Journal of Artificial Intelligence Research 69 (2020) 1473-1532 Submitted 05/2020; published 12/2020

Diagnosis of Deep Discrete-Event Systems

Gianfranco Lamperti GIANFRANCO.LAMPERTI@UNIBS.IT
Marina Zanella MARINA.ZANELLA@UNIBS.IT
Department of Information Engineering, University of Brescia
Via Branze 38, 25123 Brescia, Italy

Xiangfu Zhao XIANGFUZHAO@GMAIL.COM
School of Computer and Control Engineering, Yantai University
30, Qingquan RD, Laishan District, Yantai 264005, China

Abstract

An abduction-based diagnosis technique for a class of discrete-event systems (DESs), called deep DESs (DDESs), is presented. A DDES has a tree structure, where each node is a network of communicating automata, called an active unit (AU). The interaction of components within an AU gives rise to emergent events. An emergent event occurs when specific components collectively perform a sequence of transitions matching a given regular language. Any event emerging in an AU triggers the transition of a component in its parent AU. We say that the DDES has a deep behavior, in the sense that the behavior of an AU is governed not only by the events exchanged by the components within the AU but also by the events emerging from child AUs. Deep behavior characterizes not only living beings, including humans, but also artifacts, such as robots that operate in contexts at varying abstraction levels. Surprisingly, experimental results indicate that the hierarchical complexity of the system translates into a decreased computational complexity of the diagnosis task. Hence, the diagnosis technique is shown to be (formally) correct as well as (empirically) efficient.

1. Introduction

Diagnosis is the task of finding the possible causes of given symptoms. As such, it can be applied not only to living creatures, including humans, but also to possibly complex engineering artifacts, such as aircraft or nuclear power plants. As several other tasks, diagnosis can be automated to varying degrees based on different paradigms. Among them is model-based diagnosis (Reiter, 1987; Hamscher, Console, & de Kleer, 1992), where the discrepancy between the expected behavior of the system (inferred by a given model) and the actual observation allows for the generation of *candidate* diagnoses, each candidate being a set of *faulty* components, i.e., a minimal set such that assuming that all the components in it do not behave normally while all the others do is consistent with the observation. Although being general in nature, this *consistency-based* diagnosis technique was initially conceived for static systems, such as combinational circuits. When time-varying (dynamical) systems come into play, consistency-based diagnosis is far less adopted. Some notable exceptions can be recorded in the realm of diagnostic reasoning about formal requirements (Schuppan, 2012; Pill & Quaritsch, 2013) expressed in Linear Temporal Logic (LTL) (Pnueli, 1977) or about discrete-event systems (DESs) (Pencolé, Steinbauer, Mühlbacher, & Travé-Massuyès, 2018). Dynamical systems can be modeled as DESs (Cassandras & Lafortune, 2008), these typically being represented by finite automata (FAs), possibly communicating with one another (Brand & Zafiropulo, 1983). Model-based diagnosis of DESs is grounded on the seminal work by Sampath et al. (1996). Compared with the technique by Reiter (1987), the approach by Sampath et al. (1996) is

characterized (among others) by two peculiarities: (1) the system specification involves not only the normal behavior of the DES but also its abnormal (faulty) behavior, and (2) the diagnosis approach is not consistency-based but rather *abduction-based* as each candidate is the set of all the (component) faults included in a trajectory (i.e. a sequence of state changes) of the DES that entails the observation. In abduction-based diagnosis of DESs, the behavior of the system (or a diagnosis-oriented surrogate of it, typically a *diagnoser*) is checked against a given sequence of observable events in order to elicit the trajectory (or trajectories) of the DES that explain the observation. This way, faulty events within abnormal trajectories (that produce the given observation) are filtered out as diagnosis results.

Active systems (ASs) are a special class of asynchronous DESs, for which abduction-based diagnosis techniques were proposed more than two decades ago (Baroni, Lamperti, Pogliano, & Zanella, 1998, 1999, 2000). At that time, a novelty consisted in the diagnosis task not requiring the generation of the global model of the system, which was instead necessary to generate the diagnoser in in the approach by Sampath et al. (1996).[1]

The evolution from ASs to *deep* DESs (DDESs) was not instantaneous. It was spurred by the idea of injecting some sort of semantics into diagnosis of ASs (Lamperti & Zanella, 2010), based on the consideration that there is a gap between the model of a complex DES, which integrates several components, and its diagnosis, which is still anchored to individual components. Trivially, a (sub)AS is faulty only if it includes a faulty component. A candidate diagnosis of an AS is a set of component faults, irrespective of the role each component plays within the aggregate it belongs to and within the system. The idea of enriching diagnosis results with some semantics then evolved into the notion of *context-sensitive diagnosis*, proposed by Lamperti and Zanella (2011) and refined by Lamperti and Zhao (2014). Diagnosis is context-sensitive when faults are defined for a hierarchy of abstractions describing the system. The notion of a context-sensitive fault has progressively been extended as the notion of context has progressively been generalized. In the current paper, a context-sensitive (or contextual) fault is not necessarily a fault in the traditional sense, i.e. the record of something that has broken, instead it can correspond, for instance, to a violation of the expected function of a subsystem, a violation that has possibly taken place even if no component has broken: in this example, the context is the subsystem and the role it has to play within the considered system. ASs with a *deep behavior*, that is, where the behavior (both normal and abnormal) of lower hierarchical levels can affect the behavior of upper levels, were first introduced by Lamperti and Zhao (2013a, 2013b). From a terminological point of view, the attribute 'deep' takes inspiration from deep learning, by analogy with the structure of neural networks in Machine Learning as far as the presence of several layers is concerned. In the domain of DDESs, deep behavior is equivalently called *behavior stratification*. Considering different levels of abstraction is a common means to face the complexity of both natural systems and artifacts, for it induces a separation of concerns. For instance, in biology, the human brain is regarded as being organized in multiple layers: the primitive reptilian brain at the bottom, the emotional mammal brain in the middle, and the rational primate brain on top, with each brain being composed of several parts (amygdala, prefrontal cortex, temporal lobes, etc.). The OSI model, the layered architecture of a software system or an operating system, a cognitive architecture, all are examples of stratification in artifacts.

1. In the worst case, if the DES includes n components and each component model includes k states, then the global model of the DES will include up to k^n states, a number that can be very large even for a few tens of components. With this numbers, the generation of the global model (and thus of the diagnoser) in real systems (possibly involving thousands of components) is quite out of the question, even if performed offline.

Basically a DDES consists of several ASs; each AS is a node in a hierarchy. A child AS exerts its influence over its parent AS by sending special events to it, called *emergent events*. An AS can trivially be regarded as a DDES consisting just of one AS, in which case there is no need for emergent events. If the DDES consists of a single AS, the method for carrying out the diagnosis task is the same as for ASs. Although some existing AS models can be regarded as a DDES consisting of more than one hierarchical layer, in general the 'translation' of an AS to/from a DDES with several layers is not straightforward (and it is out of the scope of the paper). Modeling and processing an AS as a multi-layer DDES[2] is beneficial from the point of view of both the CPU time and the RAM storage, as shown by the experimental results reported in Section 6.7. However, efficiency is not the reason for DDESs have been introduced. The rationale behind the proposal of DDESs is that a deep behavior representation can be much more intuitive and natural for the design, analysis, monitoring, and diagnosis of dynamical systems operating at different abstraction levels, such as robots (Khalastchi & Kalech, 2018), as they have a dynamic context nature natively. Remarkably, a robot can perform a variety of tasks in a changing environment: its current operating context consists in the specific environment with which the robot is interacting (as sensed by the robot itself) and the specific task it is accomplishing. Observations that are legitimate under one context might not be legitimate under another: the latter are clues about contextual faults. For instance, a zero airspeed value is not legitimate while an unmanned aerial vehicle (UAV) is flying. A DDES allows for the leaf ASs of the hierarchy to represent simple contexts, while intermediate ASs in the hierarchy can represent an aggregate of contexts, along with the transitions from a simple context, dealt with as a unit, to another context, and so on. The AS at the root of the hierarchy can represent the transitions from a context aggregate to another context aggregate. The bottom-up communications in the hierarchy can represent, among other things, how (far) contextual faults can propagate.

Diagnosing a (flat) AS amounts to reconstructing its behavior by performing the asynchronous composition of the behaviors of all its components as constrained by the given observation. The diagnosis method inherited from the framework of ASs can be adapted to be used also for DDESs. Applying this method amounts to reconstructing the overall behavior of the DDES, by performing the asynchronous composition of the reconstructed behaviors of all the ASs it includes, as constrained by the given observation, and then to 'decorating' the states with the set of faults relevant to the trajectories leading to them. Being a DDES a container of ASs, its size is usually larger than that of an AS, hence the space limits are soon reached, as illustrated in Section 6.7. However, a more efficient diagnosis technique that is able to exploit the specificities of DDESs (thus avoiding the reconstruction of the overall behavior of the considered DDES) is available. In fact, in order to diagnose a DDES, the behavior of each single node in the hierarchy, as constrained by the given observation and by the events coming from its child nodes, is computed starting from the leaf nodes and going upward in the hierarchy. Once the behavior of a parent node has been constructed, the constraints relevant to this behavior are propagated to its children, so as only the diagnoses (sets of faults) that comply with the parent behavior are considered. These diagnoses are properly combined with those relevant to the parent node; therefore, once the parent node has been processed, its associated diagnoses are relevant to the whole subtree rooted in it and they comply with the observations of all the nodes in the subtree. This way, once the root node has been processed, its associated set of diagnoses turns out to be a sound and complete set of candidates relevant to the whole hierar-

2. Modeling an existing system requires adherence to reality; this means that we have not to model an AS as a multi-layer DDES if this is far from the way the real system works. Instead, modeling a new system as a multi-layer DDES can provide useful hints for the system design, in order to enhance its understandability and explainability.

chy, that is, to the whole DDES. According to this method, each single AS, is processed separately, which is usually manageable. An analogous viable technique for the diagnosis of ASs having a deep behavior was first presented by Lamperti et al. (2016, 2016b) and subsequently extended (2016a). However, neither the formalization of the technique nor support for its correctness (soundness and completeness) was provided.

This paper integrates and extends the authors' previous research by formalizing the abduction-based diagnosis technique for DDESs briefly described above, including a proof of correctness and experimental results that show that its efficiency is greater than that of the AS diagnosis method, as adapted to DDESs. To the best of our knowledge, apart from the works cited above, no approach to diagnosis of DDESs has been proposed so far. Still, several works can be related to this paper in varying degrees, as discussed at the end of the paper.

In the rest of this paper, Section 2 recalls the notion of an AS and the relevant technique for solving a diagnosis problem. Section 3 introduces the notion of an active unit (AU), the building block of a DDES, along with emergent events, the basic means of upward hierarchical communication between AUs within the DDES. Section 4 defines the notion of a DDES and a corresponding diagnosis problem. It also highlights the inadequacy of the problem-solving technique adopted for ASs if we are willing to adapt it to DDESs. Section 5 presents a method for preprocessing DDES specifications in order to support effective detection of emergent events during the diagnosis task. Section 6 outlines a technique specifically designed for solving DDES diagnosis problems, which is feasible as well as sound and complete, along with relevant experimental results. Section 7 illustrates the genesis of DDESs and their advantages. Section 8 compares the proposal with a selection of related works. Conclusions are drawn in Section 9.

2. Active Systems

This background section defines the concept of an AS, upon which the next sections will progressively build the new notion of a DDES. First, the modeling primitives of an AS are described. Later, the diagnosis problem, relevant to an AS and a specific observation, is defined, along with an abductive method for the generation of the set of candidate diagnoses.

An AS is a network of *(active) components* (ACs), with each AC being modeled by a (possibly nondeterministic) communicating automaton $M = (S, I, X, O, Y, \tau)$, where S is the set of states, I the set of input events, X the set of input terminals, O the set of output events, Y the set of output terminals, and τ the transition function, $\tau : S \times (I \times X) \times 2^{(O \times Y)} \mapsto 2^S$. In general, a transition t of a component c from a state s to a state s' is triggered by an event $e \in I$ at an input terminal $x \in X$, and generates a (possibly empty) set of output events $\{e_1(y_1), \ldots, e_k(y_k)\}$, where $\forall i \in [1\,..\,k]$, $e_i \in O$, $y_i \in Y$, denoted by the triple

$$t(c) = \langle s, (e(x), \{e_1(y_1), \ldots, e_k(y_k)\}), s' \rangle.$$

When $e(x)$ is consumed, the state s' is reached and the output events (if any) are generated at the relevant output terminals. An AC can change its state without the need for a ready event on any input terminal: this is a spontaneous transition, which is typically used for modeling state changes caused by external events (e.g. a lightning may cause the reaction of a sensor in a protection system). The absence of an event on the input terminal x is denoted by $e(x) = \varepsilon$, where ε is the *empty* event. Components within an AS are connected with one another through *links*. A link l is a pair (y, x'), where y is an output terminal of a component c and x' is an input terminal of another

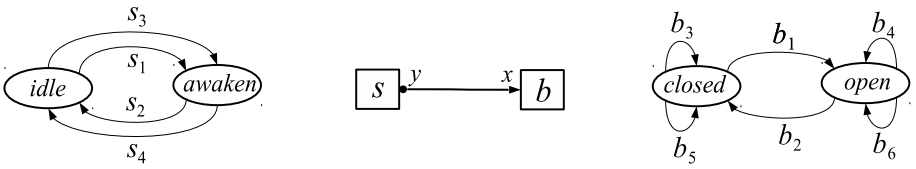


Figure 1: Active system $\mathcal{A}$ (center), and models of the sensor s (left) and of the breaker b (right).

component c', with $c' \neq c$. When a transition of c generates an output event $e(y)$, the event e is *ready* at the input terminal x', namely $e(x')$ is ready. If $e(x')$ is ready, then the link l is *loaded* and no further transition of c generating an output event at y can occur. Each terminal can be connected with (at most) one link.[3] Hence, if an (either input or output) terminal z is connected with a link, the latter can be identified as $link(z)$. If a terminal is not connected with a link, then the terminal is *dangling*. We assume that only input terminals can be dangling in an AS. If an AS $\mathcal{A}$ includes dangling terminals, then $\mathcal{A}$ is *open*. Based on this terminology, the transition $t(c)$ is *triggerable* (can occur) iff: (a) the input terminal x is dangling[4] or $e(x) = \varepsilon$ or $e(x)$ is ready, and (b) all links exiting the output terminals $y_1, \ldots, y_k$ are empty (not loaded). An AS is succinctly represented by a triple $\mathcal{A} = (C, L, D)$, where C is the array of components, L the array of links, and D the array of dangling (input) terminals.

Notice that the above modeling does not make any distinction between normal and faulty behavior. Since this distinction is crucial in order to perform abductive diagnostic reasoning (which, in our approach, means finding out all the behaviors that entail a given observation), further modeling primitives are needed: they will be introduced in subsequent sections.

Example 1 Outlined in Figure 1 is an AS $\mathcal{A}$ that includes two components, a sensor s and a breaker b. These components are connected through a link (y, x), where y is the output terminal of s and x the input terminal of b. $\mathcal{A}$ is designed to protect a device from short circuits. In a normal behavior, when a short circuit is detected by the sensor, the latter commands the breaker to open. Likewise, when the short circuit is extinguished, the sensor commands the breaker to close. The component models of s and b are depicted on the left and right side of Figure 1, respectively. Specifically, the transitions within the model of the sensor s are: $s_1 = \langle idle, (\varepsilon, \{op(y)\}), awaken\rangle$: s detects a short circuit and outputs $op(y)$; $s_2 = \langle awaken, (\varepsilon, \{cl(y)\}), idle\rangle$: s detects the end of the short and outputs $cl(y)$; $s_3 = \langle idle, (\varepsilon, \{cl(y)\}), awaken\rangle$: s detects a short circuit and outputs $cl(y)$; $s_4 = \langle awaken, (\varepsilon, \{op(y)\}), idle\rangle$: s detects the end of the short and outputs $op(y)$. Transitions within the model of breaker b are: $b_1 = \langle closed, (op(x), \emptyset), open\rangle$: b consumes event $op(x)$ and opens; $b_2 = \langle open, (cl(x), \emptyset), closed\rangle$: b consumes event $cl(x)$ and closes; $b_3 = \langle closed, (op(x), \emptyset), closed\rangle$: b consumes $op(x)$ and keeps being closed; $b_4 = \langle open, (cl(x), \emptyset), open\rangle$: b consumes $cl(x)$ and keeps being open; $b_5 = \langle closed, (cl(x), \emptyset), closed\rangle$: b consumes event $cl(x)$ and keeps being closed; $b_6 = \langle open, (op(x), \emptyset), open\rangle$: b consumes $op(x)$ and keeps being open. Notice how the transitions

3. If a component transition generates two output events that are input to the same component, then we need two distinct links between such components, with each link connecting two distinct pairs of terminals.
4. In this case, we assume that the terminal x is connected with another component outside the AS $\mathcal{A}$. This occurs when $\mathcal{A}$, which is the focus of the diagnosis task, is a portion of a larger AS. In principle, output terminals too could be dangling. However, as clarified in Section 3, in the context of a DDES, only dangling input terminals are eventually connected with special links aimed at connecting "extended" ASs to one another. That way, the DDES turns out to be "closed" (without dangling terminals).

s_3 and s_4 in the sensor, and b_3 and b_4 in the breaker, are somehow abnormal (faulty); the model of $\mathcal{A}$, however, does not identify the faulty transitions.

2.1 Space and Trajectory

Like an AC, an AS $\mathcal{A}$ can be in different states, each AS state corresponding to the array of states of its components and the array of events stored in links. Assume that $\mathcal{A}$ is in the initial state α_0 where all links are empty. At the occurrence of a transition of a component c, $\mathcal{A}$ moves from α_0 to a new state, say α_1, where the state of c is changed and possibly output events are placed within some links. The presence of events ready at the input terminals of other components causes $\mathcal{A}$ to change its state by means of new component transitions. This process continues until all links become empty anew, namely when $\mathcal{A}$ has reached a final (*quiescent*) state[5]. We say that $\mathcal{A}$ has performed a *trajectory* in its space, where the space of $\mathcal{A}$ is a deterministic finite automaton (DFA) whose language over the alphabet of all the component transitions is the set of all possible trajectories of $\mathcal{A}$ starting at α_0. All these concepts are formalized below.

Definition 1 (Space) *Let $\mathcal{A} = (C, L, D)$ be an AS, where $C = (c_1, \ldots, c_n)$ and $L = (l_1, \ldots, l_m)$. The* space *of $\mathcal{A}$, denoted $\mathcal{A}^*$, is a DFA*[6]

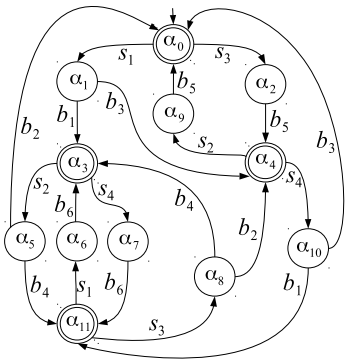
$$\mathcal{A}^* = (\Sigma, A, \tau, \alpha_0, A_f) \tag{1}$$

where the alphabet Σ is the set of transitions of components in C; A is the set of states $(\mathbb{S}, \mathbb{E})$, where $\mathbb{S}$ is the array of states of the components in C and $\mathbb{E}$ is the array of the (possibly empty) events loaded in links in L; $\alpha_0 = (\mathbb{S}_0, \mathbb{E}_0)$ is the initial state, where all events in $\mathbb{E}_0$ are empty; $A_f \subseteq A$ is the set of final states $(\mathbb{S}_f, \mathbb{E}_f)$, where all events in $\mathbb{E}_f$ are empty; τ is the transition function, $\tau : A \times \Sigma \mapsto A$, where: $\langle (\mathbb{S}, \mathbb{E}), t(c_i), (\mathbb{S}', \mathbb{E}') \rangle \in \tau$ iff:

- $t(c_i) = \left\langle \bar{s}_i, (e(x), \{\bar{e}_1(y_1), \ldots, \bar{e}_k(y_k)\}), \bar{s}'_i \right\rangle$ *is triggerable;*
- $\mathbb{S} = (s_1, \ldots, s_n)$, $\mathbb{S}' = (s'_1, \ldots, s'_n)$, $s_i = \bar{s}_i$, $\forall\, j \in [1\,..\,n]$*: if $j = i$ then $s'_j = \bar{s}'_i$, otherwise $s'_j = s_j$.*
- $\mathbb{E} = (e_1, \ldots, e_m)$, $\mathbb{E}' = (e'_1, \ldots, e'_m)$*, if x is not dangling and $e(x) \neq \varepsilon$ then $\mathbb{E}[link(x)] = e$*[7] *and $\mathbb{E}'[link(x)] = \varepsilon$, and $\forall\, j \in [1\,..\,m]$: if $\bar{e}(y)$ is an output event in $t(c_i)$ and $link(y) = l_j$ then $e'_j = \bar{e}$, otherwise $e'_j = e_j$.*

Example 2 With reference to the AS $\mathcal{A}$ defined in Example 1, assume that, in the initial state, the sensor s is *idle*, the breaker b is *closed*, and the link (y, x) is empty, namely $\alpha_0 = (\mathbb{S}_0, \mathbb{E}_0) = ((idle, closed), (\varepsilon))$. Outlined on the left side of Figure 2 is $\mathcal{A}^*$, the space of $\mathcal{A}$, where final states (double circled) are α_0, α_3, α_4, and α_{11}. Details on state contents are provided on the right side.

5. The reachability of a quiescent state (where all links are empty) is a simplifying assumption.
6. Despite the transitions of components in $\mathcal{A}$ being possibly nondeterministic, the space $\mathcal{A}^*$ is deterministic because its alphabet is the set of component transitions rather than the set of triggering events. Still, the intrinsic nondeterminism of the behavior of a component is somewhat embodied in $\mathcal{A}^*$ because two (different) component transitions exiting a state in $\mathcal{A}^*$ may be triggered by the same input event.
7. The index of $\mathbb{E}$ can be either an integer $i \in [1\,..\,m]$ or a link, as here.



State	$\mathbb{S}$	$\mathbb{E}$
α_0	(*idle*, *closed*)	(ε)
α_1	(*awaken*, *closed*)	(*op*)
α_2	(*awaken*, *closed*)	(*cl*)
α_3	(*awaken*, *open*)	(ε)
α_4	(*awaken*, *closed*)	(ε)
α_5	(*idle*, *open*)	(*cl*)
α_6	(*awaken*, *open*)	(*op*)
α_7	(*idle*, *open*)	(*op*)
α_8	(*awaken*, *open*)	(*cl*)
α_9	(*idle*, *closed*)	(*cl*)
α_{10}	(*idle*, *closed*)	(*op*)
α_{11}	(*idle*, *open*)	(ε)

Figure 2: Space $\mathcal{A}^*$ of AS $\mathcal{A}$ (left), along with state details (right).

Definition 2 (Trajectory) *A* trajectory T *within a space* $\mathcal{A}^* = (\Sigma, A, \tau, \alpha_0, A_f)$ *is a finite sequence of contiguous transitions of components in* $\mathcal{A}$*, namely* $T = [\, t_1(c_1), \ldots, t_q(c_q)\,]$*, such that*

$$\forall i \in [1\,..\,q], \langle \alpha_{i-1}, t_i(c_i), \alpha_i \rangle \in \tau, \text{ and } \alpha_q \in A_f. \tag{2}$$

The (possibly infinite) set of trajectories of $\mathcal{A}^*$ *is denoted* $\|\mathcal{A}^*\|$*. A prefix of a trajectory* T *is a* semi-trajectory *within* $\mathcal{A}^*$*. A contiguous subsequence of* T *is a* subtrajectory *within* $\mathcal{A}^*$*.*

Example 3 A trajectory within the space $\mathcal{A}^*$ outlined in Figure 2 is $T = [\, s_1, b_3, s_2, b_5\,]$.

2.2 Viewer and Temporal Observation

The modeling of an AS $\mathcal{A}$ does not provide any information on the observability of $\mathcal{A}$. Providing information about observability means defining, for each component in $\mathcal{A}$, which transitions are observable and which ones are unobservable. A transition is observable if, when occurring, it generates a label that can be perceived by an external observer; otherwise, the transition is unobservable. Therefore, specifying the observability of $\mathcal{A}$ amounts to defining a mapping from observable transitions to corresponding observable labels. This is captured by the notion of a viewer.

Definition 3 (Viewer) *Let* $\mathcal{A}$ *be an AS, and* Ω *a domain of* observable labels*. A* viewer $\mathcal{V}$ *for* $\mathcal{A}$ *is a surjective mapping from a subset of the transitions of the components in* $\mathcal{A}$ *to* Ω*.*[8] *If* $(t(c), \ell) \in \mathcal{V}$*, then the transition* $t(c)$ *is* observable*; otherwise,* $t(c)$ *is* unobservable*.*[9]

Example 4 With reference to the AS $\mathcal{A}$ introduced in Example 1, a viewer $\mathcal{V}$ is defined as follows: $\mathcal{V} = \{(s_1, awk),\ (s_2, ide), (s_3, awk), (s_4, ide), (b_1, bop), (b_2, bcl)\}$, with observable labels having the following meaning: *awk* = the sensor awakes, *ide* = the sensor becomes idle, *bop* = the breaker opens, and *bcl* = the breaker closes.

8. Since the mapping is surjective, each label in Ω is associated with (at least) one of these transitions.
9. In fact, $\mathcal{V}$ can be represented as a set of pairs $(t(c), \ell)$, where $t(c)$ is a transition of the component c and ℓ is the associated observable label.

Definition 4 (Temporal Observation) *Let T be a trajectory in $\mathcal{A}^*$ and $\mathcal{V}$ a viewer for $\mathcal{A}$. The* temporal observation *of T based on $\mathcal{V}$, written $Obs(T, \mathcal{V})$, is the sequence of observable labels associated with the observable transitions of T,*

$$Obs(T, \mathcal{V}) = [\, \ell \mid t(c) \in T, (t(c), \ell) \in \mathcal{V} \,]. \tag{3}$$

Example 5 With reference to the trajectory T in Example 3 and the viewer $\mathcal{V}$ in Example 4, the temporal observation of T based on $\mathcal{V}$ is the sequence $Obs(T, \mathcal{V}) = [\, awk, ide \,]$.

2.3 Ruler and Diagnosis

As remarked already, the modeling of an AS $\mathcal{A}$ does not make any distinction between normal and faulty behavior. Providing specification for this distinction means defining, for each component in $\mathcal{A}$, which transitions are normal and which ones are faulty. In other words, faults are associated with component transitions. This way, a transition $\langle \alpha, t(c), \alpha' \rangle$ of $\mathcal{A}$ is faulty if and only if $t(c)$ is faulty. Thus, specifying the abnormal behavior of $\mathcal{A}$ amounts to defining a mapping from faulty component transitions to faults, just as specifying the observability of $\mathcal{A}$ amounts to defining a mapping from observable component transitions to observable labels. This is captured by the notion of a ruler.

Definition 5 (Ruler) *Let $\mathcal{A}$ be an AS, and F a domain of* faults. *A* ruler *$\mathcal{R}$ for $\mathcal{A}$ is a surjective mapping from a subset of transitions of components in $\mathcal{A}$ to F. If $(t(c), \ell) \in \mathcal{R}$, then the transition $t(c)$ is* faulty*; otherwise, $t(c)$ is* normal.

Example 6 With reference to the AS $\mathcal{A}$ introduced in Example 1, a ruler $\mathcal{R}$ is defined as follows: $\mathcal{R} = \{(s_3, fso), (s_4, fsc), (b_3, fbo), (b_4, fbc)\}$, where faults have the following meaning: *fso* = failure of s in sending the opening command, *fsc* = failure of s in sending the closing command, *fbo* = failure of b in opening, and *fbc* = failure of b in closing.

Definition 6 (Diagnosis) *Let T be a trajectory within $\mathcal{A}^*$ and $\mathcal{R}$ a ruler for $\mathcal{A}$. The* diagnosis *of T based on $\mathcal{R}$, denoted $Dgn(T, \mathcal{R})$, is the set of faults involved in the faulty transitions of T, namely*

$$Dgn(T, \mathcal{R}) = \{\, f \mid t(c) \in T, (t(c), f) \in \mathcal{R} \,\}. \tag{4}$$

Since a diagnosis is a set (rather than a sequence), the same fault cannot be duplicated within the diagnosis. Consequently, two trajectories covering the same subgraph within $\mathcal{A}^*$ by traversing the same cycle a different number of times yield the same diagnosis. This is why the set of possible diagnoses is always finite, regardless the set of trajectories being possibly infinite.

Proposition 1 *Let F be the set of faults involved in the ruler of an AS $\mathcal{A}$. The set of possible diagnoses of $\mathcal{A}$ is finite, specifically a subset of the powerset 2^F.*

Example 7 With reference to the trajectory T in Example 3 and the ruler $\mathcal{R}$ in Example 6, the diagnosis of T based on $\mathcal{R}$ is the singleton $Dgn(T, \mathcal{R}) = \{fbo\}$.

A handy (binary) operator to combine sets of diagnoses is defined as follows.

Definition 7 (Join) *Let Δ_1 and Δ_2 be two sets of diagnoses. The* join *operator "$\otimes$" is defined as*

$$\Delta_1 \otimes \Delta_2 = \{\, \delta \mid \delta = \delta_1 \cup \delta_2, \delta_1 \in \Delta_1, \delta_2 \in \Delta_2 \,\}. \tag{5}$$

The join is commutative, namely $\Delta_1 \otimes \Delta_2 = \Delta_2 \otimes \Delta_1$. The neutral element of the join is the singleton $\{\emptyset\}$, since $\Delta \otimes \{\emptyset\} = \{\emptyset\} \otimes \Delta = \Delta$.

Example 8 Let $\Delta_1 = \{\{fso, fbo\}, \{fsc, fbc\}\}$ and $\Delta_2 = \{\{fso, fbc\}, \{fso\}\}$. We have $\Delta_1 \otimes \Delta_2 = \{\{fso, fbo, fbc\}, \{fso, fbo\}, \{fso, fsc, fbc\}\}$. Notice how, owing to the removal of the duplicated diagnosis $\{fso, fsc, fbc\}$, the cardinality of $\Delta_1 \otimes \Delta_2$ is less than the product of the cardinality of Δ_1 and the cardinality of Δ_2.

2.4 Diagnosis Problem

Assume that an AS $\mathcal{A}$ performs a trajectory starting from the initial state α_0. The temporal observation $\mathcal{O}$ of this trajectory is a sequence of observable labels based on a viewer $\mathcal{V}$ of $\mathcal{A}$. Assume further that $\mathcal{R}$ is the ruler for $\mathcal{A}$. Now comes the question: *What can we say about the behavior of $\mathcal{A}$ based on the information we have, namely the initial state α_0, the viewer $\mathcal{V}$, the temporal observation $\mathcal{O}$, and the ruler $\mathcal{R}$?* This is formalized by the notion of a diagnosis problem.

Definition 8 (Diagnosis Problem) *Let $\mathcal{A}$ be an AS, α_0 the initial state of $\mathcal{A}$, $\mathcal{V}$ a viewer for $\mathcal{A}$, $\mathcal{O}$ a temporal observation of $\mathcal{A}$, and $\mathcal{R}$ a ruler for $\mathcal{A}$. The quadruple*

$$\wp(\mathcal{A}) = (\alpha_0, \mathcal{V}, \mathcal{O}, \mathcal{R}) \tag{6}$$

is a diagnosis problem *for $\mathcal{A}$. The* solution *to $\wp(\mathcal{A})$, denoted $\Delta(\wp(\mathcal{A}))$, is the set of (candidate) diagnoses defined as*

$$\Delta(\wp(\mathcal{A})) = \left\{ Dgn(T, \mathcal{R}) \mid T \in \mathcal{A}^*, Obs(T, \mathcal{V}) = \mathcal{O} \right\}. \tag{7}$$

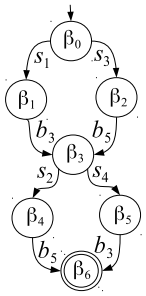
When $Obs(T, \mathcal{V}) = \mathcal{O}$, we say that T *implies* $\mathcal{O}$.

Example 9 Consider the AS $\mathcal{A}$ introduced in Example 1. A diagnosis problem for $\mathcal{A}$ is $\wp(\mathcal{A}) = (\alpha_0, \mathcal{V}, \mathcal{O}, \mathcal{R})$, where $\alpha_0 = (\mathbb{S}_0, \mathbb{E}_0) = ((idle, closed), (\varepsilon))$, the viewer $\mathcal{V}$ is defined in Example 4, $\mathcal{O} = [awk, ide]$, and the ruler $\mathcal{R}$ is defined in Example 6. According to the space $\mathcal{A}^*$ outlined in Figure 2, there are four trajectories T such that $Obs(T, \mathcal{V}) = \mathcal{O}$, namely $T_1 = [s_1, b_3, s_2, b_5]$, $T_2 = [s_1, b_3, s_4, b_3]$, $T_3 = [s_3, b_5, s_2, b_5]$, and $T_4 = [s_3, b_5, s_4, b_3]$. Based on eqn. (7), the solution to $\wp(\mathcal{A})$ is $\Delta(\wp(\mathcal{A})) = \{\delta_1, \delta_2, \delta_3, \delta_4\}$, where $\delta_1 = Dgn(T_1, \mathcal{R}) = \{fbo\}$, $\delta_2 = Dgn(T_2, \mathcal{R}) = \{fbo, fsc\}$, $\delta_3 = Dgn(T_3, \mathcal{R}) = \{fso\}$, and $\delta_4 = Dgn(T_4, \mathcal{R}) = \{fso, fsc, fbo\}$.

Still, generating the solution to a diagnosis problem based on eqn. (7) is impractical for two reasons: (a) $\mathcal{A}^*$ is not available, as its materialization is prohibitively expensive in real systems, and (b) possibly, an infinite set of trajectories should be considered. As to point (a), a more practical technique is required, which exploits the constraints imposed by the temporal observation (cf. Section 2.5). As to point (b), only a significant finite set of trajectories is considered, which is however sufficient to determine the solution to the diagnosis problem (cf. Section 2.6).

2.5 Constrained Space

According to eqn. (7), what is essential to generate a candidate diagnosis is a trajectory of $\mathcal{A}$ that implies the temporal observation $\mathcal{O}$. The set of trajectories implying the temporal observation $\mathcal{O}$ is a subset of $\|\mathcal{A}^*\|$. If the set of trajectories of $\mathcal{A}^*$ is infinite, then the subset of trajectories implying $\mathcal{O}$



State	$\mathbb{S}$	$\mathbb{E}$	$\Im$
β_0	(*idle*, *closed*)	(ε)	0
β_1	(*awaken*, *closed*)	(*op*)	1
β_2	(*awaken*, *closed*)	(*cl*)	1
β_3	(*awaken*, *closed*)	(ε)	1
β_4	(*idle*, *closed*)	(*cl*)	2
β_5	(*idle*, *closed*)	(*op*)	2
β_6	(*idle*, *closed*)	(ε)	2

Figure 3: $\mathcal{O}$-constrained space $\mathcal{A}^*_{\mathcal{O}}$ of AS $\mathcal{A}$ (left), along with state details (right).

can be infinite too. The point is, since $\mathcal{A}^*$ is a DFA embodying (in its language) a possibly infinite set of trajectories, chances are we can devise a technique for generating a DFA whose language is exactly the subset of the language of $\mathcal{A}^*$ that includes all (and only) the trajectories implying the temporal observation. We call this DFA the $\mathcal{O}$-constrained space of $\mathcal{A}$, as defined below.

Definition 9 ($\mathcal{O}$-constrained Space) *Let $\mathcal{A}$ be an AS, α_0 the initial state of $\mathcal{A}$, $\mathcal{V}$ a viewer for $\mathcal{A}$, and $\mathcal{O} = [\ell_1, \ldots, \ell_n]$ a temporal observation of $\mathcal{A}$. The* $\mathcal{O}$-constrained space *of $\mathcal{A}$ is a DFA*

$$\mathcal{A}^*_{\mathcal{O}} = (\Sigma, B, \tau, \beta_0, B_{\mathrm{f}}) \tag{8}$$

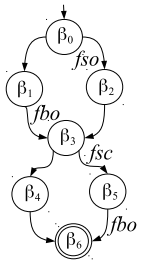
such that the set of trajectories in $\mathcal{A}^*_{\mathcal{O}}$ is a subset of the set of trajectories in $\mathcal{A}^*$, namely $\|\mathcal{A}^*_{\mathcal{O}}\| \subseteq \|\mathcal{A}^*\|$, where Σ equals the alphabet of $\mathcal{A}^*$; B is the set of states $(\mathbb{S}, \mathbb{E}, \Im)$, where $(\mathbb{S}, \mathbb{E})$ is a state of $\mathcal{A}^*$, and $\Im$ is an index *of $\mathcal{O}$, namely $\Im \in [0\,..\,n]$; $\beta_0 = (\mathbb{S}_0, \mathbb{E}_0, 0)$ is the initial state, where $(\mathbb{S}_0, \mathbb{E}_0) = \alpha_0$; $B_{\mathrm{f}} \subseteq B$ is the set of final states $(\mathbb{S}_{\mathrm{f}}, \mathbb{E}_{\mathrm{f}}, n)$, where $(\mathbb{S}_{\mathrm{f}}, \mathbb{E}_{\mathrm{f}})$ is a final state of $\mathcal{A}^*$; τ is the transition function, $\tau : B \times \Sigma \mapsto B$, where $\langle(\mathbb{S}, \mathbb{E}, \Im), t(c), (\mathbb{S}', \mathbb{E}', \Im')\rangle \in \tau$ iff $\langle(\mathbb{S}, \mathbb{E}), t(c), (\mathbb{S}', \mathbb{E}')\rangle$ is a transition in $\mathcal{A}^*$, and*

$$\Im' = \begin{cases} \Im + 1 & \text{if } (t(c), \ell) \in \mathcal{V} \text{ and } \ell = \ell_{\Im+1} \\ \Im & \text{if } t(c) \text{ is unobservable.} \end{cases} \tag{9}$$

Example 10 Let $\mathcal{A}^*$ be the space of $\mathcal{A}$ displayed in Figure 2, and $\mathcal{O} = [awk, ide]$. The $\mathcal{O}$-constrained space of $\mathcal{A}$, namely $\mathcal{A}^*_{\mathcal{O}}$, is outlined in Figure 3. Notice that $\mathcal{A}^*_{\mathcal{O}}$ includes the trajectories T_1, T_2, T_3, and T_4 defined in Example 9, each one implying $\mathcal{O}$.

2.6 Decoration and Solution

Building the $\mathcal{O}$-constrained space $\mathcal{A}^*_{\mathcal{O}}$ is the first step of the diagnosis technique. Then, we need to associate with each trajectory in $\mathcal{A}^*_{\mathcal{O}}$ the corresponding diagnosis. At a first glance, when the set of trajectories is infinite, this might seem an endless task, as we have to consider each trajectory one by one in order to generate the associated diagnosis. However, the set of diagnoses is always finite because a diagnosis is a set (rather than a multiset) of faults, so that duplicated faults within the same trajectory are removed. This means that, for the purpose of diagnosis generation, only a finite set of trajectories suffices to be considered. This finite set is composed of all the trajectories in which possible cycles are traversed once at most. This is good news because we can associate with



State	*Diagnosis set*
β_0	$\{\emptyset\}$
β_1	$\{\emptyset\}$
β_2	$\{\{fso\}\}$
β_3	$\{\{fbo\}, \{fso\}\}$
β_4	$\{\{fbo\}, \{fso\}\}$
β_5	$\{\{fbo, fsc\}, \{fso, fsc\}\}$
β_6	$\{\{fbo\}, \{fbo, fsc\}, \{fso\}, \{fso, fsc, fbo\}\}$

Figure 4: Decoration of the $\mathcal{O}$-constrained space $\mathcal{A}^*_{\mathcal{O}}$, with diagnosis sets detailed on the right side.

each state β of $\mathcal{A}^*_{\mathcal{O}}$ a finite set of diagnoses, with each diagnosis corresponding to a semi-trajectory T ending at β, such that T does not traverse a cycle more than once. That is, we can generate the set of diagnoses associated with the state β by considering only a finite subset of the (possibly infinite) set of semi-trajectories ending at β. Generating the set of diagnoses associated with each state of the constrained behavior is the second step of the diagnosis technique, as detailed below.

Definition 10 (Decorated Constrained Space) *Let $\mathcal{A}^*_{\mathcal{O}}$ be the $\mathcal{O}$-constrained space of an AS $\mathcal{A}$, and $\mathcal{R}$ a ruler for $\mathcal{A}$. The* decorated $\mathcal{O}$-constrained space *of $\mathcal{A}$, denoted $\mathcal{A}^{\mathrm{d}}_{\mathcal{O}}$, is the DFA obtained from $\mathcal{A}^*_{\mathcal{O}}$ by marking each state β of the latter with a* diagnosis set *(or* decoration*) $\Delta(\beta)$ based on the application of the following two rules:*

1. *(Basis) For the initial state β_0, $\emptyset \in \Delta(\beta_0)$;*
2. *(Induction) For each transition $\langle \beta, t(c), \beta' \rangle$, for each $\delta \in \Delta(\beta)$, if $(t(c), f) \in \mathcal{R}$, then $\delta \cup \{f\} \in \Delta(\beta')$, else $\delta \in \Delta(\beta')$.*

Based on the first rule in Definition 10, the empty diagnosis corresponds to the empty semi-trajectory. Based on the second rule, if the decoration of the state β includes a diagnosis δ, then there is at least one semi-trajectory T, ending at β, whose diagnosis is δ.[10] Consequently, there is a semi-trajectory $T \cup [\,t(c)\,]$[11] ending at β' whose diagnosis is either the extension of δ by the fault f associated with $t(c)$ in $\mathcal{R}$, when $(t(c), f) \in \mathcal{R}$, or δ, when $(t(c), f) \notin \mathcal{R}$.

Unlike the first rule, which represents the base case and is applied only once, the second rule is inductive in nature. This means that, for the sake of completeness of the decoration, the second rule should be continuously applied until the decoration of the behavior cannot be changed. As such, if β is a final state of $\mathcal{A}^{\mathrm{d}}_{\mathcal{O}}$, then, for each trajectory $T \in \|\mathcal{A}^{\mathrm{d}}_{\mathcal{O}}\|$ ending in β, the diagnosis of T, namely δ, is part of the decoration of β, that is, $\delta \in \Delta(\beta)$.

Proposition 2 *Let $\wp(\mathcal{A}) = (\alpha_0, \mathcal{V}, \mathcal{O}, \mathcal{R})$ be a diagnosis problem and $\mathcal{A}^{\mathrm{d}}_{\mathcal{O}}$ the decorated $\mathcal{O}$-constrained space of $\mathcal{A}$. The solution to $\wp(\mathcal{A})$ is*

$$\Delta(\wp(\mathcal{A})) = \bigcup \Delta(\beta), \text{ where } \beta \text{ is a final state of } \mathcal{A}^{\mathrm{d}}_{\mathcal{O}}. \tag{10}$$

10. The notion of a diagnosis defined in eqn. (4) for a trajectory is applicable to semi-trajectories too.
11. When the (overloaded) operator $\cup$ is applied to sequences, it denotes concatenation.

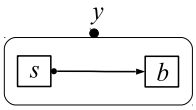
Example 11 Consider the $\mathcal{O}$-constrained space $\mathcal{A}^*_{\mathcal{O}}$ displayed in Figure 3. Outlined on the left side of Figure 4 is the same space where the identifiers of the faulty transitions are replaced with the corresponding faults. The diagnosis sets associated with each state are listed on the right side of the figure. According to Proposition 2, the solution to $\wp(\mathcal{A})$ is the diagnosis set associated with the final state β_6, namely $\Delta(\wp(\mathcal{A})) = \Delta(\beta_6) = \{\{fbo\}, \{fbo, fsc\}, \{fso\}, \{fso, fsc, fbo\}\}$, which in fact equals the solution determined in Example 9 based on eqn. (7).

3. Active Units

As described in the previous section, an AS is a network of communicating ACs. Within an AS $\mathcal{A}$, communication from a component c to a component c' is achieved by the generation of output events at the occurrence of a transition in c. Since c is assumed to be connected with c' by links, the events generated by c are available (ready) at the input terminals of c'. This allows for a temporal cascade of component transitions within $\mathcal{A}$, namely a trajectory of $\mathcal{A}$ (cf. Section 2.1).

Intuitively, an active unit (AU) is meant to allow an AS to communicate with other ASs via special events. In order to make an AS $\mathcal{A}$ communicate with another AS $\mathcal{A}'$, we need to: (1) equip $\mathcal{A}$ with output terminals, (2) equip $\mathcal{A}'$ with input terminals, (3) provide links from the output terminals of $\mathcal{A}$ to the input terminals of $\mathcal{A}'$, and (4) specify the circumstances under which the output events are generated by $\mathcal{A}$. Of the four points listed above, the last is the most critical one. Unlike an AC, whose behavior is specified by a communicating automaton, no behavior is explicitly defined for an AS. In a sense, the behavior of the AS $\mathcal{A}$ is its space $\mathcal{A}^*$, which is determined by the interaction of components. In this space, a transition of $\mathcal{A}$ is caused by a transition of one component of $\mathcal{A}$. That is, the behavior of $\mathcal{A}$ corresponds to the mode in which the components perform state transitions and interact with one another by means of exchanged events. In so doing, $\mathcal{A}$ does not generate any output event on its own, specifically directed toward another AS. In other words, *no deep behavior exists*. To enable deep behavior, we associate with $\mathcal{A}$ a set of events that are generated upon the occurrence of a *string* (sequence) of component transitions matching a given regular expression (cf. Section 3.1). To stress the mode in which these events occur, they are called *emergent events*.[12] Since each emergent event e in $\mathcal{A}$ is associated with one output terminal y of $\mathcal{A}$, the occurrence of e places the latter on the terminal y, exactly like the generation of an output event by a component c places the event on a specific output terminal of c. Therefore, in order for $\mathcal{A}$ to communicate with the AS $\mathcal{A}'$, a link from the output terminal y of $\mathcal{A}$ to an input terminal x' of $\mathcal{A}'$ should be defined (point 3 above). To this end, we require that each dangling (input) terminal of $\mathcal{A}'$ be connected with an output terminal of another AS. That is, *the input terminals of $\mathcal{A}'$ are the dangling input terminals of its components*. For instance, if a link is defined from the output terminal y of $\mathcal{A}$ to the input terminal x' of $\mathcal{A}'$, with x' being a dangling input terminal of a component c' in $\mathcal{A}'$, then the occurrence of an emergent event $e(y)$ makes e ready at the terminal x' of c'. Once ready, $e(x')$ can be consumed by a transition of c', just like any other event generated by a component in $\mathcal{A}'$. The enhancement of an AS by input terminals, output terminals, and emergent events leads to the notion of an AU defined below.

12. The qualifier "emergent" applied to these events is inherited from the science of complex systems, which inspired the present work. The basic characteristics of a complex system is the emergence of unpredictable behavior resulting from the interaction among individual components. This *emergent behavior* cannot be explained by the superimposition of the behavior of the interacting components. This paradigm is applicable to natural systems, such as animals, as well to artificial systems, such as cities, companies, and economies (Atay & Jost, 2004; Bossomaier & Green, 2007; Licata & Sakaji, 2008; Goles & Martinez, 2001).



$e(y)$	*Regular expression*	*Description*
di	b_1	Disconnection
co	b_2	Connection
nd	$s_3 \mid s_1\, b_3$	Failure in disconnecting
nc	$s_4 \mid s_2\, b_4$	Failure in connecting

Figure 5: AU $\mathcal{P}$ (left) and corresponding emergent events (right).

Definition 11 (Active Unit) *An* active unit *is a quadruple $\mathcal{U} = (\mathcal{A}, I, O, E)$ where $\mathcal{A}$ is an AS, I the set of input terminals, these being the dangling input terminals of all the components in $\mathcal{A}$, O is the set of output terminals, and E is the set of emergent events; each emergent event is defined by a pair $(e(y), r)$, where e is the event, $y \in O$, and r is a regular expression on transitions of components in $\mathcal{A}$.*

Example 12 Consider the AS $\mathcal{A}$ defined in Example 1 and depicted in Figure 1. Outlined on the left side of Figure 5 is an AU $\mathcal{P} = (\mathcal{A}, I, O, E)$ derived from $\mathcal{A}$, where $I = \emptyset$, $O = \{y\}$, and the set E of emergent events is $\{di, co, nd, nc\}$, with the following meaning: *di* = disconnected, *co* = connected, *nd* = not disconnected (failure), and *nc* = not connected (failure). The regular expressions of the emergent events are defined on the right side of Figure 5, the alphabet of each regular expression being $\Sigma = \{s_1, s_2, s_3, s_4, b_1, b_2, b_3, b_4, b_5, b_6\}$.[13] For instance, *nd* (failure in disconnecting) occurs when either the sensor s detects the short circuit but sends the wrong command to the breaker b, namely the transition s_3, or s detects the short circuit and commands b to open but b keeps being closed, namely the transition s_1 followed by the transition b_3.

The notions of viewer, ruler, and temporal observation for ASs are applicable to AUs also.

3.1 More on Emergent Events

As pointed out in Section 2, within an AS $\mathcal{A}$, events are generated upon the occurrence of component transitions. These events are called *internal events*, meaning that they are generated (and consumed) within $\mathcal{A}$. When an AU $\mathcal{U} = (\mathcal{A}, I, O, E)$ is considered, a new class of events is specified, namely the class of *emergent events*, those defined in E. Specifically,

$$E = \{(e_1(y_1), r_1), \ldots, (e_h(y_h), r_h)\} \tag{11}$$

where $\forall i \in [1\,..\,h]$, e_i is an event, y_i an output terminal of $\mathcal{U}$ ($y_i \in O$), and r_i a regular expression whose alphabet Σ is a (not necessarily strict) subset of the transitions of components in $\mathcal{A}$. Given $(e(y), r)$, the emergent event e occurs in $\mathcal{U}$ (and is placed at the output terminal y) whenever a subtrajectory S within $\mathcal{A}^*$ matches r, in other words, whenever $S \in \|r\|$ (S is a string of the regular language of r). In order to keep tracking the matching of any subtrajectory with r, a DFA is generated (cf. Section 5), called the *matcher* of $(e(y), r)$, denoted $\mu(e(y), r)$. As clarified in

13. Basically, a regular expression is defined inductively on an alphabet Σ as follows. The empty symbol ε is a regular expression. If $a \in \Sigma$, then a is a regular expression. If x and y are regular expressions, then the followings are regular expressions: $x \mid y$ (alternative), xy (concatenation), $x?$ (optionality), x^* (repetition zero or more times), and x^+ (repetition one or more times). When parentheses are missing, the concatenation has precedence over the alternative; for example, $ab \mid c$ equates to $(ab) \mid c$.

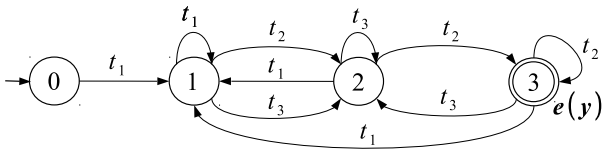


Figure 6: Matcher $\mu(e(y), r)$, where $r = t_1\,(t_2 \mid t_3)^+ t_2$, $\Sigma = \{t_1, t_2, t_3\}$.

Section 5, a matcher $\mu(e(y), r)$ is not simply the DFA recognizing the language of r, as the strings in r may overlap in a trajectory.

Example 13 Let $\mathcal{U} = (\mathcal{A}, I, O, E)$ be an AU, $(e(y), r)$ an emergent event in $\mathcal{U}$, and $\Sigma = \{t_1, t_2, t_3\}$ the alphabet of r, where $r = t_1\,(t_2 \mid t_3)^+ t_2$ is the regular expression associated with $e(y)$. Assume the following trajectory for $\mathcal{A}$:

$$T = [\,t_2, t_3, t_1, t_2, \underbrace{t_1, t_3, t_2}_{T'}, t_4, t_1, \underbrace{t_1, t_2, t_3, t_3, t_2}_{T''}, t_3\,] \tag{12}$$

where the two subtrajectories of T matching the regular expression r are $T' = [t_1, t_3, t_2]$ and $T'' = [t_1, t_2, t_3, t_3, t_2]$. Accordingly, the emergent event $e(y)$ is generated at the occurrence of the last transition in either subtrajectory, namely t_2.

The matcher $\mu(e(y), r)$ is displayed in Figure 6, where the final state 3 is marked with the emergent event $e(y)$.[14] Starting from the initial state 0, a state transition is performed in $\mu(e(y), r)$ at any component transition $t_i \in T$ belonging to the alphabet Σ. If no transition exiting the current state is marked with t_i, then such a mismatch causes the current state to become the initial state 0.

Shown in Table 1 is the association between each component transition in the trajectory T (top), the corresponding state in the matcher $\mu(e(y), r)$ (middle), and the occurrences of emergent events (bottom). The instances of the final state 3 in bold correspond to the generation of the two occurrences of the emergent event $e(y)$.

t_2	t_3	t_1	t_2	t_1	t_3	t_2	t_4	t_1	t_1	t_2	t_3	t_3	t_2	t_3
0	0	1	2	1	2	**3**	3	1	1	2	2	2	**3**	2
						$\boldsymbol{e(y)}$							$\boldsymbol{e(y)}$	

Table 1: Detection of emergent events.

3.2 Unit Space

Since an AU $\mathcal{U}$ is an extension of an AS $\mathcal{A}$ by means of a set of output terminals and a set of emergent events generated on these terminals (defined by regular expressions on component transitions), the notion of a space for $\mathcal{A}$, namely $\mathcal{A}^*$, can be naturally extended to $\mathcal{U}$, namely $\mathcal{U}^*$. In so doing, a state in $\mathcal{U}^*$ is the extension of a state in $\mathcal{A}^*$, namely $(\mathbb{S}, \mathbb{E})$, by an additional field $\mathbb{M}$ representing the array of states of the matchers of the regular expressions associated with the emergent events. Hence, a state in $\mathcal{U}^*$ is a triple $(\mathbb{S}, \mathbb{E}, \mathbb{M})$ where $(\mathbb{S}, \mathbb{E})$ is a state in $\mathcal{A}^*$. When $\mathcal{U}$ moves to a state

14. Details on the construction of a matcher are given in Section 5.

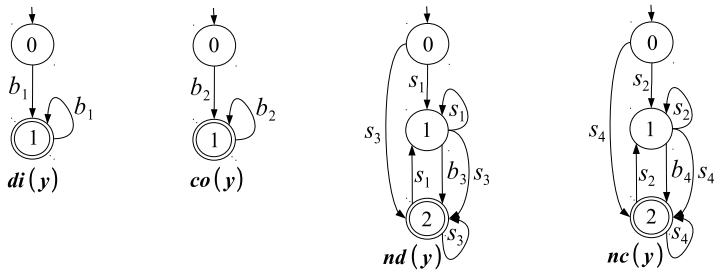


Figure 7: Matchers of the emergent events *di*, *co*, *nd*, and *nc*, at the output terminal y of the AU $\mathcal{P}$.

in which $\mathbb{M}$ includes a final state of the matcher associated with an emergent event $e(y)$, the event e is generated at the output terminal y of $\mathcal{U}$.

Definition 12 (Unit Space) *Let $\mathcal{U} = (\mathcal{A}, I, O, E)$ be an AU, where $\mathcal{A}^* = (\Sigma, A, \tau_A, \alpha_0, A_f)$, and $E = ((e_1(y_1), r_1), \ldots, (e_h(y_h), r_h))$ with corresponding array of matchers $\boldsymbol{\mu}(E) = (\mu_1, \ldots, \mu_h)$. The* space *of $\mathcal{U}$, denoted $\mathcal{U}^*$, is a DFA*

$$\mathcal{U}^* = (\Sigma, U, \tau, u_o, U_f) \tag{13}$$

where: the alphabet Σ is the set of transitions of the components in $\mathcal{A}$ (the same alphabet as that of $\mathcal{A}^*$); U is the set of states $(\mathbb{S}, \mathbb{E}, \mathbb{M})$, where $(\mathbb{S}, \mathbb{E}) \in A$, and $\mathbb{M} = (m_1, \ldots, m_h)$ is the array of states of the matchers in $\boldsymbol{\mu}(E)$; $u_0 = (\mathbb{S}_0, \mathbb{E}_0, \mathbb{M}_0)$ is the initial state, where $(\mathbb{S}_0, \mathbb{E}_0) = \alpha_0$ and $\mathbb{M}_0 = (m_{10}, \ldots, m_{h0})$ is the array of initial states of the matchers in $\boldsymbol{\mu}(E)$; $U_f \subseteq U$ is the set of final states $(\mathbb{S}_f, \mathbb{E}_f, \mathbb{M}_f)$, where $(\mathbb{S}_f, \mathbb{E}_f) \in A_f$; τ is the transition function, $\tau : U \times \Sigma \mapsto U$, where $\langle(\mathbb{S}, \mathbb{E}, \mathbb{M}), t(c), (\mathbb{S}', \mathbb{E}', \mathbb{M}')\rangle \in \tau$ iff: (1) $\langle(\mathbb{S}, \mathbb{E}), t(c), (\mathbb{S}', \mathbb{E}')\rangle \in \tau_A$ (the transition function of $\mathcal{A}^*$), and (2) $\mathbb{M} = (m_1, \ldots, m_h)$, $\mathbb{M}' = (m'_1, \ldots, m'_h)$, and $\forall i \in [1\,..\,h]$ we have

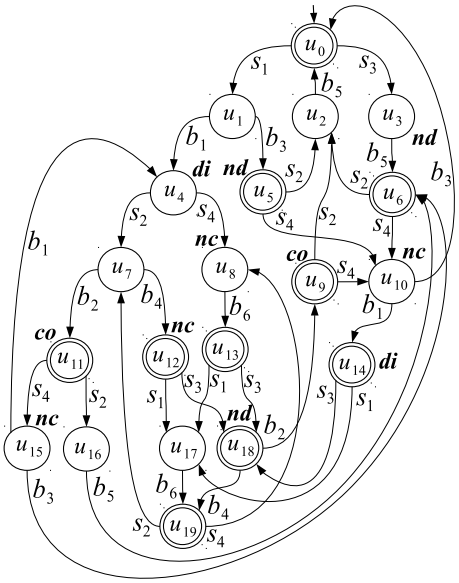
$$m'_i = \begin{cases} \bar{m}_i & \text{if } t(c) \in \Sigma(r_i) \text{ and } \langle m_i, t(c), \bar{m}_i\rangle \in \mu_i \\ m_{i0} & \text{if } t(c) \in \Sigma(r_i) \text{ and no transition marked with } t(c) \text{ exits } m_i \text{ in } \mu_i \\ m_i & \text{if } t(c) \notin \Sigma(r_i). \end{cases} \tag{14}$$

Example 14 With reference to the AU $\mathcal{P}$ defined in Example 12, displayed in Figure 7 are the matchers of the emergent events *di*, *co*, *nd*, and *nc*, where the alphabet of the regular expressions is assumed to be the totality of the transitions of the components in $\mathcal{P}$. Outlined on the left side of Figure 8 is the unit space $\mathcal{P}^*$, where relevant states are marked with the emergent events (the output terminal y is omitted). The details of the states are listed on the right side of the figure, where a bold state in $\mathbb{M}$ indicates the detection of an emergent event by the corresponding matcher (cf. Figure 7).

The notions of trajectory, semi-trajectory, and subtrajectory introduced in Section 2.1 are still valid when AU spaces are considered.

4. Deep Discrete-Event Systems

Intuitively, a DDES is a hierarchical network of AUs. Like components within an AS, AUs are connected with one another through links. Specifically, a link l connects an output terminal y of an



State	$\mathbb{S}$	$\mathbb{E}$	$\mathbb{M}$
u_0	(*idle*, *closed*)	(ε)	(0, 0, 0, 0)
u_1	(*awaken*, *closed*)	(*op*)	(0, 0, 1, 0)
u_2	(*idle*, *closed*)	(*cl*)	(0, 0, 0, 1)
u_3	(*awaken*, *closed*)	(*cl*)	(0, 0, **2**, 0)
u_4	(*awaken*, *open*)	(ε)	(**1**, 0, 0, 0)
u_5	(*awaken*, *closed*)	(ε)	(0, 0, **2**, 0)
u_6	(*awaken*, *closed*)	(ε)	(0, 0, 0, 0)
u_7	(*idle*, *open*)	(*cl*)	(0, 0, 0, 1)
u_8	(*idle*, *open*)	(*op*)	(0, 0, 0, **2**)
u_9	(*awaken*, *closed*)	(ε)	(0, **1**, 0, 0)
u_{10}	(*idle*, *closed*)	(*op*)	(0, 0, 0, **2**)
u_{11}	(*idle*, *closed*)	(ε)	(0, **1**, 0, 0)
u_{12}	(*idle*, *open*)	(ε)	(0, 0, 0, **2**)
u_{13}	(*idle*, *open*)	(ε)	(0, 0, 0, 0)
u_{14}	(*idle*, *open*)	(ε)	(**1**, 0, 0, 0)
u_{15}	(*awaken*, *closed*)	(*op*)	(0, 0, 0, **2**)
u_{16}	(*awaken*, *closed*)	(*cl*)	(0, 0, 0, 1)
u_{17}	(*awaken*, *open*)	(*op*)	(0, 0, 1, 0)
u_{18}	(*awaken*, *open*)	(*cl*)	(0, 0, **2**, 0)
u_{19}	(*awaken*, *open*)	(ε)	(0, 0, 0, 0)

Figure 8: Space of the AU $\mathcal{P}$ (left) and content of the states (right).

AU $\mathcal{U}$ with an input terminal x' of another AU $\mathcal{U}'$. When an emergent event e occurs at the terminal y and is loaded in the link l, $e(x')$ is ready to be consumed by the component in $\mathcal{U}'$ having x' as an input terminal.[15] The mode in which the AUs are linked together should be consistent with the following topological constraints (cf. Figure 9): (1) all links exiting an AU $\mathcal{U}$ enter the *same* AU $\mathcal{U}'$, where $\mathcal{U}$ is called a *child* of $\mathcal{U}'$ and $\mathcal{U}'$ the *parent* of $\mathcal{U}$, (2) every output terminal of $\mathcal{U}$ is exited by exactly one link, and (3) every input terminal of $\mathcal{U}'$ is entered by exactly one link. Constraint 1 forces the topology to be hierarchical, as in the abstract example outlined in Figure 9, where nodes denote AUs and arcs denote links from child to parent AUs.

Definition 13 (DDES) *A DDES $\mathcal{X}$ is a pair $\mathcal{X} = (\mathbf{U}, \mathbf{L})$, where $\mathbf{U}$ is an array of AUs and $\mathbf{L}$ an array of (external) links between these AUs.*

Choosing whether to represent a physical system as a DDES rather than a DES primarily depends on the complexity of the system. If the system is inherently hierarchical in structure, where each subsystem in the hierarchy is characterized by a behavior that is more abstract than the behavior of its children, then a DDES may be a better choice than a DES, because the abstractions of the physical system can be represented in a natural way in the DDES.

For instance, for the purpose of diagnosis, an aircraft (or a part of it) may be better represented by a DDES rather than a DES. For redundancy reasons related to safety, the "engine" of the aircraft may be composed of several physical engines. In the DDES, the "engine" may be an AU, with the physical engines being its child AUs. Since a fault in a physical engine does not necessarily

15. As remarked in Section 3, the input terminals of an AU $\mathcal{U} = (\mathcal{A}, I, O, E)$ are all the dangling terminals of the AS $\mathcal{A}$, namely the input terminals of the components in $\mathcal{A}$ that are not connected with a link in $\mathcal{A}$.

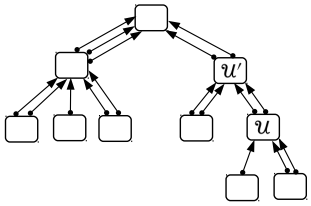


Figure 9: Hierarchy of a DDES: nodes are AUs and arcs are links.

jeopardize the behavior of the "engine" as a whole, this information should be reflected in the diagnosis information.[16]

Modeling a *complex* physical system as a DDES combines the advantages of modeling a *simple* physical system as a DES at each level of the hierarchical structure. Ideally, the specification of the behavior of each AU (each subsystem in the hierarchy) is expected to require substantially the same effort regardless of the level in which the AU is placed in the hierarchy, just as the effort for coding a function in a complex software system should not depend on the level of abstraction at which this function is placed within the software architecture. In other words, *hierarchical modularity* is the most significant feature of a physical system being modeled as a DDES. If a physical system is not hierarchically modular in nature, there is (in principle) no reason to modeling it as a DDES.

In order to model a physical system as a DDES, namely $\mathcal{X}$, the following steps are suggested: (1) define the tree of $\mathcal{X}$ based on the hierarchical structure of the physical system; (2) associate with each node of the tree an AU with relevant components and links; (3) specify the emergent events for each AU; (4) for each AU, specify the model of each component in terms of a communicating automaton; (5) for each AU, specify a viewer (observable transitions) and a ruler (faulty transitions).

In these steps, it is convenient to think with a teleological attitude, considering that the modeling of the physical system as a DDES is meant for diagnosis purposes. In other words, the model of the physical system expressed by the DDES $\mathcal{X}$ is not general-purpose in nature: it is *special-purpose*, that is, oriented to the task of diagnosis. This, however, does not prevent a priori $\mathcal{X}$ to be exploited for other purposes, such as simulation or monitoring.[17]

Example 15 In Example 1 of Section 2 we have defined an AS $\mathcal{A}$ that includes a sensor and a breaker. In Example 12 of Section 3, $\mathcal{A}$ has been extended to an AU $\mathcal{P}$. Interestingly, $\mathcal{P}$ can be part of the hierarchical structure of a DDES modeling a power transmission line. Outlined in Figure 10 is the schema of a protected power transmission line. The line is equipped with a protection apparatus on each side. On the left side, the apparatus involves the sensor s, the primary breaker b, the monitor m, and the recovery breaker r. The behavior of s and b is the one described in Example 1: if a short circuit strikes the line, then the sensor s commands the breaker b to open. To provide some sort of fault tolerance in the protection apparatus, if a misbehavior occurs (either s does not command b to open or, even if correctly commanded, b does not open), then the monitor m intervenes by commanding the recovery breaker r to open and, at the same time, by informing the monitor m' on the opposite side to perform the same recovery action on the breaker r'. For safety reasons, once opened, the recovery breakers cannot be closed again, thereby leaving the line isolated. The same

16. Compare with the notions of negative/positive paradox introduced by Lamperti and Zanella (2011).
17. In fact, monitoring-based diagnosis is a natural extension of the diagnosis task for DDESs proposed in this paper.

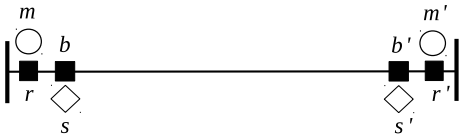


Figure 10: Schema of a protected power transmission line.

applies to the protection apparatus on the right side of the line. The protected transmission line in Figure 10 can be modeled as the DDES $\mathcal{X}$ displayed in Figure 11.

$\mathcal{X}$ incorporates four AUs, namely the protection apparata $\mathcal{P}$ and $\mathcal{P}'$ on the bottom of the hierarchy, the monitoring apparatus $\mathcal{M}$ on the middle, and the line $\mathcal{L}$ on the top. The output terminals of the AUs $\mathcal{P}$ and $\mathcal{P}'$ are connected with the input terminals of the components m and m' (in $\mathcal{M}$), respectively. Moreover, the AU $\mathcal{M}$ is equipped with two output terminals, namely y_1 and y_2, each one being connected with an input terminal of l. In order to complete the specification of $\mathcal{X}$, we need to provide the models of the monitor and the line, as well as the specification of the emergent events of $\mathcal{M}$. The model of the monitor is outlined in Figure 12, in terms of relevant input/output terminals (left) and communicating automaton (right), with transitions being detailed in Table 2. The monitor moves from state *watch* to *trigger* upon the occurrence of either the event *nd* (failure in disconnecting the side of the line) emerging from the protection apparatus or the event *rc* (request of recovery action) coming from the other monitor. In the first scenario, based on m_1, the monitor commands the corresponding recovery breaker to open and informs the other monitor. However, based on m_2, the monitor can command the corresponding recovery breaker to open without informing the other monitor. In the second scenario (consumption of *rc*), the monitor commands the corresponding recovery breaker to open (transition m_3). Transitions $m_4 \cdots m_{10}$ consume the input event (emerging from the protection apparatus) without changing state. According to Figure 11, the AU $\mathcal{M}$ is provided with the output terminals y_1 and y_2 (which have not to be confused with the homonymous terminals of both monitors m and m'). In either terminal, three emergent events can be placed, namely (1) *nr*: the corresponding side of the line cannot be reconnected, (2) *ni*: the corresponding side of the line cannot be isolated, and (3) *ps*: the short circuit persists on the corresponding side of the line. The specification of these emergent events in terms of regular expressions

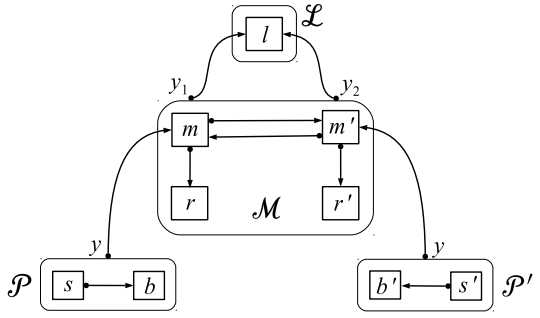


Figure 11: DDES $\mathcal{X}$, with AUs $\mathcal{L}$, $\mathcal{M}$, $\mathcal{P}$, and $\mathcal{P}'$.

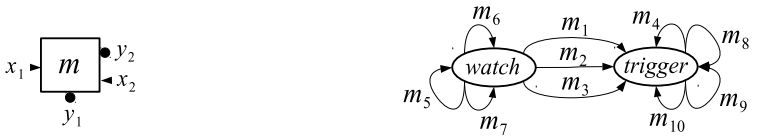


Figure 12: Model of the *monitor* component in the DDES depicted in Figure 11.

Transition	From state	Input event	Output events	To state
m_1	*watch*	$nd(x_1)$	$\{op(y_1), rc(y_2)\}$	*trigger*
m_2	*watch*	$nd(x_1)$	$\{op(y_1)\}$	*trigger*
m_3	*watch*	$rc(x_2)$	$\{op(y_1)\}$	*trigger*
m_4	*trigger*	$nd(x_1)$	$\emptyset$	*trigger*
m_5	*watch*	$di(x_1)$	$\emptyset$	*watch*
m_6	*watch*	$co(x_1)$	$\emptyset$	*watch*
m_7	*watch*	$nc(x_1)$	$\emptyset$	*watch*
m_8	*trigger*	$di(x_1)$	$\emptyset$	*trigger*
m_9	*trigger*	$co(x_1)$	$\emptyset$	*trigger*
m_{10}	*trigger*	$nc(x_1)$	$\emptyset$	*trigger*

Table 2: Transition details for the *monitor* automaton displayed on the right side of Figure 12.

is provided in Table 3.[18] The matchers of these events are depicted in Figure 13. The model of the line is shown in Figure 14, with the transitions being detailed in Table 4. The line moves from the state *normal* to *shorted*, *isolated*, or *persistent*, without generating any output event. Since the input terminals x_1 and x_2 of the line are connected with the output terminals of the AU $\mathcal{M}$, the transitions in the line are triggered by the emergent events in $\mathcal{M}$.

4.1 DDES Space and Trajectory

Since a DDES $\mathcal{X}$ is a (hierarchical) network of AUs connected by (external) links, just as an AS is a network of components connected by (internal) links, a state of $\mathcal{X}$ is identified by a pair $(\mathbb{U}, \mathbb{E})$, where $\mathbb{U}$ is the array of states of the AUs in $\mathcal{X}$, and $\mathbb{E}$ is the array of emergent events that are loaded in the (external) links.

Definition 14 (DDES Space) *Let $\mathcal{X} = (\mathbf{U}, \mathbf{L})$ be a DDES, where $\mathbf{U} = (\mathcal{U}_1, \ldots, \mathcal{U}_n)$, and $\mathbf{L} = (l_1, \ldots, l_m)$. The* space *of $\mathcal{X}$, denoted $\mathcal{X}^*$, is a DFA*

$$\mathcal{X}^* = (\Sigma, X, \tau, x_o, X_f) \tag{15}$$

where the alphabet Σ is the union of the sets of the transitions in $\mathcal{U}_i^*$, $i \in [1..n]$; X is the set of states $(\mathbb{U}, \mathbb{E})$, where $\mathbb{U} = (u_1, \ldots, u_n)$ is the array of states of the AUs in $\mathbf{U}$, $\mathbb{E} = (e_1, \ldots, e_m)$ is the array of the (possibly empty) emergent events loaded within the links in $\mathbf{L}$;[19] *$x_0 = (\mathbb{U}_0, \mathbb{E}_0)$ is*

18. The table specifies only the emergent events at the output terminal y_1. The same events generated at the terminal y_2 have a similar specification, where m and r are replaced with m' and r', respectively.

19. If nonempty, these emergent events are ready at the corresponding input terminals of the AUs in $\mathbf{U}$, so they can be consumed by the components.

$e(y_1)$	Alphabet Σ	Regular expression
$nr(y_1)$	$\{ m_7(m), m_{10}(m), b_1(r) \}$	$m_7(m) \mid m_{10}(m) \mid b_1(r)$
$ni(y_1)$	$\{ m_1(m), m_2(m), m_4(m), b_3(r) \}$	$(m_1(m) \mid m_2(m) \mid m_4(m))\, b_3(r) \mid b_3(r)\, m_4(m)$
$ps(y_1)$	$\{ m_5(m), m_6(m) \}$	$m_6(m)\, m_5(m)$

Table 3: Emergent events at output terminal y_1 of $\mathcal{M}$.

the initial state, where $\mathbb{U}_0 = (u_{10}, \ldots, u_{n0})$ *is the array of initial states of the AUs in* $\mathbf{U}$*, while* $\mathbb{E}_0 = (\varepsilon, \ldots, \varepsilon)$*, that is, all emergent events in* $\mathbb{E}_0$ *are empty;* $X_\mathrm{f} \subseteq X$ *is the set of the final states* $(\mathbb{U}_\mathrm{f}, \mathbb{E}_\mathrm{f})$*, where* $\mathbb{U}_\mathrm{f} = (u_{1_\mathrm{f}}, \ldots, u_{n_\mathrm{f}})$*,* $\forall i \in [1\,..\,n]$*,* u_{i_f} *is a final state in* $\mathcal{U}_i^*$*, while all emergent events in* $\mathbb{E}_\mathrm{f}$ *are empty;* τ *is the transition function,* $\tau : X \times \Sigma \mapsto X$*, where* $\langle (\mathbb{U}, \mathbb{E}), t(\mathcal{U}_i), (\mathbb{U}', \mathbb{E}') \rangle \in \tau$ *iff:*

- $\mathbb{U} = (u_1, \ldots, u_n)$*,* $\mathbb{U}' = (u'_1, \ldots, u'_n)$*,* $\mathbb{E} = (e_1, \ldots, e_m)$*,* $\mathbb{E}' = (e'_1, \ldots, e'_m)$*;*
- $t(\mathcal{U}_i) = \langle u_i, \bar{t}(c), \bar{u}_i \rangle$*,* $\bar{t}(c) = \langle s, (e(x), E), s' \rangle$*, where* $\{e'_1(y'_1), \ldots, e'_h(y'_h)\} \subseteq E$ *is the subset of the output events generated by* $\bar{t}(c)$ *at the output terminals of* $\mathcal{U}_i$*;*
- $\forall j \in [1\,..\,n] \left(u'_j = \left\{ \begin{array}{ll} \bar{u}_i & \text{if } i = j \\ u_j & \text{otherwise} \end{array} \right. \right)$*;*
- *If* x *is an input terminal of* $\mathcal{U}_i$*, then* $\mathbb{E}(x) = e$*;*
- $\forall j \in [1\,..\,h] \left(\mathbb{E}(y'_j) = \varepsilon \right)$*;*
- $\forall j \in [1\,..\,m] \left(\mathbb{E}'[j] = \left\{ \begin{array}{ll} e'_i & \text{if } e'_i(y'_i) \in E, i \in [1\,..\,h], \text{ and } link(y'_i) = l_j \\ \varepsilon & \text{if } x \text{ is an input terminal of } \mathcal{U}_i \text{ and } link(x) = l_j \\ \mathbb{E}[j] & \text{otherwise} \end{array} \right. \right)$.

A trajectory $T = [t_1(c_1), \ldots, t_q(c_q)]$ *within the space* $\mathcal{X}^* = (\Sigma, X, \tau, x_0, X_\mathrm{f})$ *is a finite sequence of transitions of the components within the AUs in* $\mathbf{U}$ *such that,* $\forall i \in [1\,..\,q]$*,*

$$\left\langle x_{i-1}, t(\mathcal{U}_j), x_i \right\rangle \in \tau, t(\mathcal{U}_j) = \left\langle u, t_i(c_i), u' \right\rangle, \text{ and } x_q \in X_\mathrm{f}. \tag{16}$$

The set of all the trajectories in $\mathcal{X}^*$ *is written* $\|\mathcal{X}^*\|$.

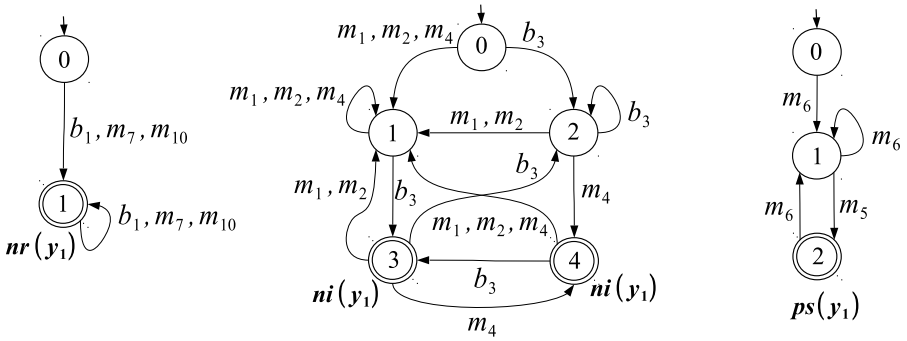


Figure 13: Matchers of emergent events *nr*, *ni*, and *ps*, generated at terminal y_1 of $\mathcal{M}$ (cf. Table 3).

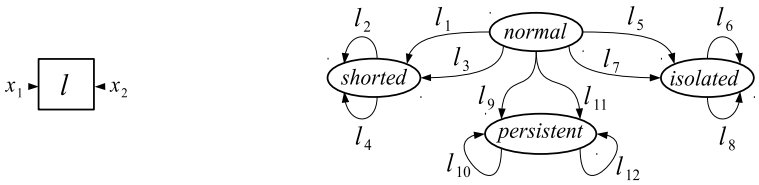


Figure 14: Model of the *line* component (cf. Figure 11).

Transition	From state	Input event	Output events	To state
l_1	*normal*	$ni(x_1)$	$\emptyset$	*shorted*
l_2	*shorted*	$ni(x_1)$	$\emptyset$	*shorted*
l_3	*normal*	$ni(x_2)$	$\emptyset$	*shorted*
l_4	*shorted*	$ni(x_2)$	$\emptyset$	*shorted*
l_5	*normal*	$nr(x_1)$	$\emptyset$	*isolated*
l_6	*isolated*	$nr(x_1)$	$\emptyset$	*isolated*
l_7	*normal*	$nr(x_2)$	$\emptyset$	*isolated*
l_8	*isolated*	$nr(x_2)$	$\emptyset$	*isolated*
l_9	*normal*	$ps(x_1)$	$\emptyset$	*persistent*
l_{10}	*persistent*	$ps(x_1)$	$\emptyset$	*persistent*
l_{11}	*normal*	$ps(x_2)$	$\emptyset$	*persistent*
l_{12}	*persistent*	$ps(x_2)$	$\emptyset$	*persistent*

Table 4: Transition details for the *line* automaton displayed on the right side of Figure 14.

Example 16 With reference to the DDES $\mathcal{X}$ in Example 15, assuming that initially the breakers are *closed*, the sensors are *idle*, the monitors are in *watch* mode, and the line is *normal*, a trajectory in $\mathcal{X}^*$ is $T = [s_1(s), b_3(b), s_1(s'), b_1(b'), m_2(m), b_1(r), l_5(l), m_5(m')]$, which can be paraphrased as follows: the sensor s detects a short circuit and commands the breaker b to open; however, b does not open; the sensor s' detects a short circuit and commands the breaker b' to open; b' opens; the monitor m detects the malfunction within the protection apparatus on the left side of the line and commands the recovery breaker r to open without however informing the monitor m'; r opens; the line l cannot be reconnected because, for safety reasons, the recovery breakers cannot be closed; m acknowledges the disconnection of the left side of the line.

4.2 DDES Viewer and Temporal Observation

The notions of a viewer and a temporal observation, defined in Section 2.2 for an AS and naturally inherited by an AU, can be extended to a DDES, as outlined below.

Definition 15 (DDES Viewer) *Let $\mathcal{X} = (\mathbf{U}, \mathbf{L})$ be a DDES, where $\mathbf{U} = (\mathcal{U}_1, \ldots, \mathcal{U}_n)$. A* viewer *for $\mathcal{X}$ is an array $\mathcal{V} = (\mathcal{V}_1, \ldots, \mathcal{V}_n)$, where $\forall i \in [1\,..\,n]$, $\mathcal{V}_i$ is a viewer for $\mathcal{U}_i$.*

Example 17 With reference to Example 15, a viewer for $\mathcal{X}$ is $\mathcal{V} = (\mathcal{V}_{\mathcal{P}}, \mathcal{V}_{\mathcal{P}'}, \mathcal{V}_{\mathcal{M}}, \mathcal{V}_{\mathcal{L}})$, where:

$$\begin{aligned}
\mathcal{V}_{\mathcal{P}} &= \{(s_1(s), awk), (s_2(s), ide), (s_3(s), awk), (s_4(s), ide), (b_1(b), bop), (b_2(b), bcl)\} \\
\mathcal{V}_{\mathcal{P}'} &= \{(s_1(s'), awk'), (s_2(s'), ide'), (s_3(s'), awk'), (s_4(s'), ide'), (b_1(b'), bop'), (b_2(b'), bcl')\} \\
\mathcal{V}_{\mathcal{M}} &= \{(m_1(m), trg), (m_2(m), trg), (m_3(m), trg), (b_1(r), rop), (b_2(r), rcl), (m_1(m'), trg'), \\
&\quad (m_2(m'), trg'), (m_3(m'), trg'), (b_1(r'), rop'), (b_2(r'), rcl')\}.
\end{aligned}$$

$\mathcal{V}_{\mathcal{L}} = \emptyset$, therefore the AU $\mathcal{L}$ (line) is "unobservable".

Definition 16 (DDES Temporal Observation) *Let T be a trajectory within a space $\mathcal{X}^*$. The* temporal observation *of T based on the viewer $\mathcal{V}$, denoted $Obs(T, \mathcal{V})$, is the array $Obs(T, \mathcal{V}) = (\mathcal{O}_1, \ldots, \mathcal{O}_n)$, where $\forall i \in [1\,..\,n]$, $\mathcal{O}_i = [\,\ell \mid t(c) \in T, c \in \mathcal{U}_i, (t(c), \ell) \in \mathcal{V}_i\,]$.*

Example 18 With reference to the trajectory T of $\mathcal{X}$ in Example 16 and the viewer $\mathcal{V}$ of $\mathcal{X}$ in Example 17, the temporal observation of T based on $\mathcal{V}$ is $Obs(T, \mathcal{V}) = ([\,awk\,], [\,awk', bop'\,], [\,trg, rop\,], [\,])$.

4.3 DDES Ruler and Diagnosis

The notions of a ruler and a diagnosis, defined in Section 2.3 for an AS and naturally inherited by an AU, can be extended to a DDES, as detailed below.

Definition 17 (DDES Ruler) *Let $\mathcal{X} = (\mathbf{U}, \mathbf{L})$ be a DDES, where $\mathbf{U} = (\mathcal{U}_1, \ldots, \mathcal{U}_n)$. A* ruler *for $\mathcal{X}$ is an array $\mathcal{R} = (\mathcal{R}_1, \ldots, \mathcal{R}_n)$, where $\forall i \in [1\,..\,n]$, $\mathcal{R}_i$ is a ruler for $\mathcal{U}_i$.*

Example 19 With reference to Example 15, a ruler for $\mathcal{X}$ is $\mathcal{R} = (\mathcal{R}_{\mathcal{P}}, \mathcal{R}_{\mathcal{P}'}, \mathcal{R}_{\mathcal{M}}, \mathcal{R}_{\mathcal{L}})$, where:

$$\begin{aligned}
\mathcal{R}_{\mathcal{P}} &= \{(s_3(s), fso), (s_4(s), fsc), (b_3(b), fbo), (b_4(b), fbc)\} \\
\mathcal{R}_{\mathcal{P}'} &= \{(s_3(s'), fso'), (s_4(s'), fsc'), (b_3(b'), fbo'), (b_4(b'), fbc')\} \\
\mathcal{R}_{\mathcal{M}} &= \{(m_2(m), fm), (b_3(r), fro), (m_2(m'), fm'), (b_3(r'), fro')\} \\
\mathcal{R}_{\mathcal{L}} &= \{(l_1(l), fls), (l_2(l), fls), (l_3(l), fls'), (l_4(l), fls'), (l_5(l), fli), (l_6(l), fli), (l_7(l), fli'), \\
&\quad (l_8(l), fli'), (l_9(l), flp), (l_{10}(l), flp), (l_{11}(l), flp'), (l_{12}(l), flp')\}.
\end{aligned}$$

The meaning of the faults is explained in Table 5. Since, by assumption, the recovery breakers r and r' cannot be closed once opened, the transition b_4 is not involved in $\mathcal{R}_{\mathcal{M}}$.

Definition 18 (DDES Diagnosis) *Let T be a trajectory in $\mathcal{X}^*$. The* diagnosis *of T based on a ruler $\mathcal{R} = (\mathcal{R}_1, \ldots, \mathcal{R}_n)$, denoted $Dgn(T, \mathcal{R})$, is the set of faults*

$$Dgn(T, \mathcal{R}) = \{\, f \mid t(c) \in T, c \in \mathcal{U}_i, (t(c), f) \in \mathcal{R}_i, i \in [1\,..\,n]\,\}. \tag{17}$$

Example 20 With reference to the trajectory T in Example 16 and the ruler $\mathcal{R}$ in Example 19, the diagnosis of T based on $\mathcal{R}$ is $Dgn(T, \mathcal{R}) = \{fbo, fm, fli\}$, that is, b fails to open, m fails to alert m', and the line l is isolated because a breaker on the left side remains open.

Fault	Meaning
fso	Sensor s in $\mathcal{P}$ commands breaker b to close instead of open
fsc	Sensor s in $\mathcal{P}$ commands breaker b to open instead of close
fbo	Breaker b in $\mathcal{P}$ keeps being closed instead of opening
fbc	Breaker b in $\mathcal{P}$ keeps being open instead of closing
fso'	Sensor s' in $\mathcal{P}'$ commands breaker b' to close instead of open
fsc'	Sensor s' in $\mathcal{P}'$ commands breaker b' to open instead of close
fbo'	Breaker b' in $\mathcal{P}'$ keeps being closed instead of opening
fbc'	Breaker b' in $\mathcal{P}'$ keeps being open instead of closing
fm	Monitor m in $\mathcal{M}$ does not alert monitor m' for recovery
fro	Recovery breaker r in $\mathcal{M}$ keeps being closed instead of opening
fm'	Monitor m' in $\mathcal{M}$ does not alert monitor m for recovery
fro'	Recovery breaker r' in $\mathcal{M}$ keeps being closed instead of opening
fls	Line l in $\mathcal{L}$ keeps being shorted since no breaker on the left side opened
fls'	Line l in $\mathcal{L}$ keeps being shorted since no breaker on the right side opened
fli	Line l in $\mathcal{L}$ keeps being isolated since a breaker on the left side remains open
fli'	Line l in $\mathcal{L}$ keeps being isolated since a breaker on the right side remains open
flp	The short on line l in $\mathcal{L}$ persists after reclosing breaker b in $\mathcal{P}$
flp'	The short on line l in $\mathcal{L}$ persists after reclosing breaker b' in $\mathcal{P}'$

Table 5: Faults defined in Example 19.

4.4 DDES Diagnosis Problem

The notion of a diagnosis problem for an AS and relevant solution, defined in Section 2.4, can be extended to a DDES.

Definition 19 (DDES Diagnosis Problem) *Let $\mathcal{X}$ be a DDES, x_0 the initial state of $\mathcal{X}$, $\mathcal{V}$ a viewer for $\mathcal{X}$, $\mathcal{O}$ a temporal observation of $\mathcal{X}$, and $\mathcal{R}$ a ruler for $\mathcal{X}$. The quadruple*

$$\wp(\mathcal{X}) = (x_0, \mathcal{V}, \mathcal{O}, \mathcal{R}) \tag{18}$$

is a diagnosis problem *for $\mathcal{X}$. The* solution *to $\wp(\mathcal{X})$ is the set of diagnoses*

$$\Delta(\wp(\mathcal{X})) = \left\{ Dgn(T, \mathcal{R}) \mid T \in \|\mathcal{X}^*\|, Obs(T, \mathcal{V}) = \mathcal{O} \right\}. \tag{19}$$

Example 21 A diagnosis problem for the DDES $\mathcal{X}$ defined in Example 15 is $\wp(\mathcal{X}) = (x_0, \mathcal{V}, \mathcal{O}, \mathcal{R})$, where in the initial state x_0 the breakers are *closed*, the sensors are *idle*, the monitors are *watch*, and the line is *normal*, while the viewer $\mathcal{V}$ is defined in Example 17, $\mathcal{O} = (\mathcal{O}_{\mathcal{P}}, \mathcal{O}_{\mathcal{P}'}, \mathcal{O}_{\mathcal{M}}, \mathcal{O}_{\mathcal{L}})$, with $\mathcal{O}_{\mathcal{P}} = [awk]$, $\mathcal{O}_{\mathcal{P}'} = [awk', bop']$, $\mathcal{O}_{\mathcal{M}} = [trg, rop]$, and $\mathcal{O}_{\mathcal{L}} = [\,]$, and the ruler $\mathcal{R}$ is defined in Example 19.

4.5 Practical Issues in Solving Diagnosis Problems for DDESs

Generating the solution to a diagnosis problem for a DDES based on eqn. (19) is overwhelmingly impractical for the same reasons given in Section 2.4 for ASs. Worse still, the technique provided in Section 2.5 and Section 2.6, namely generation and decoration of the $\mathcal{O}$-constrained space, is bound to become inadequate for solving a diagnosis problem for a DDES. In principle, we might

extend the notion of an $\mathcal{O}$-constrained space to a DDES, namely $\mathcal{X}^*_{\mathcal{O}}$. In so doing, a state of $\mathcal{X}^*_{\mathcal{O}}$ would be identified by a triple $(\mathbb{U}, \mathbb{E}, \Im)$, where $(\mathbb{U}, \mathbb{E})$ is a state in the space $\mathcal{X}^*$, while $\Im$ is the array $(\Im_1, \ldots, \Im_n)$ of indexes of the temporal observations in $\mathcal{O}$. Now, assume that, on average, each AU includes m components. In the worst case, the number of states in $\mathcal{X}^*_{\mathcal{O}}$ becomes at least[20] exponential with the product $n \cdot m$, where n is the number of AUs in $\mathcal{X}$. Hence, the worst-case complexity is at least exponential with the total number $(n \cdot m)$ of components in $\mathcal{X}$, a disturbing perspective. Therefore, solving a DDES diagnosis problem requires more sophisticated (and efficient) techniques than those adopted in diagnosis of ASs. Specifically, building something analogous to the $\mathcal{O}$-constrained space $\mathcal{X}^*_{\mathcal{O}}$ is out of the question. After all, the solution to a diagnosis problem $\wp(\mathcal{X})$ is a set of candidate diagnoses, not a set of candidate trajectories within $\mathcal{X}^*_{\mathcal{O}}$. Therefore, chances are that the solution to $\wp(\mathcal{X})$ can be determined *without* generating $\mathcal{X}^*_{\mathcal{O}}$. The challenge is to design a technique for solving a diagnosis problem for $\mathcal{X}$ that is, on the one hand, *viable*, thereby not requiring the generation of the $\mathcal{O}$-constrained space of $\mathcal{X}$ (hence, neither the space $\mathcal{X}^*$ nor the $\mathcal{O}$-constrained space $\mathcal{X}^*_{\mathcal{O}}$ should be generated) and, on the other, *sound* and *complete*, thereby providing exactly the set $\Delta(\wp(\mathcal{X}))$ of candidate diagnoses defined in eqn. (19).

5. Preprocessing

To speed up the task performed by the diagnosis engine *online*, that is, while the DDES is being operated, a certain amount of preprocessing is to be carried out *offline*, before the DDES starts being operated. Typically, the specification of $\mathcal{X}$ is compiled into a variety of data structures, among which are the matchers of the regular expressions associated with the emergent events defined in the AUs (Section 3.1). In general, for each AU $\mathcal{U}$ in $\mathcal{X}$, a set E of emergent events is defined as in eqn. (11). In order to trace the occurrences of each emergent event $e_i(y_i) \in E$, $i \in [1\,..\,h]$, based on the regular expression r_i, a DFA $\mu(e_i(y_i), r_i)$ needs to be generated offline, with each final state being marked with $e_i(y_i)$. However, since strings of component transitions generating the same emergent event may overlap, the regular language of the matcher is in general wider than the language of the regular expression. To clarify the point, consider the example below.

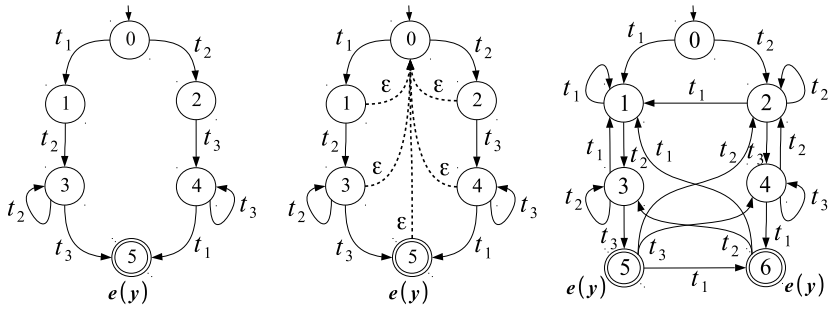
Example 22 Let $(e(y), r)$ be an emergent event in the AU $\mathcal{U}$, with $\Sigma = \{t_1, t_2, t_3\}$ being the alphabet of the regular expression $r = t_1\, t_2^+\, t_3 \,|\, t_2\, t_3^+\, t_1$. Assume the following trajectory T in $\mathcal{U}^*$:

$$T = [\, t_3, \overbrace{t_1, \underbrace{t_2, t_3}, t_1}^{T'}, t_3\,]. \tag{20}$$
$$T''$$

T includes two overlapping subtrajectories matching r, namely $T' = [\, t_1, t_2, t_3\,]$ and $T'' = [\, t_2, t_3, t_1\,]$, where the suffix $[\, t_2, t_3\,]$ of T' is a prefix of T''. Hence, two occurrences of the emergent event $e(y)$ are expected in T in correspondence of the last transition of T' and T'', respectively.

Assume further to trace the occurrences of $e(y)$ based on the *recognizer* of r on the left side of Figure 15, a DFA with the same regular language as r, whose final state is marked with the emergent event $e(y)$. When the final state 5 is reached, the emergent event e is generated at the output terminal y of $\mathcal{U}$. The sequence of states of this recognizer is outlined in the second row of Table 6, where each state is reached by the corresponding transition of T listed in the first row of

20. For more accurate figures (and even increasing complexity), we also need to account for the combinations of arrays of events in links.



μ state	NFA states
0	{0}
1	{0, 1}
2	{0, 2}
3	{0, 2, 3}
4	{0, 4}
5	{0, 4, 5}
6	{0, 1, 5}

Figure 15: From left to right: (1) recognizer of $r = t_1\, t_2^+\, t_3 \,|\, t_2\, t_3^+\, t_1$, where $\Sigma = \{t_1, t_2, t_3\}$; (2) NFA obtained by the insertion of the ε-transitions; (3) matcher $\mu(e(y), r)$ resulting from the determinization of the NFA; and (4) state details for the matcher μ.

the table. As indicated in the third row, the emergent event $e(y)$ is correctly generated at the fourth transition in T, which is the last transition of the subtrajectory T'. At this point, since no transition exits the final state 5, the recognizer starts again from the initial state 0 in order to keep matching. It first changes state to 1 in correspondence of t_1, and with t_3 (mismatch) it returns to 0. The result is that, owing to the overlapping of the subtrajectories T' and T'', *the second occurrence of* $e(y)$ *is not detected*.

Given the pair $(e(y), r)$, in order to recognize all (possibly overlapping) instances of r, we need to transform the recognizer of r into a *matcher* of r, as follows:

1. An ε-transition is inserted into the recognizer from each non-initial state to the initial state;[21]
2. The nondeterministic finite automaton (NFA) obtained in step 1 is determinized into an equivalent DFA;
3. The DFA generated in step 2 is minimized, thereby obtaining the matcher of r.

The final states of the minimized DFA (the matcher) are marked with the emergent event $e(y)$.

Example 23 With reference to Example 22, consider the recognizer of the regular expression r displayed on the left side of Figure 15. Outlined on the right of the recognizer is the NFA obtained by inserting five ε-transitions (dashed arrows) toward the initial state (step 1). The DFA resulting from the determinization of the NFA is displayed next to the NFA (step 2), with states being detailed in the table. Incidentally, this DFA is also minimal, thus minimization (step 3) is not applied. In conclusion, the DFA on the right side of Figure 15 is the matcher $\mu(e(y), r)$, with the final states 5 and 6 being marked with $e(y)$. The dynamics of the matching performed by $\mu(e(y), r)$ on the trajectory T is outlined in the last two rows of Table 6. In sharp contrast with the recognizer of r, which produces only one $e(y)$, the matcher correctly detects the emergent event twice, based on the two overlapping subtrajectories of T, namely $T' = [t_1, t_2, t_3]$ and $T'' = [t_2, t_3, t_1]$. Unlike the recognizer of r, after reaching the state 5 and generating $e(y)$, the next transition t_1 moves the matcher to the other final state 6, thereby generating the second occurrence of $e(y)$ correctly.

Transitions in T	t_3	t_1	t_2	t_3	t_1	t_3
States of recognizer of r	0	1	3	**5**	1	0
Detected emergent events				$\boldsymbol{e(y)}$		
States of matcher $\mu(e(y), r)$	0	1	3	**5**	**6**	0
Detected emergent events				$\boldsymbol{e(y)}$	$\boldsymbol{e(y)}$	

Table 6: Detection of emergent events with overlapping.

A formal peculiarity of a matcher is that, since it is a DFA, it is always in a single state regardless of the trajectory under consideration. Specifically, whatever the number of overlapping events, the mode in which the matcher is derived from the NFA (where the ε-transitions are inserted) allows the matcher to account for the recognition of all these events by means of a single state (simultaneously). But, is the set of emergent events detected by a matcher actually sound and complete? The answer to this question is given by the following proposition.

Proposition 3 *Let $\mu(e(y), r)$ be a matcher for an active unit $\mathcal{U}$. The set of emergent events detected by $\mu(e(y), r)$ is sound and complete.*

Proof. Grounded on Lemma 3.1, Lemma 3.2, Lemma 3.3, and Lemma 3.4.

Lemma 3.1 *Let $\mathcal{N}$ be the NFA from which the matcher $\mu(e(y), r)$ is generated by* Subset Construction*. Each state of $\mu(e(y), r)$ includes the initial state of $\mathcal{N}$.*

Proof. Since $\mathcal{N}$ is the NFA obtained from the recognizer ρ of r by the insertion of an ε-transition from each state of ρ to the initial state of ρ, each state of the matcher generated by *Subset Construction* by means of an ε-closure will include the initial state of $\mathcal{N}$. □

Lemma 3.2 *Let m be a state of the matcher $\mu(e(y), r)$ and T a string in the language of r. There exists a sequence of transitions in the matcher, from m to a final state, that generates T.*

Proof. According to Lemma 3.1, the state m includes the initial state n_0 of the NFA $\mathcal{N}$. Thus, m includes the ε-closure of n_0, namely the initial state m_0 of the matcher. Let $T = [t_1, \ldots, t_k]$. Since t_1 exits m_0, t_1 exits m also. By induction on T, since T leads to a final state m_f of the matcher, the state m' reached by m with T will include the set of $\mathcal{N}$ states identifying m_f. Hence, m' is final also; in other words, T is generated starting from m. □

Lemma 3.3 (*Soundness*) *Let $u = (\mathbb{S}, \mathbb{E}, \mathbb{M})$ be a state of $\mathcal{U}^*$ where the state m in $\mathbb{M}$ relevant to $\mu(e(y), r)$ is final. There exists a subtrajectory T_{ij} entering u such that the subsequence $T_{ij}(\Sigma)$ of T_{ij} that includes the transitions in the alphabet Σ of r is a string in the language of r.*

Proof. By contradiction. Assume that there is no such T_{ij} in $\mathcal{U}^*$. If so, either there is a set of subtrajectories entering u such that the subsequence of a prefix of them that includes the transitions in Σ is a prefix of a string in the language of r or not. If these subtrajectories exist, the state m in $\mathbb{M}$ relevant to $\mu(e(y), r)$ cannot be final. If no such subtrajectory exists, based on Definition 12, m equals the initial state of $\mu(e(y), r)$. Both cases contradict the assumption that m is final. □

21. So, if the recognizer includes n states, then the number of ε-transitions will be $n - 1$.

Lemma 3.4 (*Completeness*) *Let $T_{ij} = [t_i, t_{i+1}, \ldots, t_j]$ be a subtrajectory of $\mathcal{U}$ entering a state $(\mathbb{S}, \mathbb{E}, \mathbb{M})$ of the unit space $\mathcal{U}^*$. Let $T_{ij}(\Sigma)$ be the subsequence of T_{ij} that includes the transitions in the alphabet Σ of r. If $T_{ij}(\Sigma)$ is a string in the language of r, then the state m in $\mathbb{M}$ relevant to the matcher $\mu(e(y), r)$ is marked with the emergent event $e(y)$.*

Proof. Let m be the state of the matcher $\mu(e(y), r)$ when $T_{ij}(\Sigma)$ starts being recognized. By virtue of Lemma 3.2, $T_{ij}(\Sigma)$ is generated from m to a final state of the matcher, upon which the emergent event $e(y)$ is generated. □

Eventually, the proof of Proposition 3 comes from Lemma 3.3 and Lemma 3.4. □

6. Lazy Problem-Solving

As pointed out in Section 4.5, solving a diagnosis problem for a DDES $\mathcal{X}$ by adapting the technique designed for solving diagnosis problems of (plain) ASs is impractical. Specifically, we can no longer rely on the generation of the $\mathcal{O}$-constrained space of $\mathcal{X}$ (cf. Section 2.5). On the other hand, whatever the alternative problem-solving technique, soundness and completeness are required. In other words, any (viable) alternative technique is expected to generate the solution to the diagnosis problem defined in eqn. (19), that is, exactly the same set of candidate diagnoses.

This section introduces a *lazy* technique for solving a DDES diagnosis problem. The technique is called lazy inasmuch it does not require the generation of the $\mathcal{O}$-constrained space of the (whole) DDES $\mathcal{X}$. Instead, it generates an extended constrained space of each AU incorporated in $\mathcal{X}$.

Basically, the idea stems from the following considerations. Since the AUs in $\mathcal{X}$ are accommodated hierarchically, and since the events emerging in one unit $\mathcal{U}$ are consumed by the parent unit $\mathcal{U}'$ only, it follows that $\mathcal{U}$ can influence the behavior of $\mathcal{U}'$. Conversely, since an emergent event in $\mathcal{U}$ cannot be generated if the corresponding link is not empty, it follows that $\mathcal{U}'$ (which may or may not consume the emergent event already stored in the link) may influence the behavior of $\mathcal{U}$. Hence, the behavior of each AU is constrained by the behavior of other units only loosely, that is, through emergent events.

This opens the way to the possibility of focusing on the generation of the subspace of each single AU $\mathcal{U}$ based on the constraints coming from its child units. In generating this new sort of constrained space of $\mathcal{U}$, a subset of the behavior of any child unit of $\mathcal{U}$ can be inconsistent and, as such, it is pruned. The same applies to the behavior of $\mathcal{U}$ when considered in the generation of the constrained space of its parent unit, up to the root node of the hierarchy.

In summary, we can envisage a viable technique that generates the constrained behavior of each AU rather than the constrained behavior of the whole system. The behavior is constrained not only by the temporal observation of the unit, but also by the emergent events generated by the child AUs.

The question now becomes: *How is it possible to focus on the constrained space of each single AU and, at the same time, guarantee the soundness and completeness of the diagnosis result?* The answer to this question stems from the mode in which the constraints from child AUs are enforced. In fact, it is not the constrained space of a child AU that is considered but, rather, its *interface*. The interface of a constrained space carries information on the emergent events generated by the child AU $\mathcal{U}$ and the associated candidate diagnoses relevant to all the AUs in the DDES rooted in $\mathcal{U}$.[22]

It is of utmost importance noticing that, in addition to the emergent events, the interface carries absolutely no information on the trajectories relevant to the sub-DDESs rooted in the child units, but

22. Roughly, any subtree in the DDES hierarchy is a (sub-)DDES on its own.

only the diagnoses relevant to these trajectories. Thus, the top-down influence of a parent unit on its child units translates to the removal of the set of diagnoses associated with inconsistent emergent events. This way, both types of influence on the behavior, namely from children to parent and from parent to children, can be managed in a single bottom-up traversing of the DDES, where the influence of the parent unit on the child units is accounted for by pruning the inconsistent emergent events along with the associated diagnosis sets (as underlined above, the diagnosis sets associated with the emergent events coming from a child unit $\mathcal{U}$ are relevant to all the AUs in the DDES rooted in $\mathcal{U}$). Eliminating the set of diagnoses associated with the inconsistent emergent events is equivalent to removing the inconsistent trajectories of the sub-DDESs rooted in the child units.

As a result, when generating the constrained space of the root node $\mathcal{U}_x$ of $\mathcal{X}$, the transitions are marked with the candidate diagnoses of the subtrees rooted in the child units of $\mathcal{U}_x$. By exploiting this extra piece of diagnosis information on child units, the eventual decoration of the constrained space of $\mathcal{U}_x$ will generate the (sound and complete) set of candidate diagnoses for $\mathcal{X}$, as proven in Theorem 1 (Appendix A). The formalization of the lazy diagnosis technique is presented in the next sections, where the (circular) notions of *interface constraint* (Section 6.1), *constrained unit space* (Section 6.2), *fragment* (Section 6.3), *interface* (Section 6.4), and *hidden set* (Section 6.5) are introduced.

6.1 Interface Constraint

The lazy problem-solving technique generates an extended constrained space for each AU of the DDES. This is carried out bottom-up, starting from leaf AUs up to the root of the hierarchy. For each AU $\mathcal{U}$, the corresponding constrained space is generated based on an *interface constraint*.

Definition 20 (Interface Constraint) *Let $\mathcal{U}$ be an AU, u_0 the initial state of $\mathcal{U}$, $\mathcal{V}$ a viewer for $\mathcal{U}$, $\mathcal{O}$ a temporal observation of $\mathcal{U}$, $\mathcal{R}$ the array of rulers for the AUs in the subtree rooted in $\mathcal{U}$, $(\mathcal{U}_1, \ldots, \mathcal{U}_k)$ the (possibly empty) array of child AUs of $\mathcal{U}$, and $\mathbf{I}$ the array of interfaces of the constrained AU spaces $\mathcal{U}^*_{1\xi_1}, \ldots, \mathcal{U}^*_{k\xi_k}$.*[23] *The 5-tuple*

$$\xi = (u_0, \mathcal{V}, \mathcal{O}, \mathcal{R}, \mathbf{I}) \tag{21}$$

is an interface constraint *for $\mathcal{U}$.*

The interface constraint ξ may be thought of as a sort of extended diagnosis problem for $\mathcal{U}$, with the additional field $\mathbf{I}$. This way, the unit space constrained by ξ (Section 6.2) accounts not only for the model and the temporal observation of $\mathcal{U}$, but for the mode in which emergent events are generated by the child AUs and consumed by $\mathcal{U}$. Notice that, if $\mathcal{U}$ is a leaf node, then the array $\mathbf{I}$ is null.

Example 24 Consider the AU $\mathcal{P}$ within the DDES $\mathcal{X}$ defined in Example 15 (Figure 11). An interface constraint for $\mathcal{P}$ is $\xi_{\mathcal{P}} = (u_0, \mathcal{V}_{\mathcal{P}}, \mathcal{O}_{\mathcal{P}}, (\mathcal{R}_{\mathcal{P}}), \mathbf{I}_{\mathcal{P}})$, where $u_0 = (idle, closed)$, $\mathcal{V}_{\mathcal{P}}$ is the viewer defined in Example 17, $\mathcal{O}_{\mathcal{P}}$ is the temporal observation defined in Example 21, $\mathcal{R}_{\mathcal{P}}$ is the ruler defined in Example 19, and $\mathbf{I}_{\mathcal{P}}$ is null ($\mathcal{P}$ is a leaf AU in the hierarchy of $\mathcal{X}$). Similarly, an interface constraint for the AU $\mathcal{P}'$ is $\xi_{\mathcal{P}'} = (u'_0, \mathcal{V}_{\mathcal{P}'}, \mathcal{O}_{\mathcal{P}'}, (\mathcal{R}_{\mathcal{P}'}), \mathbf{I}_{\mathcal{P}'})$, where $u'_0 = (idle, closed)$, $\mathcal{V}_{\mathcal{P}'}$ is the viewer defined in Example 17, $\mathcal{O}_{\mathcal{P}'}$ is the temporal observation defined in Example 21, $\mathcal{R}_{\mathcal{P}'}$ is the ruler defined in Example 19, and $\mathbf{I}_{\mathcal{P}'}$ is null ($\mathcal{P}'$ is a leaf AU in $\mathcal{X}$).

23. As detailed in Section 6.4, an interface is a DFA derived from a constrained unit space (Section 6.2).

6.2 Constrained Unit Space

As outlined in Section 6.1, within an interface constraint $\xi = (u_0, \mathcal{V}, \mathcal{O}, \mathcal{R}, \mathbf{I})$, the i-th element of the array $\mathbf{I}$, $i \in [1\,..\,k]$, is the interface of the constrained space of the i-th child AU of $\mathcal{U}$, denoted $Int(\mathcal{U}^*_{i\,\xi_i})$. As formalized in Section 6.4, $Int(\mathcal{U}^*_{i\,\xi_i})$ is a DFA whose alphabet is a set of pairs $(\mathbb{E}, \Delta)$, where $\mathbb{E}$ is a set of (output) emergent events in $\mathcal{U}_i$, while Δ is a set of diagnoses associated with the trajectories of the DDES rooted in $\mathcal{U}_i$. The interfaces of the child AU constrained spaces are exploited for the generation of the constrained space of the parent AU $\mathcal{U}$ by maintaining the configurations of the links exiting the output terminals of the child units, thereby enforcing the constraints imposed by the interaction of the child units with the parent unit. In particular, a transition of the constrained space of $\mathcal{U}$ is marked either by a transition of a component in $\mathcal{U}$ (along with the associated emergent output events in $\mathcal{U}$ and associated fault, if any), or by the pair marking a transition in one of the child-unit constrained-space interfaces. In other words, the constrained space of $\mathcal{U}$ carries *mixed* information in its transitions, specifically either:

- A triple $(t(c), \mathbb{E}, \{\delta\})$, where $t(c)$ is a component transition in $\mathcal{U}$, $\mathbb{E}$ is the set of emergent output events of $\mathcal{U}$ occurring at $t(c)$, and δ is either $\emptyset$, if $t(c)$ is normal, or $\{f\}$, if f is the fault associated with the faulty transition $t(c)$; or

- A pair $(\mathbb{E}, \Delta)$, where $\mathbb{E}$ is the set of emergent events occurring in one child AU, and Δ is the set of diagnoses of the trajectories of the DDES rooted in the child unit that are associated with the occurrence of $\mathbb{E}$.

Compared with a state of the unit space defined in Section 3.2, namely $(\mathbb{S}, \mathbb{E}, \mathbb{M})$, a state of the constrained unit space will involve three additional fields, namely:

- $\mathbb{H}$, the array of (possibly empty) emergent events stored in the links exiting the output terminals of the child AUs;

- $\mathbb{I}$, the array of states of the interfaces of the constrained child units;

- $\Im$ the index of the temporal observation.

Definition 21 (Constrained Unit Space) *Let $\mathcal{U}^* = (\Sigma_u, U, \tau_u, u_0, U_f)$ be the space of an AU $\mathcal{U}$ and $\xi = (u_0, \mathcal{V}, \mathcal{O}, \mathcal{R}, \mathbf{I})$ an interface constraint for $\mathcal{U}$, where $\mathcal{O} = [\ell_1, \ldots, \ell_n]$ and $\mathcal{R}_\mathcal{U} \in \mathcal{R}$ the ruler relevant to $\mathcal{U}$. The* spurious space *of $\mathcal{U}$* constrained *by ξ is a DFA*

$$\bar{\mathcal{U}}^*_\xi = (\Sigma, W, \tau, w_0, W_f). \tag{22}$$

Specifically, the alphabet Σ is $\big(\mathbf{T} \times 2^{\mathbf{E}} \times \{\{\delta\}\}\big) \cup \Sigma_1 \cup \ldots \cup \Sigma_k$, where $\mathbf{T}$ is the set of transitions of components in $\mathcal{U}$, $\mathbf{E}$ the set of emergent events defined in $\mathcal{U}$, δ either $\emptyset$ or the singleton $\{f\}$, with f being a fault of the ruler $\mathcal{R}_\mathcal{U}$, and Σ_i, $i \in [1\,..\,k]$, the alphabet of $Int(\mathcal{U}^*_{i\,\xi_i})$. W is the set of states $(\mathbb{S}, \mathbb{E}, \mathbb{M}, \mathbb{H}, \mathbb{I}, \Im)$, where $(\mathbb{S}, \mathbb{E}, \mathbb{M})$ is a state in U, $\mathbb{H} = (e_1, \ldots, e_h)$ the array of the (possibly empty) emergent events ready at input terminals of $\mathcal{U}$, $\mathbb{I} = (I_1, \ldots, I_k)$ the array of states of the interfaces in $\mathbf{I}$, and $\Im$ an index of $\mathcal{O}$. The initial state is $w_0 = (\mathbb{S}_0, \mathbb{E}_0, \mathbb{M}_0, \mathbb{H}_0, \mathbb{I}_0, 0)$, where $(\mathbb{S}_0, \mathbb{E}_0, \mathbb{M}_0)$ is the initial state u_0 of $\mathcal{U}^*$, $\mathbb{H}_0 = (\varepsilon, \ldots, \varepsilon)$, and $\mathbb{I}_0 = (I_{1\,0}, \ldots, I_{k\,0})$ the array of initial states of the interfaces in $\mathbf{I}$. $W_f \subseteq W$ is the set of final states $(\mathbb{S}, \mathbb{E}, \mathbb{M}, \mathbb{H}, \mathbb{I}, n)$, where

$(\mathbb{S}, \mathbb{E}, \mathbb{M}) \in U_{\mathrm{f}}$, $\mathbb{H} = (\varepsilon, \ldots, \varepsilon)$, $\mathbb{I} = (I_1, \ldots, I_k)$ *such that* $\forall i \in [1\,..\,k]$, I_i *is final in the* i*-th interface in* $\mathbf{I}$. *The transition function is* $\tau : W \times \Sigma \mapsto W$, *where either:*

$$\left\langle (\mathbb{S}, \mathbb{E}, \mathbb{M}, \mathbb{H}, \mathbb{I}, \Im), \left(\mathbb{E}_j, \Delta_j\right), (\mathbb{S}, \mathbb{E}, \mathbb{M}, \mathbb{H}', \mathbb{I}', \Im)\right\rangle \in \tau \tag{23}$$

if and only if:

- $\left\langle I_j, (\mathbb{E}_j, \Delta_j), I'_j \right\rangle$ *is a transition in the* j*-th interface in* $\mathbf{I}$, $j \in [1\,..\,k]$;
- $\forall e(y) \in \mathbb{E}_j$, $\mathbb{H}[link(y)] = \varepsilon$;
- $\forall i \in [1\,..\,h], \mathbb{H}'[i] = \begin{cases} e & \text{if } e(y) \in \mathbb{E}_j \text{ and } link(y) = l_i \\ \mathbb{H}[i] & \text{otherwise;} \end{cases}$
- $\forall i \in [1\,..\,k], \mathbb{I}'[i] = \begin{cases} I'_j & \text{if } i = j \\ \mathbb{I}[i] & \text{otherwise;} \end{cases}$

or

$$\left\langle (\mathbb{S}, \mathbb{E}, \mathbb{M}, \mathbb{H}, \mathbb{I}, \Im), \left(t(c_j), \bar{\mathbb{E}}, \bar{\Delta}\right), (\mathbb{S}', \mathbb{E}', \mathbb{M}', \mathbb{H}', \mathbb{I}, \Im')\right\rangle \in \tau \tag{24}$$

if and only if:

- $\left\langle (\mathbb{S}, \mathbb{E}, \mathbb{M}), t(c_j), (\mathbb{S}', \mathbb{E}', \mathbb{M}')\right\rangle$ *is a transition in* $\mathcal{U}^*$;
- $\bar{\mathbb{E}}$ *is the set of emergent events generated by the above transition in* $\mathcal{U}^*$, *on condition that* $\bar{\mathbb{E}}$ *includes at most one event for the same output terminal, that is,* $\{e(y), e'(y)\} \not\subseteq \bar{\mathbb{E}}$;
- $t(c_j) = \langle s, (e(x), \{e_1(y_1), \ldots, e_v(y_v)\}), s'\rangle$;
- $e(x) = \varepsilon$, *or* x *is not linked with an input terminal of* $\mathcal{U}$[24], *or* x *is linked with an input terminal of* $\mathcal{U}$ *and* $\mathbb{H}[link(x)] = e$;
- *If* x *is not linked with an input terminal of* $\mathcal{U}$ *then* $\mathbb{H}' = \mathbb{H}$, *else*

$$\forall i \in [1\,..\,k], \mathbb{H}'[i] = \begin{cases} \varepsilon & \text{if } link(x) = l_i \\ \mathbb{H}[i] & \text{otherwise;} \end{cases}$$

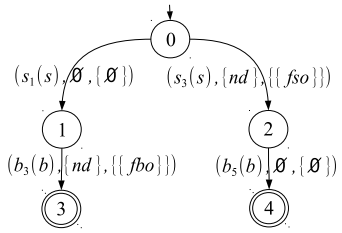
- $\Im' = \begin{cases} \Im + 1 & \text{if } (t(c_j), \ell) \in \mathcal{V} \text{ and } \ell = \ell_{\Im+1} \\ \Im & \text{if } (t(c_j), \ell) \notin \mathcal{V}; \end{cases}$
- $\bar{\Delta} = \begin{cases} \{\{f\}\} & \text{if } (t(c_j), f) \in \mathcal{R}_{\mathcal{U}} \\ \{\emptyset\} & \text{otherwise.} \end{cases}$

The space *of* $\mathcal{U}$ constrained *by* ξ, *namely* $\mathcal{U}^*_\xi$, *is the DFA obtained from the (spurious)* $\bar{\mathcal{U}}^*_\xi$ *by pruning all states and transitions that are not connected with any final state. A* path p *within* $\mathcal{U}^*_\xi = (\Sigma, W, \tau, w_0, W_{\mathrm{f}})$ *is a finite sequence of contiguous transitions in* τ,

$$p = \left[\langle w_0, \sigma_1, w_1\rangle, \langle w_1, \sigma_2, w_2\rangle, \ldots, \langle w_{q-1}, \sigma_q, w_q\rangle\right] \tag{25}$$

where $\sigma_i \in \Sigma$, $i \in [1\,..\,q]$, *and* $w_q \in W_{\mathrm{f}}$. *The (finite) sequence* $[\,\sigma_1, \ldots, \sigma_q\,]$ *of symbols marking the transitions of* p *is denoted* $\Sigma(p)$. *The (possibly infinite) set of paths within* $\mathcal{U}^*_\xi$ *is denoted* $\|\mathcal{U}^*_\xi\|$.

24. In this case, it means that x is connected with a link embedded in $\mathcal{U}$.



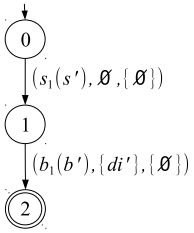
State	$\mathbb{S}$	$\mathbb{E}$	$\mathbb{M}$	$\Im$
0	(*idle*, *closed*)	(ε)	(0, 0, 0, 0)	0
1	(*awaken*, *closed*)	(*op*)	(0, 0, 1, 0)	1
2	(*awaken*, *closed*)	(*cl*)	(0, 0, **2**, 0)	1
3	(*awaken*, *closed*)	(ε)	(0, 0, **2**, 0)	1
4	(*awaken*, *closed*)	(ε)	(0, 0, 0, 0)	1

Figure 16: Constrained space $\mathcal{P}^*_{\xi_\mathcal{P}}$ of the AU $\mathcal{P}$ (left); the state contents are detailed in the table on the right side (bold states in $\mathbb{M}$ indicate the detection of the output emergent event).

Example 25 Consider the interface constraint $\xi_\mathcal{P}$ defined in Example 24. The space of $\mathcal{P}$ constrained by $\xi_\mathcal{P}$, namely $\mathcal{P}^*_{\xi_\mathcal{P}}$ is displayed in Figure 16 (left), along with state details (right). Since $\mathcal{P}$ is a leaf AU in $\mathcal{X}$, the fields $\mathbb{I}$ and $\mathbb{H}$ are empty (that's why they are not considered). Likewise, displayed in Figure 17 is the space of the AU $\mathcal{P}'$ constrained by $\xi_{\mathcal{P}'}$, which is defined in Example 24.

6.3 Fragment

To define the interface of a constrained AU space (Section 6.4), we first need to introduce the notion of a fragment of a constrained AU space $\mathcal{U}^*_\xi$. This is because a state of the interface is a set of fragments. A fragment is associated with a state of $\mathcal{U}^*_\xi$, starting from the initial state w_0. Roughly, given a state $\bar{w}$ of $\mathcal{U}^*_\xi$, the corresponding fragment $\mathcal{U}^*_\xi(\bar{w})$ is the subgraph of $\mathcal{U}^*_\xi$ that is reachable from $\bar{w}$ by transitions either marked with a pair $(\mathbb{E}, \Delta)$ or a triple $(t(c), \emptyset, \{\delta\})$, that is, where no emergent event is generated upon $t(c)$. Besides, each state w incorporated in the fragment is extended with a diagnosis information, namely the set of diagnoses relevant to the paths connecting $\bar{w}$ with w. These diagnoses are generated by joining the diagnoses marking the transitions in each path (by means of the join operator "$\otimes$" defined in Section 2.3). This way, each state of the fragment carries (partial) diagnosis information of the entire (sub-)DDES rooted in $\mathcal{U}$.



State	$\mathbb{S}$	$\mathbb{E}$	$\mathbb{M}$	$\Im$
0	(*idle*, *closed*)	(ε)	(0, 0, 0, 0)	0
1	(*awaken*, *closed*)	(*op*)	(0, 0, 1, 0)	1
2	(*awaken*, *open*)	(ε)	(**1**, 0, 0, 0)	2

Figure 17: Constrained space $\mathcal{P}'^*_{\xi_{\mathcal{P}'}}$ of the AU $\mathcal{P}'$ (left); the state contents are detailed in the table on the right side (bold states in $\mathbb{M}$ indicate the detection of the output emergent event).

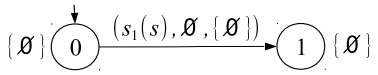


Figure 18: Fragment relevant to the initial state of the constrained space of AU $\mathcal{P}$ in Figure 16.

Definition 22 (Fragment) *Let $\mathcal{U}^*_\xi = (\Sigma, W, \tau', w_0, W_\mathrm{f})$ be a constrained unit space, and $\bar{w} \in W$. The* fragment *of $\mathcal{U}^*_\xi$* rooted *in state $\bar{w}$ is a DFA*

$$\mathcal{U}^*_\xi(\bar{w}) = (\Sigma, \Phi, \tau, \varphi_0). \tag{26}$$

The set of states is $\Phi \subseteq W \times 2^{\boldsymbol{\delta}}$, where $\boldsymbol{\delta}$ is the domain of the possible diagnoses including the faults involved in the array of rulers in the constraint ξ.

The transition function $\tau : \Phi \times \Sigma \mapsto \Phi$ is defined as follows. Each state $(w, \Delta) \in \Phi$ is such that Δ is the minimal set defined by the following two rules:

1. *If $w = \bar{w}$, then $\emptyset \in \Delta$;*
2. *If $\big\langle (w', \Delta'), \big(\mathbb{E}, \bar{\Delta}\big), (w, \Delta)\big\rangle \in \tau$ or $\big\langle (w', \Delta'), \big(t(c), \emptyset, \bar{\Delta}\big), (w, \Delta)\big\rangle \in \tau$, then $\big(\Delta' \otimes \bar{\Delta}\big) \subseteq \Delta$.*

$\varphi_0 = (\bar{w}, \bar{\Delta})$ is the initial state. The transition function is defined as follows:

$$\big\langle (w, \Delta), \big(\mathbb{E}, \bar{\Delta}\big), (w', \Delta')\big\rangle \in \tau \;\; \textit{if and only if} \;\; \big\langle w, \big(\mathbb{E}, \bar{\Delta}\big), w'\big\rangle \in \tau'$$

or

$$\big\langle (w, \Delta), \big(t(c), \emptyset, \bar{\Delta}\big), (w', \Delta')\big\rangle \in \tau \;\; \textit{if and only if} \;\; \big\langle w, \big(t(c), \emptyset, \bar{\Delta}\big), w'\big\rangle \in \tau'.$$

Example 26 With reference to the constrained space $\mathcal{P}^*_{\xi_\mathcal{P}}$ displayed in Figure 16 (Example 25), the fragment relevant to the initial state, namely $\mathcal{P}^*_{\xi_\mathcal{P}}(0)$, is shown in Figure 18.

6.4 Interface

As outlined in Section 6.2, the constrained space of an AU must comply with the interfaces of the child AU constrained spaces. The interface of a constrained space of an AU $\mathcal{U}$ is a DFA where states are sets of fragments of the constrained space, while transitions are marked with pairs $(\mathbb{E}, \Delta)$, where $\mathbb{E}$ is a set of output emergent events, and Δ is the associated set of diagnoses for the DDES rooted in $\mathcal{U}$. Thus, each diagnosis in Δ may involve faults relevant to any unit in the subtree rooted in $\mathcal{U}$. More precisely, each diagnosis in Δ is relevant to a subtrajectory of the DDES $\bar{\mathcal{U}}$ rooted in $\mathcal{U}$. As such, Δ is the diagnosis surrogate of the (possibly infinite) set of semi-trajectories of the DDES $\bar{\mathcal{U}}$ within the fragment (including the component transition in $\mathcal{U}$ exiting the fragment).

Definition 23 (Interface) *Let $\mathcal{U}^*_\xi = (\Sigma', W, \tau', \bar{w}_0, W_\mathrm{f})$ be a constrained unit space, where $\xi = (u_0, \mathcal{V}, \mathcal{O}, \mathcal{R}, \mathbf{I})$. The* interface *of $\mathcal{U}^*_\xi$, namely $Int(\mathcal{U}^*_\xi)$, is a DFA:*

$$Int(\mathcal{U}^*_\xi) = (\Sigma, I, \tau, i_0, I_\mathrm{f}). \tag{27}$$

Specifically, the alphabet is $\Sigma = 2^{\mathbf{E}} \times 2^{\boldsymbol{\delta}}$, where $\mathbf{E}$ is the set of emergent events in $\mathcal{U}$, and $\boldsymbol{\delta}$ is the domain of possible diagnoses that include faults relevant to $\mathcal{R}$. The set of states is $I =$

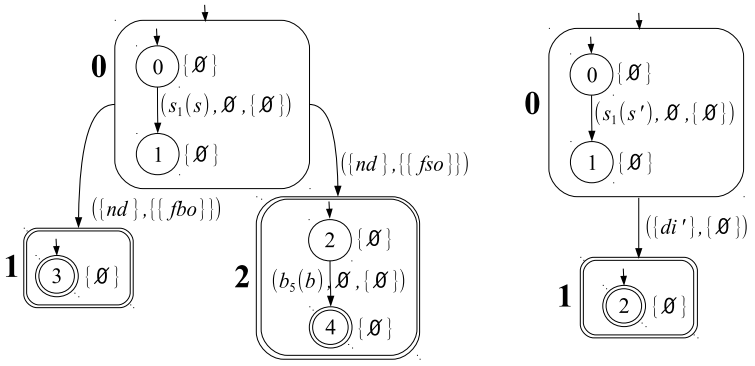


Figure 19: Interface of the constrained space of the AU $\mathcal{P}$ in Figure 16, namely $Int(\mathcal{P}^*_{\xi_\mathcal{P}})$ (left) and interface of the constrained space of the AU $\mathcal{P}'$ in Figure 17, namely $Int(\mathcal{P}'^*_{\xi_{\mathcal{P}'}})$ (right).

$2^{\boldsymbol{\Phi}}$, *where* $\boldsymbol{\Phi}$ *is the set of all fragments of* $\mathcal{U}^*_\xi$. *The initial state is the singleton including the fragment relevant to* w_0, *namely* $i_0 = \{\, \mathcal{U}^*_\xi(w_0) \,\}$. *The set of the final states is* $I_\mathrm{f} \subseteq I$, *where* $\forall i_\mathrm{f} \in I_\mathrm{f}\,(\varphi \in i_\mathrm{f}, (w, \Delta) \in \Phi(\varphi), w \in W_\mathrm{f})$. *The transition function is* $\tau : I \times \Sigma \mapsto I$, *where:* $\langle i, (\mathbb{E}, \Delta), i' \rangle \in \tau$ *iff* $\varphi \in i$, $(w, \bar{\Delta}) \in \Phi(\varphi)$, $\langle w, (t(c), \mathbb{E}, \Delta^*), w' \rangle \in \tau'$, $\mathbb{E} \neq \emptyset$, $\Delta = \bar{\Delta} \otimes \Delta^*$, *and* $\mathcal{U}^*_\xi(w') \in i'$. *A* path p *within an interface* $Int(\mathcal{U}^*_\xi)$ *is a finite contiguous sequence of transitions in* τ, $p = \big[\langle i_0, (\mathbb{E}_1, \Delta_1), i_1 \rangle, \langle i_1, (\mathbb{E}_2, \Delta_2), i_2 \rangle, \ldots, \big\langle i_{q-1}, (\mathbb{E}_q, \Delta_q), i_q \big\rangle\big]$, *where* $i_q \in I_\mathrm{f}$. *The (finite) sequence* $[\,(\mathbb{E}_1, \Delta_1), \ldots, (\mathbb{E}_q, \Delta_q)\,]$ *of symbols marking the transitions of* p *is denoted* $\Sigma(p)$. *The (possibly infinite) set of paths within* $Int(\mathcal{U}^*_\xi)$ *is denoted* $\|Int(\mathcal{U}^*_\xi)\|$.

Example 27 With reference to Example 25, consider the constrained spaces $\mathcal{P}^*_{\xi_\mathcal{P}}$ and $\mathcal{P}'^*_{\xi_{\mathcal{P}'}}$ displayed in Figure 16 and Figure 17, respectively. The corresponding interfaces $Int(\mathcal{P}^*_{\xi_\mathcal{P}})$ and $Int(\mathcal{P}'^*_{\xi_{\mathcal{P}'}})$ are shown in Figure 19. Based on these interfaces, we can determine the constrained space of the AU $\mathcal{M}$, the monitoring apparatus (Figure 11), along with the corresponding interface. An interface constraint for $\mathcal{M}$ is $\xi_\mathcal{M} = (m_0, \mathcal{V}_\mathcal{M}, \mathcal{O}_\mathcal{M}, (\mathcal{R}_\mathcal{P}, \mathcal{R}_{\mathcal{P}'}, \mathcal{R}_\mathcal{M}), \mathbf{I}_\mathcal{M})$, where $m_0 = (watch, closed, watch, closed)$, $\mathcal{V}_\mathcal{M}$ is the viewer defined in Example 17, $\mathcal{O}_\mathcal{M}$ is the temporal observation defined in Example 21, $\mathcal{R}_\mathcal{P}$, $\mathcal{R}_{\mathcal{P}'}$, and $\mathcal{R}_\mathcal{M}$ are the rulers defined in Example 19, and $\mathbf{I}_\mathcal{M} = (Int(\mathcal{P}^*_{\xi_\mathcal{P}}), Int(\mathcal{P}'^*_{\xi_{\mathcal{P}'}}))$. The space of $\mathcal{M}$ constrained by $\xi_\mathcal{M}$, namely $\mathcal{M}^*_{\xi_\mathcal{M}}$ is shown in Figure 20, with details of the states being listed in Table 7. The interface, namely $Int(\mathcal{M}^*_{\xi_\mathcal{M}})$ is outlined in Figure 21.

6.5 Hidden Set

As pointed out in Section 6.2, the transitions within the constrained space of an AU $\mathcal{U}$ are marked with sets of diagnoses relevant either to $\mathcal{U}$ or to the DDESs rooted in the child units of $\mathcal{U}$. Thus, joining together the set of diagnoses relevant to a path in the constrained space is bound to generate a set of candidate diagnoses for the DDES rooted in $\mathcal{U}$. In particular, if $\mathcal{U}$ is the root node of the hierarchy of the DDES $\mathcal{X}$, then these should be the candidate diagnoses of $\mathcal{X}$, that is, a subset of the solution to the diagnosis problem for $\mathcal{X}$.[25] However, if we consider how the constrained space

25. To obtain all the candidate diagnoses, we need to consider every path in the constrained space.

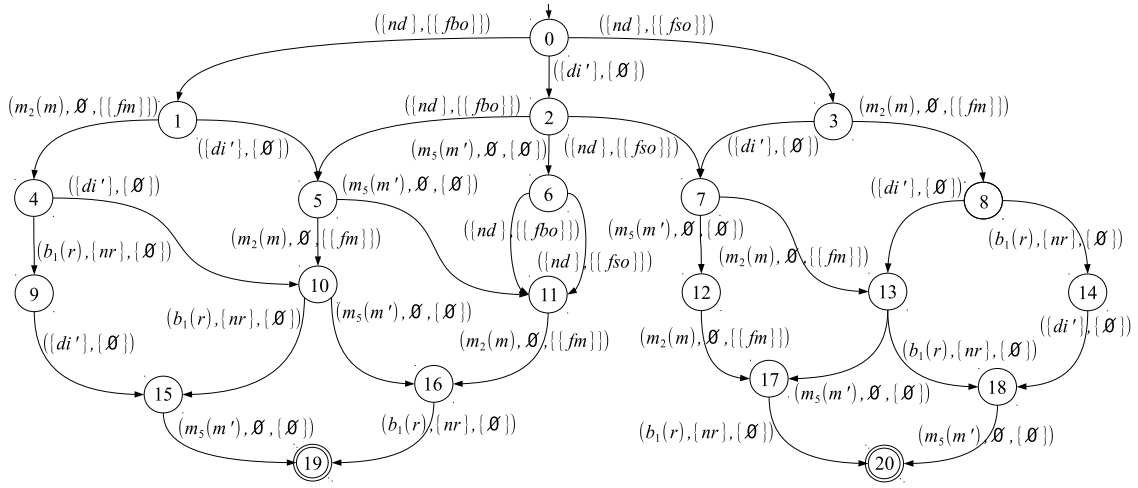


Figure 20: Constrained space $\mathcal{M}^*_{\xi_\mathcal{M}}$ of the AU $\mathcal{M}$; the content of the states is detailed in Table 7.

is generated, we will find out that some diagnosis information is missing. Specifically, there is a set of diagnoses that is hidden in the constrained space, which we call the *hidden set*. To simplify the picture, consider a DDES $\mathcal{X}$, with $\mathcal{U}$ being the root node, and $\mathcal{U}_1$ and $\mathcal{U}_2$ the child units of $\mathcal{U}$. No other AU is included in $\mathcal{X}$ (hence, $\mathcal{U}_1$ and $\mathcal{U}_2$ are leaf nodes). Consider a path p within the constrained space of $\mathcal{U}$, from the initial state w_0 to a final state w_q. Let $[\Delta_0, \Delta_1, \ldots, \Delta_q]$ be the sequence of diagnosis sets involved in p. The combination of such diagnosis sets, namely $\Delta^* = \Delta_0 \otimes \Delta_1 \otimes \cdots \otimes \Delta_q$, generates a set of diagnoses for $\mathcal{X}$. However, in order for a set δ of faults to be a candidate diagnosis of $\mathcal{X}$, δ shall be relevant to a *complete* trajectory of $\mathcal{X}$. To this end, we need to consider the tail of the trajectories of $\mathcal{X}$ that are encapsulated within the fragments included in the final states I_1 and I_2 of the interfaces of the constrained spaces of $\mathcal{U}_1$ and $\mathcal{U}_2$, which are part of the final state w_q of the path p. Considering a set of faults $\delta^* \in \Delta^*$, in order for δ^* to be a candidate diagnosis of $\mathcal{X}$, we need to extend δ^* with a diagnosis in $\bar{\Delta}_1$ such that $(\bar{w}_1, \bar{\Delta}_1)$ is a state of a fragment in I_1 where $\bar{w}_1$ is final, and a diagnosis in $\bar{\Delta}_2$ such that $(\bar{w}_2, \bar{\Delta}_2)$ is a state of a fragment in I_2 where $\bar{w}_2$ is final. This way, we enrich δ^* with the set of faults relevant to the transitions of the components in $\mathcal{U}_1$ and $\mathcal{U}_2$ that complete a trajectory in $\mathcal{X}$. To obtain the complete set of candidate diagnoses, this operation should be performed for each $\delta^* \in \Delta^*$ and for each state (w, Δ) within the fragments relevant to I_1 and I_2, respectively, such that w is final in the corresponding constrained space.

Definition 24 (Hidden Set) *Let $\mathcal{X}$ be a DDES, $\mathcal{U}$ an AU in $\mathcal{X}$, $\mathcal{U}^*_\xi = (\Sigma, W, \tau, w_0, W_\mathrm{f})$ the unit space constrained by $\xi = (u_0, \mathcal{V}, \mathcal{O}, \mathcal{R}, \mathbf{I})$, and $w = (\mathbb{S}, \mathbb{E}, \mathbb{M}, \mathbb{H}, \mathbb{I}, \mathbb{S})$ a state in W. The* hidden set *of w, namely $\nabla(w)$, is the set of diagnoses recursively defined as follows:*

$$\nabla(w) = \begin{cases} \{\emptyset\} & \text{if } \mathcal{U} \text{ is a leaf node in } \mathcal{X} \\ \bigotimes_{i \in \mathbb{I}} \nabla(i) & \text{otherwise} \end{cases} \tag{28}$$

State	$\mathbb{S}$	$\mathbb{E}$	$\mathbb{M}$	$\mathbb{H}$	$\mathbb{I}$	$\Im$
0	$(watch, closed, watch, closed)$	$(\varepsilon, \varepsilon, \varepsilon, \varepsilon)$	$(0,0,0,0,0,0)$	$(\varepsilon, \varepsilon)$	$(0,0)$	0
1	$(watch, closed, watch, closed)$	$(\varepsilon, \varepsilon, \varepsilon, \varepsilon)$	$(0,0,0,0,0,0)$	(nd, ε)	$(1,0)$	0
2	$(watch, closed, watch, closed)$	$(\varepsilon, \varepsilon, \varepsilon, \varepsilon)$	$(0,0,0,0,0,0)$	(ε, di')	$(0,1)$	0
3	$(watch, closed, watch, closed)$	$(\varepsilon, \varepsilon, \varepsilon, \varepsilon)$	$(0,0,0,0,0,0)$	(nd, ε)	$(2,0)$	0
4	$(trigger, closed, watch, closed)$	$(op, \varepsilon, \varepsilon, \varepsilon)$	$(0,1,0,0,0,0)$	$(\varepsilon, \varepsilon)$	$(1,0)$	1
5	$(watch, closed, watch, closed)$	$(\varepsilon, \varepsilon, \varepsilon, \varepsilon)$	$(0,0,0,0,0,0)$	(nd, di')	$(1,1)$	0
6	$(watch, closed, watch, closed)$	$(\varepsilon, \varepsilon, \varepsilon, \varepsilon)$	$(0,0,0,0,0,0)$	$(\varepsilon, \varepsilon)$	$(0,1)$	0
7	$(watch, closed, watch, closed)$	$(\varepsilon, \varepsilon, \varepsilon, \varepsilon)$	$(0,0,0,0,0,0)$	(nd, di')	$(2,1)$	0
8	$(trigger, closed, watch, closed)$	$(op, \varepsilon, \varepsilon, \varepsilon)$	$(0,1,0,0,0,0)$	$(\varepsilon, \varepsilon)$	$(2,0)$	1
9	$(trigger, open, watch, closed)$	$(\varepsilon, \varepsilon, \varepsilon, \varepsilon)$	$(1,1,0,0,0,0)$	$(\varepsilon, \varepsilon)$	$(1,0)$	2
10	$(trigger, closed, watch, closed)$	$(op, \varepsilon, \varepsilon, \varepsilon)$	$(0,1,0,0,0,0)$	(ε, di')	$(1,1)$	1
11	$(watch, closed, watch, closed)$	$(\varepsilon, \varepsilon, \varepsilon, \varepsilon)$	$(0,0,0,0,0,0)$	(nd, ε)	$(1,1)$	0
12	$(watch, closed, watch, closed)$	$(\varepsilon, \varepsilon, \varepsilon, \varepsilon)$	$(0,0,0,0,0,0)$	(nd, ε)	$(2,1)$	0
13	$(trigger, closed, watch, closed)$	$(op, \varepsilon, \varepsilon, \varepsilon)$	$(0,1,0,0,0,0)$	(ε, di')	$(2,1)$	1
14	$(trigger, open, watch, closed)$	$(\varepsilon, \varepsilon, \varepsilon, \varepsilon)$	$(1,1,0,0,0,0)$	$(\varepsilon, \varepsilon)$	$(2,0)$	2
15	$(trigger, open, watch, closed)$	$(\varepsilon, \varepsilon, \varepsilon, \varepsilon)$	$(1,1,0,0,0,0)$	(ε, di')	$(1,1)$	2
16	$(trigger, closed, watch, closed)$	$(op, \varepsilon, \varepsilon, \varepsilon)$	$(0,1,0,0,0,0)$	$(\varepsilon, \varepsilon)$	$(1,1)$	1
17	$(trigger, closed, watch, closed)$	$(op, \varepsilon, \varepsilon, \varepsilon)$	$(0,1,0,0,0,0)$	$(\varepsilon, \varepsilon)$	$(2,1)$	1
18	$(trigger, open, watch, closed)$	$(\varepsilon, \varepsilon, \varepsilon, \varepsilon)$	$(1,1,0,0,0,0)$	(ε, di')	$(2,1)$	2
19	$(trigger, open, watch, closed)$	$(\varepsilon, \varepsilon, \varepsilon, \varepsilon)$	$(1,1,0,0,0,0)$	$(\varepsilon, \varepsilon)$	$(1,1)$	2
20	$(trigger, open, watch, closed)$	$(\varepsilon, \varepsilon, \varepsilon, \varepsilon)$	$(1,1,0,0,0,0)$	$(\varepsilon, \varepsilon)$	$(2,1)$	2

Table 7: Transition details for the states in the constrained space of $\mathcal{M}$ displayed in Figure 20.

where $\nabla(i)$ *is the* hidden set *of an interface state* i*, defined as follows:*[26]

$$\nabla(i) = \bigcup_{\varphi \in i, (w_{\mathrm{f}}, \Delta_{\mathrm{f}}) \in \Phi(\varphi), w_{\mathrm{f}} \in W_{\mathrm{f}}} \Big(\Delta_{\mathrm{f}} \otimes \nabla(w_{\mathrm{f}}) \Big) \,. \tag{29}$$

Example 28 In order to see a meaningful example, we assume a hypothetical scenario in which the constrained space of the AU $\mathcal{L}$ includes a final state w such that $\mathbb{I} = (i)$, where i is a state of the interface of the constrained space of $\mathcal{M}$. We assume that i includes just one fragment involving two final states, namely w_1 and w_2, with associated diagnosis sets $\Delta(w_1) = \{\{fm, fro\}\}$ and $\Delta(w_2) = \{\emptyset\}$. On their turn, w_1 involves the field $\mathbb{I} = (i_1, i_1')$, while w_2 involves the field $\mathbb{I} = (i_2, i_2')$. We assume that the states i_1, i_1', i_2, and i_2' include one fragment each, involving one final state, namely $\bar{w}_1$, $\bar{w}_2$, $\bar{w}_1'$, and $\bar{w}_2'$, respectively, where $\Delta(\bar{w}_1) = \{\{fso\}, \{fbo\}\}$, $\Delta(\bar{w}_2) = \{\emptyset\}$, $\Delta(\bar{w}_1') = \{\{fsc', fbo'\}\}$, and $\Delta(\bar{w}_2') = \{\{fsc'\}, \{fso', fbo'\}\}$. According to eqn. (28) and eqn. (29), the hidden

26. Note the circular dependency between $\Delta(w)$ and $\nabla(i)$.

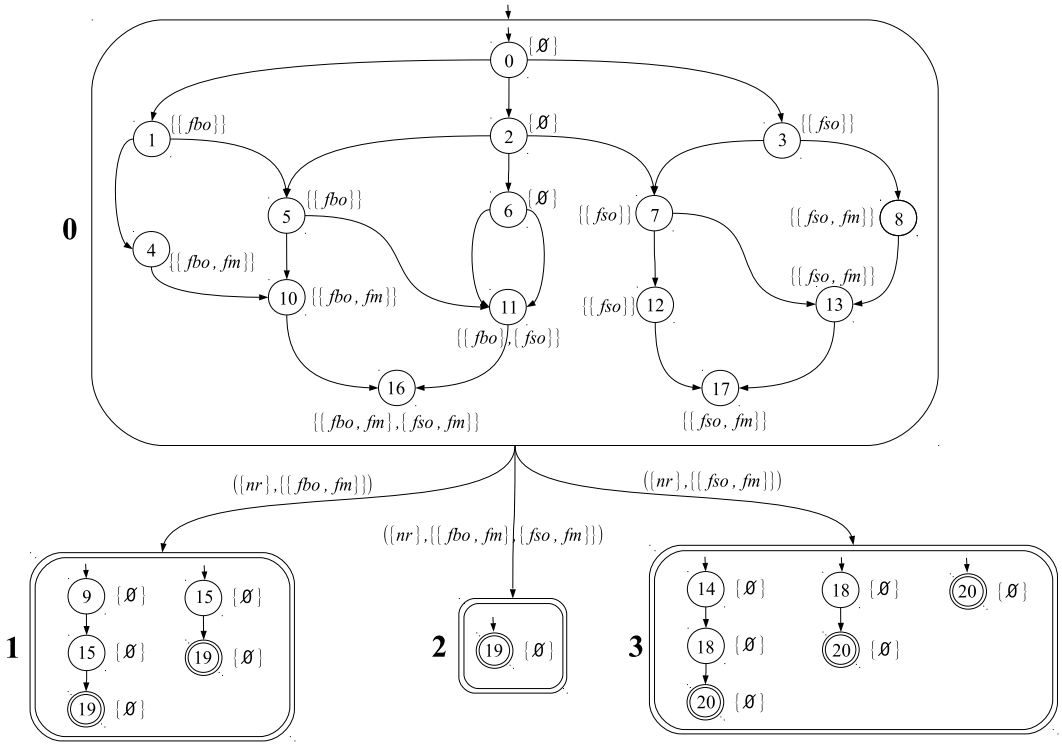


Figure 21: Interface of $\mathcal{M}^*_{\xi,\mathcal{M}}$ (Figure 20); in fragments, only the diagnosis sets associated with the states are outlined, with the labels associated with the transitions being omitted.

set of w is

$$\begin{aligned}
\nabla(w) = \nabla(i) &= (\Delta(w_1) \otimes \nabla(w_1)) \cup (\Delta(w_2) \otimes \nabla(w_2)) \\
&= \big(\Delta(w_1) \otimes \big(\nabla(i_1) \otimes \nabla(i'_1)\big)\big) \cup \big(\Delta(w_2) \otimes \big(\nabla(i_2) \otimes \nabla(i'_2)\big)\big) \\
&= \Delta(w_1) \otimes \big((\Delta(\bar{w}_1) \otimes \{\emptyset\}) \otimes \big(\Delta(\bar{w}'_1) \otimes \{\emptyset\}\big)\big) \cup \\
&\quad \Delta(w_2) \otimes \big((\Delta(\bar{w}_2) \otimes \{\emptyset\}) \otimes \big(\Delta(\bar{w}'_2) \otimes \{\emptyset\}\big)\big) \\
&= \{\{fm, fro\}\} \otimes (\{\{fso\}, \{fbo\}\} \otimes \{\emptyset\}) \otimes (\{\{fsc', fbo'\}\} \otimes \{\emptyset\}) \cup \\
&\quad \{\emptyset\} \otimes (\{\emptyset\} \otimes \{\emptyset\}) \otimes (\{\{fsc'\}, \{fso', fbo'\}\} \otimes \{\emptyset\}) \\
&= \{\{fm, fro\}\} \otimes \{\{fso\}, \{fbo\}\} \otimes \{\{fsc', fbo'\}\} \cup \{\{fsc'\}, \{fso', fbo'\}\} \\
&= \{\{fm, fro, fso, fsc', fbo'\}, \{fm, fro, fbo, fsc', fbo'\}, \{fsc'\}, \{fso', fbo'\}\}.
\end{aligned}$$

Each diagnosis in the hidden set of w includes the faults relevant to the inferior AUs in the hierarchy, namely $\mathcal{M}$, $\mathcal{P}$, and $\mathcal{P}'$.

6.6 Final Decoration

Assume that $\mathcal{U}$ is the root node of a DDES $\mathcal{X}$. Once the constrained space of $\mathcal{U}$, namely $\mathcal{U}^*_\xi$, has been generated, we are ready to derive the solution to the diagnosis problem for $\mathcal{X}$ by means of a decoration technique similar to that outlined in Section 2.6 for (plain) ASs. Intuitively, we have to

consider each path $p \in \|\mathcal{U}_\xi^*\|$, and combine the diagnosis sets marking p along with the hidden set of the final state of p. Practically, we perform a decoration of $\mathcal{U}_\xi^*$, where the final states are marked with the candidate diagnoses of $\mathcal{X}$.

Definition 25 (Decorated Constrained Space) *Let $\mathcal{X}$ be a DDES, $\mathcal{U}$ the root node of $\mathcal{X}$, and $\mathcal{U}_\xi^* = (\Sigma, W, \tau, w_0, W_\mathrm{f})$ the unit space of $\mathcal{U}$ constrained by $\xi = (u_0, \mathcal{V}, \mathcal{O}, \mathcal{R}, \mathbf{I})$. The* decorated constrained space *of $\mathcal{U}$ is the DFA $\mathcal{U}_\xi^\mathrm{d}$ obtained from $\mathcal{U}_\xi^*$ by marking each state w with the minimal set $\Delta(w)$ of diagnoses based on the following two rules:*

1. $\emptyset \in \Delta(w_0)$*;*
2. *If either $\langle w, (\mathbb{E}, \bar{\Delta}), w' \rangle \in \tau$ or $\langle w, (t(c), \mathbb{E}, \bar{\Delta}), w' \rangle \in \tau$, then $\Delta(w') \subseteq \left(\Delta(w) \otimes \bar{\Delta} \right)$.*

The diagnosis set *of $\mathcal{U}_\xi^\mathrm{d}$, $\Delta(\mathcal{U}_\xi^\mathrm{d})$, is the set of diagnoses*

$$\Delta\left(\mathcal{U}_\xi^\mathrm{d}\right) = \bigcup_{w_\mathrm{f} \in W_\mathrm{f}} \left(\Delta(w_\mathrm{f}) \otimes \nabla(w_\mathrm{f}) \right) . \tag{30}$$

Remarkably, $\Delta(\mathcal{U}_\xi^\mathrm{d})$ is the solution to the diagnosis problem $\wp(\mathcal{X})$ defined in eqn. (19) of Section 4.4, as claimed in Theorem 1.

Theorem 1 *Let $\wp(\mathcal{X}) = (x_0, \mathcal{V}, \mathcal{O}, \mathcal{R})$ be a diagnosis problem for the DDES $\mathcal{X}$, $\mathcal{U}$ the root node in $\mathcal{X}$, and $\mathcal{U}_\xi^\mathrm{d}$ the decorated constrained space of $\mathcal{U}$. We have*

$$\Delta(\wp(\mathcal{X})) = \Delta\left(\mathcal{U}_\xi^\mathrm{d}\right). \tag{31}$$

A proof of Theorem 1 is given in Appendix A. The proof makes use of four additional definitions, namely Definition 26 (diagnosis set resulting from the join of a sequence of sets of diagnoses), Definition 27 (path of a constrained unit space $\mathcal{U}_\xi^*$ and relevant diagnosis set, along with the diagnosis set of $\mathcal{U}_\xi^*$), Definition 28 (path of an interface $Int(\mathcal{U}_\xi^*)$ and relevant diagnosis set, along with the diagnosis set of $Int(\mathcal{U}_\xi^*)$), and Definition 29 (projection of a diagnosis problem on a subtree of the DDES). Besides, two ancillary propositions are introduced, namely Proposition 4 and Proposition 5. Proposition 4 claims that the projection $\bar{T}$ of a trajectory T in $\mathcal{X}^*$ on the subtree $\bar{\mathcal{U}}$ of $\mathcal{X}$ rooted on an active unit $\mathcal{U}$, with $\mathcal{O}$ being the temporal observation relevant to T, is such that $\bar{T} \in \bar{\mathcal{U}}^*$ and

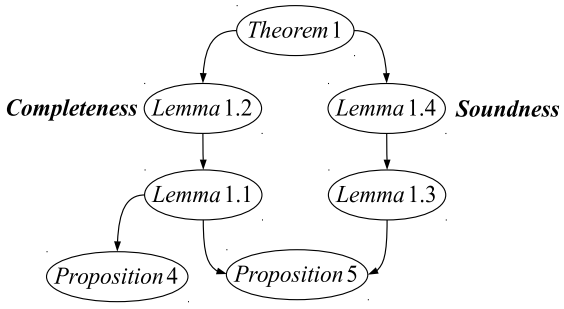


Figure 22: Dependency graph of the proof of Theorem 1 (cf. Appendix A).

the temporal observation relevant to $\bar{T}$ equals the projection of $\mathcal{O}$ on $\bar{\mathcal{U}}$. Proposition 5 claims that the diagnosis set of a unit space $\mathcal{U}^*_\xi$ equals the diagnosis set of $Int(\mathcal{U}^*_\xi)$. In other words, the set of diagnoses associated with the paths in the respective graphs are the same.

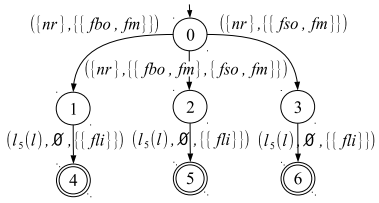
The actual proof of Theorem 1 is based on four lemmas, namely Lemma 1.1, Lemma 1.2, Lemma 1.3, and Lemma 1.4. The dependencies between the theorem, the relevant lemmas, and the ancillary propositions are depicted in Figure 22.

Lemma 1.1 claims that, given a diagnosis problem $\wp(\mathcal{X})$, a relevant projection $\wp(\bar{\mathcal{U}})$, and a constraint ξ for $\mathcal{U}$ involving the parameters of $\wp(\bar{\mathcal{U}})$, if $\delta \in \Delta(\wp(\bar{\mathcal{U}}))$, with T being a trajectory generating δ, then (1) $\mathcal{U}^*_\xi$ contains a path p whose diagnosis set equals δ, and (2) the sequence of component transitions in $\mathcal{U}$ within the triples marking p equals the subsequence of T involving the transitions of components in $\mathcal{U}$ only. Roughly, this means that each diagnosis within the solution to $\wp(\bar{\mathcal{U}})$ is included in the diagnosis set of the constrained space $\mathcal{U}^*_\xi$ (completeness of $\Delta(\mathcal{U}^*_\xi)$). As such, this lemma supports the proof of Lemma 1.2, which states the completeness of $\Delta(\mathcal{U}^{\mathrm{d}}_\xi)$, that is, $\Delta(\mathcal{U}^{\mathrm{d}}_\xi) \supseteq \Delta(\wp(\mathcal{X}))$.

In an analogous way, Lemma 1.3 claims that, given a diagnosis problem $\wp(\mathcal{X})$, a relevant projection $\wp(\bar{\mathcal{U}})$, and a constraint ξ for $\mathcal{U}$ involving the parameters of $\wp(\bar{\mathcal{U}})$, if $\delta \in \Delta(\mathcal{U}^*_\xi)$, with δ being a diagnosis relevant to a path p in $\mathcal{U}^*_\xi$, then (1) $\delta \in \Delta(\wp(\bar{\mathcal{U}}))$, with T being the trajectory in $\bar{\mathcal{U}}$ generating δ, and (2) the sequence of component transitions in $\mathcal{U}$ within the triples marking p equals the subsequence of T involving the transitions of components in $\mathcal{U}$ only. Roughly, this means that each diagnosis within the diagnosis set of the constrained space $\mathcal{U}^*_\xi$ is included in the solution to $\wp(\bar{\mathcal{U}})$ (soundness of $\Delta(\mathcal{U}^*_\xi)$). As such, this lemma supports the proof of Lemma 1.4, which states the soundness of $\Delta(\mathcal{U}^{\mathrm{d}}_\xi)$, that is, $\Delta(\mathcal{U}^{\mathrm{d}}_\xi) \subseteq \Delta(\wp(\mathcal{X}))$.

Eventually, soundness (Lemma 1.3) and completeness (Lemma 1.1) of $\Delta(\mathcal{U}^{\mathrm{d}}_\xi)$ prove the correctness of Theorem 1.

Example 29 Consider the interface of the constrained space of the AU $\mathcal{M}$, namely $Int(\mathcal{M}^*_{\xi_\mathcal{M}})$, displayed in Figure 21. Based on this interface, we can determine the constrained space of the AU $\mathcal{L}$, the line (cf. Figure 11), along with its decoration. An interface constraint for $\mathcal{L}$ is $\xi_\mathcal{L} = (l_0, \mathcal{V}_\mathcal{L}, \mathcal{O}_\mathcal{L}, \mathcal{R}, \mathbf{I}_\mathcal{L})$, where $l_0 = (normal)$, $\mathcal{V}_\mathcal{L} = \emptyset$, $\mathcal{O}_\mathcal{L}$ is empty, $\mathcal{R}$ is the ruler defined in Example 19, and $\mathbf{I}_\mathcal{L} = (Int(\mathcal{M}^*_{\xi_\mathcal{M}}))$. The space of $\mathcal{L}$ constrained by $\xi_\mathcal{L}$, namely $\mathcal{L}^*_{\xi_\mathcal{L}}$, is shown on



State	$\mathbb{S}$	$\mathbb{H}$	$\mathbb{I}$	$\mathfrak{I}$
0	(*normal*)	$(\varepsilon, \varepsilon)$	(0)	0
1	(*normal*)	(nr, ε)	(1)	0
2	(*normal*)	(nr, ε)	(2)	0
3	(*normal*)	(nr, ε)	(3)	0
4	(*isolated*)	$(\varepsilon, \varepsilon)$	(1)	0
5	(*isolated*)	$(\varepsilon, \varepsilon)$	(2)	0
6	(*isolated*)	$(\varepsilon, \varepsilon)$	(3)	0

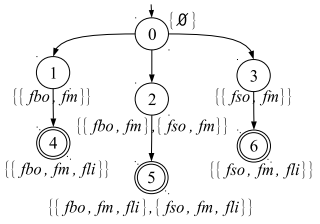


Figure 23: Constrained space of the AU $\mathcal{L}$ (left), details of states where the empty fields $\mathbb{E}$ and $\mathbb{M}$ are omitted (center), and corresponding decorated constrained space where labels associated with transitions are omitted (right).

the left side of Figure 23. The decorated space $\mathcal{L}^{\mathrm{d}}_{\xi_{\mathcal{L}}}$ is depicted on the right side. According to eqn. (30), the relevant diagnosis set is $\Delta(\mathcal{L}^{\mathrm{d}}_{\xi_{\mathcal{L}}}) = (\Delta(4) \otimes \nabla(4)) \cup (\Delta(5) \otimes \nabla(5)) \cup (\Delta(6) \otimes \nabla(6))$, where, based on the state details in Figure 23, $\nabla(4) = \Delta(1)$, $\nabla(5) = \Delta(2)$, and $\nabla(6) = \Delta(3)$, where 1, 2, and 3 are the final states of $Int(\mathcal{M}^{*}_{\xi_{\mathcal{M}}})$ (cf. Figure 21). Specifically, $\Delta(1)$ is $(\Delta(19) \otimes \nabla(19)) \cup (\Delta(19) \otimes \nabla(19)) \cup (\Delta(19) \otimes \nabla(19)) = (\Delta(19) \otimes \nabla(19)) = \{\emptyset\} \otimes \{\emptyset\} = \{\emptyset\}$. Likewise, $\Delta(2) = \Delta(3) = \{\emptyset\}$. Hence, $\Delta(\mathcal{L}^{\mathrm{d}}_{\xi_{\mathcal{L}}}) = \{\{fbo, fm, fli\}\} \otimes \{\emptyset\} \cup \{\{fbo, fm, fli\}, \{fso, fm, fli\}\} \otimes \{\emptyset\} \cup \{\{fso, fm, fli\}\} \otimes \{\emptyset\} = \{\{fbo, fm, fli\}, \{fso, fm, fli\}\}$. Based on Theorem 1 in Appendix A, $\Delta(\mathcal{L}^{\mathrm{d}}_{\xi_{\mathcal{L}}})$ is the solution to the diagnosis problem $\wp(\mathcal{X})$ defined in Example 21, embodying two candidate diagnoses. Since the subset $\{fm, fli\}$ is included in both candidates, it follows that we know for sure that the monitor m failed to alert the monitor m' and the line l was isolated. Instead, as to the cause of the malfunction in $\mathcal{P}$, we are uncertain: either the breaker b failed to open (*fbo*) or the sensor s sent the wrong command to the breaker b (*fso*).

6.7 Experimental Results

In Section 2, the usual technique adopted to solve a diagnosis problem relevant to an AS (which is a net of interconnected ACs) has been described. This technique reconstructs the overall AS behavior as constrained by the given observation, then it decorates the states in the FA representing this behavior with the relevant sets of faults: the collection of all the sets of faults relevant to the final states is the global diagnosis result. In Section 3, the notion of an AU has been introduced, this basically being an AS that can both receive and generate emerging events. Section 4 has defined the notion of a DDES, this being a hierarchy of AUs, which means that each AU in a DDES can receive emerging events only from its child AUs and can send emerging events only to its parent AU. An emergent event is generated by an AU whenever the ACs contained in it have followed a pattern of state transitions; a DFA, called a matcher, is able to detect whether a pattern has been matched (Section 5). A technique similar to that adopted for ASs can be applied to solve diagnosis problems relevant to DDESs. Basically, the behavior of each AU as constrained by the given observation is reconstructed, where each state includes also the states of the external links exiting the AU itself. Then, the constrained behaviors of all the AUs is combined into a global behavior, which is later decorated. Finally, the sets of faults relevant to the final states are extracted in the global diagnosis result. Here, this technique is referred to as *greedy*.

However, a new technique, which is able to find a sound and complete set of candidates relevant to a DDES, has been described in the current Section 6. This technique is here referred to as *lazy*, as it does not compute the global trajectories of the DDES, these being unnecessary in order to find the global candidates. Both the greedy and the lazy techniques encompass the same preprocessing step in order to build the matchers, each relevant to a distinct pattern (regular expression) given as input. Both techniques perform the reconstruction of the behavior of each AU, as constrained by the given local observation. As briefly mentioned in Section 4.5, the time needed for this reconstruction has an upper bound that is exponential in the number of ACs and (internal) links of the considered AU. The other steps are different in the two techniques, however. An activity was carried out in order to experimentally show that the lazy technique outperforms the greedy one.

It is well known that the task of model-based diagnosis, even in the form defined by Reiter (1987) for static systems, suffers from two potential combinatorial problems: computing a diagnosis is NP-hard, and the number of diagnoses can be exponential in the number of distinct faults. Lamperti and Zanella (2013) proved that, for distributed DESs, checking the existence of a trajec-

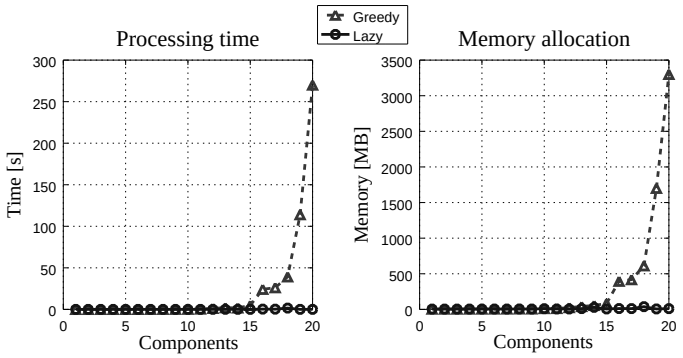


Figure 24: Comparison between greedy diagnosis and lazy diagnosis (CPU time and memory).

tory that produces the given observation is NP-hard. The experimental results recorded by solving a number of diagnosis problems relevant to DDESs by adopting the lazy technique may look somewhat surprising, since the computation time is linear in the number of ACs included in each considered DDES. An interpretation of these results is given in Section 9.

Diagnosis of DDESs has been prototyped in C++ under Linux Ubuntu 15.10, on a notebook with an *Intel Core i5* CPU (2.40GHz) and 4GB of RAM, using a single thread. The diagnosis framework includes a compiler for offline preprocessing and an online diagnosis engine. A language was designed to specify DDESs in terms of ACs, AUs, and emerging events, as well as diagnosis problems in terms of viewer, observation, and ruler. The simulated systems have been generated semi-automatically based on a given template. The diagnosis engine can run in either *greedy* or *lazy* mode, thereby enabling the comparison of the performances of the two techniques. As already explained, the greedy mode requires the reconstruction of the overall DDES behavior as constrained by the given observation, exactly like in diagnosis of ASs (Lamperti, Zanella, & Zhao, 2018b), whereas the lazy mode embodies the diagnosis method specific to DDESs described in the (previous subsections of the) current section of the paper. The implementation and experimentation have been conducted in a master thesis work at the University of Brescia.

Each DDES adopted for the experiments represents a power transmission system consisting of several contiguous protected lines. The schema of a protected line is a variant of that outlined in Figure 10; in this variant, the protection apparatus at either end of the line includes two breakers and (only) one sensor, the same as in the paper by Lamperti and Zhao (2013a). Each leaf AU in the DDES hierarchy represents a protection apparatus, hence it includes three ACs. A leaf AU sends the emergent events *di*, *co*, *nc*, and *nc* to its parent AU. Each pair of leaf AUs are the children of a distinct AU, positioned in the next upper level; this AU represents a kind of monitoring process inherent to a single protected line. Each monitoring AU sends the emergent events *ni*, *nr*, and *ps* to its parent AU, which includes either two or three ACs, each representing a line. Hence, altogether, each parent AU represents either two or three contiguous lines. Each pair of these AUs are the children of a higher level AU (if any), which thus cumulatively represents either four or six contiguous lines. An agglomerate of several lines is called a *superline*. For each of the next higher levels of the hierarchy up to the root, each AU (if any) represents the union of the pair of its child superlines. Specific rulers have been defined for each superline, whose faults are based on the emergent events coming from the child AUs (be they lines or superlines).

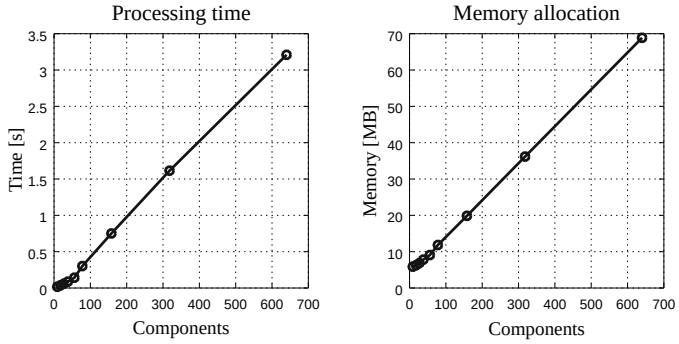


Figure 25: Experimental results with lazy diagnosis only (CPU time and memory).

The above modeling choices allowed for a modular construction of DDESs representing power transmission systems of increasing size. As already remarked, in the third layer from the bottom, AUs aggregating either two or three lines can be adopted. If in this layer only aggregates of two lines are built, the overall system represents a number of lines which is a power of two, e.g. $n = 2, 4, 8, \ldots, 64$. For instance, with 64 lines, the resulting DDES is composed of 255 AUs and 638 components (i.e. ACs), which are accommodated within a balanced binary tree having 8 layers. Every AU in the bottom layer has three ACs, while every AU in the other layers has two ACs. If instead, in the third layer from the bottom, agglomerates of three lines are created, despite the DDES hierarchy being a binary tree, the overall system represents a number of protected lines that is not a power of two. All the DDES hierarchies in the experiments has a binary tree structure, however the tree is not necessarily a balanced one.

Comparison results are shown in Figure 24, both in CPU processing time (left) and in memory allocation (right). The horizontal axis indicates DDESs of increasing size, up to 20 components (ACs). The size of the DDES is limited because, as indicated by the figure, at a certain point (around 15 components), both the processing time and the memory allocation in greedy mode explode. For instance, in the last (and largest) DDES, the processing time in greedy mode is 270 seconds, while in lazy mode it is 0.032 seconds. Similarly, in the last DDES, the allocated memory in greedy mode is 3300 MB, while in lazy mode it is 10.5 MB.

In order to test the experimental complexity of lazy diagnosis only, another set of experiments was done, as shown in Figure 25. Nine DDESs have been considered, from 9 to 638 components. The number of AUs in each DDES ranges from 4 to 255. As shown in the graphs, both processing time and memory allocation are practically linear in the number of components. For instance, with 318 components (127 AUs), the processing time is 1.61 seconds and the memory requires 36.18 MB; with 638 components (255 AUs), the processing time is 3.20 seconds and the required memory is 68.87 MB.

It is worth highlighting that the comparison of the outputs produced by greedy diagnosis and lazy diagnosis in the experiments have also indicated that both diagnosis engines invariably generate the same set of candidates, which is in fact the solution to the diagnosis problem. This outcome was expected and reinforces in empirical terms the theoretical claim of Theorem 1, which proves the soundness and completeness of lazy diagnosis.

Evidence from the experiments points out that diagnosing a DDES of significant size would be impossible adopting the classical abduction-based technique requiring the generation of the $\mathcal{O}$-

constrained space. Whether an application for "real" systems might be actually developed using the lazy technique is still an open question. Yet, given the reported results, the feeling is that lazy diagnosis of DDESs may be a viable solution.

7. Discussion

In the 1990s, a new class of DESs was introduced, coined *active* systems, as they are meant to represent systems that act in order to affect the external world and to react to events coming from it. Although conceived independently, when ASs appeared in the literature (Baroni et al., 1998), a major contribution about diagnosability and diagnosis of DESs, namely the diagnoser approach (Sampath, Sengupta, Lafortune, Sinnamohideen, & Teneketzis, 1995; Sampath et al., 1996), had already been delivered. The diagnoser approach proposed to model a distributed DES as a network of synchronous FAs, where each FA represents the behavior of a component, this being a monolithic DES. In other words, according to the diagnoser approach, a component can influence the behavior of other components since some state transitions occur at the same time in distinct components. ASs, instead, are asynchronous by nature, that is, in a distributed AS, each component (a monolithic DES whose behavior is modeled by an FA, the same as in the diagnoser approach) can influence the behavior of other components by communicating asynchronously with them: this is achieved by making the components exchange (internal) events through finite directed buffers, called (internal) links. Hence, the notion of an AS relies on the notion of a link: a specific modeling primitive is used for representing a link, and a diagnostic engine relevant to ASs has to be able to process it correctly. However, including links in a DES model is a matter of expressive power, not of computational power. A link, whichever its capacity and its policy (FIFO, LIFO, etc.)[27], could be represented as a (synchronous) component itself (Lamperti & Zanella, 2013), that is, as an FA. The class of DESs described by using links, that is, the class of ASs, is a subset of the class of DESs described as interacting synchronously[28], since an asynchronous communication can be modeled through the primitive for synchronous communication (i.e. the synchronous transition). Adopting specific primitives for asynchronous automata communication comes with several advantages. First, the link primitive is generic (it specifies just the capacity and the policy), that is, independent of the number and types of exchanged events, while the component model of a link depends on them; second, the component model of a link includes a (possibly large) number of states that equals $\sum_{i=0}^{n} d^i$, where d is the cardinality of the set of communication events and n is the capacity of the link; third, the link primitive is processed more efficiently (by ad hoc algorithms) than the component model of a link. These advantages are not the reason for ASs were proposed.

This paper has introduced the class of DDESs, which adopts additional links, the external ones, to convey a new kind of events, the emergent events. Hence, new primitives have been added to the ones already available to model ASs. Specifically, in order to represent an emergent event, we need to specify regular expressions since an emergent event is generated whenever a subtrajectory matches a given regular expression. Hence, the notion of a DDES relies on the notions of AS and regular expression. Once again, including ad hoc primitives for such notions in a DES model is a

27. In the previous sections, it has been assumed that every link can store a single event (so its capacity is one) and that no event can be sent to the link when the link is full (this policy is called WAIT). However, more generically, other finite capacities and different policies can be adopted for links in the AS approach.

28. The class of DESs interacting synchronously does not equal the class of DESs defined in the diagnoser approach, instead it is a superset of it. In fact, the diagnoser approach assumes the liveliness of both the language of transitions and the observational language. Such assumptions are relaxed in the AS approach.

matter of expressive power, not of computational power. All the ASs included in a DDES can be represented as distributed synchronous DESs; external links could be represented as synchronous DES components; each regular expression could be represented by its matcher, an FA that could be dealt with in the same way as all the behavioral descriptions of synchronous DES components. Hence, the class of DDESs is a subset of the class of synchronous DESs (and the class of DDESs includes the class of ASs since each AS can be seen as DDES trivially consisting in just one AU). Modeling an existing AS as a DDES with multiple layers[29] comes with advantages, first of all a more efficient diagnosis method. Still, these advantages are not the reason for DDESs were proposed.

So, what is the rationale behind ASs and DDESs? It is, first of all, the simplicity, intuitiveness, readability, and understandability of the models. If you have to model a DES whose components interact asynchronously, describing it as an AS is much more natural and easier than representing it as a distributed DES, where components interact synchronously and some components are really active whereas other are indeed passive as they are a trick to model buffers. If you have to model the hierarchical abstractions of a DES, specifying it as a DDES is more natural and easier than describing it as a flat DES consisting of synchronous components or even as a flat AS.

In order to substantiate this allegation, take the DES (power transmission line) described in Example 15. If it were represented as an AS, then $\mathcal{P}$ would not be an AU that generates emergent events *di*, *co*, *nd*, *nc* directed to monitor m, instead it would be a pair of components (b and s) sending internal events to component m. Each emergent event communicates a distinct circumstance the target component has to take care of, where such a circumstance may be the result of an evolution involving several components (as it is possibly the case with emergent events *nd* and *nc*). The circumstances m is interested in are disconnection (emergent event *di*), connection (*co*), failure in disconnecting (*nd*), and failure in connecting (*nc*). If the DES is represented as an AS, such circumstances are not communicated to the monitor as emergent events, hence the monitor has to be able to intercept them based on the internal events coming separately from the breaker and the sensor. This means that the model of the monitor would not be that displayed on the right of Figure 12, instead it would be more complex and less readable as it should represent also the behavior needed to discriminate between the four different circumstances of the protection apparatus instead of just the behavior to react to them. Analogously, if the DES were described as an AS, the four components m, m', r and r' would not be regarded as a unit (AU $\mathcal{M}$) that sends to line l the emergent events specifying that a side of the line cannot be reconnected (*nr*) or isolated (*ni*) or that it is affected by a persisting short circuit (*ps*). Instead, line l would receive internal events coming separately from components m, m', r and r'; therefore, the model of l would not be that depicted on the right of Figure 14, instead it should include also the behavior needed to discriminate what is going on on either side of the line. The behavior (displayed in Figure 14) of l when the system is represented as a DDES is quite simple to design and understand: it includes just four states, that clarify the real situation of the line (which can be normal, isolated, or shorted since a breaker has not opened, or even persistently shorted after a breaker has been closed). If the DES were modeled as an AS, the model of the line l would be quite larger (compactness of representation is an asset of DDESs), and its states would be less understandable since there would be intermediate states just

29. An AS can easily be represented as a DDES if the set of its components can be partitioned such that each part is a node in a hierarchy where the components in each node communicate, through links, only with the components in a single node in the next upper level and the communication is unidirectional and each link can store just one kind of events.

inherent to the process of discriminating the actual situation of the line and link states representing the actual situation that has been found out.

The diagnosis of a flat AS is provided at the component level (e.g. breakers and protection devices): the whole system is faulty only if it includes some faulty component(s). Since no function-based interpretation of what has happened is given, it may be difficult for an operator to decide the correct actions to carry out. By contrast, the context-sensitive diagnosis provided for a DDES is bound to result in a clearer understanding of the risks associated with the actions. For instance, the faults relevant to the line in the DDESs described in Example 15 are not actually faults in the traditional sense that something in the line has broken. In fact, nothing in the line can break; the faults associated with the AU including just the line component, which is the root of the hierarchy, provide an interpretation of what has happened at system level (e.g. fault *fli* denotes that the line keeps being isolated since a breaker on the left side remains open). This considerably enhances the explainability of the diagnosis results.

In addition, modeling complex systems as DDESs provides a support for integration. If we have several DESs, each one devoted to a specific task, and we want to integrate them into a new DES in order to control all DESs at a higher abstraction level, then the new DES can be conveniently modeled as a DDES, which is sensitive to specific behavioral patterns of the integrated DESs (emergent events). This way, monitoring and diagnosing the DDES is far more natural than interpreting (possibly overwhelming) streams of low-level events generated by individual DESs, such as the alarms detected in a control room. The key point is that the DESs are integrated in a new system not just by connecting them with one another by means of new (internal) links between components belonging to different DESs. Instead, besides this physical integration, a semantic integration is achieved too, so that the new DDES is not just the union of the DESs: it is a higher-level system with its proper behavior.

Semantic patterns specified as regular expressions were first introduced in the AS approach by Lamperti and Zanella (2010). The work, in order to assign a different status to diagnosis candidates that cause a (sub)system to misbehave with respect to candidates that do not cause any misbehavior, defines the concept of *pattern rule*. A pattern rule is a regular expression over the alphabet of the transitions of the components contained in the considered (sub)system, where distinct subsystems may overlap. Any sequence of transitions matching the same expression denotes a specific malfunction. There are distinct rules for distinct malfunctions. In this respect, a pattern rule resembles the regular expression for the generation of an emergent event: a detected malfunction is indeed an example of emergent event, that can propagate higher in the hierarchy since a misbehaving subsystem can cause the misbehavior of another. In fact, in the paper by Lamperti and Zanella (2010) a DAG, called dependency graph, is adopted in order to mimic this malfunction propagation. Still, the generation of emergent events in DDESs can represent further phenomena in addition to malfunction propagation; for instance, it can represent causal chains of faults.

The next step in the research leading to DDESs is related to the work by Lamperti and Zanella (2011), which remarks that the topology of a complex DES, as defined in another work of the same authors (2010), is organized in a hierarchy, where the root corresponds to the whole system, leaves to components, and intermediate nodes to subsystems. This topology, which is called *context hierarchy* by Lamperti and Zhao (2014), suggests to organize the diagnosis rules within a hierarchy that parallels the structure of the DES: each subsystem has its proper set of diagnosis rules, which may or may not depend on the rules of inner subsystems (nodes in the next lower layer of the hierarchy). Candidate diagnoses can be generated at the different levels of the hierarchy. This

approach, called *context-sensitive* diagnosis, provides the example of a possible DDES: the above complex topology can be accommodated in the hierarchical structure of a DDES. However, the hierarchical structure of a DDES can be exploited to represent several abstractions other than a containment hierarchy.

The awareness that a hierarchical structure can represent any abstraction that is profitable for a given purpose is gained in the works by Lamperti and Zhao (2013a, 2013b), where the notion of *behavior stratification*, which is adopted in DDESs, was introduced. Hence a context is not necessarily a (sub)system in a containment hierarchy, instead, it is a node in an abstraction hierarchy. Behavior stratification means that the behavior of each context, placed in a layer of the hierarchy, possibly depends on the behaviors of its child contexts, placed in the adjacent lower layer. This is a significant change w.r.t. to the work by Lamperti and Zanella (2011), where it is the diagnosis rules of a (sub)system that can be affected by the diagnosis rules of lower layer (sub)systems, whereas the behavior of a (sub)system is invariably implicitly given by the composition of the behaviors of its components. In the works by Lamperti and Zhao (2013a, 2013b), instead, each node in the hierarchy has its own behavioral model, which does not coincide with the composition of the behavioral models of its components, as two distinct nodes consisting of the same components can be affected by the behavior of their lower level nodes in different ways. The communication between subsystems at different levels relies on *complex events*, occurring when specific patterns of transitions are matched. These patterns are defined based on the transitions of contained subsystems, where each subsystem has its own modeled behavior.

The current notion of DDESs can support any hierarchical abstraction. When a context is made up of components only, a regular expression defines a pattern of transitions for such components. In the general case, when a context is made up of subcontexts, the regular expression defines a pattern of interface symbols associated with its subcontexts, with each interface symbol being associated with a regular expression in the subcontexts. Basically, its strength consists in providing the modeler with the capabilities for (*i*) representing the behavior of a subsystem/context concisely and effectively, as the reaction (also) to complex phenomena that have taken place in other parts/contexts of the system, and (*ii*) separating the model of this behavior from the representation of such phenomena. The former model can include (the representation of) faults that cannot be simply inferred from the faults of lower-level contexts, while the latter models are free of faults. A DDES highlights that each subsystem has a combined behavior, partly depending on its components and partly on the function this subsystem is assigned within the system.

8. Related Work

In this work, each event emerging from a layer in a DDES hierarchy, directed to the adjacent upper layer, corresponds to the occurrence of a string of component transitions in a regular language. In other words, an emergent event is generated when a behavioral pattern is recognized. Superficially, a behavioral pattern looks like a *fault supervision pattern*, defined by Jéron et al. (2006). However, the two categories of patterns are orthogonal: supervision patterns are meant to generalize the concept of a fault for both diagnosis and diagnosability of flat DESs, while patterns for the generation of emergent events aim to support behavior stratification in DDESs.

In the work by Console, Picardi, and Theseider Dupré (2003), temporal patterns about observables are represented by so-called *temporal decision trees*. These trees are knowledge compilation schemes that can be used for both qualitative and quantitative temporal constraints relevant to the

history of observables once a fault has been detected without being isolated. The values of the observables are assumed to be acquired at discrete times. The compiled knowledge is exploited to perform fault isolation so as to carry out a recovery action before a given hard deadline, which has been set by keeping safety and integrity of the physical system into account. Although a temporal decision tree is a means to represent a temporal pattern, the same as a matcher in the current approach, its alphabet, semantics, and purpose are quite different. The alphabet of a temporal decision tree is the set of values of an observable, whereas the alphabet of a matcher is a set of transitions. A temporal decision tree is based on the assumption that observations are performed at regular times, whereas the current approach assumes to intercept each value change whenever it occurs. The temporal decision tree represents a finite temporal horizon (the deadline within which the recovery action has to be carried out), whereas a matcher is not constrained by any time limit. A temporal decision tree is the result of a knowledge compilation process, whereas a matcher is not. Finally, the purpose of a temporal decision tree is to discriminate a fault, whereas the purpose of a matcher is to detect the occurrence of an emergent event.

DDESs are endowed with a tree-shaped hierarchical structure similar to that of *hierarchical finite state machines* (HFSMs), which are inspired by state-charts (Harel, 1987). Diagnosis of HFSMs is considered by Idghamishi and Zad (2004) and by Paoli and Lafortune (2008). However, in contrast with DDESs, the complexity of HFSMs is confined to the structure, without emergent events or behavior stratification.

A jointree structure that is similar to the hierarchical structure of DDESs is mentioned by Darwiche (1998). *Jointrees*, also called *junction trees*, *cluster trees*, *clique trees*, or *trees of belief universes* in the literature (Huang & Darwiche, 1996), are adopted in various AI fields, including probabilistic reasoning (Shenoy & Shafer, 1986; Huang & Darwiche, 1996) and constraint processing (Dechter, 2003). Darwiche (1998) proposes a symbolic logic characterization of consistency-based diagnosis of static systems that is alternative to the classical one (Reiter, 1987; de Kleer & Williams, 1987; de Kleer, Mackworth, & Reiter, 1992). The traditional behavioral system description is augmented with a system structure (a DAG explicating the interconnections between components): this pair of representations is altogether called a *structured system description*. By exploiting the novel characterization, the paper proposes a new diagnostic method and tries to minimize the computational complexity of such a method by graphically transforming the system structure into a jointree. The contribution includes a number of guidelines for creating models that can be processed by diagnostic algorithms efficiently, and provides computational guarantees based on the topology of the system structure. However, the addressed systems are static, the diagnosis method is consistency-based, and no deep behavior is conceived, as the jointree structure is solely meant to reduce the computational effort.

Darwiche and Provan (1996) apply the notion of a structured system description (Darwiche, 1998) to continuous dynamic systems, each governed by a discrete controller, a device that issues control commands at discrete time points. While for static systems the structure is a DAG with nodes representing the inputs and outputs of the system components, and arcs depicting their interconnections, for dynamic systems the structure has been extended with directed cycles (satisfying specific conditions). The behavior of each system component is specified by a propositional temporal logic with quantification over discrete time (temporal sentences have to fulfill some conditions), and component descriptions are accommodated in a symbolic causal network, which represents the overall system model. The pair (system model, system structure), called dynamic system description, is then turned into the system description defined by Darwiche (1998) in linear time. This

phase, called expansion, can be carried out offline and consists in creating a causal network for each time point in the time-step cycle of the system and connecting each network with the next time-point network. After that, the diagnosis approach for static systems described by Darwiche (1998) is exploited. The most computationally intensive phase depends on the topology of the system structure. Like in the work by Darwiche (1998), in order to enhance the efficiency of the diagnosis process one can first convert the system structure into a jointree. The notion of a dynamic system adopted by Darwiche and Provan (1996) is far from that of a DES of the current paper, let alone from that of a DDES. Once again, the jointree is not regarded as an abstraction hierarchy, instead it is exploited to reduce the diagnosis computation time.

Likewise, no deep behavior is conceived by Stumptner and Wotawa (2001), where an algorithm for computing minimal diagnoses of tree-structured systems is presented. The goal is to improve the efficiency of the diagnosis task by exploiting the structure of the system. Subdiagnoses, which are generated by traversing the tree top-down, are eventually combined to yield the candidate diagnoses of the whole system. The considered systems are static while DDESs are dynamic, and the diagnosis method is consistency-based, whereas the method described in this paper for DDES diagnosis is abductive and processes the tree structure bottom-up.

Jointrees are exploited also in DES diagnosability analysis. In contrast with the technique for diagnosability-checking of DESs proposed by Sampath et al. (1995), which suffers from exponential complexity, the *twin plant* method (Jiang, Huang, Chandra, & Kumar, 2001; Yoo & Lafortune, 2002) exhibits a polynomial complexity. However, the construction of a global twin plant, which corresponds to the synchronization based on (all) observable events of two identical instances of the automaton representing the whole DES behavior, is often impractical. The scalability of the approach is increased by Schumann and Huang (2008) by taking advantage of the distribution of a DES to build a local twin plant for each component. Then, the DES components (and their relevant local twin plants) are organized into a jointree. Both the method by Schumann and Huang (2008) and that presented in this paper are distributed. However, the tasks accomplished by the two methods are profoundly different, as the transformation of the DES into a jointree by Schumann and Huang (2008) is an artifice for reducing the complexity of the diagnosability analysis task, whereas the tree-shaped structure is an integral part of the topological and behavioral abstraction in DDES modeling.

A decentralized/distributed approach to diagnosis of DESs was introduced by Kan John and Grastien (2008), with the aim of computing local diagnoses that are globally consistent. By definition, a diagnosis is a trajectory that complies with the given observation. The observation is a set of sequences of observable events, each sequence being inherent to a distinct component. The paper enhances the proposal by Su and Wonham (2005) in order to achieve global consistency in local diagnoses of distributed DESs without generating either any global diagnosis or the global system model. It is well known that local (pairwise) consistency does not ensure global consistency, and that an algorithm that iteratively refines local diagnoses pairwisely until stability is reached (that is, no local diagnosis changes anymore) may not terminate. However, local consistency ensures global consistency when the connections between components form a tree. Thus, the proposal by Kan John and Grastien (2008) is to convert the connections between the components of the distributed DES, which, in the general case, form a graph, into a jointree. Each node in the tree is a cluster of components (a subsystem) of the DES, where all such components are interconnected in the system, that is, each pair of them share at least one event. Although clusters differ from one another, nonetheless distinct clusters may partially overlay and their union gives the set of all system components. Two

subsystems (nodes of the jointree) that share an event are connected through an edge of the jointree, or through a chain of edges, where intermediate subsystems also share this event. Building an optimal jointree is NP-complete but, by exploiting heuristics (Huang & Darwiche, 1996), the algorithm that transforms a graph into a jointree achieves polynomial-time while still producing a high quality tree (the fewest nodes, associated with the smallest clusters). Diagnosis (the construction of all the trajectories) is performed locally on each cluster by the classical synchronous composition of component models and observations. After a cluster has been picked to be the root of the jointree, the consistency of each local diagnosis with the local diagnoses inherent to neighbor nodes in the tree is achieved according to a strategy, called global propagation (Huang & Darwiche, 1996), which consists of two phases. In the former, called gather phase, local consistency starts from leaf nodes and is performed up to the root. In the latter, called distribute phase, local consistency is performed in the opposite direction, from the root down to the leaves. The set of all globally-consistent local-diagnoses is the distributed diagnosis of the system. The aim of Kan John and Grastien (2008) is slightly different from that of the current paper: the former computes (globally consistent) local diagnoses whereas we compute (globally consistent) global diagnoses. The notion of diagnosis is different in the two papers: for Kan John and Grastien (2008), a diagnosis is (a possibly infinite) set of trajectories whereas in our paper a diagnosis is a (finite) set of candidates, a candidate being a (finite) set of faults. Although the DESs dealt with in both papers are distributed, they differ as in the work by Kan John and Grastien (2008) they are plain, synchronous, with no deep behavior, and they are not explicitly modeled, that is, a DES (component) is a regular language; instead, in this paper, the systems are asynchronous, they feature behavior stratification and a DES component is explicitly modeled as an FA. While for Kan John and Grastien (2008) any node of the tree (and then the root also) is a subsystem, possibly structurally overlapping with the other subsystems, in the current approach a node is the representation of a (sub)system at some abstraction level and there is no overlapping. Finally, the root of a DDES is the representation of the whole system at the highest abstraction level, whereas the root of a jointree is chosen arbitrarily.

An approach to consistency-based diagnosis supported by structural abstraction (which aggregates components to represent a static system at different levels of structural detail) is described by Chittaro and Ranon (2004) as a remedy for the high computational complexity of model-based diagnosis. Evidence from experimental evaluation indicates that, on average, the technique performs favorably with the algorithm of Mozetič (1992). Still, differently from the topic of this paper, the approach is consistency-based, the considered systems are static, and no emergent behavior is conceived.

Siddiqi and Huang (2011) exploit structural abstraction in order to reduce the cost of sequential diagnosis, this being a model-based diagnosis task where, once the candidates relevant to a given observation have been computed, faults are isolated by exploiting the evidence provided by a sequence of measurements. The diagnostic cost is the number of the measurements that are needed until all the actual faults are identified. The measurements and the order in which they have to be taken are selected heuristically. The structural abstraction dealt with in the paper is based on hierarchical diagnosis (Siddiqi & Huang, 2007), which can decompose the system into self-contained subsystems, each of which is regarded as a single component at a higher abstraction level. This means that a single health variable and failure probability is adopted for the entire subsystem, thus scaling the approach to handle larger systems. If a subsystem is identified as faulty in the top abstraction level, it can then be processed separately in a recursive fashion. In fact, a subsystem may contain further subsystems, leading to a hierarchy of subsystems. A system can be abstracted by

treating all its maximal subsystems as black boxes. In addition, Siddiqi and Huang (2011) proposed a method that systematically modifies the structure of a system to reduce the size of its abstraction. The attention of such a method is focused on the components that are not part of any subsystem and hence cannot be abstracted away in hierarchical diagnosis. The idea is to create a sufficient number of clones of each of such components so that the original component and each of its clones become part of some subsystems and hence can participate in the abstraction. The abstraction dealt with by Siddiqi and Huang (2007, 2011), unlike the one proposed in the current approach, is purely structural and is meant to scale diagnosis to large systems. Moreover, the examples of systems taken into account by Siddiqi and Huang (2007, 2011) are combinational circuits, hence static systems, although the authors claim that their sequential diagnosis framework applies as well to other types of systems as long as the system behavior is described by a probabilistic model.

9. Conclusion

The shift from ASs to DDESs, which is motivated by ergonomics in the modeling task, does not come with any additional cost in the diagnosis task, which is not only sound and complete, but also viable. Since a state of the AS includes the states of *all* components and the states of *all* links, the complexity of the abduction of the whole AS in diagnosis of ASs is exponential both in the number of components and in the number of links. This theoretical expectation is confirmed by the experimental results presented in Section 6.7. Two diagnosis engines have been implemented in order to diagnose DDESs. The former engine makes use of the same technique adopted in diagnosis of ASs (Lamperti et al., 2018b), while the latter engine operates according to this paper. The results clearly show that the processing time of the latter engine increases almost linearly with the size of the system, in contrast with the processing time of the former engine, which grows exponentially.

We believe that there are two reasons for this surprising difference in the performances of the two implementations: (*a*) the new technique is able to distill from each AS the essential constraints to be propagated upward in the hierarchy in order to guarantee that the trajectory reconstruction relevant to an AS complies with the global observation as projected on the AS itself and all its descendant ASs, without any need for performing the asynchronous product of the reconstructed behaviors of distinct ASs, and (*b*) the new method, being specific for DDESs, is able to take advantage of their hierarchical structure, while the previous method, being general, handles all systems the same way, independently of their structure. The latter point possibly corroborates several findings in the literature (Darwiche & Provan, 1996; Darwiche, 1998; Schumann & Huang, 2008; Kan John & Grastien, 2008), according to which the computation time of diagnosis and diagnosability analysis can be reduced by transforming the system structure into a tree structure. In turn, this suggests that the performance of existing methods for DES diagnosis can be enhanced by modeling the DES as a DDES. Moreover, we expect that the technique proposed in this paper can still save more computation time by performing a powerful knowledge compilation, as suggested in previous works (Lamperti, Zanella, & Zhao, 2018a, 2018c), and by processing online independent AUs in parallel. These remarks are the spurs to future investigation on DDESs.

Further research paths can be envisaged. First, the tree-based topology of the DDES can be relaxed to a directed acyclic graph (DAG), where an AU can have several parent units. Second, the diagnosis task, which in this paper is assumed to be *a posteriori*, that is, carried out once a complete temporal observation has been gathered, can be made *reactive*, where diagnosis is performed as soon as a piece of observation is received. Third, knowledge compilation can be adopted

in order to generate a fast *diagnoser*, that is, a DFA whose language includes all possible temporal observations of the DDES. Fourth, the patterns defining emergent events can be extended beyond regular languages, based on more powerful grammars, possibly enriched by semantic rules. Finally, in order to avoid the generation of the whole set of candidate diagnoses, the search space might be pruned based on domain-specific criteria.

Acknowledgments

The authors would like to acknowledge the anonymous reviewers for their constructive comments and inspiring suggestions, which resulted in a considerable improvement of the quality of the final manuscript. Gianfranco Lamperti also expresses his gratitude to Giulio Quarenghi, who implemented the compiler of the DDES specifications and the diagnosis engine in C++, thereby providing the experimental results in his master thesis. This work was supported in part by Regione Lombardia (Smart4CPPS, Linea Accordi per Ricerca, Sviluppo e Innovazione, POR-FESR 2014-2020 Asse I) and by the National Natural Science Foundation of China (grant number 61972360).

Appendix A. Proof of Theorem 1

In this appendix, a proof of soundness and completeness of the technique presented in Section 6 is provided (Theorem 1). This guarantees that the diagnosis set based on eqn. (30) equals the solution to the diagnosis problem of a DDES, as defined in eqn. (19). In so doing, four definitions are given (Definition 26, Definition 27, Definition 28, and Definition 29), along with two ancillary propositions (Proposition 4 and Proposition 5).

Definition 26 *Let* $\boldsymbol{\Delta} = [\Delta_1, \ldots, \Delta_n]$ *be a (possibly empty) sequence of sets of diagnoses. The* diagnosis set *of* $\boldsymbol{\Delta}$ *is a set of diagnoses defined as follows:*

$$\Delta(\boldsymbol{\Delta}) = \begin{cases} \{\emptyset\} & \text{if } \boldsymbol{\Delta} = [\,] \\ \Delta_1 & \text{if } \boldsymbol{\Delta} = [\Delta_1] \\ \Delta_1 \otimes \Delta_2 \otimes \ldots \otimes \Delta_n & \text{otherwise.} \end{cases} \tag{32}$$

Definition 27 *Let* p *be a path within a constrained unit space* $\mathcal{U}_\xi^* = (\Sigma, W, \tau, w_0, W_\mathrm{f})$*, namely*

$$p = \left[\langle w_0, \sigma_1, w_1\rangle, \langle w_1, \sigma_2, w_2\rangle, \ldots, \langle w_{q-1}, \sigma_q, w_q\rangle\right] \tag{33}$$

and $\boldsymbol{\Delta}$ *the sequence of diagnosis sets* Δ_i *involved in* σ_i*, where either* $\sigma_i = (\mathbb{E}_i, \Delta_i)$ *or* $\sigma_i = (t(c), \mathbb{E}_i, \Delta_i)$*,* $i \in [1\,..\,q]$*. The* diagnosis set *of* p*, denoted* $\Delta(p)$*, is defined as*[30]

$$\Delta(p) = \Delta(\boldsymbol{\Delta}) \otimes \nabla(w_q). \tag{34}$$

The diagnosis set *of* $\mathcal{U}_\xi^*$ *is defined as*

$$\Delta\left(\mathcal{U}_\xi^*\right) = \bigcup_{p \in \|\mathcal{U}_\xi^*\|} \Delta(p). \tag{35}$$

30. $\Delta(\boldsymbol{\Delta})$ and $\nabla(w)$ are defined in eqn. (32) and eqn. (28), respectively.

Definition 28 *Let p be a path within an interface Int($\mathcal{U}_\xi^*$),*

$$p = \left[\langle i_0, (\mathbb{E}_1, \Delta_1), i_1\rangle, \langle i_1, (\mathbb{E}_2, \Delta_2), i_2\rangle, \ldots, \left\langle i_{q-1}, (\mathbb{E}_q, \Delta_q), i_q\right\rangle\right] \tag{36}$$

and $\boldsymbol{\Delta} = [\Delta_1, \ldots, \Delta_q]$. The diagnosis set *of p, denoted $\Delta(p)$, is defined as*[31]

$$\Delta(p) = \Delta(\boldsymbol{\Delta}) \otimes \nabla(i_q). \tag{37}$$

The diagnosis set *of Int($\mathcal{U}_\xi^*$) is defined as*

$$\Delta\left(Int\left(\mathcal{U}_\xi^*\right)\right) = \bigcup_{p \in \|Int(\mathcal{U}_\xi^*)\|} \Delta(p). \tag{38}$$

Definition 29 *Let $\wp(\mathcal{X}) = (x_0, \mathcal{V}, \mathcal{O}, \mathcal{R})$ be a diagnosis problem for a DDES $\mathcal{X} = (\mathbf{U}, \mathbf{L})$, $\mathcal{U} \in \mathbf{U}$, $x_0 = (\mathbb{U}_0, \mathbb{E}_0)$, and let $\bar{\mathcal{U}} = (\bar{\mathbf{U}}, \bar{\mathbf{L}})$ be the DDES corresponding to the subtree of $\mathcal{X}$ rooted in $\mathcal{U}$. The* projection *of $\wp(\mathcal{X})$ on $\bar{\mathcal{U}}$ is the diagnosis problem $\wp(\bar{\mathcal{U}}) = (\bar{u}_0, \bar{\mathcal{V}}, \bar{\mathcal{O}}, \bar{\mathcal{R}})$, where $\bar{u}_0 = (\bar{\mathbb{U}}_0, \bar{\mathbb{E}}_0)$, $\bar{\mathbb{U}}_0$ is the restriction of $\mathbb{U}_0$ on the states of AUs in $\bar{\mathbf{U}}$, $\bar{\mathbb{E}}_0$ is the restriction of $\mathbb{E}_0$ on the events within the links between AUs in $\bar{\mathbf{U}}$, $\bar{\mathcal{V}}$ is the restriction of $\mathcal{V}$ on the viewers of the AUs in $\bar{\mathbf{U}}$, $\bar{\mathcal{O}}$ is the restriction of $\mathcal{O}$ on the temporal observations of the AUs in $\bar{\mathbf{U}}$, and $\bar{\mathcal{R}}$ is the restriction of $\mathcal{R}$ on the rulers of the AUs in $\bar{\mathbf{U}}$.*

Proposition 4 *Let $\wp(\mathcal{X}) = (x_0, \mathcal{V}, \mathcal{O}, \mathcal{R})$ be a diagnosis problem for the DDES $\mathcal{X}$, $\mathcal{U}$ an AU within $\mathcal{X}$, $\bar{\mathcal{U}}$ the DDES rooted in $\mathcal{U}$, $\wp(\bar{\mathcal{U}}) = (\bar{u}_0, \bar{\mathcal{V}}, \bar{\mathcal{O}}, \bar{\mathcal{R}})$ the projection of $\wp(\mathcal{X})$ on $\bar{\mathcal{U}}$, $T \in \|\mathcal{X}^*\|$ such that $Obs(T, \mathcal{V}) = \mathcal{O}$, and $\bar{T}$ the subsequence of T relevant to transitions of components in $\bar{\mathcal{U}}$. We have $\bar{T} \in \|\bar{\mathcal{U}}^*\|$, $Obs(\bar{T}, \bar{\mathcal{V}}) = \bar{\mathcal{O}}$.*

Proof. Based on the definitions of the space of $\mathcal{X}$ in Section 4.1 and the temporal observation of T in Section 4.2, the component transitions in T are constrained by the configurations of the links in $\mathcal{X}$ and by the array of the temporal observations in $\mathcal{O}$. Since $\bar{T}$ is the projection of T on $\bar{\mathcal{U}}$, in particular, $\bar{T}$ is constrained by the configurations of the links in $\bar{\mathcal{U}}$ and by the array of the temporal observations in $\bar{\mathcal{O}}$, where $\bar{\mathcal{O}}$ is the array obtained by removing from $\mathcal{O}$ the temporal observation relevant to the root AU of $\mathcal{X}$. That is, $\bar{T} \in \|\bar{\mathcal{U}}^*\|$ and $Obs(\bar{T}, \bar{\mathcal{V}}) = \bar{\mathcal{O}}$, where $\bar{\mathcal{V}}$ is the array obtained by removing from $\mathcal{V}$ the viewer relevant to the root AU of $\mathcal{X}$. □

Proposition 5 *Let $\mathcal{U}_\xi^*$ be a constrained unit space, where $\xi = (\bar{u}_0, \bar{\mathcal{V}}, \bar{\mathcal{O}}.\bar{\mathcal{R}}, \mathbf{I})$. Then,*

$$\Delta(\mathcal{U}_\xi^*) = \Delta\left(Int\left(\mathcal{U}_\xi^*\right)\right). \tag{39}$$

Proof. Grounded on Lemma 5.1 and Lemma 5.2.

Lemma 5.1 *If $\delta \in \Delta(\mathcal{U}_\xi^*)$ then $\delta \in \Delta\left(Int\left(\mathcal{U}_\xi^*\right)\right)$.*

Proof. According to Definition 27, there is a path $p \in \|\mathcal{U}_\xi^*\|$ such that $\delta \in \Delta(p)$,

$$p = \left[\langle w_0, \sigma_1, w_1\rangle, \langle w_1, \sigma_2, w_2\rangle, \ldots, \left\langle w_{q-1}, \sigma_q, w_q\right\rangle\right] \tag{40}$$

31. $\nabla(i)$ is defined in eqn. (29).

where, according to eqn. (34), $\Delta(p) = \Delta(\boldsymbol{\Delta}) \otimes \nabla(w_q)$, with $\boldsymbol{\Delta}$ being the sequence of the diagnosis sets Δ relevant to the elements in $\Sigma(p) = [\,\sigma_1, \ldots, \sigma_q\,]$, and $\nabla(w_q)$ is the hidden set of w_q defined in eqn. (28). Let

$$\Sigma(p) = \Sigma_1 \cup [\sigma'_1] \cup \Sigma_2 \cup [\sigma'_2] \cup \ldots \cup \Sigma_{q'} \cup [\sigma'_{q'}] \cup \Sigma_{q'+1} \tag{41}$$

such that $[\,\sigma'_1, \ldots, \sigma'_{q'}\,]$ is $[\,\sigma \mid \sigma \in \Sigma(p), \sigma = (t(c), \mathbb{E}, \Delta), \mathbb{E} \neq \emptyset\,] = [\,(t_1(c_1), \mathbb{E}_1, \Delta_1), \ldots, (t_{q'}(c_{q'}), \mathbb{E}_{q'}, \Delta_{q'})\,]$. Hence, based on the definitions of a fragment (Section 6.3) and an interface (Section 6.4), there is a path $p' \in \|Int(\mathcal{U}^*_\xi)\|$,

$$p' = \Big[\langle i_0, (\mathbb{E}'_1, \Delta'_1), i_1\rangle, \langle i_1, (\mathbb{E}'_2, \Delta'_2), i_2\rangle, \ldots \Big\langle i_{q'-1}, (\mathbb{E}'_{q'}, \Delta'_{q'}), i_{q'}\Big\rangle\Big] \tag{42}$$

such that $\forall\, j \in [1\,..\,q']$, $\mathbb{E}'_j = \mathbb{E}_j$, where $\mathbb{E}_j$ is the (nonempty) set of emergent events involved in σ'_j. Moreover, since each state i_j of p', $j \in [1\,..\,q']$, includes (at least) one fragment (Section 6.3) φ of $\mathcal{U}^*_\xi$ in which each state is marked with the set of diagnoses relevant to all the subpaths of p starting in the initial state of φ, we also have

$$\forall\, j \in [1\,..\,q'] \left(\Delta'_j \supseteq \Delta(\boldsymbol{\Delta}_j) \otimes \Delta_j\right) \tag{43}$$

where $\boldsymbol{\Delta}_j = [\,\Delta \mid \Delta \text{ is involved in } \sigma, \sigma \in \Sigma_j\,]$, while Δ_j is relevant to $\sigma'_j = (t_j(c_j), \mathbb{E}_j, \Delta_j)$. Finally, we have

$$\nabla(i_{q'}) \supseteq \Delta(\boldsymbol{\Delta}_{q'+1}) \otimes \nabla(w_q). \tag{44}$$

In fact, on the one hand, $\Sigma_{q'+1}$ is relevant to a subpath in a fragment in $i_{q'}$, up to the state (w_q, Δ_q). On the other, based to eqn. (29), $\nabla(i_{q'}) \supseteq \Delta_q \otimes \nabla(w_q)$. Hence, based to Definition 28, eqn. (43), and eqn. (44), we have $\delta \in \Delta(p')$ and, hence, based on eqn. (38), $\delta \in \Delta\left(Int\left(\mathcal{U}^*_\xi\right)\right)$. □

Lemma 5.2 *If* $\delta \in \Delta\left(Int\left(\mathcal{U}^*_\xi\right)\right)$, *then* $\delta \in \Delta(\mathcal{U}^*_\xi)$.

Proof. Based on Definition 28, there is a path $p \in \|Int(\mathcal{U}^*_\xi)\|$ such that $\delta \in \Delta(p)$,

$$p = \big[\langle i_0, (\mathbb{E}_1, \Delta_1), i_1\rangle, \langle i_1, (\mathbb{E}_2, \Delta_2), i_2\rangle, \ldots, \big\langle i_{q-1}, (\mathbb{E}_q, \Delta_q), i_q\big\rangle\big] \tag{45}$$

where, according to eqn. (37), $\Delta(p) = \Delta(\boldsymbol{\Delta}) \otimes \nabla(i_q)$, with $\boldsymbol{\Delta} = [\,\Delta_1, \ldots, \Delta_q\,]$, while $\nabla(i_q)$ is the hidden set of i_q defined in eqn. (29). Based on the definitions of a fragment (Section 6.3) and an interface (Section 6.4), there is a path $p' \in \|\,\mathcal{U}^*_\xi\,\|$,

$$p' = \big[\langle w_0, \sigma_1, w_1\rangle, \langle w_1, \sigma_2, w_2\rangle, \ldots \big\langle w_{q'-1}, \sigma_{q'}, w_{q'}\big\rangle\big] \tag{46}$$

such that $\Sigma(p') = \Sigma_1 \cup [\sigma'_1] \cup \Sigma_2 \cup [\sigma'_2] \cup \ldots \cup \Sigma_q \cup [\sigma'_q] \cup \Sigma_{q+1}$, where $[\,\sigma'_1, \ldots, \sigma'_q\,]$ is $[\,\sigma' \mid \sigma' \in \Sigma(p'), \sigma' = (t'(c), \mathbb{E}', \Delta'), \mathbb{E}' \neq \emptyset\,] = [\,(t'_1(c_1), \mathbb{E}'_1, \Delta'_1), \ldots, (t'_q(c_q), \mathbb{E}'_q, \Delta'_q)\,]$ such that $\forall\, j \in [1\,..\,q]$, $\mathbb{E}'_j = \mathbb{E}_j$, with $\mathbb{E}_j$ being the (nonempty) set of the emergent events involved in the j-th transition of the interface path p in eqn. (45). Moreover, since each state i_j of the interface path p, $j \in [1\,..\,q]$, includes (at least) one fragment φ of $\mathcal{U}^*_\xi$ in which each state is marked with the set of diagnoses relevant to all the subpaths of constrained unit-space path p' starting in the initial state of φ, we also have

$$\forall\, j \in [1\,..\,q] \left(\Delta_j \supseteq \Delta(\boldsymbol{\Delta}_j) \otimes \Delta'_j\right) \tag{47}$$

where $\boldsymbol{\Delta}_j = [\, \Delta \mid \Delta \text{ is involved in } \sigma, \sigma \in \Sigma_j \,]$, while Δ'_j is relevant to $\sigma'_j = (t'_j(c_j), \mathbb{E}'_j, \Delta'_j)$. Finally, we have

$$\nabla(i_q) \supseteq \Delta(\boldsymbol{\Delta}_{q+1}) \otimes \nabla(w_{q'}). \tag{48}$$

In fact, on the one hand, Σ_{q+1} is relevant to a subpath within a fragment in i_q, up to the state $(w_{q'}, \Delta_{q'})$. On the other, according to eqn. (29), $\nabla(i_q) \supseteq \Delta_{q'} \otimes \nabla(w_{q'})$. In conclusion, according to Definition 28 and based on eqn. (47) and eqn. (48), we have $\delta \in \Delta(p')$ and, hence, based on eqn. (35), $\delta \in \Delta(\mathcal{U}^*_\xi)$. □

Theorem 1 *Let $\wp(\mathcal{X}) = (x_0, \mathcal{V}, \mathcal{O}, \mathcal{R})$ be a diagnosis problem for the DDES $\mathcal{X}$, $\mathcal{U}$ the root node in $\mathcal{X}$, and $\mathcal{U}^{\mathrm{d}}_\xi$ the decorated constrained space of $\mathcal{U}$ (defined in Section 6.6). We have*

$$\Delta(\wp(\mathcal{X})) = \Delta\left(\mathcal{U}^{\mathrm{d}}_\xi\right). \tag{49}$$

Proof. Grounded on Lemma 1.1, Lemma 1.2, Lemma 1.3, and Lemma 1.4.

Lemma 1.1 *Let $\wp(\mathcal{X}) = (x_0, \mathcal{V}, \mathcal{O}, \mathcal{R})$ be a diagnosis problem, $\mathcal{U}$ an AU in $\mathcal{X}$, $\bar{\mathcal{U}}$ the DDES rooted in $\mathcal{U}$, $\wp(\bar{\mathcal{U}}) = (\bar{u}_0, \bar{\mathcal{V}}, \bar{\mathcal{O}}, \bar{\mathcal{R}})$ the projection of $\wp(\mathcal{X})$ on $\bar{\mathcal{U}}$, and $\xi = (\bar{u}_0, \bar{\mathcal{V}}, \bar{\mathcal{O}}, \bar{\mathcal{R}}, \mathbf{I})$ an interface constraint for $\mathcal{U}$. If $\delta \in \Delta(\wp(\bar{\mathcal{U}}))$, with $\delta = Dgn(T, \bar{\mathcal{R}})$, then there is a path $p \in \|\mathcal{U}^*_\xi\|$ such that: (1) $\delta \in \Delta(p)$, and (2) the sequence $[\, t(c) \mid (t(c), \mathbb{E}, \Delta) \in \Sigma(p) \,]$ equals the subsequence of T involving only transitions of components in $\mathcal{U}$.*

Proof. By induction on the tree of AUs in $\mathcal{X}$.

(*Basis*) If $\mathcal{U}$ is a leaf node in $\mathcal{X}$ then $\mathcal{U}^*_\xi$ is such that $\xi = (\bar{u}_0, \bar{\mathcal{V}}, \bar{\mathcal{O}}, \bar{\mathcal{R}}, \mathbf{I})$ where $\mathbf{I}$ is empty (no interface constraints). Hence, $\mathcal{U}^*_\xi$ is constrained by $\bar{\mathcal{O}}$ only, in other words, $\mathcal{U}^*_\xi = \bar{\mathcal{U}}^*_{\bar{\mathcal{O}}}$. Thus, if $\delta \in \Delta(\wp(\bar{\mathcal{U}}))$, where $\delta = Dgn(T, \bar{\mathcal{R}})$, $T = [\, t_1(c_1), \ldots, t_q(c_q) \,]$, then there will be a path $p \in \|\bar{\mathcal{U}}^*_{\bar{\mathcal{O}}}\|$,

$$p = \left[\langle w_0, (t_1(c_1), \mathbb{E}_1, \Delta_1), w_1\rangle, \langle w_1, (t_2(c_2), \mathbb{E}_2, \Delta_2), w_2\rangle, \ldots, \left\langle w_{q-1}, (t_q(c_q), \mathbb{E}_q, \Delta_q), w_q\right\rangle\right] \tag{50}$$

such that

$$\forall i \in [1\,..\,q] \left(\Delta_i = \begin{cases} \{\{f_i\}\} & \text{if } (t_i(c_i), f_i) \in \bar{\mathcal{R}} \\ \{\emptyset\} & \text{otherwise} \end{cases} \right). \tag{51}$$

According to eqn. (34) and eqn. (28), $\Delta(p)$ is $\Delta(\boldsymbol{\Delta}) \otimes \nabla(w_q) = \Delta_1 \otimes \Delta_2 \otimes \ldots \otimes \Delta_q \otimes \{\emptyset\} = \{\delta\}$. In other words, $\delta \in \Delta(p)$ (condition 1 of the lemma). Condition 2 of the lemma is grounded on eqn. (50), namely $[\, t(c) \mid (t(c), \mathbb{E}, \Delta) \in \Sigma(p)] = [\, t_1(c_1), \ldots, t_q(c_q) \,]$.

(*Induction*) Assume that $\mathcal{U}$ is an internal node of $\mathcal{X}$ with children $\mathcal{U}_1, \ldots, \mathcal{U}_k$, such that, $\forall\, j \in [1\,..\,k]$, if $\delta_j \in \Delta(\wp(\bar{\mathcal{U}}_j))$, $\delta_j = Dgn(T_j, \bar{\mathcal{R}}_j)$, then $p_j \in \|\mathcal{U}^*_{j\,\xi_j}\|$ such that: (1) $\delta_j \in \Delta(p_j)$, and (2) the sequence $\left[\, t(c) \mid (t(c), \mathbb{E}, \Delta) \in \Sigma(p_j) \,\right]$ equals the subsequence of trajectory T_j involving only transitions of components in AU $\mathcal{U}_j$.

Assume that $\delta \in \Delta(\wp(\bar{\mathcal{U}}))$, where $\delta = Dgn(T, \bar{\mathcal{R}})$, with T being a trajectory within the space $\bar{\mathcal{U}}^*$. Let T_j, $j \in [1\,..\,k]$, denote the subsequence of T including the transitions relevant to the components within the DDES $\bar{\mathcal{U}}_j$. Based on Proposition 4, T_j is a trajectory in $\bar{\mathcal{U}}^*_j$ such that $Obs(T_j, \bar{\mathcal{V}}_j) = \bar{\mathcal{O}}_j$, where $\bar{\mathcal{O}}_j$ is the temporal observation relevant to $\wp(\bar{\mathcal{U}}_j)$. Consequently, according to eqn. (19), T_j is also a trajectory involved in the solution to $\wp(\bar{\mathcal{U}}_j)$, in other words,

$\delta_j \in \Delta(\wp(\bar{\mathcal{U}}_j))$, where $\delta_j = Dgn(T_j, \bar{\mathcal{R}}_j)$, with $\bar{\mathcal{R}}_j$ being the ruler relevant to $\wp(\bar{\mathcal{U}}_j)$. Hence, by the induction hypothesis, there is a path $p_j \in \|\mathcal{U}^*_{j\,\xi_j}\|$ such that: (1) $\delta_j \in \Delta(p_j)$, and (2) $T_{j\,\mathcal{U}_j} = \big[\, t(c) \mid (t(c), \mathbb{E}, \Delta) \in \Sigma(p_j) \,\big]$ equals the subsequence of trajectory T_j involving transitions of components in $\mathcal{U}_j$ only. Based on Proposition 5, there is a path $p'_j \in \|Int(\mathcal{U}^*_{j\,\xi_j})\|$ such that $\delta_j \in \Delta(p'_j)$, where

$$p'_j = \Big[\big\langle i_0, (\mathbb{E}_{1_j}, \Delta_{1_j}), i_1\big\rangle, \big\langle i_1, (\mathbb{E}_{2_j}, \Delta_{2_j}), i_2\big\rangle, \ldots, \Big\langle q'_{j-1}, (\mathbb{E}_{q'_j}, \Delta_{q'_j}), i_{q'_j}\Big\rangle\Big] \tag{52}$$

and, according to eqn. (34), $\Delta(p'_j)$ equals

$$\Delta(\boldsymbol{\Delta}) \otimes \nabla(i_{q'_j}) = \Delta_{1_j} \otimes \Delta_{2_j} \otimes \ldots \otimes \Delta_{q'_j} \otimes \nabla(i_{q'_j}). \tag{53}$$

Moreover, $\forall\, j' \in [1\,..\,q']$, we have $\mathbb{E}_{j'_j} \neq \emptyset$, where $\mathbb{E}_{j'_j}$ is the set of emergent events generated by a corresponding transition in $T_{j\,\mathcal{U}_j}$. Let $T^e_j = [\,t^e_{1_j}(c_{1_j}) \ldots, t^e_{q'_j}(c_{q'_j})\,]$ denote the subsequence of $T_{j\,\mathcal{U}_j}$ involving the transitions generating emergent events. Since all transitions in T_j are included in T, $j \in [1\,..\,k]$, T includes all transitions in T^e_j as well. Let T^e denote the subsequence of T involving the transitions in all T^e_j, $j \in [1\,..\,k]$. As such, the transitions in T^e are constrained by the temporal observations in $\bar{\mathcal{O}}$ relevant to the AUs $\mathcal{U}_1, \ldots, \mathcal{U}_k$, respectively, and by the load configuration of the links entering the input terminals of $\mathcal{U}$. Since the generation of $\mathcal{U}^*_\xi$ too is based on these two constraints, there is a path $p \in \|\mathcal{U}^*_\xi\|$ involving all the diagnosis sets marking the transitions in all p'_j, $j \in [1\,..\,k]$, namely $\Delta_{1_1}, \ldots, \Delta_{q'_1}, \Delta_{1_2}, \ldots, \Delta_{q'_2}, \ldots, \Delta_{1_k}, \ldots, \Delta_{q'_k}$,

$$p = \big[\langle w_0, \sigma_1, w_1\rangle, \langle w_1, \sigma_2, w_2\rangle, \ldots, \big\langle w_{q-1}, \sigma_q, w_q\big\rangle\big]. \tag{54}$$

Let $T_{\mathcal{U}}$ denote the subsequence of T composed of the transitions relevant to the components in $\mathcal{U}$ only. As such, the transitions in $T_{\mathcal{U}}$ are constrained by the temporal observation in $\bar{\mathcal{O}}$ relevant to the AU $\mathcal{U}$, and by the load configuration of the links entering the input terminals of $\mathcal{U}$. Since the generation of $\mathcal{U}^*_\xi$ too is based on these two constraints, the path $p \in \|\mathcal{U}^*_\xi\|$ in eqn. (54) involves all transitions in $T_{\mathcal{U}}$. Finally, according to eqn. (34),

$$\Delta(p) = \Delta(\boldsymbol{\Delta}) \otimes \nabla(i_q) = \Delta_1 \otimes \ldots \otimes \Delta_q \otimes \nabla(w_q) \tag{55}$$

where $\Delta_1, \ldots, \Delta_q$ are the diagnosis sets involved in $\sigma_1, \ldots, \sigma_q$, respectively. Still, comparing eqn. (55) to eqn. (53), with $j \in [1\,..\,k]$, in order for $\delta \in \Delta(p)$, we need to show that $\nabla(w_q) = \nabla(i_{q'_1}) \otimes \nabla(i_{q'_2}) \otimes \ldots \otimes \nabla(i_{q'_k})$. Based on eqn. (28), as $\mathcal{U}$ is assumed not to be a leaf node, we have

$$\nabla(w_q) = \bigotimes_{i \in \mathbb{I}} \nabla(i) \tag{56}$$

where, according to Section 6.2, $w_q = (\mathbb{S}, \mathbb{E}, \mathbb{M}, \mathbb{H}, \mathbb{I}, \Im)$, with $\mathbb{I} = (i_1, \ldots, i_k)$ being the array of final states in the interfaces $Int(\mathcal{U}^*_{1\,\xi_1}), \ldots, Int(\mathcal{U}^*_{k\,\xi_k})$, respectively. The conclusion of the proof of the induction step, as well as of Lemma 1.1, lies in that each state $i_j \in \mathbb{I}$, $j \in [1\,..\,k]$, is in fact the final state $i_{q'_j}$ of the path p'_j in eqn. (52). □

Lemma 1.2 (*Completeness*) *Let $\mathcal{U}$ be the root node of $\mathcal{X}$. If $\delta \in \Delta(\wp(\mathcal{X}))$, then $\delta \in \Delta\left(\mathcal{U}^{\mathrm{d}}_\xi\right)$.*

Proof. According to eqn. (30), the diagnosis set of the decorated constrained space $\mathcal{U}_\xi^{\mathrm{d}}$ is

$$\Delta\left(\mathcal{U}_\xi^{\mathrm{d}}\right) = \bigcup_{w_{\mathrm{f}} \in W_{\mathrm{f}}} \left(\Delta(w_{\mathrm{f}}) \otimes \nabla(w_{\mathrm{f}})\right) \tag{57}$$

where W_{f} is the set of final states of $\mathcal{U}_\xi^*$, $\Delta(w_{\mathrm{f}})$ the diagnosis set (decoration) of w_{f}, and $\nabla(w_{\mathrm{f}})$ the hidden set of w_{f} defined in eqn. (28). Specifically, each state w in $\mathcal{U}_\xi^*$ is decorated based on the following two rules: (1) if w_0 is the initial state, then $\emptyset \in \Delta(w_0)$, and (2) if $\langle w, \sigma, w' \rangle$ is a transition in $\mathcal{U}_\xi^*$, with $\bar{\Delta}$ being the diagnosis set involved in σ, then $\Delta(w') \supseteq \left(\Delta(w) \otimes \bar{\Delta}\right)$. From Lemma 1.1, if $\delta \in \Delta(\wp(\mathcal{X}))$, then there is $p \in \|\mathcal{U}_\xi^*\|$ such that $\delta \in \Delta(p)$, where

$$\Delta(p) = \Delta(\boldsymbol{\Delta}) \otimes \nabla(w_q) \tag{58}$$

where $\boldsymbol{\Delta} = [\,\Delta_1, \ldots, \Delta_q\,]$ is the sequence of diagnosis sets marking the transitions in p and w_q is the final state of p. By induction on p we show that, for each prefix $\hat{p}$ of p ending at state $\hat{w}$, if $\hat{\delta} \in \Delta(\hat{\boldsymbol{\Delta}})$, then $\hat{\delta} \in \Delta(\hat{w})$, where $\hat{\boldsymbol{\Delta}}$ is the sequence of diagnosis sets marking the transitions in $\hat{p}$.

(*Basis*) If $\hat{w} = w_0$ then, according to eqn. (32), $\Delta(\hat{\boldsymbol{\Delta}}) = \{\emptyset\}$, where $\hat{\boldsymbol{\Delta}} = [\,]$. On the other hand, based on the first decoration rule, $\emptyset \in \Delta(w_0)$.

(*Induction*) Assume that, for a prefix $\hat{p}$ of p, with $\hat{p}$ ending at the state $\hat{w}$, if $\hat{\delta} \in \Delta(\hat{\boldsymbol{\Delta}})$, then $\hat{\delta} \in \Delta(\hat{w})$. Let $\hat{p}'$ be the prefix of p obtained by extending $\hat{p}$ with the transition exiting $\hat{w}$ in p, namely $\langle \hat{w}, \sigma', \hat{w}' \rangle$, with Δ' being the diagnosis set involved in σ'. Let $\hat{\delta}'$ be a diagnosis in $\Delta(\hat{\boldsymbol{\Delta}}')$, where $\hat{\boldsymbol{\Delta}}' = \hat{\boldsymbol{\Delta}} \cup [\Delta']$. As such, $\hat{\delta}'$ is obtained by extending $\hat{\delta}$ by a diagnosis $\delta' \in \Delta'$. On the other hand, according to the second decoration rule, the same applies for $\Delta(\hat{w}')$, based on that $\Delta(\hat{w}') \supseteq \left(\Delta(\hat{w}) \otimes \Delta'\right)$. Considering the final state w_q of p, based on eqn. (58), δ is obtained by joining a diagnosis $\hat{\delta}_q \in \Delta(\boldsymbol{\Delta})$ with a diagnosis in the hidden set $\nabla(w_q)$, which is exactly the way a diagnosis in $\Delta(\mathcal{U}_\xi^*)$ is obtained based on eqn. (57). □

Lemma 1.3 *Let $\wp(\mathcal{X}) = (x_0, \mathcal{V}, \mathcal{O}, \mathcal{R})$ be a diagnosis problem, $\mathcal{U}$ an AU in $\mathcal{X}$, $\bar{\mathcal{U}}$ the DDES rooted in $\mathcal{U}$, $\wp(\bar{\mathcal{U}}) = (\bar{u}_0, \bar{\mathcal{V}}, \bar{\mathcal{O}}, \bar{\mathcal{R}})$ the projection of $\wp(\mathcal{X})$ on $\bar{\mathcal{U}}$, and $\xi = (\bar{u}_0, \bar{\mathcal{V}}, \bar{\mathcal{O}}, \bar{\mathcal{R}}, \mathbf{I})$ an interface constraint for $\mathcal{U}$. If $\delta \in \Delta(\mathcal{U}_\xi^*)$, where $\delta \in \Delta(p)$, $p \in \|\mathcal{U}_\xi^*\|$, then: (1) $\delta \in \Delta(\wp(\bar{\mathcal{U}}))$, where $\delta = Dgn(T, \bar{\mathcal{R}})$, and (2) the sequence $[\,t(c) \mid (t(c), \mathbb{E}, \Delta) \in \Sigma(p)\,]$ equals the subsequence of T involving only transitions of components in $\mathcal{U}$.*

Proof. By induction on the tree of AUs in $\mathcal{X}$.

(*Basis*) If $\mathcal{U}$ is a leaf node in $\mathcal{X}$ then $\mathcal{U}_\xi^*$ is such that $\xi = (\bar{u}_0, \bar{\mathcal{V}}, \bar{\mathcal{O}}, \bar{\mathcal{R}}, \mathbf{I})$ where $\mathbf{I}$ is empty (no interface constraints). Hence, $\mathcal{U}_\xi^*$ is constrained by $\bar{\mathcal{O}}$ only, in other words, $\mathcal{U}_\xi^* = \bar{\mathcal{U}}_{\bar{\mathcal{O}}}^*$. Thus, path $p \in \|\mathcal{U}_\xi^*\|$ relevant to δ is such that

$$p = \left[\langle w_0, (t_1(c_1), \mathbb{E}_1, \Delta_1), w1\rangle, \langle w_1, (t_2(c_2), \mathbb{E}_2, \Delta_2), w2\rangle, \ldots, \langle w_{q-1}, (t_q(c_q), \mathbb{E}_q, \Delta_q), w_q\rangle\right] \tag{59}$$

where

$$\forall i \in [1\,..\,q]\left(\Delta_i = \begin{cases} \{\{f_i\}\} & \text{if } (t_i(c_i), f_i) \in \bar{\mathcal{R}} \\ \{\emptyset\} & \text{otherwise} \end{cases}\right). \tag{60}$$

According to eqn. (34) and eqn. (28), $\Delta(p)$ is $\Delta(\boldsymbol{\Delta})\otimes\nabla(w_q) = \Delta_1\otimes\Delta_2\otimes\ldots\otimes\Delta_q\otimes\{\emptyset\} = \{\delta\}$. On the other hand, this also means that there is a trajectory $T \in \|\bar{\mathcal{U}}^*\|$ such that $Obs(T, \bar{\mathcal{V}}) = \bar{\mathcal{O}}$ and $Dgn(T, \bar{\mathcal{R}}) = \delta$. In other words, based on eqn. (19), $\delta \in \Delta(\wp(\bar{\mathcal{U}}))$.

(*Induction*) Assume that $\mathcal{U}$ is an internal node of $\mathcal{X}$ with children $\mathcal{U}_1, \ldots, \mathcal{U}_k$, such that, $\forall\, j \in [1\,..\,k]$, if $\delta_j \in \Delta(\mathcal{U}^*_{j\,\xi_j})$, where $\delta_j \in \Delta(p_j)$, $p_j \in \|\mathcal{U}^*_{j\,\xi_j}\|$, then: (1) $\delta_j \in \Delta(\wp(\bar{\mathcal{U}}_j))$, where $\delta_j = Dgn(T_j, \bar{\mathcal{R}}_j)$, with $\bar{\mathcal{R}}_j$ being the ruler of $\wp(\bar{\mathcal{U}}_j)$, and (2) the sequence $\big[\, t(c) \mid (t(c), \mathbb{E}, \Delta) \in \Sigma(p_j) \,\big]$ equals the subsequence of T_j involving only transitions of components in AU $\mathcal{U}_j$. Assuming $\delta \in \Delta(\mathcal{U}^*_\xi)$, specifically $\delta \in \Delta(p)$, $p \in \|\mathcal{U}^*_\xi\|$, we have

$$p = \big[\langle w_0, \sigma_1, w_1\rangle, \langle w_1, \sigma_2, w_2\rangle, \ldots, \langle w_{q-1}, \sigma_q, w_q\rangle\big] \tag{61}$$

such that, based on eqn. (34), $\Delta(p) = \Delta(\boldsymbol{\Delta}) \otimes \nabla(w_q)$, where $\boldsymbol{\Delta} = [\,\Delta_1, \ldots, \Delta_q\,]$ is the sequence of the diagnosis sets involved in $\sigma_1, \ldots, \sigma_q$, respectively, while, based on eqn. (28), the hidden set of $w_q = (\mathbb{S}, \mathbb{E}, \mathbb{M}, \mathbb{H}, \mathbb{I}, \Im)$, where $\mathbb{I} = (i_1, \ldots, i_k)$, is $\nabla(w_q) = \nabla(i_1) \otimes \nabla(i_2) \otimes \ldots \otimes \nabla(i_k)$. In summary, $\delta \in \Delta(p)$, where $\Delta(p) = \Delta_1 \otimes \ldots \otimes \Delta_q \otimes \nabla(i_1) \otimes \nabla(i_2) \otimes \ldots \otimes \nabla(i_k)$. So, $\Delta(p)$ can be expressed as the join of $k+1$ join expressions $\Delta(p) = \Delta' \otimes \Delta'_1 \otimes \Delta'_2 \otimes \ldots \otimes \Delta'_k$, where

$$\Delta' = \{\delta'\} = \bigotimes_{(t(c),\mathbb{E},\Delta)\in\Sigma(p)} \Delta \tag{62}$$

and, $\forall\, j \in [1\,..\,k]$, we have

$$\Delta'_j = \left(\bigotimes_{\sigma\in\Sigma(p),\sigma=(\mathbb{E},\Delta),\sigma \text{ relevant to } Int\left(\mathcal{U}^*_{j\,\xi_j}\right)} \Delta \right) \otimes \nabla(i_j). \tag{63}$$

Hence, δ can be expressed as the union of $k+1$ sets of faults, $\delta = \delta' \cup \delta'_1 \cup \delta'_2 \cup \ldots \cup \delta'_k$, where $\delta' \in \Delta' = \{\delta'\}$, while $\delta'_j \in \Delta'_j$, $j \in [1\,..\,k]$. Moreover, based on the definition of constrained unit space (Section 6.2), $\forall\, j \in [1\,..\,k]$, there is a path $p'_j \in \|Int(\mathcal{U}^*_{j\,\xi_j})\|$ such that $\delta'_j \in \Delta(p'_j)$. According to Proposition 5, there is a path $p_j \in \|\mathcal{U}^*_{j\,\xi_j}\|$ such that $\delta'_j \in \Delta(p_j)$, in other words, $\delta'_j \in \Delta(\mathcal{U}^*_{j\,\xi_j})$. Hence, by point 1 of the induction step, $\delta'_j \in \Delta(\wp(\bar{\mathcal{U}}_j))$. According to eqn. (19), δ'_j is the diagnosis of a trajectory $T_j \in \|\bar{\mathcal{U}}^*_j\|$ based on the ruler $\bar{\mathcal{R}}_j$, such that $Obs(T_j, \bar{\mathcal{V}}_j) = \bar{\mathcal{O}}_j$, with $\bar{\mathcal{O}}_j$ being the temporal observation and $\bar{\mathcal{V}}_j$ the viewer in $\wp(\bar{\mathcal{U}}_j)$. In order for T_j to be a subsequence of a trajectory $T \in \|\bar{\mathcal{U}}^*_{\bar{\mathcal{O}}}\|$, the constraints on load configuration of links entering the input terminals of $\mathcal{U}$ should be respected.[32] Since these constraints are in fact respected in the generation of $\mathcal{U}^*_\xi$ too, all the transitions of the trajectory T_j are interspersed (with same relative order) within T. Besides, based on point 2 of the induction step, the elements $(\mathbb{E}, \Delta)$ in $\Sigma(p)$ are relevant to the transitions of the components in $\mathcal{U}_j$, where $\mathbb{E} \neq \emptyset$ is the set of emergent events in $\mathcal{U}_j$, $j \in [1\,..\,k]$. Consequently, the mode in which emergent events are generated in $\mathcal{U}_j$ and consumed in the construction of $\mathcal{U}^*_\xi$ is the same as in T. In other words, the sequence of transitions $t(c)$ involved in the triples of eqn. (62) is also the projection of T on the transitions of the components in $\mathcal{U}$. In summary, we have

$$Dgn(T, \bar{\mathcal{R}}) = \delta' \cup \delta'_1 \cup \delta'_2 \cup \ldots \cup \delta'_k = \delta. \tag{64}$$

32. Specifically, no transition in $\mathcal{U}_j$ generating an emergent event on a nonempty terminal can be triggered.

Hence, the first condition of Lemma 1.3 holds. Finally, the second condition of Lemma 1.3 is grounded on eqn. (62). □

Lemma 1.4 (*Soundness*) *Let $\mathcal{U}$ be the root node of $\mathcal{X}$. If $\delta \in \Delta\left(\mathcal{U}_\xi^{\mathrm{d}}\right)$, then $\delta \in \Delta(\wp(\mathcal{X}))$.*

Proof. According to Section 6.6 and eqn. (30), we have

$$\Delta(\mathcal{U}_\xi^{\mathrm{d}}) = \bigcup_{w_{\mathrm{f}} \in W_{\mathrm{f}}} \Big(\Delta(w_{\mathrm{f}}) \otimes \nabla(w_{\mathrm{f}}) \Big) \tag{65}$$

where W_{f} is the set of final states of $\mathcal{U}_\xi^*$, $\Delta(w_{\mathrm{f}})$ the diagnosis set of w_{f}, and $\nabla(w_{\mathrm{f}})$ the hidden set of w_{f} defined in eqn. (28). Specifically, each state w in $\mathcal{U}_\xi^*$ is decorated based on the following two rules: (1) if w_0 is the initial states of $\mathcal{U}_\xi^*$ then $\emptyset \in \Delta(w_0)$, and (2) if $\langle w, \sigma, w' \rangle$ is a transition in $\mathcal{U}_\xi^*$, with $\bar{\Delta}$ being the diagnosis set involved in σ, then $\Delta(w') \supseteq \left(\Delta(w) \otimes \bar{\Delta} \right)$. Based on these rules, if $\delta \in \Delta(\mathcal{U}_\xi^{\mathrm{d}})$, then there is a path $p \in \|\mathcal{U}_\xi^*\|$, namely

$$p = \big[\langle w_0, \sigma_1, w_1 \rangle, \langle w_1, \sigma_2, w_2 \rangle, \ldots, \langle w_{q-1}, \sigma_q, w_q \rangle \big] \tag{66}$$

where w_q is one of final states w_{f} in eqn. (65), such that $\delta \in \{\emptyset\} \otimes \Delta_1 \otimes \Delta_2 \otimes \ldots \otimes \Delta_q \otimes \nabla(w_q)$, where, $\forall i \in [1 \,..\, q]$, Δ_i is involved in σ_i, that is $\delta \in \Delta(\boldsymbol{\Delta}) \otimes \nabla(w_q)$, where $\boldsymbol{\Delta} = [\, \Delta_1, \ldots, \Delta_q \,]$. Hence, according to eqn. (34), $\delta \in \Delta(p)$. Also, based on Lemma 1.3, $\delta \in \Delta(\wp(\bar{\mathcal{U}}))$. Since $\mathcal{U}$ is the root node of $\mathcal{X}$, we have $\bar{\mathcal{U}} = \mathcal{X}$ and, therefore, $\delta \in \Delta(\wp(\mathcal{X}))$. □

Eventually, the proof of Theorem 1 comes from Lemma 1.2 and Lemma 1.4. □

References

Atay, F., & Jost, J. (2004). On the emergence of complex systems on the basis of the coordination of complex behaviors of their elements: synchronization and complexity. *Complexity*, *10*(1), 17–22.

Baroni, P., Lamperti, G., Pogliano, P., & Zanella, M. (1998). Diagnosis of active systems. In *Thirteenth European Conference on Artificial Intelligence (ECAI 1998)*, pp. 274–278, Brighton, United Kingdom.

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

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

Bossomaier, T., & Green, D. (2007). *Complex Systems*. Cambridge University Press, Cambridge.

Brand, D., & Zafiropulo, P. (1983). On communicating finite-state machines. *Journal of the ACM*, *30*(2), 323–342.

Cassandras, C., & Lafortune, S. (2008). *Introduction to Discrete Event Systems* (second edition). Springer, New York.

Chittaro, L., & Ranon, R. (2004). Hierarchical model-based diagnosis based on structural abstraction. *Artificial Intelligence*, *155*(1–2), 147–182.

Console, L., Picardi, C., & Dupré, D. T. (2003). Temporal decision trees: Model-based diagnosis of dynamic systems on-board. *Journal of Artificial Intelligence Research*, *19*, 469–512.

Darwiche, A. (1998). Model-based diagnosis using structured system descriptions. *Journal of Artificial Intelligence Research*, *8*, 165–222.

Darwiche, A., & Provan, G. (1996). Exploiting system structure in model-based diagnosis of discrete-event systems. In *7th International Workshop on Principles of Diagnosis (DX 1996)*, pp. 95–105, Montreal, CDN.

de Kleer, J., Mackworth, A. K., & Reiter, R. (1992). Characterizing diagnoses and systems. *Artificial Intelligence*, *56*(2-3), 197–222.

de Kleer, J., & Williams, B. (1987). Diagnosing multiple faults. *Artificial Intelligence*, *32*(1), 97–130.

Dechter, R. (2003). *Constraint Processing*. The Morgan Kaufmann Series in Artificial Intelligence. Morgan Kaufmann Publishers Inc., San Francisco, CA.

Goles, E., & Martinez, S. (Eds.). (2001). *Complex Systems*. Springer, Dordrecht, The Netherlands.

Hamscher, W., Console, L., & de Kleer, J. (Eds.). (1992). *Readings in Model-Based Diagnosis*. Morgan Kaufmann, San Mateo, CA.

Harel, D. (1987). Statecharts: a visual formalism for complex systems. *Science of Computer Programming*, *8*(3), 231–274.

Huang, C., & Darwiche, A. (1996). Inference in belief networks: A procedural guide. *International Journal of Approximate Reasoning*, *15*(3), 225–263.

Idghamishi, A., & Zad, S. (2004). Fault diagnosis in hierarchical discrete-event systems. In *43rd IEEE Conference on Decision and Control*, pp. 63–68, Paradise Island, Bahamas.

Jéron, T., Marchand, H., Pinchinat, S., & Cordier, M. (2006). Supervision patterns in discrete event systems diagnosis. In *Seventeenth International Workshop on Principles of Diagnosis (DX 2006)*, pp. 117–124, Peñaranda de Duero, Spain.

Jiang, S., Huang, Z., Chandra, V., & Kumar, R. (2001). A polynomial algorithm for testing diagnosability of discrete event systems. *IEEE Transactions on Automatic Control*, *46*(8), 1318–1321.

Kan John, P., & Grastien, A. (2008). Local consistency and junction tree for diagnosis of discrete-event systems. In *Eighteenth European Conference on Artificial Intelligence (ECAI 2008)*, pp. 209–213, Patras, Greece. IOS Press, Amsterdam.

Khalastchi, E., & Kalech, M. (2018). On fault detection and diagnosis in robotic systems. *ACM Comput. Surv.*, *51*(1).

Lamperti, G., & Quarenghi, G. (2016). Intelligent monitoring of complex discrete-event systems. In Czarnowski, I., Caballero, A., Howlett, R., & Jain, L. (Eds.), *Intelligent Decision Technologies 2016*, Vol. 56 of *Smart Innovation, Systems and Technologies*, pp. 215–229. Springer International Publishing Switzerland.

Lamperti, G., & Zanella, M. (2010). Injecting semantics into diagnosis of discrete-event systems. In *21st International Workshop on Principles of Diagnosis (DX 2010)*, pp. 233–240, Portland, OR.

Lamperti, G., & Zanella, M. (2011). Context-sensitive diagnosis of discrete-event systems. In Walsh, T. (Ed.), *Twenty-Second International Joint Conference on Artificial Intelligence (IJCAI 2011)*, Vol. 2, pp. 969–975, Barcelona, Spain. AAAI Press.

Lamperti, G., & Zanella, M. (2013). Preliminaries on complexity of diagnosis of discrete-event systems. In *24th International Workshop on Principles of Diagnosis (DX 2013)*, pp. 192–197, Jerusalem, Israel.

Lamperti, G., Zanella, M., & Zhao, X. (2018a). Abductive diagnosis of complex active systems with compiled knowledge. In Thielscher, M., Toni, F., & Wolter, F. (Eds.), *Principles of Knowledge Representation and Reasoning: Proceedings of the Sixteenth International Conference (KR 2018)*, pp. 464–473, Tempe, Arizona. AAAI Press.

Lamperti, G., Zanella, M., & Zhao, X. (2018b). *Introduction to Diagnosis of Active Systems*. Springer, Cham.

Lamperti, G., Zanella, M., & Zhao, X. (2018c). Knowledge compilation techniques for model-based diagnosis of complex active systems. In Holzinger, A., Kieseberg, P., Tjoa, A. M., & Weippl, E. (Eds.), *Machine Learning and Knowledge Extraction*, Vol. 11015 of *Lecture Notes in Computer Science*, pp. 43–64. Springer, Cham.

Lamperti, G., & Zhao, X. (2013a). Diagnosis of higher-order discrete-event systems. In Cuzzocrea, A., Kittl, C., Simos, D., Weippl, E., & Xu, L. (Eds.), *Availability, Reliability, and Security in Information Systems and HCI*, Vol. 8127 of *Lecture Notes in Computer Science*, pp. 162–177. Springer, Berlin, Heidelberg.

Lamperti, G., & Zhao, X. (2013b). Specification and model-based diagnosis of higher-order discrete-event systems. In *IEEE International Conference on Systems, Man, and Cybernetics (SMC 2013)*, pp. 2342–2347, Manchester, United Kingdom.

Lamperti, G., & Zhao, X. (2014). Diagnosis of active systems by semantic patterns. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *44*(8), 1028–1043.

Lamperti, G., & Zhao, X. (2016a). Diagnosis of complex active systems with uncertain temporal observations. In Buccafurri, F., Holzinger, A., Tjoa, A. M., & Weippl, E. (Eds.), *Availability, Reliability, and Security in Information Systems*, Vol. 9817 of *Lecture Notes in Computer Science*, pp. 45–62. Springer International Publishing AG Switzerland.

Lamperti, G., & Zhao, X. (2016b). Viable diagnosis of complex active systems. In *IEEE International Conference on Systems, Man, and Cybernetics (SMC 2016)*, pp. 457–462, Budapest.

Licata, I., & Sakaji, A. (2008). *Physics of Emergence and Organization*. World Scientific.

Mozetič, I. (1992). Hierarchical diagnosis. In Hamscher, W., Console, L., & de Kleer, J. (Eds.), *Readings in Model-Based Diagnosis*. Morgan Kaufmann.

Paoli, A., & Lafortune, S. (2008). Diagnosability analysis of a class of hierarchical state machines. *Journal of Discrete Event Dynamic Systems: Theory and Applications*, *18*(3), 385–413.

Pencolé, Y., Steinbauer, G., Mühlbacher, C., & Travé-Massuyès, L. (2018). Diagnosing discrete event systems using nominal models only. In Zanella, M., Pill, I., & Cimatti, A. (Eds.),

28th International Workshop on Principles of Diagnosis (DX'17), Vol. 4, pp. 169–183. Kalpa Publications in Computing.

Pill, I., & Quaritsch, T. (2013). Behavioral diagnosis of LTL specifications at operator level. In *Twenty-Third International Joint Conference on Artificial Intelligence (IJCAI 2013)*, pp. 1053–1059. AAAI Press.

Pnueli, A. (1977). The temporal logic of programs. In *8th Annual Symposium on Foundations of Computer Science*, SFCS '77, pp. 46–57, Washington, DC, USA. IEEE Computer Society.

Reiter, R. (1987). A theory of diagnosis from first principles. *Artificial Intelligence*, *32*(1), 57–95.

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

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

Schumann, A., & Huang, J. (2008). A scalable jointree algorithm for diagnosability. In *Twenty-Third National Conference on Artificial Intelligence (AAAI 2008)*, pp. 535–540, Chicago, IL.

Schuppan, V. (2012). Towards a notion of unsatisfiable and unrealizable cores for LTL. *Sci. Comput. Program.*, *77*(7–8), 908–939.

Shenoy, P., & Shafer, G. (1986). Propagating belief functions with local computations. *IEEE Expert*, *1*(3), 43–52.

Siddiqi, S., & Huang, J. (2007). Hierarchical diagnosis of multiple faults. In *20th International Joint Conference on Artificial Intelligence (IJCAI 2007)*, pp. 581–586, Hyderabad, India.

Siddiqi, S., & Huang, J. (2011). Sequential diagnosis by abstraction. *Journal of Artificial Intelligence Research*, *41*, 329–365.

Stumptner, M., & Wotawa, F. (2001). Diagnosing tree-structured systems. *Artificial Intelligence*, *127*(1), 1–29.

Su, R., & Wonham, W. (2005). Global and local consistencies in distributed fault diagnosis for discrete-event systems. *IEEE Transactions on Automatic Control*, *50*(12), 1923–1935.

Yoo, T., & Lafortune, S. (2002). Polynomial-time verification of diagnosability of partially observed discrete-event systems. *IEEE Transactions on Automatic Control*, *47*(9), 1491–1495.