conclude that the complexity of *Diagnoser* essentially depends on the dominant terms of T_μ and $T_\Im$, namely $B_\mu^{R_\mu}$ and $B^{R_\Im}$, respectively. Observe that both terms depends exponentially on the maximum extent of the routes within the corresponding graph. Since a route must include all the messages relevant to the sub-part of the system considered, it turns out that, at least, the final complexity depends exponentially on the number of messages available. On the other hand, the less observable the behavioral models, the larger the extent of routes, because, besides those observable, several additional silent transitions may be included in a route.

However, the complexity analysis given above is extremely pessimistic since it does not account for the constraints imposed by the topology of the active system at hand and, therefore, the mode in which components actually interact with one another. These are expected to decrease the computation time considerably, as one can perceive by looking at the example given in Section 2.

6. Sample application domain

This section provides some hints on how applying the modeling principles of the diagnostic method to a simplified view of the power transmission network domain. Interestingly enough, this sample domain offers evidence that even components, such as transmission lines, that are normally considered as characterized by passive behavior and continuous variables, can be conveniently modeled as discrete-event components for the purpose of diagnosis.

A power transmission network is a meshed graph where nodes and edges correspond to the high voltage sub-stations and to the high voltage three-phase lines, respectively. This structure aims to guarantee great robustness in case of accidents, mainly represented by short circuits on transmission lines. The protection system is designed to detect dangerous

conditions, to disconnect a component (such as a line, a bus, a transformer or a generation group) as soon as it starts operating in a dangerous way, and to keep in operation nonfaulty components as much as possible, in order to avoid a black-out. This is achieved by tripping the circuit breaker associated with each protection.

Several sorts of protective devices are used in power transmission networks, depending on the electrical apparatus to be protected. Among them are distance protections, which recognize the presence of short circuits on lines and command the opening of relevant breakers. Each protection is devoted to one component, but also works as a backup to the other protections nearby. Distance protections are located at both ends of each line and measure current and voltage in order to evaluate the impedance and, possibly, recognize the presence and the distance of a short circuit on the network. The protection is directional, as it only "sees" the variations of impedance on the protected line and on the lines which are connected to that line in the same direction.

A transmission sub-network is shown on the right of Fig. 6, which is composed of lines L_1 and L_2, breakers B_{l1}, B_{r1}, B_{l2}, and B_{r2}, and relevant distance protections P_{l1}, P_{r1}, P_{l2}, and P_{r2}. As such, P_{l1} and P_{r2} are sensitive to impedance variations on both lines L_1 and L_2, while P_{r1} and P_{l2} only see the variation of impedance on line L_1 and L_2, respectively.

According to the measurements, the protection determines the timing of its intervention: the shorter the estimated distance of the short circuit, the faster the opening command sent to the breaker. This timing is discretized in steps defined by fixed impedance thresholds, which are set in order to discriminate the location of the short circuit: a tripping of the protection at the first step corresponds to a short circuit on the protected line; a tripping at the second step, excluding a few exceptions, to a short circuit on a line which is directly connected to the protected line, and so on. In a simplified view, it is possible to consider two steps only, the second step corresponding to a localization of the short circuit on a line which is not the protected one. A characteristics of the distance protection device is depicted on the left of Fig. 6. According to it, the domain of the impedance is partitioned into three segments, namely $0 \leqslant Z_{low} \leqslant Z_1$, $Z_1 < Z_{low2} \leqslant Z_2$, and $Z_{ok} > Z_2$, where Z_{low} and Z_{low2} correspond to the tripping at first and second step, respectively, while Z_{ok} is the impedance region for which no distance protection intervention is required.

When reacting to a short circuit, all the protective components send logical signals (messages) to a *Regional Control Center*. These messages consist of a unique address

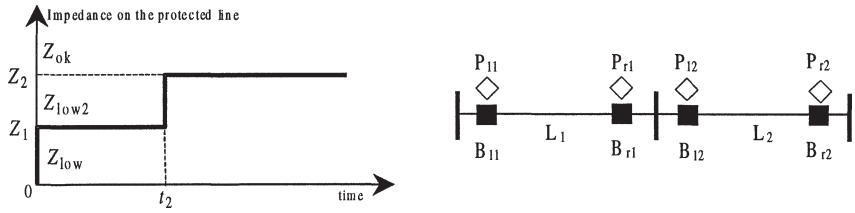

Fig. 6. Simplified impedance vs. time characteristics of distance protection (left) and power transmission sub-system (right).

of the event source, an event code, and possibly a timestamp. Operators have to decide within one minute where the short circuit is located and what recovery actions have to be applied. When the protection system reaction is faulty, such a localization is generally not straightforward at all, since several lines, instead of the shorted one only, are isolated. All the same, the shorted line along with the protection devices that reacted abnormally are certainly within the isolated sub-network. Consequently, the localization of the short circuit, together with the diagnosis of the corresponding part of the protection system, may be carried out by focusing on the portion of the network involved in the isolation.

The active system corresponding to the power transmission sub-network of Fig. 6 is shown on the top of Fig. 7. The three types of components, namely *power transmission line* (L), *protection* (P) and *breaker* (B), are depicted on the center of Fig. 7, along with their input and output terminals (identifiers of input and output terminals start with I and O, respectively). As such, components $\{L_1, L_2\}$, $\{P_{l1}, P_{r1}, P_{l2}, P_{r2}\}$, and $\{B_{l1}, B_{r1}, B_{l2}, B_{r2}\}$ are instances of type L, P, and B, respectively, whose models are represented on the bottom of Fig. 7.

Typically, when the system is quiescent, the state of lines, protections, and breakers is *Normal*, *Standby*, and *Closed*, respectively. The occurrence of a short circuit on a line corresponds to an event triggering the reaction, which may cause the isolation corresponding to the sub-network displayed in Fig. 6. For example, if the shorted line is L_1, transition T_{L1} is fired, which generates four output events, namely, two *zlow* events to protections P_{l1} and P_{r1}, respectively, and two *zlow*2 events to the neighboring lines, respectively, (the line on the left of L_1 is not considered in the reconstruction process, since it does not belong to the isolated sub-system). These are expected to generate the *zlow*2 events, which start the relevant protections.

When the reaction finishes, the sub-system becomes quiescent again, but in a different (final) state. Typically, lines and protections return to state *Normal* and *Standby*, respectively. Based on the available messages, the application of the diagnostic method presented in Sections 2–5 leads to the reconstruction of the sub-system history and to the final diagnosis of the protection system.

Fault diagnosis in power transmission networks is a very complex problem, of huge practical relevance. Many attempts have been made to develop automated support tools for this task, based on a wide variety of approaches and very different technological solutions, including model-based diagnosis [4,39], neural networks [21], Petri nets [40], and fuzzy systems [8]. The advantages of the model-based approach are discussed in [4], where component models associate a qualitative fault localization with each component state. To some extent, the approach of [4] shares some commonalities with ours. However, the component models adopted are qualitative in nature and do not capture the dynamic behavior of the system. Consequently, the set of generated diagnoses may include physically implausible diagnoses, which have to be pruned by exploiting additional knowledge about faulty behavior. In our approach, instead, a more thorough modeling results in a more restricted set of diagnoses, which accounts for additional constraints on the way in which events occur over time and are exchanged among components.

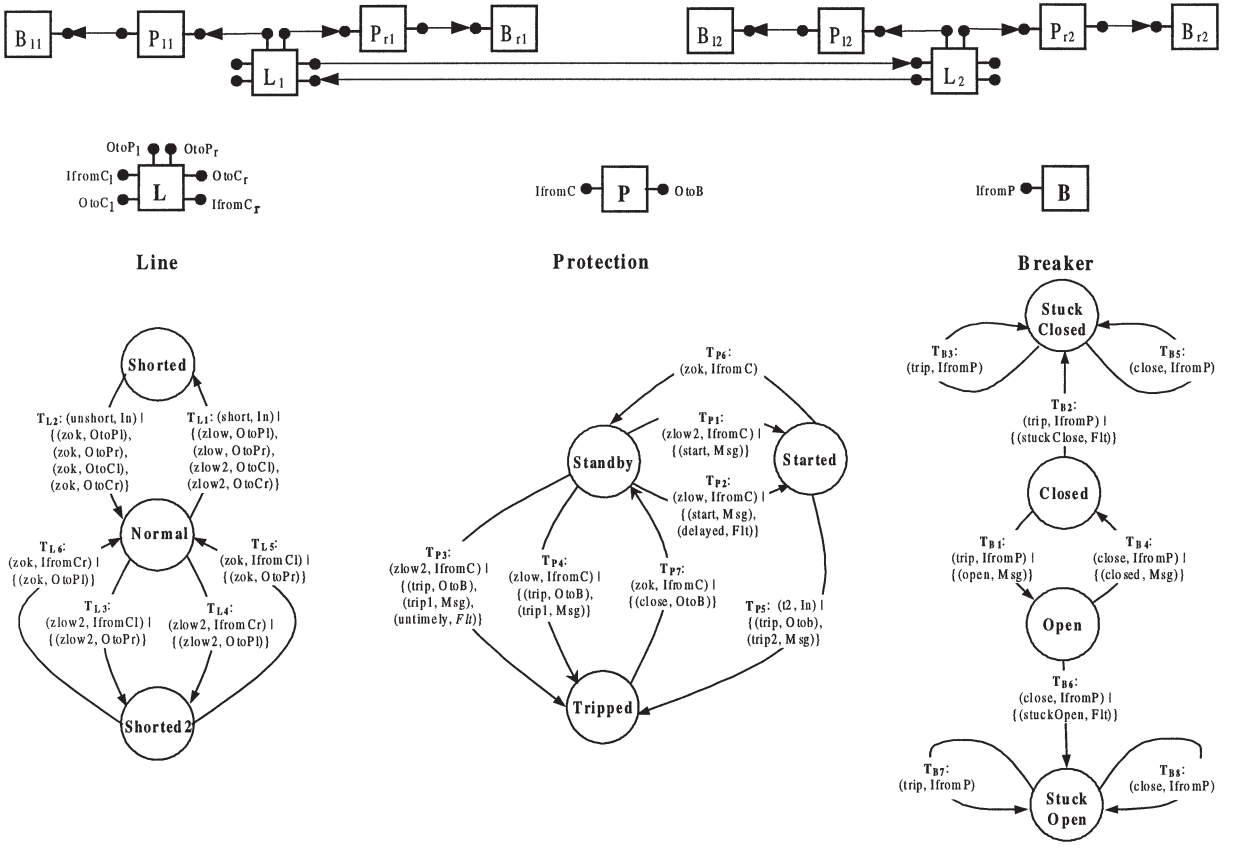

Fig. 7. Active system (top), structural models of transmission line, distance protection, and breaker (center), and corresponding models (bottom).

7. Comparison with related work

In this section we summarize the main features of the proposed approach and discuss their relationship with related work in the area of temporal diagnosis. Four subsections deal with the main conceptual aspects of our work, namely the class of considered systems, the underlying notion of diagnosis, the modeling technique, and the diagnostic method.

7.1. Active systems

The diagnosis of active systems from a systematic and theoretical point of view has received limited attention in past research. However, on the one hand, the practical importance of the topic is confirmed by some application-oriented works concerning the diagnosis of systems that can quite naturally be modeled as active, such as power transmission networks [2,3,25], power distribution networks [11,22], and telecommunication networks [33,34]. On the other, such works fail to provide a general analysis and understanding, since they just consider specific application problems, and also feature important differences in the adopted modeling and diagnostic methods, as shown below.

The work of Williams and Nayak [41] has some significant similarities with ours, since it concerns systems that are able to modify their configuration in response to a failure event. However, the approach is different from ours in several aspects. First, it has different goals. In fact, the work in [41] is devoted to configuration management, not to diagnosis, and focuses on the problem of detecting a failure at a global level and, then, of deciding the optimal system reconfiguration according to the current conditions, which are assumed not to evolve over time until a system reconfiguration is carried out. Second, compared to ours, this work relies on some restrictive assumptions, which limit the generality of the approach: system components are assumed to perform synchronous transitions, and the identification of the current operation mode of the system takes into account the previous state of the system only.

7.2. Notion of diagnosis

The significant difference between active and passive systems gives rise to a very different statement of the diagnosis problem in the two cases. If the system is passive, as in the classical diagnosis theory [32] and in many subsequent approaches, it is necessary to detect faulty operation and then to find an explanation for it.

If the system is active, as in our case, no faulty operation is detected based on the observation, instead diagnosis consists in finding a complete and coherent explanation of the observed dynamic system reaction. In order to verify whether this explanation is complete (i.e., accounts for all the messages in the observation) and consistent (i.e., does not entail contradictions), it is not enough to find a set of faulty components: a temporal reconstruction of what happened within the system is necessary.

Therefore, the diagnostic approach proposed in this paper lends itself to a notion of diagnosis as explanation of the dynamic behavior of the considered system, rather than as the simple identification of a list of faulty components. The notion of explanatory (abductive) diagnosis [27] contrasts with the classical notion of diagnosis for static

systems. In [27] and [10] it is stressed that, in some contexts, the question a diagnostic solver has to answer is "what happened to the system?" rather than "what is wrong with the system?". In the case of active systems the two questions are so strictly related that it is generally impossible to consider them separately: in order to find out what is wrong with an active system, it is necessary to reconstruct what happened to it. In particular, we strongly agree with the view of "temporal diagnoses as sequences of events rather than as states or sequences of states" [10]. Following this direction, our work represents a confirmation of the importance of the notion of explanatory diagnosis for many classes of dynamic systems and suggests that such a concept deserves further analysis and a general formalization.

With respect to other works sharing the notion of abductive diagnosis, our approach offers significant advantages as far as the type of knowledge exploited and the complexity of the abductive reasoning are concerned. The basic knowledge endowment required by our approach is an FSM model of the behavior of each type of system component. This contrasts with the rather common association of abductive diagnosis with the use of causal models [6,31]. Such models are less modular than FSMs and usually encompass compiled knowledge about interactions among different components, which, in some cases, is rather difficult to elicit. FSM models allow the history reconstruction process to be modular, and thus keep under control the complexity of the algorithm, which deals with a limited amount of information at each step. This issue is further discussed in Section 7.3.

7.3. Modeling technique

Our approach adopts FSMs as a basic modeling formalism and supports the construction of system models through the composition of separate FSMs representing the behaviors of components. Such composition is guided by the structure of the system at hand. This guarantees several important properties, including

(i) modularity in the definition of the global system model,
(ii) clear separation between behavioral model, concerning the operation of components, and structural model, concerning how components are connected together in a specific system, and
(iii) expressiveness and compactness of the obtained representation.

The advantages offered by our modeling approach as well as its limitations can be better pointed out by comparing it with the formalisms adopted by other approaches to the diagnosis of discrete dynamic systems. The comparison is carried out on the two aspects that most typically characterize the modeling of dynamic systems: the representation of temporal information and the representation of behavioral transitions.

From the point of view of the representation of temporal information, there are approaches that adopt some form of temporally qualified *if–then* rules [18–20] or logical clauses [9,13] in order to represent the temporal behavior of the system. However, this representation has a limited expressiveness, does not allow a simple modular construction of the system model from component models, and often implicitly incorporates knowledge about system structure and interactions among components.

The work described in [28] uses both an abductive model, which integrates qualitative temporal constraints, and a mode constraint graph, which states the possible temporal relations between behavioral modes. The use of a simplified version of Allen's interval

algebra [1] provides a richer representation of temporal constraints than in our proposal: the extension of our approach in this direction is planned for future work.

Considering the representation of behavioral transitions, significant similarities can be found in works that explicitly introduce either graph (network) or FSM models for diagnosis of discrete systems.

In [16] a causal network model is adopted, which, however, does not allow a modular representation of components and relies on simple Boolean equations to express relationships holding between variable values at different time instants.

In [10] a graph is used to model the global system behavior: each node in the graph represents a global state (the combination of the states of individual components) whereas each link represents a global state transition and is associated with the event which actually triggers the transition. The issue of providing a separate representation for the behavior of each system component and then of combining component behaviors is not considered, since the global graph implicitly encompasses also the structural model of the system.

Finally, some approaches [22,34,36,37] use communicating automata models which are very similar to our representation. In these works, a separate FSM model is defined for each component, and the global system model is then obtained through an operation of synchronous composition [24,26], thereby guaranteeing both satisfactory expressiveness and modularity in the definition of the representation. Our approach is more general and, thereby, more complex, since it considers buffered events and adopts asynchronous composition.

7.4. Diagnostic method

In this subsection the diagnostic approach we propose is compared with other existing methods in the field of the diagnosis of dynamic systems. Here we focus on the general features of the diagnostic methods and try to keep, as far as possible, an abstract view that does not take into account application or representation aspects covered in previous subsections.

A large group of approaches to diagnosis of dynamic systems [9,13,14,20,23,29,30] (most of which are mainly based on modifications and extensions to Reiter's theory) tend to keep reasoning about temporal behavior separated from diagnostic reasoning. In fact, in these attempts, attention is mainly focused on the problem of fault detection in the context of dynamic behavior, whereas, once a fault has been detected, a diagnostic algorithm, normally not including significant temporal reasoning features, is applied. Basically, they adopt the same approach: some kind of simulation is used to make a prediction of system behavior which is then matched against the observations. If a discrepancy is detected, a fault hypothesis is generated and simulation is used in turn to verify whether the hypothesis produces predictions consistent with observations or not. Therefore reasoning about temporal behavior in these approaches mainly concerns system monitoring, this being the activity of looking after the system and detecting symptoms over time, rather than true system diagnosis, that is, the activity of finding explanations for the symptoms. In other words diagnosis is kept out of the temporal reasoning process.

The works reported in [18,28] present some concepts that are closer to our ideas, since diagnosis is considered as the task of reconstructing the history of the system. However,

both works feature significant differences from ours as far as the modeling principles and the adopted diagnostic method are concerned, as remarked in the previous section.

Friedrich and Lackinger [17] carry out a general analysis of the concept of temporal misbehavior and examine the distinction between permanent and transient faults. According to this analysis they claim that (and we strongly agree with this remark) "In order to distinguish between temporal properties of failures we have to model them explicitly. This is due to the fact that we cannot draw conclusions about the temporal nature of failures by solely considering the temporal properties of their manifestations". Failures possibly affecting a component are therefore explicitly modeled and temporal diagnosis is then defined as a set of temporally located failure states, associated with components, such that they are consistent with observations. However, the behavior associated with each failure state is represented through a set of first order sentences that are independent of time. Therefore this approach does not encompass failures whose consequences span over time, i.e., temporally evolving faulty behaviors, as it is the case in active systems.

A recent work of Struss [38] proposes an original view about diagnosis of dynamic systems which is based on a significantly different standpoint from that of the works surveyed above. In fact, a common feature of all these works is that a simulation of the behavior of the dynamic system is carried out in order to compare the system outputs produced by the simulated behavior with the actual observations. In contrast with this traditional approach, called simulation-based diagnosis, Struss argues that simulation is not necessary for diagnosis of dynamic systems, and proposes state-based diagnosis, by showing, both at theoretical level and in a practical application, that it is possible to discriminate different behavioral modes by considering only a single observation about the state of the system, without relating it to the system temporal evolution.

However, the approach proposed in [38] relies on the assumption that the behavioral mode of the system does not vary during the time interval considered for observation: in fact, the observation is sufficient to identify the actual behavioral model (possibly a faulty one) of the current state. This limits the applicability of state-based diagnosis to a specific class of dynamic systems.

Other limitations of the approach proposed in [38] concern some restrictive assumptions about the nature of the temporal constraints within behavioral models. It is therefore possible to conclude that results about state-based diagnosis, though providing a substantial advancement for the diagnosis of a specific class of dynamic systems, should not be regarded as generally applicable and, in particular, are not adequate for active systems.

Tighter relationships can be found with works concerning diagnosis of discrete event systems. In [10] diagnosis is defined as a sequence of events that might have been followed by the system so as to yield the observations gathered at successive time instants. Each diagnosis corresponds to a path in the graph representing state transitions of the system. The diagnostic procedure exploits each observation in order to derive the set of global states the system could be in when such observation was collected. Then the system graph is exploited in order to trace the sequences of events that could have determined the transitions between the states corresponding to different observations.

This approach may run into difficulties when dealing with large systems, where each observation is compatible with a huge set of global system states. The way our approach copes with this complexity problem is twofold. On the one hand, we allow diagnosis to be

carried out modularly, considering different levels of component aggregations within the system: this is an original feature of our approach, ensuring significant advantages when dealing with large systems. On the other, as is common in many systems, we consider observations associated with state transitions, which are very useful to direct and constrain the reconstruction procedure.

In the works of Sampath et al. [36,37], separate automata representing components are combined together in order to form the global system model. The global system model is then exploited to produce another FSM, called diagnoser. This approach necessarily requires the generation of the global system model, which is the starting point for the automatic synthesis of a diagnoser. Such a diagnoser is an FSM, created off-line, taking as input the sequence of observations of the system, is able to produce an estimate of the current state of the system and information on failures that may have occurred.

The approach outlined above is further extended in [7] where Timed Discrete Event Systems (TDES) are considered: each transition is associated with an interval specifying upper and lower time bounds for its occurrence. Explicit management of quantitative temporal information is a potentially useful extension which is, however, beyond the scope of the present work. The timed system representation is then translated into an "untimed" equivalent version which fits in the modeling framework of [36,37], so that theoretical results and algorithms concerning such framework can be directly extended to the TDES case.

These approaches share however also a basic problem: the need to produce a global system model and then a global diagnoser significantly reduces the advantage of using a modular and compositional representation and makes the approach by Sampath et al. not applicable in practice to large systems, since it runs into significant computational difficulties, as pointed out in [34].

In order to tackle this difficulty, Rozé [34] exploits the hierarchical structure of the system and applies the diagnoser approach to a selected group of components. However, this approach is only applicable to systems where a rigid hierarchical structure and several independent sub-systems can be identified. Our method does not require a rigid and a priori partitioning of the system into groups of components, but, rather, is able to exploit separate partial diagnoses obtained at the level of individual components or groups of components in order to produce a diagnosis at the level of any arbitrary component assembly. Diagnoses at the level of groups can then be exploited at the further level of component group aggregation and so on until the level of the global system is reached. This dynamic and modular diagnostic process is particularly suitable for large systems and gives further advantages when the system is physically distributed (as occurs in many relevant application domains). In fact, our method allows partial diagnoses to be produced separately at a local level, thereby exploiting local computational resources, speeding up the initial diagnostic process and minimizing the amount of data to be exchanged among different sites.

Still based on the diagnoser approach, the work by Laborie and Krivine [22] considers the possibility of generating exhaustively off-line the set of chronicles, i.e., all the sequences of observations a system may produce, so that diagnostic activity is reduced to the identification of the actual observed sequence of events within the complete set of precompiled chronicles.

It should be remarked however that even the off-line generation of all possible chronicles may give rise to a prohibitive computational burden. For this reason, in [22] the method is applied to separate groups of components, called cross-sections, and the quite restrictive assumption is made that there is no temporal nondeterminism in the model, so that the sequences of observations produced at the level of the global system are not affected by the different delays occurring in component operation and inter-component communication.

It is worth highlighting two further aspects that distinguish our proposal from other FSM based approaches for diagnosis of discrete event systems. First, we explicitly model the communication links between automata representing different components and perform—on the fly—asynchronous composition, thereby allowing delayed message transmission and the existence of a queue of dangling messages to be processed by the receiver component. This contrasts with the operation of synchronous composition [24,26] adopted in [34,36,37], where output events produced by a component are instantaneously associated with input events of another component. Moreover, in contrast with [36,37], our method is able to cope with cycles of transitions producing no external observations within the system model.

Finally, in [35], the authors of [34,36,37] propose a further development of their theory concerning the so-called "active diagnosis". Despite the use of the same keyword "active", the focus of [35] is very different from ours, since it is relevant to "the design of diagnosable systems by appropriate design of the system controller", which is out of the scope of the present work.

8. Conclusion

A technique for model-based diagnosis of large active systems has been presented. The systems considered in this paper are complex since no significant limiting assumptions are made either on structure (i.e., topology and/or size) or on behavior, and, to make the scenario even more complicated, events are generated, buffered within communication links, and consumed asynchronously. Because of this, the state of the active system depends not only on the state of each single component but also on the configuration of events within links. The main challenge of diagnosis of large active systems is the huge size of the search space. To summarize, the main contributions of the proposed approach include:

(1) the ability to cope with asynchronism,
(2) the ability to cope with silent behavioral cycles,
(3) the definition of a coherent conceptual framework that integrates the problems of history reconstruction and fault diagnosis,
(4) the definition of a modular method for history reconstruction amenable to parallel implementation, and
(5) the use of a graph-based representation for different diagnostic information, namely diagnoses, explanations, and histories.

From a critical perspective, several limitations of the approach and possible enhancements can be envisaged. For instance, the paper never discusses how dealing with possible loss of messages, a common situation in real applications. The solution to this problem can be confined to the modeling task, by defining, for each observable transition T whose message may be lost, a further silent transition T' where the message is missing.

Moreover, the assumption on the final quiescence of the system after the reaction may seem too restrictive. Minor changes to the history reconstruction algorithm would enable it to cope with reactions that never reach quiescence, yet yield a finite observation.

Considering models from the temporal point of view, it turns out that every component behavior is endowed with restricted temporal information, consisting just in the constrained ordering of state transitions, as described by the edges of the corresponding automaton. Additional temporal information is implicit in the link model, which is shared by all links. This shortage of temporal information in the model of the system comes with a scarcity of temporal information on observations, which are just ordered on a component-by-component basis. However, the diagnostic technique has already been extended so as to account also for the possibly total ordering of messages relevant to any group of components. Furthermore, the approach would be more flexible if the behavior of links were modeled explicitly and if several temporal behavioral models were envisaged, so as to better reflect real distinct link behaviors.

A more substantial effort is instead needed in order to modify the modeling technique and the diagnostic algorithm so as to account for possible timestamps associated with messages, which is the case in several real application domains.

Considering the core problem of diagnosis of large systems, that is tractability, criteria for systematic clusterization could be devised in order to improve efficiency. The problem of optimal clusterization is related also to reuse of interpretations, as discussed in Section 2.6. The topic of reuse has already been tackled in our research, even if the issue is not delved into in this paper.

Besides, the efficiency of the proposed approach depends also on the available processing architecture, since several interpretation and merging steps can be carried out in parallel. Therefore, optimization techniques based on a given computational architecture constitute another research topic.

A point of interest not discussed in the paper is how extending the diagnostic technique for real-time monitoring. In this scenario, the occurrence of a new observable event should update the reconstructed behavior and the diagnostic hierarchy on the basis of the constraints imposed by the new event only. This corresponds to extending the modularity of the technique from the spatial to the temporal dimension.

Lastly, the complexity analysis might be refined so as to take into consideration further system parameters that explicitly mirror the constraints imposed by the interaction among components.

The development of a software system that implements the diagnostic method is under way, based on a language for system modeling, a compiler, a persistent repository, and a real-time diagnostic engine.

Acknowledgement

We are most grateful to Benedict Mulvihill, who gave the manuscript meticulous attention, and to the anonymous reviewers, whose comments and suggestions considerably improved the quality of the paper.

Appendix A. Proof of Theorem 1

The equivalence of the two automata, namely the interpretation automaton and the merging automaton, can be stated by showing the isomorphism among respective states and transitions. Considering states, the definition of a state, let it be S_μ, as given in Definition 11 (merging) can be brought back to an instance of state as given in Definition 10 (interpretation) by means of simple transformations. We assume that $\Xi = \{\xi_1, \ldots, \xi_k\}$ is a decomposition of cluster ξ incorporating the set of components $\{C_1, \ldots, C_n\}$ and denote with $m_1, \ldots, m_k$ the number of components of the k clusters, respectively. We denote by $(D_1, \ldots, D_{r_i})$ the dangling set corresponding to the links within each cluster ξ_i, with $(d_1, \ldots, d_I)$ the dangling set of *Interf*(Ξ), and with $(d_1, \ldots, d_\ell)$ the dangling set of ξ. This allows us to write the following equalities:

$$S_\mu = (S_1, \ldots, S_m, (d_1, \ldots, d_I)) \tag{A.1}$$

$$= ((\sigma_1, (k_1, \ldots, k_{m_1}), (d_1, \ldots, d_{r_1})), \ldots, (\sigma_k, (k_h, \ldots, k_{m_h}), (d_1, \ldots, d_{r_h})), (d_1, \ldots, d_I)) \tag{A.2}$$

$$= (((S_{11}, \ldots, S_{1m}), (k_1, \ldots, k_{m_1}), (d_1, \ldots, d_{r_1})), \ldots, ((S_{h1}, \ldots, S_{hm}), (k_h, \ldots, k_{m_h}), (d_1, \ldots, d_{r_h})), (d_1, \ldots, d_I)) \tag{A.3}$$

$$= ((S_1, \ldots, S_n), (k_1, \ldots, k_n), (d_1, \ldots, d_\ell)), \tag{A.4}$$

where the last expression is in the form required by Definition 10.

According to Definition 10, a transition of the merging automaton is of the kind $S \xrightarrow{T_{ij}} S'$ and belongs to a $\mathcal{T}_i$, which is the transition set of the interpretation of a single cluster. This transition corresponds to a transition in the automaton of the whole set of components connecting the two states isomorphic to S and S'.

A further notion to be introduced in order to prove the theorem is the restriction of a state and the restriction of a transition belonging to the interpretation of a cluster.

Definition A.1 (*Restriction of a state*). A state $S = (\sigma_i, (k_1, \ldots, k_n), (D_1, \ldots, D_m))$ of an interpretation of a cluster ξ corresponds to a state $S_r = (\sigma'_i, (k'_1, \ldots, k'_{j'}, \ldots, k'_{n'}), (D_1, \ldots, D'_{h'}, \ldots, D'_{k'}))$ *restricted* to $\xi' \subseteq \xi$ where σ'_i, $k'_{j'}$, and $D'_{h'}$ are obtained from the corresponding components in S by deleting all the terms not referring to the sub-cluster ξ'.

Definition A.2 (*Restriction of a transition*). A transition

$$T_i = S_1 \xrightarrow{T_{ij}} S_2 \in \mathcal{T}$$

in an interpretation of a cluster ξ, with T_{ij} belonging to the model of a component included in $\xi' \subseteq \xi$, corresponds to a transition

$$T'_i = S'_1 \xrightarrow{T_{ij}} S'_2$$

restricted to ξ', where S'_1 and S'_2 are states restricted to ξ'. T_i is also said to be *restrictable* to ξ'.

Lemma A.1. *If $\Im(Obs(\xi), \xi_0)$ is the interpretation of a cluster $\xi \supseteq \xi'$, any transition in it that is restrictable to ξ' belongs to $\Im(Obs(\xi'), \xi'_0)$, where ξ'_0 is the restriction of ξ_0 and $Obs(\xi')$ is the restriction of $Obs(\xi)$ to ξ'.*

Proof. The initial and final states of $\Im(Obs(\xi), \xi_0)$, when restricted to ξ', are also initial and final states of $\Im(Obs(\xi'), \xi'_0)$, by Definition 10. Let C_i be a component in ξ' and $\mathcal{T}_i$ the corresponding transition function. We can prove the lemma by observing that, if $T_{ij} \in \mathcal{T}_i$ is a transition corresponding to a transition $T \in \mathcal{T}_\xi$ in the interpretation of ξ, then $T_r \in \mathcal{T}'_\xi$ in the interpretation of ξ', where T_r is the reduced transition. If $T \in \mathcal{T}_\xi$, the three points from Definition 10 defining the transition function must be proved. Thus, the same transition T_{ij} generates a restricted transition in $\Im(Obs(\xi'), \xi'_0)$ since:

(1) the first point in Definition 10 holds as $T_{ij} \in \mathcal{T}_i$ both for $\Im(Obs(\xi), \xi_0)$ and $\Im(Obs(\xi'), \xi'_0)$;
(2) the second point is verified as T belongs to a history $h \in \Im(Obs(\xi), \xi_0)$ and we can build a path of the restricted history $\wp'$ by collecting all the restrictable transitions in h: however, $\wp'$ must be a history in $\Im(Obs(\xi'), \xi'_0)$ as it connects the initial state with the final state obtained restricting the final state of h;
(3) the third condition both applies to $\Im(Obs(\xi), \xi_0)$ and $\Im(Obs(\xi'), \xi'_0)$, as any internal event for $\Im(Obs(\xi'), \xi'_0)$ is also an internal event for $\Im(Obs(\xi), \xi_0)$. □

Corollary A.1. *Given a partition of a set of components, the union of the transitions in the interpretation $\Im$ of each cluster is a superset of the transitions included in the interpretation of the whole set of components.*

Proof. Each transition in $\Im$ can be reduced to one of the clusters in the partition by definition and, as a consequence of Lemma A.1, the restricted transition must belong to the interpretation of that cluster. □

In order to prove Theorem 1 we need to prove the following claims:

(a) the transitions in the merging automaton and in the interpretation automaton are the same, namely $\mathcal{T}_\mu = \mathcal{T}_\Im$;
(b) the set of events is the same, namely $\mathcal{E}_\mu = \mathcal{E}_\Im$;
(c) the initial state is the same, namely $S_{0\mu} = S_{0\Im}$; and
(d) the set of final states coincides, namely $\mathcal{S}_{f\mu} = \mathcal{S}_{f\Im}$.

Point (b) is true as $\mathcal{E}_\Im$ is the union of all the transitions of all the component models and $\mathcal{E}_\mu$ is the union on all the clusters (which cover the whole set of components) of the transitions included in the models of their components.

Points (c) and (d) are easily verified from Definitions 10, 11, and the above stated isomorphism among states.

Point (a) can be proved as follows.

(a_1) $\mathcal{T}_\mu \supseteq \mathcal{T}_\Im$. If we admit that a transition $T \in \mathcal{T}_\Im$ has not a corresponding transition in $\mathcal{T}_\mu$, then either:
  (1) T is not included in any interpretation of the clusters; or
  (2) T has been discarded as it is not consistent.

But, point (1) cannot hold for Lemma A.1. Similarly, point (2) cannot hold as the transitions composing a history in $\Im$ and including T must have corresponding restricted transitions in the interpretations of the clusters (see Corollary A.1). Given the isomorphism established between initial and final states in the interpretation and merging automata, these corresponding transitions constitute a history in the merging automaton as well. Therefore, it is not possible that any of these transitions, and specifically T, be discarded as not belonging to a history.

(a_2) $\mathcal{T}_\mu \subseteq \mathcal{T}_\Im$. It is not possible that a transition $T \in \mathcal{T}_\mu$ has not a corresponding transition in $\mathcal{T}_\Im$ as T must belong to a history by definition. This history is composed of transitions whose isomorphic transitions in $\mathcal{T}_\Im$ also form a history, due to the isomorphism between starting and final states. So, this history (and the included transition) must be in $\mathcal{T}_\Im$ by Definition 10. □

References

[1] J. Allen, Maintaining temporal knowledge about temporal intervals, Comm. ACM 26 (11) (1983) 832–843.
[2] P. Baroni, U. Canzi, G. Guida, Fault diagnosis through history reconstruction: An application to power transmission networks, Expert Systems with Applications 12 (1) (1997) 37–52.
[3] P. Baroni, G. Lamperti, P. Pogliano, G. Tornielli, M. Zanella, Automata-based reasoning for short circuit diagnosis in power transmission networks, in: Proc. 12th International Conference on Applications of Artificial Intelligence in Engineering, Capri, Italy, 1997.
[4] A. Beschta, O. Dressler, H. Freitag, M. Montag, P. Struss, A model-based approach to fault localisation in power transmission networks, Intelligent Systems Engineering (1993) 3–14.
[5] D. Brand, P. Zafiropulo, On communicating finite-state machines, J. ACM 30 (2) (1983) 323–342.
[6] V. Brusoni, L. Console, P. Terenziani, D. Theseider Dupré, A spectrum of definitions for temporal model-based diagnosis, Artificial Intelligence 102 (1) (1998) 39–80.
[7] Y.L. Chen, G. Provan, Modeling and diagnosis of timed discrete event systems—A factory automation example, in: American Control Conference, Albuquerque, NM, 1997, pp. 31–36.
[8] H.-J. Cho, J.-K. Park, H.-J. Lee, A fuzzy expert system for fault diagnosis of power systems, in: Proc. International Conference on Intelligent System Application to Power Systems, Montpellier, France, 1994, pp. 217–222.
[9] L. Console, L. Portinale, D. Theseider Dupré, P. Torasso, Diagnostic reasoning across different time points, in: Proc. ECAI-92, Vienna, Austria, 1992, pp. 369–373.
[10] M.O. Cordier, S. Thiébaux, Event-based diagnosis for evolutive systems, in: Proc. 5th International Workshop on Principles of Diagnosis, New Paltz, NY, 1994, pp. 64–69.
[11] M.O. Cordier, S. Thiébaux, O. Jehl, J.P. Krivine, Supply restoration in power distribution systems: A reference problem in diagnosis and reconfiguration, in: Proc. 8th International Workshop on Principles of Diagnosis, Mont St. Michel, France, 1997.
[12] J. de Kleer, B.C. Williams, Diagnosing multiple faults, Artificial Intelligence 32 (1) (1987) 97–130.
[13] O. Dressler, H. Freitag, Prediction sharing across time and contexts, in: Proc. AAAI-94, Seattle, WA, 1994, pp. 1136–1141.
[14] D.L. Dvorak, Monitoring and diagnosis of continuous dynamic systems using semiquantitative simulation, Ph.D. Thesis, University of Texas at Austin, TX, 1992.
[15] D.L. Dvorak, B. Kuipers, Model-based monitoring of dynamic systems, in: Proc. IJCAI-89, Detroit, MI, 1989, pp. 1238–1243.
[16] Y. El Fattah, G. Provan, Modeling temporal behavior in the model-based diagnosis of discrete-event systems (a preliminary note), in: Proc. 8th International Workshop on Principles of Diagnosis, Mont St. Michel, France, 1997.
[17] G. Friedrich, F. Lackinger, Diagnosing temporal misbehavior, in: Proc. IJCAI-91, Sydney, Australia, 1991, pp. 1116–1122.

[18] T. Guckenbiehl, G. Schäfer-Richter, Sidia: Extending prediction based diagnosis to dynamic models, in: Proc. First International Workshop on Principles of Diagnosis, Stanford, CA, 1990, pp. 74–82.
[19] W. Hamscher, Modeling digital circuits for troubleshooting, Artificial Intelligence 51 (1–3) (1991) 223–271.
[20] W. Hamscher, R. Davis, Diagnosing circuits with state: an inherently underconstrained problem, in: Proc. AAAI-84, Austin, TX, 1984, pp. 142–147.
[21] M. Kezunovic, I. Rikalo, D.J. Sobajic, Neural network applications to real-time and off-line fault analysis, in: Proc. International Conference on Intelligent System Application to Power Systems, Montpellier, France, 1994, pp. 29–36.
[22] P. Laborie, J.P. Krivine, Automatic generation of chronicles and its application to alarm processing in power distribution systems, in: Proc. 8th International Workshop on Principles of Diagnosis, Mont St. Michel, France, 1997.
[23] F. Lackinger, W. Nejdl, Integrating model-based monitoring and diagnosis of complex dynamic systems, in: Proc. IJCAI-91, Sydney, Australia, 1991, pp. 2893–2898.
[24] S. Lafortune, E. Chen, A relational algebraic approach to the representation and analysis of discrete-event systems, in: Proc. American Control Conference, Boston, MA, 1991.
[25] G. Lamperti, P. Pogliano, Event-based reasoning for short circuit diagnosis in power transmission networks, in: Proc. IJCAI-97, Nagoya, Japan, 1997, pp. 446–451.
[26] N.A. Lynch, M.R. Tuttle, An introduction to input/output automata, CWI Quarterly 2 (3) (1989) 219–246.
[27] S.A. McIlraith, Explanatory diagnosis: conjecturing actions to explain observations, in: Proc. 8th International Workshop on Principles of Diagnosis, Mont St. Michel, France, 1997.
[28] W. Nejdl, J. Gamper, Harnessing the power of temporal abstractions in modelbased diagnosis of dynamic systems, in: Proc. ECAI-94, Amsterdam, 1994, pp. 667–671.
[29] H.T. Ng, Model-based, multiple-fault diagnosis of dynamic, continuous physical devices, IEEE Expert 6 (6) (1991) 38–43.
[30] J.Y.-C. Pan, Qualitative reasoning with deep-level mechanism models for diagnosis of mechanism failures, in: Proc. First IEEE Conference on AI Applications, Denver, CO, 1984, pp. 295–301.
[31] D. Poole, Normality and faults in logic-based diagnosis, in: Proc. IJCAI-89, Detroit, MI, 1989, pp. 1129–1135.
[32] R. Reiter, A theory of diagnosis from first principles, Artificial Intelligence 32 (1) (1987) 57–95.
[33] M. Riese, Diagnosis of communicating systems: dealing with incompleteness and uncertainty, in: Proc. IJCAI-93, Chambéry, France, 1993, pp. 1480–1485.
[34] L. Rozé, Supervision of telecommunication network: a diagnoser approach, in: Proc. 8th International Workshop on Principles of Diagnosis, Mont St. Michel, France, 1997.
[35] M. Sampath, S. Lafortune, D.C. Teneketzis, Active diagnosis of discrete-event systems, IEEE Trans. Autom. Control 43 (7) (1998) 908–929.
[36] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, D.C. Teneketzis, Diagnosability of discrete-event systems, IEEE Trans. Autom. Control 40 (9) (1995) 1555–1575.
[37] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, D.C. Teneketzis, Failure diagnosis using discrete-event models, IEEE Trans. Control Systems Technology 4 (2) (1996) 105–124.
[38] P. Struss, Fundamentals of model-based diagnosis of dynamic systems, in: Proc. IJCAI-97, Nagoya, Japan, 1997, pp. 480–485.
[39] G. Tornielli, L. Capetta, S. Cermignani, A.S. Fabiano, R. Schinco, An interval-based approach to fault diagnosis of power systems, in: Proc. International Conference on Intelligent System Application to Power Systems, Montpellier, France, 1994, pp. 809–816.
[40] J.P. Wang, J. Trecat, A parallel fault diagnosis expert system for one dispatch center, in: Proc. International Conference on Intelligent System Application to Power Systems, Montpellier, France, 1994, pp. 201–208.
[41] B.C. Williams, P.P. Nayak, A model-based approach to reactive self-configuring systems, in: Proc. 7th International Workshop on Principles of Diagnosis, Montreal, Quebec, 1996.