Received: 20 May 2019 | Revised: 30 October 2019 | Accepted: 30 October 2019

DOI: 10.1002/eng2.12060

RESEARCH ARTICLE

WILEY

Twin-engined diagnosis of discrete-event systems

Nicola Bertoglio[1] | **Gianfranco Lamperti**[1] | **Marina Zanella**[1] | **Xiangfu Zhao**[2]

[1]Department of Information Engineering, University of Brescia, Brescia, Italy

[2]College of Mathematics, Physics and Information Engineering, Zhejiang Normal University, Jinhua, China

Correspondence
Gianfranco Lamperti, Department of Information Engineering, University of Brescia, Via Branze 38, 25123 Brescia, Italy.
Email: gianfranco.lamperti@unibs.it

Funding information
Smart4CPPS, Grant/Award Number: POR-FESR 2014-2020 Asse I; National Natural Science Foundation of China, Grant/Award Number: 61972360

Diagnosis of discrete-event systems (DESs) is computationally complex. This is why a variety of knowledge compilation techniques have been proposed, the most notable of them rely on a *diagnoser*. However, the construction of a diagnoser requires the generation of the whole system space, thereby making the approach impractical even for DESs of moderate size. To avoid total knowledge compilation while preserving efficiency, a twin-engined diagnosis technique is proposed in this paper, which is inspired by the two operational modes of the human mind. If the symptom of the DES is part of the knowledge or experience of the diagnosis engine, then *Engine 1* allows for efficient diagnosis. If, instead, the symptom is unknown, then *Engine 2* comes into play, which is far less efficient than *Engine 1*. Still, the experience acquired by *Engine 2* is then integrated into the *symptom dictionary* of the DES. This way, if the same diagnosis problem arises anew, then it will be solved by *Engine 1* in linear time. The symptom dictionary can also be extended by specialized knowledge coming from *scenarios*, which are the most critical/probable behavioral patterns of the DES, which need to be diagnosed quickly.

KEYWORDS

communicating automata, discrete-event systems, knowledge compilation, model-based diagnosis

1 | INTRODUCTION

According to Daniel Kahneman, psychologist and Nobel laureate in Economic Sciences, two modes of thinking coexist in the human brain, which correspond to two systems in the mind, called *System 1* and *System 2*.[1] *System 1* operates automatically and quickly, with little if any effort, and no sense of voluntary control, such as when orienting to the source of a sudden sound or driving a car in an empty road. By contrast, *System 2* operates consciously and slowly, with attention being focused on demanding mental activities, possibly including complex computations or inferences, such as when filling out an intricate application form or checking the validity of a complex argument. Intriguingly, an activity initially performed by *System 2*, such as driving a car or playing the piano, may be subsequently operated by *System 1* after appropriate training.

This "twin-engined" architecture of the mind is a metaphor for the diagnosis approach for discrete-event systems (DESs) presented in this paper. The proposed diagnosis engine (DE) operates in two different modes resembling *System 1* and *System 2* in the human mind, called *Engine 1* and *Engine 2*. If the diagnosis problem to be solved is part of the knowledge or experience of the DE, then *Engine 1* can solve this problem quickly. If, instead, the problem is not part of the knowledge or experience of the DE, then comes into play *Engine 2*, which requires deep model-based reasoning and, therefore, operates far more slowly than *Engine 1*. Still, the experience acquired by *Engine 2* in solving the diagnosis problem can be integrated into the knowledge of the DE, so that, in the future, the same diagnosis problem can be solved by *Engine 1* efficiently. Besides, the DE is not necessarily born naked, that is, without any knowledge except the model of the DES, otherwise *Engine 2* would operate far more frequently than *Engine 1* for a possibly long time. Instead, the DE can start working being already equipped with specialized knowledge on domain-dependent *scenarios* that are considered either most probable or most critical for the safety of the DES (or the surrounding environment) and, as such, need to be coped with efficiently.

2 | DIAGNOSIS AND THE MYTH OF TOTAL KNOWLEDGE COMPILATION

Diagnosis aims to explain the abnormal behavior of a system based on the observations relevant to its operation, which are perceived from an external observer. In the Artificial Intelligence community, the formal definition of the diagnosis task[2] led, in the '80s, to the *model-based* paradigm,[3] according to which the normal behavior of the system to be diagnosed is described by a model and the diagnosis results have to explain the discrepancies between the output of the system and what we were supposed to observe based on the system model. The diagnosis task generates a collection of sets of faulty components, where each set, named a *candidate*, is an explanation of the observation. Each candidate explains the observation in the sense that assuming all the components in the candidate are faulty and all the others are normal is consistent with the observation. This model-based diagnosis approach was initially conceived for static systems, such as combinational circuits, and then extended to dynamic systems.

Recently, another approach that relies on the exploitation of Artificial Intelligence techniques has been adopted, *data-driven* diagnosis.[4,5] In fact, machine learning can be used to automatically process sensor data and create models for fault prediction and classification. Typically, nominal data (ie, data relevant to the normal behavior) and data from different faults can be used to train fault classifiers. This relieves the diagnostician from the burden of creating a model of the system/process to be diagnosed, sometimes even from the burden of identifying proper features since deep learning techniques offer potential tools for automatic feature extraction from (physical) signals.[6] However, in many industrial applications, faults are rare events and available training data from faulty conditions is usually limited.[7] Hence, collecting a sufficient amount of data from relevant faulty situations for data-driven diagnosis is a time-consuming and expensive process.

Alternatively, the behavior of the system to be diagnosed can be described as a set of cases, where each case is labeled as either normal or faulty. The case base can include the diagnostician's expertise, starting from a limited amount of cases (that is, an incomplete knowledge), and its size can increase incrementally over time, as such an expertise is growing. The human diagnostician is assisted in decision making about diagnosis, maintenance and reconfiguration by the interactive case-based reasoning system, for instance in dealing with automated production systems.[8]

In some contexts, such as rotating machinery, which is among the most important equipments in modern industry, fault diagnosis can be regarded as a pattern recognition problem. Due to the variability and richness of the response signals relevant to the rotating machinery condition, it is almost impossible to recognize fault patterns directly: Artificial Intelligence techniques, encompassing a preprocessing step for feature extraction and an online step for fault recognition, are very promising.[9] Specifically, several methods have been used for performing the latter step, including convex optimization, mathematical optimization, as well as classification, statistical learning, and probability-based algorithms.

An advantage of model-based methods, with respect to data-driven methods, is that fault detection and isolation can be achieved without the need for data from different faults. In this paper, we deal with a model-based approach as applied to dynamical systems. For a dynamical system, a DES[10] model can be adopted. This model can be described as a Petri net,[11,12] or as a finite automaton, where each state of the automaton represents a state of the dynamical system (at some abstraction level), like in this paper. The model is typically distributed, consisting of several automata that communicate with one another.[13] Each automaton represents the behavior of a component, which is a DES itself, where some state transitions are faulty (and, as such, each of them is marked with a fault) and the occurrence of some transitions can be perceived from the outside of the system (and, as such, each of these transitions is marked with an observable event). Diagnosing a DES means finding out all the candidates relevant to a given sequence of observed events, where a candidate is a set of faults that can explain such a sequence.

Model-based diagnosis of DESs represented as networks of finite automata is grounded on the seminal work of Sampath et al,[14-16] which, as relying on a compiled-knowledge data structure called a diagnoser, has been coined the *diagnoser approach*. The knowledge embedded in the DES component models is totally compiled offline, thus producing the diagnoser, which is then exploited online to draw the candidates relevant to any given sequence of observed events. Two major requirements are assumed by the diagnosis method embodied in the diagnoser approach.

- To guarantee the soundness and completeness of the diagnosis results, the DES must have a property called *diagnosability*.
- To generate the diagnoser, the materialization of the whole behavior space of the DES is needed.

These requirements are strongly interrelated as the creation of the whole behavior space is needed not only for fulfilling the second requirement (as the diagnoser is drawn from the system behavioral space) but also for checking whether the first requirement is met (that is, for performing the diagnosability analysis). However, the materialization of the whole behavioral space is most critical since, even for DESs of moderate size, the behavior space may become overwhelmingly large, being exponential in the number of components. The memory demands for such a space are difficult to satisfy, and producing it is a computationally expensive and inefficient process (see, for instance, the work of Rozé[17] about the computational difficulties of the diagnoser approach, or the worst case computational complexity analysis in the work of Baroni et al,[18] or the discussions in the works of Basile[11] and Kurien and Nayak[19]). Nowadays, the diagnoser approach can still be regarded as a reference conceptual framework for the definition of both the task of diagnosis and the notion of diagnosability of DESs described as networks of finite automata. However, from an operational point of view, subsequent contributions in the literature about diagnosis and/or diagnosability of DESs described as networks of finite automata do not follow the original diagnoser approach method as the awareness has soon grown that total knowledge compilation is definitely a myth.

Notwithstanding the computational drawbacks of the diagnoser approach, the topic of diagnosability has unsurprisingly received considerable attention in the literature. In the work of Sampath et al,[14] the property of diagnosability is defined for (partially observable) DESs, whose prefix-closed language of the events is live and does not include any cycle of unobservable events. Roughly, a system is diagnosable if, given any occurrence of any failure event, it is possible to detect and isolate that failure within a finite number of events (ie, a finite number of transitions of the system model) following it. In other words, the system is diagnosable if it is always possible to detect within a finite delay the occurrence of a failure event (and uniquely identify its type) by using the record of observed events. A necessary and sufficient condition is proposed in the work of Sampath et al[14] to check diagnosability based on the diagnoser construction. However, testing diagnosability is performed in polynomial time in the cardinality of the state space of the diagnoser, which, in the worst case, is exponential in the cardinality of the state space of the system model. The most famous approach to checking whether the diagnosability property holds for a given DES is the *twin plant* method,[20] whose time complexity is polynomial in the global number of the states of the system model. In fact, this method relies on the construction of the twin plant, this being the synchronous product of a nondeterministic finite automaton (or NFA, drawn from the global model of the DES, and generating the DES observable language) by itself over the alphabet of observable events. A similar approach to diagnosability checking, whose complexity is still polynomial in the number of states of the global behavioral space, is presented in the work of Yoo and Lafortune.[21] Finding out whether the condition for diagnosability (as restated by the twin plant method) holds was formulated also as a model-checking problem[22,23] or a satisfiability problem.[24]

In the work of Console et al,[25] the analogy between modeling concurrent processes (for the purpose of performance evaluation) and modeling distributed physical systems (for the purpose of diagnosis and diagnosability analysis) is pointed out. The concepts of diagnosis and diagnosability are then characterized based on a process algebra formalism (PEPA).

Most of existing works focused on how to verify that the intrinsic diagnosability of a DES assume that candidates are computed by a diagnostic algorithm that takes as input a completely certain sequence of observable events. The ability to disambiguate DES candidates for a system with uncertain observations is discussed in the work of Su et al.[26] The uncertainty is measured by a parameter, which allows for studying the level of noise that can affect the observed sequence of events without impacting the performance of diagnosis. A definition of DES diagnosability that extends the original one[14] is provided, as well as a method to check whether the newly defined property holds for a given DES. This method extends the original twin plant method[20] by considering two kinds of uncertainty in the sequence of observable events.

Diagnosability of fuzzy DESs (FDESs), namely, fuzzy diagnosability, is faced in the work of Liu and Qiu.[27] The FDESs are DESs whose states and events, instead of being crisp, are fuzzy (according to the fuzzy set theory) to represent the vagueness, impreciseness, and subjectivity that are typical features in several real-world domains, such as biomedicine. Specifically, in the quoted paper, FDESs are modeled as max-min systems, and observability is assumed to be fuzzy. A fuzzy diagnosability function to characterize the diagnosability degree of a FDES is introduced, a necessary and sufficient condition for diagnosability of FDESs, which generalizes that for crisp DESs introduced in the work of Sampath et al,[14] is presented, and a method to check the fuzzy diagnosability of FDESs is given that is based on the diagnoser in the aforementioned work.[14]

In the work of Paoli and Lafortune,[28] the property of diagnosability for DESs tailored as hierarchical finite state machines (HFSMs) is introduced. According to this notion, a fault can be detected using only a high-level description of the HFSM, and diagnosability can be tested by relying on a so-called *extended diagnoser*. If some sufficient conditions hold, the extended diagnoser can be built from particular projections of the HFSM. For faults that cannot be proven to be diagnosable in an HFSM using the aforementioned projections, if another sufficient condition holds, a refined version of the diagnoser can be used. The whole approach is aimed at checking diagnosability while avoiding the construction of the behavioral model of the whole system, as instead required by both the diagnoser approach and the twin plant method.

In order to avoid the construction of the behavioral model of the whole system, another research line performs a distributed diagnosability analysis.[29,30] Basically, by taking advantage of the distributed modeling of the considered DES, a (small) local twin plant is built for each component, instead of building a (huge) twin plant relevant to the whole DES. Local twin plants are then synchronized with each other until diagnosability is decided. Unfortunately, the synchronization of local twin plants can remain a bottleneck for large systems. The scalability of the approach is increased in the work of Schumann and Huang,[31] where DES components (and their relevant local twin plants) are organized into a jointree, a classical tool adopted in various fields of Artificial Intelligence. Once the jointree has been constructed, only the twin plants in each jointree node need to be synchronized, and all the remaining computation takes the form of message passing along the edges of the jointree.

Fault detection in DESs was generalized[32] to the recognition of a pattern, which can represent the occurrence of a single fault as well as of multiple faults, the ordered occurrence of significant events, the multiple occurrences of the same fault, etc. Consequently, the notion of fault diagnosability was generalized to pattern diagnosability, and the task of checking pattern diagnosability was faced in a distributed way.[33]

Diagnosability analysis was addressed for DESs represented as labeled Petri nets also, both bounded[34,35] and unbounded.[36]

Coming back to the diagnosis task, as already remarked, solving a DES diagnosis problem instance means finding out all the sets of faults entailed by the system evolutions that produce the sequence of events that have been observed. Based on the method for tracking the evolutions of the system that explain a given sequence of observable events, two approaches to diagnosis of DESs were singled out in the work of Basile.[11]

- The former, coined *compiled diagnoser*, generates offline (a concise model of) all possible evolutions and then retrieves online only the evolutions that explain the observation; it basically consists in the diagnoser approach.[14,15]
- The latter, coined *interpreted diagnoser*, generates online in one shot the evolutions, explaining the sequence of observations; this is traditionally the case with the so-called *active system approach* by the authors,[18,37-41] and the works that stem from it.[42,43]

Offline computation means preprocessing, which is a computation performed when no diagnostic processing is ongoing: it generates compiled knowledge once and for all, with this knowledge being possibly exploited online several times. The offline computation as well as the space requirements of the methods in the former approach are prohibitive, whereas the online computation to solve a diagnosis problem instance is quite efficient (a sequence of observable events is processed in linear time in the length of the sequence itself). The methods in the latter category do not perform any offline computation, they only carry out an online model-based reasoning, whose high cost is limited as it is focused on a single sequence of observable events. Still, this alternative “diagnoserless” approach comes with a significant cost in computation because of the extended model-based reasoning required. Therefore, shall we abandon the idea of diagnosing DESs efficiently?

In order to face this question, recent work by the authors has tried to combine the two mentioned approaches and to make them cooperate with each other, both for a posteriori diagnosis[44] and diagnosis during monitoring[45]: this paper moves a further step in this direction as far as a posteriori diagnosis is concerned. The main findings can be summarized as follows.

- The notion of a *symptom dictionary*, this being a compiled knowledge structure for a posteriori diagnosis, is defined.
- A method for performing a posteriori diagnosis that is based on an (extensible) portion of the symptom dictionary that includes its initial state, called *open (symptom) dictionary*, is presented.
- The notion of a *symptom pattern*, this being a deterministic finite automaton (DFA) that recognizes the language consisting of some symptoms of the considered DES, is introduced.

- The way a *scenario*, which is an NFA representing some behaviors of interest (the most likely or the most critical ones), can be processed so as to draw its corresponding symptom pattern is shown.
- The pseudocode of an algorithm for extending the open (symptom) dictionary based on a symptom pattern is provided.

Using the terminology of Section 1, the open (symptom) dictionary is the repository of the compiled knowledge exploited by *Engine 1*, where this knowledge can be incomplete. If the compiled knowledge is insufficient to solve a new diagnosis problem, an online computation is performed, that is, *Engine 2* is run, so as to (permanently) add to the dictionary the chunks of knowledge that are missing inherently to such a problem. The diagnosis method is sound and complete although the compiled knowledge is incomplete since it is able to generate the needed chunks of compiled knowledge on the fly (by drawing them from the "raw" knowledge in the DES component models and in the DES topology). After a problem has been solved inefficiently (by *Engine 2*) once, it will be solved efficiently (by *Engine 1*) thereafter. In previous work relevant to the interpreted diagnoser, instead, each diagnosis problem was solved anew every time. Moreover, the added chunks of compiled knowledge can enable *Engine 1* to solve efficiently several additional problems besides the one that has spurred the open (symptom) dictionary extension.

3 | DESS AND DIAGNOSIS PROBLEMS

A DES is assumed to be a network of *components*, where each component, endowed with input and output *pins*, is modeled as a communicating automaton.[13] Each output pin of a component is connected with an input pin of another component by a *link*. The way a transition is triggered in a component is threefold: (1) spontaneously (by the empty event ε), (2) by an (external) event coming from the extern of the DES, or (3) by an (internal) event coming from another component of the DES. When a component performs a transition based on a (possibly empty) input event e, it may generate a set E of new events on its output pins, which possibly trigger the transitions of other components, with the triggering events being consumed. This cascading process continues until the DES becomes quiescent anew. A transition generating an event on an output pin can occur only if this pin is not occupied by another event already. Assuming that only one component transition at a time can occur (asynchronism), the process that moves a DES from the initial quiescent state to the final quiescent state can be represented by a sequence of component transitions, called a *trajectory* of the DES. At the occurrence of each transition, a DES $\mathcal{X}$ changes state, with each state x of $\mathcal{X}$ being a pair (C, L), where C is the array of current states of components and L is the array of (possibly empty) current events placed in links. Formally, the (possibly infinite) set of trajectories of $\mathcal{X}$ is specified by a DFA, namely, the *space* of $\mathcal{X}$,

$$\mathcal{X}^* = (\Sigma, X, \tau, x_0, X_\mathrm{f}), \tag{1}$$

where Σ (the alphabet) is the set of component transitions, X is the set of states, τ is the (deterministic) transition function, $\tau : X \times \Sigma \mapsto X$, x_0 is the initial (quiescent) state, and X_f is the set of final states. A state (C, L) of $\mathcal{X}$ is final when all links in L are empty (no event is placed in the links). The DES model adopted does not make any assumption about the liveliness of the language of events/transitions of the DES. Moreover, the space of the DES may include unobservable cycles.[*]

Example 1. Outlined in the center of Figure 1 is a DES called $\mathcal{P}$ (protection), which includes two components, a sensor s and a breaker b, and one link connecting the (single) output pin of s with the (single) input pin of b. The model of the sensor s (outlined on the left side of Figure 1) involves three states (denoted by circles) and four transitions (denoted by arcs). The model of the breaker b (outlined on the right side of Figure 1) involves two states and six transitions. No final states are defined for component models. Each component transition t from a state p to a state p', triggered by an input event e, and generating a set of output events E, is denoted by the angled triple $t = \langle p, (e, E), p' \rangle$, as detailed in Table 1.[†] The space of $\mathcal{P}$, namely, $\mathcal{P}^*$, is depicted on the left side of Figure 2, where each state is identified by a triple (s_s, s_b, e), with s_s being the state of the sensor, s_b the state of the breaker, and e the internal event in the link (ε means no event). The initial state is $(0, 0, \varepsilon)$; final states are double circled. To ease referencing, the states of $\mathcal{P}$ are renamed by numbers $0 \cdots 12$. Owing to cycles, the set of possible trajectories of $\mathcal{P}$ is infinite, one of them being

[*]An unobservable cycle makes the DES undiagnosable when it translates into a so-called *indeterminate hidden cycle* within the special diagnoser introduced in the work of Basilio and Lafortune.[46] However, the diagnosis method described in the paper (and the active system approach in general) is not affected by the possible non diagnosability induced by unobservable cycles, as it copes with both diagnosable and undiagnosable DESs the same way.
[†]As such, the alphabet of the automaton model is a set of pairs (e, E),where e is the (possibly empty) input event and E is the (possibly empty) set of output events.

TABLE 1 Transition details for the components s and b of the discrete-event system (DES) $\mathcal{P}$ outlined in Figure 1

Transition	***Description***
$s_1 = \langle 0, (ko, \{op\}), 1\rangle$	The sensor s detects a threatening event *ko* and generates the *op* (open) output event
$s_2 = \langle 1, (ok, \{cl\}), 0\rangle$	The sensor s detects a liberating event *ok* and generates the *cl* (close) output event
$s_3 = \langle 0, (ko, \{cl\}, 2\rangle$	The sensor s detects a threatening event *ko*, yet generates the *cl* (close) output event
$s_4 = \langle 1, (ok, \{op\}), 2\rangle$	The sensor s detects a liberating event *ok*, yet generates the *op* (open) output event
$b_1 = \langle 0, (op, \emptyset), 1\rangle$	The breaker b reacts to the *op* (open) input event by opening
$b_2 = \langle 1, (cl, \emptyset), 0\rangle$	The breaker b reacts to the *cl* (close) input event by closing
$b_3 = \langle 0, (op, \emptyset), 0\rangle$	The breaker b does not react to the *op* (open) input event and remains closed
$b_4 = \langle 1, (cl, \emptyset), 1\rangle$	The breaker b does not react to the *cl* (close) input event and remains open
$b_5 = \langle 0, (cl, \emptyset), 0\rangle$	The breaker b reacts to the *cl* (close) input event by remaining closed
$b_6 = \langle 1, (op, \emptyset), 1\rangle$	The breaker b reacts to the *op* (open) input event by remaining open

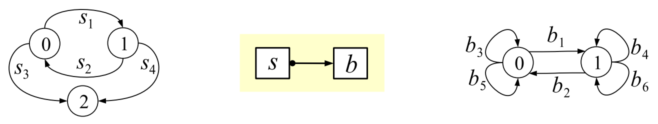
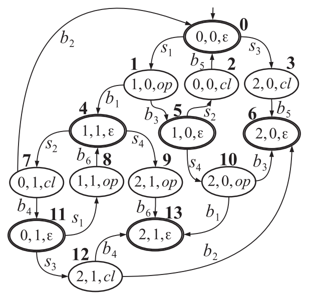


FIGURE 1 Discrete-event system (DES) $\mathcal{P}$ (center), including the sensor s and the breaker b, and models of sensor (left) and breaker (right)



t	o	f
s_1	*act*	ε
s_2	*sby*	ε
s_3	*act*	*fos*
s_4	*sby*	*fcs*
b_1	*opn*	ε
b_2	*cls*	ε
b_3	ε	*fob*
b_4	ε	*fcb*
b_5	ε	ε
b_6	ε	ε

o	*Observation description*
act	The sensor is in activation
sby	The sensor is in standby
opn	The breaker opens
cls	The breaker closes

f	*Fault description*
fos	The sensor fails to send the *op* command
fcs	The sensor fails to send the *cl* command
fob	The breaker fails to open
fcb	The breaker fails to close

FIGURE 2 From left to right: space $\mathcal{P}^*$, mapping table $\mu(\mathcal{P})$, and observation (top) and fault (bottom) descriptions

$[s_1, b_1, s_2, b_4]$, which ends in state 11, described as follows: s detects a threatening event and commands b to open; b opens; s detects a liberating event and commands b to close; eventually, however, b does not close.

For diagnosis purposes, we characterize a DES $\mathcal{X}$ with its *observability* (whether each transition is observable or unobservable) and *normality* (whether each transition is normal or faulty). No assumption is made about the nature of faults: each individual fault can either be permanent or not, independently of the other faults.

Let $\mathbf{T}$ be the set of component transitions in $\mathcal{X}$, $\mathbf{O}$ a finite set of *observations*, and $\mathbf{F}$ a finite set of *faults*. The *mapping table* μ of $\mathcal{X}$ is a function

$$\mu(\mathcal{X}) : \mathbf{T} \mapsto (\mathbf{O} \cup \{\varepsilon\}) \times (\mathbf{F} \cup \{\varepsilon\}, \tag{2}$$

where ε is the *empty* symbol. Table $\mu(\mathcal{X})$ can be represented as a finite set of triples (t, o, f), where $t \in \mathbf{T}$, $o \in \mathbf{O} \cup \{\varepsilon\}$, and $f \in \mathbf{F} \cup \{\varepsilon\}$. The triple (t, o, f) defines the observability and normality of t: if $o \neq \varepsilon$, then t is *observable*, else t is *unobservable*;[‡] if $f \neq \varepsilon$, then t is *faulty*, else t is *normal*.

Based on $\mu(\mathcal{X})$, each trajectory T in $\mathcal{X}^*$ can be associated with a *symptom* and a *diagnosis*. The *symptom* $\mathcal{O}$ of T is the finite *sequence* of observations involved in T,

$$\mathcal{O} = [o \,|\, t \in T, (t, o, f) \in \mu(\mathcal{X}), o \neq \varepsilon]. \tag{3}$$

[‡]In a physical DES, an observable transition manifests itself to an external observer by the generation of the observation associated with it (when the transition is triggered). In other words, if the transition is observable, then its occurrence generates the observation; otherwise (if the transition is unobservable), no observation is generated.

The *diagnosis* δ of T is the finite *set* of faults involved in T, ie,

$$\delta = \{ f \mid t \in T, (t, o, f) \in \mu(\mathcal{X}), f \neq \varepsilon \}. \tag{4}$$

Since a diagnosis is a set, only one instance of each fault f can be in δ. Hence, the domain of possible diagnoses is bounded by the (finite) powerset $2^{\mathbf{F}}$. By contrast, several instances of the same observation o can be in symptom $\mathcal{O}$; therefore, the domain of possible symptoms is in general infinite. Trajectory T involved in Equation (3) is said to *imply* $\mathcal{O}$, denoted $T \Rightarrow \mathcal{O}$. Likewise, trajectory T involved in Equation (4) is said to *imply* δ, denoted $T \Rightarrow \delta$. A trajectory of $\mathcal{X}$ is observed as a symptom and, since the observed symptom can be implied by several (possibly infinite[§]) trajectories, several diagnoses can be associated with the same symptom, which are collectively called the explanation of the symptom. Let $\mathcal{O}$ be a symptom of $\mathcal{X}$. The *explanation* Δ of $\mathcal{O}$ is the finite set

$$\Delta(\mathcal{O}) = \{ \delta \mid T \in \mathcal{X}^*, T \Rightarrow \mathcal{O}, T \Rightarrow \delta \}. \tag{5}$$

That is, the explanation of $\mathcal{O}$ is the set of diagnoses (called *candidates*) implied by the trajectories of $\mathcal{X}$ that imply $\mathcal{O}$. It is worth remarking that a trajectory implying a given symptom does not necessarily end with an observable transition. In fact, in a trajectory in Equation (5), the transition generating the last observable event in $\mathcal{O}$ can be followed by unobservable transitions. In other words, the candidates produced by the diagnosis method presented in this paper (and in the active system approach in general) do not account solely for trajectories generating the given symptom and ending with an observable transition (as is the case with the diagnoser approach[14]) but also for trajectories generating the given symptom and ending with (a chain of) unobservable transitions, possibly involving unobservable cycles.

Example 2. With reference to the DES $\mathcal{P}$ introduced in Example 1, displayed on the center of Figure 2 is the mapping table $\mu(\mathcal{P})$, where the symbols are described on the right side of the figure, namely, the observations (top) and the faults (bottom). Let $\mathcal{O} = [act, opn, sby]$ be a symptom of $\mathcal{P}$. Based on $\mathcal{P}^*$ in Figure 2, two trajectories imply $\mathcal{O}$, namely, $T_1 = [s_1, b_1, s_2, b_4]$ and $T_2 = [s_1, b_1, s_4, b_6]$, where T_1 involves the faulty transition b_4, whereas T_2 involves the faulty transition s_4. Hence, the explanation of $\mathcal{O}$ includes two (singleton) candidates, namely, $\Delta(\mathcal{O}) = \{\{fcb\}, \{fcs\}\}$. In plain words, two scenarios are possible: either the breaker failed to close (*fcb*) or the sensor failed to send the closing command (*fcs*).

In theory, it is always possible to generate online the explanation of a symptom $\mathcal{O}$ of $\mathcal{X}$ by *abducing* the subspace of $\mathcal{X}^*$ involving all and only the trajectories implying $\mathcal{O}$. However, the application domain may require the explanation to be given under stringent time constraints, thereby making this process impractical. In theory (and ideally), we could generate offline a data structure representing all possible symptoms of $\mathcal{X}$ and associating with each symptom the corresponding explanation. This way, given a symptom, the explanation could be immediately known online from the data structure.

4 | SYMPTOM DICTIONARY

In this section, we present a technique for preprocessing a DES to generate a data structure, called the *symptom dictionary*, which associates each possible symptom of the DES with the corresponding explanation. Still, it should be clear that the symptom dictionary is introduced for formal reasons only and is never materialized (completely). To this end, we first introduce the notion of an *extended space*.

Definition 1. Let $\mathcal{X}^* = (\Sigma, X, \tau, x_0, X_{\mathrm{f}})$ be the space of a DES $\mathcal{X}$, and $\mathbf{F}$ the domain of faults within the mapping table $\mu(\mathcal{X})$. The *extended space* of $\mathcal{X}$ is a DFA

$$\mathcal{X}^+ = (\Sigma, X^+, \tau^+, x_0^+, X_{\mathrm{f}}^+), \tag{6}$$

where Σ is the alphabet (the same as in $\mathcal{X}^*$), X^+ is the set of states (x, δ), $x \in X$, $\delta \subseteq \mathbf{F}$; $x_0^+ = (x_0, \emptyset)$; X_{f}^+ is the set of final states (x, δ), where $x \in X_{\mathrm{f}}$; $\tau^+ : X^+ \times \Sigma \mapsto X^+$ is the transition function, where $\tau^+((x, \delta), t) = (x', \delta')$ iff $\tau(x, t) = x'$ and $\delta' = \delta \cup \{f\}$ if $(t, o, f) \in \mu(\mathcal{X})$, $f \neq \varepsilon$ (when t is faulty), otherwise $\delta' = \delta$ (when t is normal).

[§]If a trajectory implies a symptom and includes a cycle with no observable component transitions, then there is an unbounded number of trajectories implying this symptom, each one is obtained by iterating the cycle a different number of times. However, since the diagnosis of a trajectory is a set of faults (without duplicates), additional iterations of the same cycle do not extend the diagnosis with new faults.

Proposition 1. *The regular language*[¶] *of $\mathcal{X}^+$ equals the regular language of $\mathcal{X}^*$.*[#] *Besides, if $x^+ = (x, \delta)$ is a state in $\mathcal{X}^+$, then the following two properties hold:* (1) *for each trajectory T in $\mathcal{X}^*$ ending in x, there is a trajectory T^+ in $\mathcal{X}^+$ ending in x^+ such that $T^+ = T$ and $T \Rightarrow \delta$; and* (2) *for each trajectory T^+ in $\mathcal{X}^+$ ending in x^+, there is a trajectory T in $\mathcal{X}^*$ ending in x such that $T = T^+$ and $T \Rightarrow \delta$.*

Proof. First, we prove that the regular language of $\mathcal{X}^+$ equals the regular language of $\mathcal{X}^*$. (*Soundness*) If $T \in \mathcal{X}^+$, then $T \in \mathcal{X}^*$. The proof is by induction on the sequence of transitions in T. (*Basis*) Trivially, if $T_0 = [\,] \in \mathcal{X}^+$, then $T_0 \in \mathcal{X}^*$. (*Induction*) If a prefix $T_i = [t_1, \dots, t_i] \in \mathcal{X}^+$ and $T_i \in \mathcal{X}^*$, then, if $T_{i+1} = [t_1, \dots, t_i, t_{i+1}] \in \mathcal{X}^+$, then $T_{i+1} \in \mathcal{X}^*$. In fact, the state reached by T_i in $\mathcal{X}^+$, namely, $x_i^+ = (x_i, \delta_i)$, is such that x_i is the state reached by T_i in $\mathcal{X}^*$, as the sequence of component transitions in T_i is the same in both $\mathcal{X}^+$ and $\mathcal{X}^*$. Hence, if t_{i+1} is triggerable in x_i^+, then it is triggerable in x_i also, as the triggerability of t_{i+1} depends on x_i only. (*Completeness*) If $T \in \mathcal{X}^*$, then $T \in \mathcal{X}^+$. Again, the proof is by induction on the sequence of transitions in T. (*Basis*) Trivially, if $T_0 = [\,] \in \mathcal{X}^*$, then $T_0 \in \mathcal{X}^+$. (*Induction*) If a prefix $T_i = [t_1, \dots, t_i] \in \mathcal{X}^*$ and $T_i \in \mathcal{X}^+$, then if $T_{i+1} = [t_1, \dots, t_i, t_{i+1}] \in \mathcal{X}^*$, then $T_{i+1} \in \mathcal{X}^+$. In fact, if x_i is the state reached by T_i in $\mathcal{X}^*$, then the state reached by T_i in $\mathcal{X}^+$ will be $x_i^+ = (x_i, \delta_i)$. Therefore, if t_{i+1} is triggerable in x_i, then it is triggerable in x_i^+ also, as the triggerability of t_{i+1} depends on x_i only. In other words, $\mathcal{X}^+$ has the same regular language as $\mathcal{X}^*$. Then, we prove property (1) of the proposition. Assume $T = [t_1, \dots, t_n] \in \mathcal{X}^*$ ending in x_n. Since X^* and $\mathcal{X}^+$ share the same regular language, there is a trajectory $T^+ = T$ in $\mathcal{X}^+$ ending in $x_n^+ = (x_n, \delta_n)$, as the sequence of component transitions in T is the same as in T^+. Moreover, $T \Rightarrow \delta$, as, according to Definition 1, δ is constructed by accumulating the set of (nonempty) faults associated with the transitions in T. Finally, we prove property (2) of the proposition. Assume $T^+ = [t_1, \dots, t_n] \in \mathcal{X}^+$ ending in $x^+ = (x_n, \delta_n)$. Since X^+ and $\mathcal{X}^*$ share the same regular language, there is a trajectory $T = T^+$ in $\mathcal{X}^*$ ending in x_n, as the sequence of component transitions in T is the same as in T^+. Moreover, $T \Rightarrow \delta$, as, according to Definition 1, δ is constructed by accumulating the set of (nonempty) faults associated with the transitions in T. This concludes the proof of Proposition 1. □

Example 3. With reference to the DES $\mathcal{P}$ in Example 1, displayed in Figure 3 is the extended space $\mathcal{P}^+$, with states renamed by numbers $0 \cdots 47$ and final states double circled. In accordance with Proposition 1, comparing the extended space $\mathcal{P}^+$ with the space $\mathcal{P}^*$ displayed in Figure 2, it turns out that the regular language of $\mathcal{P}^+$ equals the regular language of $\mathcal{P}^*$. Besides, in accordance with property (1) of the proposition, considering the trajectory $T = [s_1, b_1, s_2, b_4, s_3, b_2]$ in $\mathcal{P}^*$, which ends in state 6, there is a trajectory $T^+ = T$ in $\mathcal{P}^+$ that ends in the state $\mathbf{18} = (6, \{fcb, fos\})$ such that $T \Rightarrow \{fcb, fos\}$. Moreover, in accordance with property (2) of the proposition, considering the trajectory $T^+ = [s_1, b_1, s_2, b_4, s_1, b_6, s_2, b_2, s_3, b_5]$ in $\mathcal{P}^+$, which ends in state $\mathbf{18} = (6, \{fcb, fos\})$, there is a trajectory $T = T^+$ in $\mathcal{P}^*$ that ends in state 6 such that $T \Rightarrow \{fcb, fos\}$.

Definition 2. Let $\mathcal{X}^+$ be an extended space. Let $\mathcal{X}_{\mathrm{n}}^+$ be the NFA obtained from $\mathcal{X}^+$ by substituting the symbol t (component transition), marking each transition in $\mathcal{X}^+$, with the (possibly empty) observation o, where $(t, o, f) \in \mu(\mathcal{X})$.[‖] The *symptom dictionary* of $\mathcal{X}$ is the DFA $\mathcal{X}^\oplus$ obtained by determinization of $\mathcal{X}_{\mathrm{n}}^+$, where each final state $x^\oplus$ of $\mathcal{X}^\oplus$ is marked with the set of diagnoses associated with the final states of $\mathcal{X}_{\mathrm{n}}^+$ included[**] in $x^\oplus$, denoted $\Delta(x^\oplus)$.

Proposition 2. *The regular language of $\mathcal{X}^\oplus$ is equal to the set of possible symptoms of $\mathcal{X}$. Besides, if $\mathcal{O}$ is a symptom with accepting state $x^\oplus$ in $\mathcal{X}^\oplus$, then $\Delta(x^\oplus) = \Delta(\mathcal{O})$.*

Proof. First, we prove that the regular language of the symptom dictionary $\mathcal{X}^\oplus$ is equal to the set of symptoms of $\mathcal{X}$. Notice that, since $\mathcal{X}^\oplus$ is obtained by determinization of the NFA $\mathcal{X}_{\mathrm{n}}^+$, it suffices to show that the language of $\mathcal{X}_{\mathrm{n}}^+$ is equal to the set of symptoms of $\mathcal{X}$. (*Soundness*) If $\mathcal{O} \in \mathcal{X}_{\mathrm{n}}^+$, then $\mathcal{O}$ is a symptom of $\mathcal{X}$. In fact, according to Definition 2, $\mathcal{O} = [o \mid (t, o, f) \in \mu(\mathcal{X}), o \neq \varepsilon]$, where t is the component transition marking an arc in $\mathcal{X}^+$. Based on Proposition 1, the sequence of such component transitions, namely, the corresponding trajectory T^+, is a trajectory in $\mathcal{X}^*$ also. Hence, according to Equation (3), $\mathcal{O}$ is a symptom of $\mathcal{X}$. (*Completeness*) If $\mathcal{O}$ is a symptom of $\mathcal{X}$, then $\mathcal{O} \in \mathcal{X}_{\mathrm{n}}^+$. In fact, by

[¶]The regular language of a finite automaton $\mathcal{A}$ (be it deterministic or nondeterministic) is the set of strings accepted (recognized) by$\mathcal{A}$.[47]
[#] Hence, the notion of a trajectory is applicable to the strings in the regular language of$\mathcal{X}^+$ also.
[‖]The substitution of the component transition t with the observation o associated in the mapping table yields in general an NFA (namely, $\mathcal{X}_{\mathrm{n}}^+$) because of two reasons: (1) the observation o may be the empty symbol ε, and (2) there may be two arcs exiting the same state that are marked with the same observation o.
[**]According to the *Subset Construction* algorithm for NFA determinization,[47]each state of the DFA is identified by a subset ofthe states of the NFA.

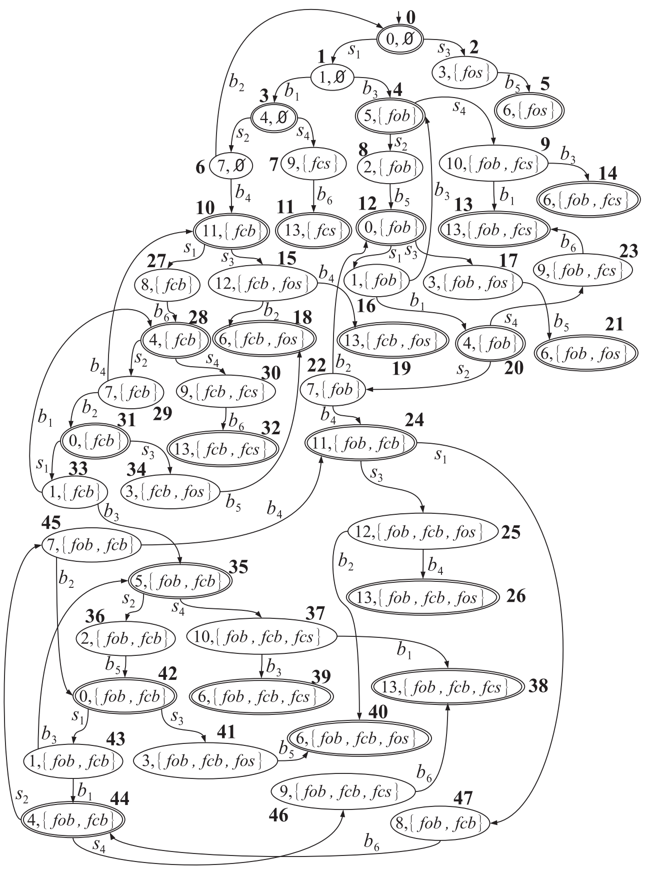


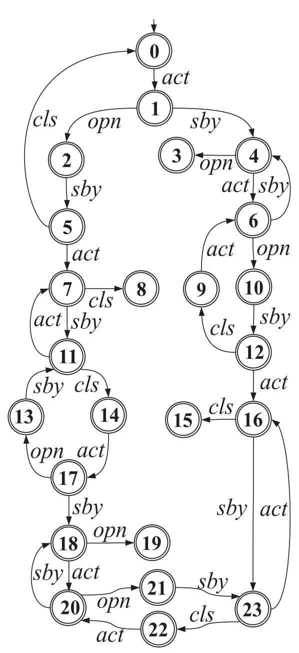
FIGURE 3 Extended space $\mathcal{P}^+$

definition of a symptom, there is a trajectory $T \in \mathcal{X}^*$, where $\mathcal{O} = [o \,|\, (t, o, f) \in \mu(\mathcal{X}), o \neq \varepsilon]$. By virtue of Proposition 1, $T \in \mathcal{X}^+$. Hence, after the transformation of $\mathcal{X}^+$ into $\mathcal{X}^+_{\mathrm{n}}$ in Definition 2, T is mapped to $[o|(t, o, f) \in \mu(\mathcal{X}), o \neq \varepsilon]$, namely, $\mathcal{O}$. In other words, $\mathcal{O} \in \mathcal{X}^+_{\mathrm{n}}$.

Now, we show that, if $\mathcal{O}$ is a symptom with accepting state $x^\oplus$ in $\mathcal{X}^\oplus$, then $\Delta(x^\oplus) = \Delta(\mathcal{O})$. First, if $\delta \in \Delta(x^\oplus)$, then $\delta \in \Delta(\mathcal{O})$. In fact, based on Definition 2, there is a final state $x^+ = (x, \delta)$ in $\mathcal{X}^+$ such that x is final in $\mathcal{X}^*$. According to Proposition 1, there is a trajectory $T \in \mathcal{X}^*$ such that $T \Rightarrow \delta$. Moreover, since $x^\oplus$ is the accepting state of $\mathcal{O}$, $T \Rightarrow \mathcal{O}$. In other words, based on Equation (5), $\delta \in \Delta(\mathcal{O})$. Second, if $\delta \in \Delta(\mathcal{O})$, then $\delta \in \Delta(x^\oplus)$. In fact, based on Equation (5), there is $T \in \mathcal{X}^*$ such that $T \Rightarrow \mathcal{O}$ and $T \Rightarrow \delta$. According to Proposition 1, $T \in \mathcal{X}^+$. Hence, there is a (final) state $x^+ = (x, \delta)$ in $\mathcal{X}^+$ such that x is the (final) state reached by T in $\mathcal{X}^*$. Based on Definition 2, when $\mathcal{X}^+$ is transformed into $\mathcal{X}^+_{\mathrm{n}}$, the trajectory T is transformed into a sequence of observations that, once the ε symbols are removed, equals $\mathcal{O}$, with accepting state x^+.

To conclude the proof, it suffices to show that $x^+ \in x^\oplus$. This comes from the mode in which the *Subset Construction* determinization algorithm operates: if a string s in accepted by an NFA in a state n, then the accepting state d of s in the DFA (equivalent to the NFA) generated by *Subset Construction* includes n. The proof is by induction on the string s. (*Basis*) If s is empty, then the accepting state n of s belongs to the ε-closure of the initial state n_0. Since the initial state of the DFA is in fact the ε-closure of n_0, the NFA state n belongs to the initial state of the DFA, which is in fact the accepting state of the empty string. (*Induction*) If n is the accepting state of a string s in the NFA and n belongs to the accepting state d of the DFA, then, if n' is the accepting state of the extended string $s' = s \cup [o]$ in the NFA, then n' belongs to the accepting state d' of s' in the DFA. In fact, n' is necessarily reached by a sequence of transitions of the NFA (exiting n) that involves just the symbol o. Therefore, all the NFA states preceding the transition marked by o in such a sequence will belong to the DFA state d, whereas n' will belong to the ε-closure of the set of states reached by the transitions exiting the states of d that are marked by o, which, based on *Subset Construction*, are in fact the set of states belonging to d'. Hence, n' belongs to d'. This concludes the proof of Proposition 2. □

Example 4. With reference to $\mathcal{P}^+$ in Figure 3, the symptom dictionary $\mathcal{P}^\oplus$ is outlined on the left side of Figure 4. Incidentally, all states in $\mathcal{P}^\oplus$ are final. On the right side of Figure 4, each state $p^\oplus$ of $\mathcal{P}^\oplus$ is described in terms of the $\mathcal{P}^+$ states included in $p^\oplus$ (with the final states being underlined) and the associated set of diagnoses, which, based on



$p^\oplus$	$\mathcal{P}^+$ states	Candidates in $\Delta(p^\oplus)$
0	$\underline{0}$	$\emptyset$
1	$1, 2, \underline{4}, \underline{5}$	$\{fob\}, \{fos\}$
2	$\underline{3}$	$\emptyset$
3	$\underline{13}$	$\{fob, fcs\}$
4	$8, 9, \underline{12}, \underline{14}$	$\{fob\}, \{fob, fcs\}$
5	$6, 7, \underline{10}, \underline{11}$	$\{fcb\}, \{fcs\}$
6	$\underline{4}, 16, 17, \underline{21}$	$\{fob\}, \{fob, fos\}$
7	$15, \underline{19}, 27, \underline{28}$	$\{fcb\}, \{fcb, fos\}$
8	$\underline{18}$	$\{fcb, fos\}$
9	$\underline{12}$	$\{fob\}$
10	$\underline{20}$	$\{fob\}$
11	$\underline{10}, 29, 30, \underline{32}$	$\{fcb\}, \{fcb, fcs\}$
12	$\underline{13}, 22, 23, \underline{24}$	$\{fob, fcb\}, \{fob, fcs\}$
13	$\underline{28}$	$\{fcb\}$
14	$\underline{31}$	$\{fcb\}$
15	$\underline{40}$	$\{fob, fcb, fos\}$
16	$25, \underline{26}, \underline{44}, 47$	$\{fob, fcb\}, \{fob, fcb, fos\}$
17	$\underline{18}, 33, 34, \underline{35}$	$\{fob, fcb\}, \{fcb, fos\}$
18	$36, 37, \underline{39}, \underline{42}$	$\{fob, fcb\}, \{fob, fcb, fcs\}$
19	$\underline{38}$	$\{fob, fcb, fcs\}$
20	$\underline{35}, \underline{40}, 41, 43$	$\{fob, fcb\}, \{fob, fcb, fos\}$
21	$\underline{44}$	$\{fob, fcb\}$
22	$\underline{42}$	$\{fob, fcb\}$
23	$\underline{24}, \underline{38}, 45, 46$	$\{fob, fcb\}, \{fob, fcb, fcs\}$

FIGURE 4 Symptom dictionary $\mathcal{P}^\oplus$ (left) and relevant state details (right)

Proposition 2, is in fact the explanation of each symptom with accepting state $p^\oplus$. In accordance with Proposition 2, one can take any string in the regular language of $\mathcal{X}^\oplus$ and find out that this string is a symptom of $\mathcal{P}$ by considering the trajectories in the space $\mathcal{P}^*$ displayed in Figure 2. For instance, the string $[act, opn, sby]$ in $\mathcal{P}^\oplus$ is the symptom relevant to two trajectories of $\mathcal{P}$, namely, $T_1 = [s_1, b_1, s_2, b_4]$ and $T_2 = [s_1, b_1, s_4, b_6]$ (cf Example 2). Moreover, still in accordance with Proposition 2, considering again the symptom $\mathcal{O} = [act, opn, sby]$, which has accepting state **5** in $\mathcal{P}^\oplus$, we have $\Delta(\mathbf{5}) = \{\{fcb\}, \{fcs\}\}$, which in fact equals the explanation $\Delta(\mathcal{O})$ determined in Example 2.

A symptom dictionary $\mathcal{X}^\oplus$ is an extremely efficient tool for a posteriori diagnosis of DESs, where candidates are generated after the reception of the complete symptom. Given a symptom $\mathcal{O}$, the computation of $\Delta(\mathcal{O})$ boils down to matching $\mathcal{X}^\oplus$ against $\mathcal{O}$, a simple operation with complexity that is linear in the length of $\mathcal{O}$. In theory, the symptom dictionary allows the DE to operate always in quick mode by *Engine 1*, with *Engine 2* never coming into play. However, as recalled in Section 2, like for the diagnoser, the symptom dictionary requires total knowledge compilation, which is out of dispute. Therefore, in order to escape from the myth of total knowledge compilation and somewhat retaining the advantage of *Engine 1*, we propose a restricted dictionary that expands over time either by the "experience" acquired from the solution of new problems or by the injection of specific knowledge. In other words, we propose an *open dictionary*.

5 | OPEN DICTIONARY

Roughly, an open dictionary is a subgraph of the symptom dictionary. Hence, the (regular) language of an open dictionary is a subset of the (regular) language of the symptom dictionary, which, based on Proposition 2, is the set of symptoms of the DES. This means that only a (possibly tiny) fraction of the symptoms of the DES are cataloged in the open dictionary. Still, each symptom $\mathcal{O}$ that is cataloged in the open dictionary is associated with exactly $\Delta(\mathcal{O})$, the *sound and complete* set of candidates that explain $\mathcal{O}$. Hence, despite being sound (but not complete) in the set of symptoms (as it contains only a subset of the symptoms of the DES), the open dictionary is sound and complete in the explanation of the symptom, provided that the symptom is included in the language of the open dictionary.[††] What is the initial configuration of the open

[††]It is like having an "open" dictionary of the English language that contains only a subset of the English words. Still, each English word that is included in the open dictionary is explained exactly as it is in the (complete) English dictionary. In that sense, the open English dictionary is sound in the explanation of the words included, yet incomplete in the words of the English language. Therefore, "openness" refers to the set of words cataloged, not to the explanation of these words.

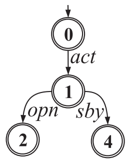


FIGURE 5 Prefix $\mathcal{P}^{\oplus}_{[2]}$ of the symptom dictionary $\mathcal{P}^{\oplus}$

dictionary? Although the open dictionary can be initially empty, we suggest to initialize it with a *prefix* of the symptom dictionary, as specified in Definition 3.

Definition 3. Let $\mathcal{X}^{\oplus}$ be a symptom dictionary. The *distance* of a state $x^{\oplus}$ in $\mathcal{X}^{\oplus}$ is the minimum number of transitions connecting the initial state of $\mathcal{X}^{\oplus}$ with $x^{\oplus}$. The *prefix* of $\mathcal{X}^{\oplus}$ up to a distance $d \geq 0$, denoted $\mathcal{X}^{\oplus}_{[d]}$, is the subgraph of $\mathcal{X}^{\oplus}$ comprehending all the states at a distance $\leq d$ and all the transitions along paths (starting from the initial state of the symptom dictionary) that have length $\leq d$.

Example 5. With reference to Figure 4, the prefix of $\mathcal{P}^{\oplus}$ up to distance 2, namely, $\mathcal{P}^{\oplus}_{[2]}$, is displayed in Figure 5 (cf Figure 4 for state details).

A prefix $\mathcal{X}^{\oplus}_{[d]}$ provides the explanation of every symptom that is not longer than d. If $\mathcal{X}^{\oplus}_{[d]}$ embodies a cycle (which is not the case in $\mathcal{P}^{\oplus}_{[2]}$), it also provides the explanation of the infinite set of symptoms encompassing this cycle. However, any symptom longer than d may not belong to the language of $\mathcal{X}^{\oplus}_{[d]}$, such as $\mathcal{O} = [act, sby, opn]$ in $\mathcal{P}^{\oplus}_{[2]}$. In this case, comes into play *Engine 2*, which generates the explanation $\Delta(\mathcal{O})$ based on the *abduction* of $\mathcal{O}$, namely, a DFA whose language is the subset of the trajectories of $\mathcal{X}$ implying $\mathcal{O}$. To this end, *Engine 2* performs model-based reasoning to reconstruct the subspace of $\mathcal{X}^*$ required. Once provided the explanation $\Delta(\mathcal{O})$, the "experience" acquired by the DE can be integrated into the open dictionary based on the *symptom pattern* of $\mathcal{O}$. However, the notion of a symptom pattern goes beyond a (plain) symptom, as specified below.

Definition 4. A *symptom pattern* of a DES $\mathcal{X}$ is a DFA whose alphabet is the set of observations of $\mathcal{X}$. A string in the language of the symptom pattern that is not a symptom of $\mathcal{X}$ is a *spurious symptom*.

A special (and very simple) case of symptom pattern is associated with each symptom $\mathcal{O}$, denoted $\mathcal{O}^*$, which is the DFA recognizing $\mathcal{O}$ (the language of $\mathcal{O}^*$ is the singleton $\{\mathcal{O}\}$).

Example 6. Displayed in Figure 6 is the symptom pattern $\mathcal{O}^*$ of $\mathcal{O} = [act, sby, opn]$. Despite the fact that the states of $\mathcal{O}^*$ are identified by natural numbers, these states should not be confused with the homonymous states in the symptom dictionary (cf Figure 4). Another (circular) symptom pattern is shaded on the bottom-right side of Figure 9 (cf Example 10).

Given a symptom pattern $\mathcal{O}^*$, the language of an open dictionary $\mathcal{X}^{\oplus}$ can be extended by the (nonspurious) symptoms in the language of $\mathcal{O}^*$ by means of the *Dictionary Extension* algorithm listed below (cf Algorithm 1, lines 1 to 38). We assume that each state $x^{\oplus}$ in $\mathcal{X}^{\oplus}$ is equipped with a *labeling set*, denoted $\Omega(x^{\oplus})$ (initially empty), which is instantiated by the algorithm with states in $\mathcal{O}^*$, and that ω is an unmarked label in this set. Roughly, the algorithm aims to match $\mathcal{O}^*$ with the language of $\mathcal{X}^{\oplus}$. When the matching of an observation o succeeds, the labeling set of the state reached in $\mathcal{X}^{\oplus}$ is extended by the state reached in $\mathcal{O}^*$(provided that it is not included already). If the matching fails, then $\mathcal{X}^{\oplus}$ is extended by a new transition and, possibly, by a new state. Let $\langle x^{\oplus}, o, x'^{\oplus}\rangle$ be the missing transition in $\mathcal{X}^{\oplus}$. Based on lines 15 to 18, the new state $x'^{\oplus}$ is generated first by determining the set X^+_o of $\mathcal{X}^+$ states reached by a transition exiting a state in $x^{\oplus}$ and marked with o and, then, by extending X^+_o with all the states that are reached by a sequence of unobservable component transitions. It should be clear that $\mathcal{X}^+$ is not materialized: only the states required are actually generated starting from the $\mathcal{X}^+$ states within $x^{\oplus}$ and stored in $\mathcal{X}^{\oplus}$. Once all the component transitions relevant to all the observable events marking the transitions exiting the considered state ω have been processed, ω is marked (line 34). Given $\omega \in \Omega(x^{\oplus})$, two cases are possible. If ω is not marked, then the transition function of $x^{\oplus}$ in $\mathcal{X}^{\oplus}$ needs to be checked against $\mathcal{O}^*$. If, instead, ω is marked, then the update of the transition function of $x^{\oplus}$ is completed. Since it is impossible to insert ω into $\Omega(x^{\oplus})$ if included already, once ω has been marked, the processing of ω is inhibited, thereby preventing the infinite matching of cycles in $\mathcal{O}^*$.

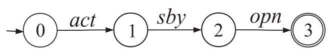


FIGURE 6 Symptom pattern $\mathcal{O}^*$, where $\mathcal{O} = [act, sby, opn]$

Algorithm 1 *Dictionary Extension* (takes in input an open dictionary $\mathcal{X}^\oplus$ and a symptom pattern $\mathcal{O}^*$, extending $\mathcal{X}^\oplus$ based on $\mathcal{O}^*$)

```
1:  procedure DICTIONARY EXTENSION($\mathcal{X}^\oplus$, $\mathcal{O}^*$)
2:      $\mathcal{X}^\oplus = (\Sigma, X^\oplus, \tau^\oplus, x_0^\oplus, X_f^\oplus)$: an open dictionary of $\mathcal{X}$,
3:      $\mathcal{O}^* = (\Sigma_\omega, \Omega, \tau_\omega, \omega_0, \Omega_f)$: a symptom pattern of $\mathcal{X}$;
4:  begin
5:      Insert $\omega_0$ into the labeling set $\Omega(x_0^\oplus)$;
6:      repeat
7:          Choose an $x^\oplus \in X^\oplus$ such that there is an unmarked $\omega \in \Omega(x^\oplus)$;
8:          for all unmarked $\omega \in \Omega(x^\oplus)$ do
9:              for all transitions $\langle \omega, o, \omega' \rangle \in \tau_\omega$ do
10:                 if $\langle x^\oplus, o, x'^\oplus \rangle \in \tau^\oplus$ then
11:                     if $\omega' \notin \Omega(x'^\oplus)$ then
12:                         Insert $\omega'$ into $\Omega(x'^\oplus)$
13:                     end if
14:                 else
15:                     $X_o^+ \leftarrow \{x'^+ \mid x^+ \in x^\oplus, \langle x^+, o, x'^+ \rangle \in \tau(\mathcal{X}^+)\}$;
16:                     if $X_o^+ \neq \emptyset$ then
17:                         $\hat{X}_o^+ \leftarrow$ the set of states in $\mathcal{X}^+$ that are reachable from a state in $X_o^+$ by a sequence of
                                transitions $\langle x_i^+, t, x_j^+ \rangle$ where $t$ is unobservable;
18:                         $\bar{X}_o^+ \leftarrow X_o^+ \cup \hat{X}_o^+$;
19:                         if $X^\oplus$ includes a state $x'^\oplus = \bar{X}_o^+$ then
20:                             Insert into $\tau^\oplus$ the new transition $\langle x^\oplus, o, x'^\oplus \rangle$;
21:                             Insert $\omega'$ into $\Omega(x'^\oplus)$ only if $\omega' \notin \Omega(x'^\oplus)$
22:                         else
23:                             Insert into $X^\oplus$ the new state $x'^\oplus = \bar{X}_o^+$;
24:                             if there is $(x, \delta) \in x'^\oplus$ such that $x$ is quiescent then
25:                                 Insert $x'^\oplus$ into $X_f^\oplus$;
26:                                 $\Delta(x'^\oplus) \leftarrow \{\delta \mid (x, \delta) \in x'^\oplus, x \text{ is quiescent}\}$
27:                             end if
28:                             Label $x'^\oplus$ with the singleton $\Omega(x'^\oplus) = \{\omega'\}$;
29:                             Insert into $\tau^\oplus$ the new transition $\langle x^\oplus, o, x'^\oplus \rangle$
30:                         end if
31:                     end if
32:                 end if
33:             end for
34:             Mark $\omega$ within the labeling set $\Omega(x^\oplus)$
35:         end for
36:     until there is no $x^\oplus \in X^\oplus$ such that $\Omega(x^\oplus)$ includes an unmarked state;
37:     Empty all the nonempty labeling sets $\Omega(x^\oplus)$
38: end procedure
```

Example 7. Consider the open dictionary $\mathcal{P}_{[2]}^\oplus$ in Figure 5 and the symptom pattern $\mathcal{O}^*$ in Figure 6. The extension of $\mathcal{P}_{[2]}^\oplus$ based on $\mathcal{O}^*$ by Algorithm 1 is performed as follows (to distinguish from $\mathcal{O}^*$ states, the states of the open dictionary are in bold). Initially, the labeling set $\Omega(\mathbf{0})$ is $\{0\}$, where 0 is the initial state of $\mathcal{O}^*$. Since both *act* and *sby* are matched, the labeling sets of the involved states become $\Omega(\mathbf{1}) = \{1\}$ and $\Omega(\mathbf{4}) = \{2\}$. Now, since no transition marked by *opn* exits **4**, the missing dictionary state $x'^\oplus = \mathbf{3}$ is generated first computing $X_{opn}^+ = \{13\}$, where 13 is the state reached by the $\mathcal{X}^+$ state 9 (cf Figure 3). However, since no transition exits 13 in $\mathcal{X}^+$, we have $\hat{X}_{opn}^+ = \emptyset$ and, hence, $\bar{X}_{opn}^+ = \{13\} = \mathbf{3}$. Since it is missing, the state **3** is inserted into $\mathcal{P}_{[2]}^\oplus$. Based on lines 23 to 25, **3** is also inserted into

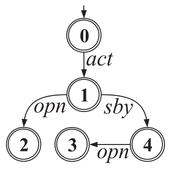


FIGURE 7 Expansion of the prefix $\mathcal{P}^{\oplus}_{[2]}$ (cf Figure 5) into the open dictionary $\mathcal{P}^{\oplus}_{[2,\mathcal{O}]}$

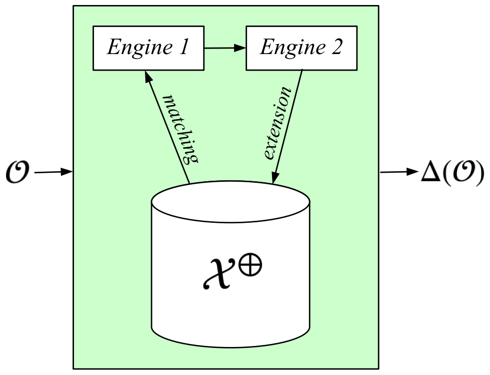


FIGURE 8 Architecture of the diagnosis engine based on *Engine 1*, *Engine 2*, and the open dictionary $\mathcal{X}^{\oplus}$

the final states and marked by the explanation $\Delta(\mathbf{3}) = \{\{fob, fcs\}\}$. Eventually, **3** is labeled with $\Omega(\mathbf{3}) = \{3\}$ and the transition $\langle \mathbf{4}, opn, \mathbf{3} \rangle$ is created. Since no transition exits the state 3 in $\mathcal{O}^*$, the processing of $\omega = 3$ has no effect, and the condition of termination in line 36 is true, thereby ending the loop. The updated open dictionary, namely, $\mathcal{P}^{\oplus}_{[2,\mathcal{O}]}$, is shown in Figure 7.

Based on Example 7, one may argue that, since the prefix of the symptom $\mathcal{O} = [act, sby, opn]$ up to the second observation, namely, $[act, sby]$, is already in the language of $\mathcal{P}^{\oplus}_{[2]}$, it might be convenient to avoid generating the abduction of $\mathcal{O}$ by *Engine 2*. Instead, the extension of the dictionary might be performed *on the fly* to eventually obtain the explanation from the state **3** created. Actually, this is reasonable in general. Indeed, the complete separation between *Engine 1* and *Engine 2* is more conceptual than physical. To clarify this, displayed in Figure 8 is the architecture of the DE based on *Engine 1*, *Engine 2*, and the open dictionary $\mathcal{X}^{\oplus}$. Given the symptom $\mathcal{O}$ in input, *Engine 1* aims to perform the matching of $\mathcal{O}$ with $\mathcal{X}^{\oplus}$ (cf Algorithm 1). If the next observation is not matched, then *Engine 2* comes into play, which extends the open dictionary with a new transition and, possibly, a new state, thereby allowing *Engine 1* to continue the matching of $\mathcal{O}$. As such, the task of *Engine 1* is intertwined with the task of *Engine 2* to match symptom $\mathcal{O}$ and to eventually provide the explanation $\Delta(\mathcal{O})$. If *Engine 2* is actually involved in the process, the open dictionary $\mathcal{X}^{\oplus}$ will be extended appropriately. If not, the open dictionary is simply matched and will remain unchanged.

6 | SCENARIOS

An open dictionary $\mathcal{X}^{\oplus}$ can be extended with (a possibly infinite number of) new symptoms. The simplest way is adding a symptom $\mathcal{O}$ that was previously explained by *Engine 2*, as in Example 7. Remarkably, if $\mathcal{O}$ is generated in $\mathcal{X}^{\oplus}$ by a path of transitions involving a cycle, then the language of $\mathcal{X}^{\oplus}$ will be extended not only with $\mathcal{O}$ but also with the infinite symptoms involved in the circular path. For example, extending $\mathcal{P}^{\oplus}_{[2]}$ in Figure 5 with the symptom $[act, opn, sby, cls]$ actually extends $\mathcal{P}^{\oplus}_{[2]}$ with the infinite set of symptoms generated by the circular path $0 \to 1 \to 2 \to 5 \to 0$.

The dictionary can also be extended based on particular behavioral patterns of the DES, called *scenarios*. A scenario is a behavior of the DES that is considered either most probable or most critical and, hence, is required to be explained efficiently. The idea is to generate the symptom pattern of the scenario and to extend the language of the open dictionary with its language. This way, each symptom generated from now on by a trajectory that conforms with the scenario will be explained by *Engine 1* quickly.

Definition 5. A *scenario* of a DES $\mathcal{X}$ is a pair $\mathcal{S} = (\Sigma, \mathcal{L})$, where Σ is a subset of the component transitions in $\mathcal{X}$ and $\mathcal{L}$ a regular language on Σ.

Since Σ is a subset of the component transitions in $\mathcal{X}$, all the transitions not included in Σ are irrelevant to the scenario. Therefore, in general, a string in $\mathcal{L}$ is not a trajectory of $\mathcal{X}$.

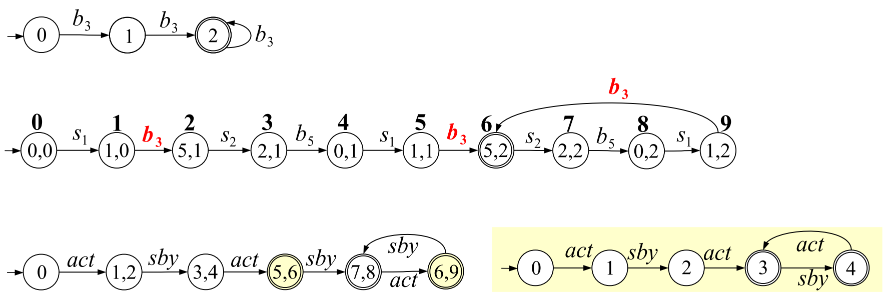


FIGURE 9 $\hat{\mathcal{L}}$ (top), $\mathcal{P}^*_{\mathcal{S}}$ (middle), and $\mathcal{O}^*_{\mathcal{S}}$ (bottom)

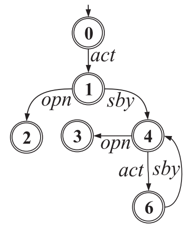


FIGURE 10 Expansion of the open dictionary $\mathcal{P}^{\oplus}_{[2,\mathcal{O}]}$ (cf Figure 7) into the open dictionary $\mathcal{P}^{\oplus}_{[2,\mathcal{O},\mathcal{S}]}$

Example 8. The scenario in which the breaker is stuck closed can be defined as $\mathcal{S} = (\Sigma, \mathcal{L})$, where $\Sigma = \{s_3, s_4, b_1, b_2, b_3, b_4\}$ and $\mathcal{L}$ is specified by the regular expression $b_3 b_3^+$ (namely, b_3 repeated at least twice).[‡‡]

Definition 6. Let $\mathcal{S} = (\Sigma, \mathcal{L})$ be a scenario of $\mathcal{X}$. The *restriction* of a trajectory T in $\mathcal{X}^*$ on Σ is the sequence $T_\Sigma = [t \mid t \in T, t \in \Sigma]$. The *abduction* of $\mathcal{S}$, denoted $\mathcal{X}^*_{\mathcal{S}}$, is a DFA whose language is the set $\{T \mid T \in \mathcal{X}^*, T_\Sigma \in \mathcal{L}\}$.

In other words, the abduction of a scenario $\mathcal{S}$ is a subspace of $\mathcal{X}^*$, where each trajectory T conforms to one string of the scenario, in the sense that the subsequence of the component transitions in T that are in Σ is a string in $\mathcal{L}$.

Example 9. Consider the scenario $\mathcal{S}$ defined in Example 8. The generation of the abduction $\mathcal{P}^*_{\mathcal{S}}$ is based on the DFA recognizing the language $\mathcal{L}$, namely $\hat{\mathcal{L}}$, shown on the top of Figure 9. The DFA representing $\mathcal{P}^*_{\mathcal{S}}$ is displayed in the middle of the same figure, where each state is a pair (x, ℓ), where x is a state of $\mathcal{X}^*$ and ℓ a state of $\hat{\mathcal{L}}$. A state is final when both x and ℓ are final, in our example, only state $\mathbf{6} = (5, 2)$.

Definition 7. Let $\mathcal{S} = (\Sigma, \mathcal{L})$ be a scenario of a DES $\mathcal{X}$ and $\mathcal{X}^*_{\mathcal{S}}$ the abduction of $\mathcal{O}$. Let $\mathcal{N}$ be the NFA obtained from $\mathcal{X}^*_{\mathcal{S}}$ by substituting $\langle x, o, x' \rangle$ for every transition $\langle x, t, x' \rangle$, where $(t, o, f) \in \mu(\mathcal{X})$. The *symptom pattern* of the scenario $\mathcal{S}$, denoted $\mathcal{O}^*_{\mathcal{S}}$, is the minimum DFA equivalent to $\mathcal{N}$.

Example 10. With reference to abduction $\mathcal{P}^*_{\mathcal{S}}$ determined in Example 9 (cf middle of Figure 9), shown on the bottom-left side of Figure 9 is the DFA obtained by determinization of $\mathcal{N}$ (cf Definition 7), where the states $\{5, 6\}$ and $\{6, 9\}$ are equivalent. The minimal DFA, namely, the symptom pattern $\mathcal{O}^*_{\mathcal{S}}$, is shown on the bottom-right side of Figure 9.

The nonspurious part of the language of the symptom pattern $\mathcal{O}^*_{\mathcal{S}}$ of a scenario $\mathcal{S}$ is composed of all the symptoms with which $\mathcal{S}$ manifests itself to the observer. However, any such symptom can be implied not only by the trajectories that conform with the scenario but also by other trajectories. The extension of the open dictionary based on $\mathcal{O}^*_{\mathcal{S}}$ allows for the sound and complete explanation of any (nonspurious) symptom in $\mathcal{O}^*_{\mathcal{S}}$.

Example 11. Based on Algorithm 1, extending the open dictionary $\mathcal{P}^{\oplus}_{[2,\mathcal{O}]}$, displayed in Figure 7, with the symptom pattern $\mathcal{O}^*_{\mathcal{S}}$ results in the new open dictionary $\mathcal{P}^{\oplus}_{[2,\mathcal{O},\mathcal{S}]}$ shown in Figure 10.

[‡‡]A regular expression is defined inductively on an alphabet Σ. 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), $x y$ (concatenation), $x?$ (optionality), x^* (repetition zero or more times), and x^+ (repetition one or more times).

7 | DISCUSSION

The research presented in this paper is based on recent work by the authors. In the work of Bertoglio et al,[44] both the addressed task, that is, a posteriori diagnosis, and the notions of symptom dictionary, dictionary prefix, and scenario are the same as in this paper. However, the techniques adopted for the progressive expansion of the dictionary are different: in the aforementioned work,[44] the open dictionary is extended by merging it with some so-called *constrained dictionaries*, each inherent to a distinct scenario, whereas algorithm *Dictionary Extension*, which takes as input parameters the open dictionary to be expanded and a symptom pattern, is run here. The notion of a constrained dictionary is quite different from that of a symptom pattern. A constrained dictionary is a dictionary that encompasses all and only the symptoms relevant to the trajectories of the DES that are compliant with a given scenario and provides the sound and complete diagnoses relevant to them. A symptom pattern, instead, is generically the recognizer of a language defined over the alphabet of the observations of the DES (and it does not include any information about faults). Basically, in case the open dictionary has to be expanded with a dictionary path (relevant to a symptom) that is partially covered by a path already existing in the current version of the dictionary (that is, some states and transitions in the new path already belong to the dictionary), the method in the work of Bertoglio et al[44] for extending the open dictionary requires constructing the whole path, whereas algorithm *Dictionary Extension* constructs only the missing chunks of it. Moreover, the method in the aforementioned[44] for extending the open dictionary performs a determinization, whereas no such operation is carried out by the method described in this paper. Finally, the open dictionary generated by the algorithm *Dictionary Extension* in Section 5 is always a subgraph of the symptom dictionary, whereas this is not guaranteed by the method presented in the work of Bertoglio et al.[44]

The notion of a symptom pattern has been introduced in another work of Bertoglio et al,[45] where a method analogous to that presented in this current paper is adopted to expand the open dictionary. However, Bertoglio et al[45] addressed the task of diagnosis during monitoring and enforced this task to fulfill a so-called consistency requirement, whereas the task considered in the current paper is a posteriori diagnosis. Consequently, the dictionaries handled by the two papers are slightly different: a *temporal* dictionary is managed for diagnosis during monitoring, whereas a symptom dictionary is managed both here and in our different work.[44]

The symptom dictionary resembles the compiled knowledge structure called diagnoser in the diagnoser approach[14,15]; however, several differences can be singled out. First of all, it is worth underlining that the diagnoser approach and the active system approach (which the work presented in this paper stems from) make different modeling assumptions. The diagnoser approach (and other works that have adopted its definition of a DES) assumes that every faulty component transition is unobservable. This assumption is made on the ground that the diagnosis of faults would be trivial if they were observable. The aforementioned justification is objected by the authors of the active system approach, since the observable event relevant to a faulty transition can be shared with (several) normal and/or faulty transitions, which does not make the diagnosis task easier. Hence, in the twin-engine method presented in this paper (the same as in the whole active system approach), faulty transitions can be observable. Moreover, the diagnoser approach assumes that both the language of the transitions of the DES and the language of the observable events of the DES are live, whereas the active system approach does not make any such assumption. As to the symptom dictionary and the diagnoser, both of them are DFAs that recognize the language of all the symptoms of the DES. However, the language of symptoms in the diagnoser approach is prefix closed, whereas the language of symptoms in the a posteriori diagnosis task described in the current paper is not prefix closed, as a symptom is relevant to a behavioral evolution, called a trajectory[§§] in Section 3, leading from the initial state to a final state of the DES, where a state is final if all the links are empty. The notion of a final state does not apply to the DES models taken into account by the diagnoser approach, as such models, supporting a synchronous communication between components, do not include any link. Leaving apart this distinctive feature, another major difference can be caught if we consider a path p in the diagnoser/dictionary and the symptom s associated with it. In the diagnoser, p represents all the DES trajectories that meet two conditions: (a) their projection on the set of observable events is s, and (b) their final transition is observable. In the dictionary, p represents all the DES trajectories (leading from the initial state to a final state) whose projection on the set of observable events is s, where the final component transition can be either observable or not. Consequently, given the same symptom s, the set of candidates output by the diagnoser approach is different with respect to that output by the current approach.

[§§]This notion of a trajectory is specific to the task of a posteriori diagnosis of active systems; in fact, the constraint that a trajectory has to end in a final state is removed when the task of diagnosis during monitoring is performed, as in our other work.[45]

One may also speculate that scenarios look like supervision patterns.[32] In fact, this similarity is only superficial because supervision patterns are meant to extend the notion of a fault to a relevant event that arises when a subtrajectory of the DES matches a given pattern, like in the works of Lamperti et al.[48,49] As such, the relevant event is part of the candidate diagnosis. By contrast, scenarios are meant to polarize knowledge compilation on some specific domain-dependent behavior that is considered either most probable or most critical, so that it can be diagnosed efficiently (by *Engine 1*). In fact, scenarios are orthogonal to supervision patterns, inasmuch scenarios could account for supervision-patterns also, should they be defined in the (extended) mapping table of the DES.

Some resemblances can be envisaged also between the extended space and the automaton, introduced in the work of Carvalho et al,[50] resulting from the composition of the dilation automaton and the label automaton. Basically, the dilation automaton represents the behavior of a plant including a sensor whose reading can either be recorded (normal behavior) or not (abnormal behavior). Such an automaton can include unobservable behavioral cycles, the same as the DES models considered in this paper. The label automaton, instead, represents the mode changes of the considered sensor, where such modes are actually three labels, N corresponding to the normal behavior when no fault has occurred, F to the faulty behavior, and R to the recover behavior, which is a normal behavior after some (intermittent) fault(s) have occurred. Therefore, each state in the automaton resulting from the composition of the dilation automaton with the label automaton includes a label, which represents the health situation, similarly to the way field δ represents the health situation of each state in the extended space. However, while the approach described in the work of Carvalho et al[50] is meant to detect only one kind of sensor fault, where such a fault is intermittent, as well as to highlight whether the sequence of observable events that has been gathered is consistent with a (temporarily) recover, the approach presented in this paper can cope with several kinds of faults, it does not make any distinction between persistent and intermittent faults, and it is not able to find out whether the system may have recovered from an intermittent fault. The health situation in the aforementioned work[50] is a simple label, whereas it is a set of faults in the method described in the current paper.

The NFA $\mathcal{X}_n^+$ introduced by Definition 2 in this paper is not to be confused with the NFA G_o defined in the twin plant method,[20] although it is similar to it. In fact, each state in both automata is a DES state accompanied by the set of faults along a path (or several paths, all sharing the same set of faults) from the initial state to the DES state itself. However, every transition in G_o is observable (the same as in the NFA G', called the generator, in the work of Sampath et al[14]), whereas, in $\mathcal{X}_n^+$, a transition is either observable or not. Consequently, every state in G_o corresponds to a DES state that has an entering observable transition, whereas no observability constraint holds for the transitions entering the DES state inherent to a state in $\mathcal{X}_n^+$. Moreover, the modeling assumptions adopted in the work of Jiang et al[20] are the same as in the diagnoser approach,[14] that is, liveness of the language of events, absence of any cycle of unobservable events, and unobservability of faulty events, all of which are relaxed here (and in the whole active system approach).

In the work of Rajaoarisoa and Sayed-Mouchaweh,[51] a so-called diagnoser[¶¶] (model) is first built based just on the normal behavior of the considered DES, since this is assumed to be the only behavior that is initially known and it is hard to include in advance all abnormal behaviors in the models. Later on, while the DES is operating, this model is progressively integrated with new specific faulty behaviors: on account of this, the approach, which has been experimented with (the actuator faults in) a manufacturing system, is said to be "adaptive." This adaptivity may seem similar to the extensibility of the symptom dictionary presented in this paper. However, there are some major differences, first of all, relevant to the nature of the structure that is progressively extended: the diagnoser (model) in the work of Rajaoarisoa and Sayed-Mouchaweh[51] is actually a behavioral model inherent to the whole DES (basically corresponding to the automaton called *space* in this paper), whereas the dictionary is a compiled data structure. Assuming that the diagnoser model encompasses all normal behaviors is demanding since it requires that the whole normal behavioral model is built, which may be computationally unrealistic. In addition, the diagnostic reasoning presented in the aforementioned work[51] is performed online based on such a model (that is, no knowledge compilation is exploited), whereas the knowledge in the dictionary, when available, is used by the approach presented in this paper to achieve better performances. Another major difference is relevant to the portion of the diagnoser/dictionary to be built beforehand. While in the work of Rajaoarisoa and Sayed-Mouchaweh[51] the initial portion of the diagnoser model encompasses all normal behaviors, no such constraint exists for the dictionary. The (open) dictionary could even be initially empty, algorithm *Dictionary Extension* can cope with this: the content of the dictionary just affects the time needed to compute a diagnosis. A further difference is that our approach assumes that the component models include both normal and abnormal behaviors (that is, the models are

[¶¶]Although the name is the same, this diagnoser is different from that defined in the work of Sampath et al[14] and does not include any compiled knowledge.

complete, no component behavior is initially unknown): a dictionary extension is needed when a system behavior, although known in principle, has not been processed before, that is, no knowledge compilation has been performed inherently to such a behavior. The diagnoser in the work of Rajaoarisoa and Sayed-Mouchaweh,[51] instead, is adapted whenever a behavior that was unknown has occurred. This can be seen as a sort of learning, and an intervention by a human expert is needed, in an interactive online environment, whereas our approach faces a posteriori diagnosis, with no man-in-the-loop. Minor differences are inherent to the DES models, which are pure in the current paper (ie, the time notion consists just in a temporal ordering), whereas time instants are explicitly taken into account in the aforementioned work.[51]

Sometimes, as it is the case with the work of Rish et al,[52] the adjective "adaptive", accompanying diagnosis, is used to refer to the selection of measurements to achieve a fast diagnostic inference and restrict the number of candidate diagnoses. Example of measurements are tests of hardware or software systems, inspections of system components, questions to an expert, or probes. Most often, this process is called *sequential* diagnosis, and its goal is to reduce the diagnostic cost, defined as the number of measurements to be taken until the faults (or the most probable candidates) are identified. As one can surmise, the term "adaptivity" has quite a different meaning in the aforementioned work[52] with respect to both the same word in the work of Rajaoarisoa and Sayed-Mouchaweh[51] and the concept of extensibility of the symptom dictionary presented in this paper.

8 | CONCLUSION

A technique for the a posteriori diagnosis of DESs (each represented as a network of asynchronously communicating finite automata) based on an open dictionary and two cooperating DEs was presented. The technique is viable and becomes increasingly efficient without requiring the generation of the whole space of the DES; that is, it works while avoiding total knowledge compilation. The open dictionary is possibly initialized before the DES is being operated, starting, for instance, from a prefix of the symptom dictionary, which is then integrated with the symptoms and the candidate diagnoses relevant to a set of scenarios of the DES that are considered worth being diagnosed efficiently.

When the DES is being operated, the open dictionary can be enlarged at any time in three ways: (a) by coping with new symptoms explained by *Engine 2*, (b) by incorporating new compiled knowledge based on the symptoms inherent to additional scenarios, and (c) by extending its border with new transitions and states using a relaxed variation of Algorithm 1, where the search for new transitions is unconstrained by any symptom pattern. The remark that the system model may lack some behaviors[51] (ie, events and/or states in the automata representing the behavior of the components) is interesting (although not new[53]). The idea of extending the open dictionary also based on previously unknown behaviors is a challenge for future work.

In this paper, the same as in all papers dealing with the active system approach, a candidate diagnosis is the set of faults relevant to a trajectory in the DES space that entails the given symptom. As such, the approach does not make any distinction between permanent faults (ie, faults having a permanent duration) and transient faults (ie, temporally finite deviations from the normal behavior, also called intermittent faults, a qualification that emphasizes the repeatability of their occurrence). If a candidate includes a fault, this means that this fault occurs at least once in a trajectory that produces the given symptom (and there are possibly several trajectories that produce the given symptom). However, in case we are interested either in one occurrence of a fault or in multiple occurrences of the same fault, the mechanism of scenarios allows us to make the open dictionary encompass (only) the compiled knowledge about the trajectories that satisfy such constraints. Providing diagnosis outputs that can differentiate permanent and intermittent faults and that can help in understanding whether the DES behavior has possibly recovered from some intermittent faults is another interesting research topic for the future.

We have implemented the twin-engined diagnosis technique in C++ and we are also expanding it to cope with monitoring-based diagnosis, without requiring the diagnosability of the DES. As additional future research, we plan to extend the technique proposed in this paper to complex DESs.[54-58]

ACKNOWLEDGEMENTS

We would like to acknowledge the anonymous reviewers for their constructive comments and suggestions, which allowed us to improve the quality of the final manuscript considerably. This work was supported in part by the 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 61972360).

CONFLICT OF INTEREST

The authors declare no potential conflict of interest.

ORCID

Nicola Bertoglio https://orcid.org/0000-0002-7905-5957
Gianfranco Lamperti https://orcid.org/0000-0002-1915-6932
Marina Zanella https://orcid.org/0000-0003-3896-3913
Xiangfu Zhao https://orcid.org/0000-0001-5870-5730

REFERENCES

1. Kahneman D. *Thinking Fast and Slow*. New York, NY: Farrar, Straus and Giroux; 2011.
2. Reiter R. A theory of diagnosis from first principles. *Artificial Intelligence*. 1987;32(1):57-95.
3. Hamscher W, Console L, de Kleer J. *Readings in Model-Based Diagnosis*. San Mateo, CA: Morgan Kaufmann; 1992.
4. Aldrich C, Auret L. *Unsupervised Process Monitoring and Fault Diagnosis with Machine Learning Methods*. London, UK: Springer; 2013.
5. Yin S, Ding S, Xie X, Luo H. A review on basic data-driven approaches for industrial process monitoring. *IEEE Trans Ind Electron*. 2014;61(11):6418-6428.
6. Jia F, Lei Y, Guo L, Lin J, Xing S. A neural network constructed by deep learning technique and its application to intelligent fault diagnosis of machines. *Neurocomputing*. 2018;272:619-628. https://doi.org/10.1016/j.neucom.2017.07.032
7. Jung D, Ng K, Frisk E, Krysander M. Combining model-based diagnosis and data-driven anomaly classifiers for fault isolation. *Control Eng Pract*. 2018;80:146-156.
8. Rabah NB, Saddem R, Hmida FB, Carré-Ménétrier V, Tagina M. Intelligent case based decision support system for online diagnosis of automated production system. *J Phys Conf Ser*. 2017;783. https://doi.org/10.1088/1742-6596/783/1/012009
9. Liu R, Yang B, Zio E, Chen X. Artificial intelligence for fault diagnosis of rotating machinery: a review. *Mech Syst Signal Process*. 2018;108:33-47. https://doi.org/10.1016/j.ymssp.2018.02.016
10. Cassandras C, Lafortune S. *Introduction to Discrete Event Systems*. 2nd ed. New York, NY: Springer; 2008.
11. Basile F. Overview of fault diagnosis methods based on Petri net models. In: Proceedings of the 2014 European Control Conference (ECC); 2014; Strasbourg, France.
12. Cong X, Fanti M, Mangini A, Li Z. Decentralized diagnosis by Petri nets and integer linear programming. *IEEE Trans Syst Man Cybern Syst*. 2018;48(10):1689-1700.
13. Brand D, Zafiropulo P. On communicating finite-state machines. *J ACM*. 1983;30(2):323-342. https://doi.org/10.1145/322374.322380
14. Sampath M, Sengupta R, Lafortune S, Sinnamohideen K, Teneketzis D. Diagnosability of discrete-event systems. *IEEE Trans Autom Control*. 1995;40(9):1555-1575.
15. Sampath M, Sengupta R, Lafortune S, Sinnamohideen K, Teneketzis D. Failure diagnosis using discrete-event models. *IEEE Trans Control Syst Technol*. 1996;4(2):105-124.
16. Sampath M, Lafortune S, Teneketzis D. Active diagnosis of discrete-event systems. *IEEE Trans Autom Control*. 1998;43(7):908-929.
17. Rozé L. Supervision of telecommunication network: a diagnoser approach. Paper presented at: 8th International Workshop on Principles of Diagnosis (DX); 1997; Mont St. Michel, France.
18. Baroni P, Lamperti G, Pogliano P, Zanella M. Diagnosis of large active systems. *Artificial Intelligence*. 1999;110(1):135-183. https://doi.org/10.1016/S0004-3702(99)00019-3
19. Kurien J, Nayak P. Back to the future for consistency-based trajectory tracking. Paper presented at: 11th International Workshop on Principles of Diagnosis (DX); 2000; Morelia, Mexico.
20. Jiang S, Huang Z, Chandra V, Kumar R. A polynomial algorithm for testing diagnosability of discrete-event systems. *IEEE Trans Autom Control*. 2001;46(8):1318-1321.
21. Yoo T, Lafortune S. Polynomial-time verification of diagnosability of partially observed discrete-event systems. *IEEE Trans Autom Control*. 2002;47(9):1491-1495.
22. Cimatti A, Pecheur C, Cavada R. Formal verification of diagnosability via symbolic model checking. Paper presented at: 18th International Joint Conference on Artificial Intelligence (IJCAI); 2003; Acapulco, Mexico.
23. Grastien A. Symbolic testing of diagnosability. Paper presented at: 20th International Workshop on Principles of Diagnosis (DX); 2009; Stockholm, Sweden.
24. Rintanen J, Grastien A. Diagnosability testing with satisfiability algorithms. Paper presented at: 20th International Joint Conference on Artificial Intelligence (IJCAI); 2007; Hyderabad, India.
25. Console L, Picardi C, Ribaudo M. Diagnosis and diagnosability analysis using PEPA. Paper presented at: 14th European Conference on Artificial Intelligence (ECAI); 2000; Berlin, Germany.
26. Su X, Zanella M, Grastien A. Diagnosability of discrete-event systems with uncertain observations. Paper presented at: 25th International Joint Conference on Artificial Intelligence (IJCAI); 2016; New York, NY.

27. Liu F, Qiu D. Diagnosability of fuzzy discrete-event systems: a fuzzy approach. *IEEE Trans Fuzzy Syst*. 2009;17(2):372-384. https://doi.org/10.1109/TFUZZ.2009.2013840
28. Paoli A, Lafortune S. Diagnosability analysis of a class of hierarchical state machines. *Discrete Event Dyn Syst Theory Appl*. 2008;18(3):385-413.
29. Pencolé Y. Diagnosability analysis of distributed discrete event systems. Paper presented at: 16th European Conference on Artificial Intelligence (ECAI); 2004; Valencia, Spain.
30. Schumann A, Pencolé Y. Scalable diagnosability checking of event-driven systems. In: Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI); 2007; Hyderabad, India.
31. Schumann A, Huang J. A scalable jointree algorithm for diagnosability. Paper presented at: 23rd National Conference on Artificial Intelligence (AAAI); 2008; Chicago, IL.
32. Jéron T, Marchand H, Pinchinat S, Cordier M. Supervision patterns in discrete event systems diagnosis. Paper presented at: 2006 8th International Workshop on Discrete Event Systems (WODES); 2006; Ann Arbor, MI.
33. Ye L, Dague P, Yan Y. An incremental approach for pattern diagnosability in distributed discrete event systems. Paper presented at: 2009 21st IEEE International Conference on Tools with Artificial Intelligence; 2009; Newark, NJ.
34. Ran N, Su H, Giua A, Seatzu C. Codiagnosability analysis of bounded Petri nets. *IEEE Trans Autom Control*. 2018;63(4):1192-1199.
35. Li B, Khlif-Bouassida M, Toguyéni A. Reduction rules for diagnosability analysis of complex systems modeled by labeled Petri nets. *IEEE Trans Autom Sci Eng*. 2019. https://doi.org/10.1109/TASE.2019.2933230
36. Yin X, Lafortune S. On the decidability and complexity of diagnosability for labeled Petri nets. *IEEE Trans Autom Control*. 2017;62(11):5931-5938.
37. Lamperti G, Zanella M. Diagnosis of discrete-event systems from uncertain temporal observations. *Artificial Intelligence*. 2002;137(1-2):91-163. https://doi.org/10.1016/S0004-3702(02)00123-6
38. Lamperti G, Zanella M. A bridged diagnostic method for the monitoring of polymorphic discrete-event systems. *IEEE Trans Syst Man Cybern B Cybern*. 2004;34(5):2222-2244.
39. Cerutti S, Lamperti G, Scaroni M, Zanella M, Zanni D. A diagnostic environment for automaton networks. *Softw: Pract Exper*. 2007;37(4):365-415. https://doi.org/10.1002/spe.773
40. Lamperti G, Zanella M. Monitoring of active systems with stratified uncertain observations. *IEEE Trans Syst Man Cybern A Syst Hum*. 2011;41(2):356-369. https://doi.org/10.1109/TSMCA.2010.2069096
41. Lamperti G, Zanella M, Zhao X. *Introduction to Diagnosis of Active Systems*. Cham, Switzerland: Springer International Publishing; 2018.
42. Pencolé Y, Cordier M. A formal framework for the decentralized diagnosis of large scale discrete event systems and its application to telecommunication networks. *Artificial Intelligence*. 2005;164(1-2):121-170.
43. Grastien A, Cordier M, Largouët C. Incremental diagnosis of discrete-event systems. Paper presented at: 16th International Workshop on Principles of Diagnosis (DX); 2005; Monterey, CA.
44. Bertoglio N, Lamperti G, Zanella M. A posteriori diagnosis of discrete-event systems with symptom dictionary and scenarios. In: Wotawa F, Friedrich G, Pill I, Koitz-Hristov R, Ali M, eds. *Advances and Trends in Artificial Intelligence. From Theory to Practice: 32nd International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE 2019, Graz, Austria, July 9-11, 2019, Proceedings*. Cham, Switzerland: Springer; 2019:325-333. *Lecture Notes in Artificial Intelligence*; vol. 11606.
45. Bertoglio N, Lamperti G, Zanella M. Temporal diagnosis of discrete-event systems with dual knowledge compilation. In: Holzinger A, Kieseberg P, Tjoa AM, Weippl E, eds. *Machine Learning and Knowledge Extraction: Third IFIP TC 5, TC 12, WG 8.4, WG 8.9, WG 12.9 International Cross-Domain Conference, CD-MAKE 2019, Canterbury, UK, August 26-29, 2019, Proceedings*. Cham, Switzerland: Springer; 2019:333-352. *Lecture Notes in Computer Science*; vol. 11713.
46. Basilio J, Lafortune S. Robust codiagnosability of discrete event systems. In: Proceedings of the American Control Conference; 2009; St. Louis, MO. https://doi.org/10.1109/ACC.2009.5160208
47. Hopcroft J, Motwani R, Ullman J. *Introduction to Automata Theory Languages, and Computation*. 3rd ed. Boston, MA: Addison-Wesley; 2006.
48. Lamperti G, Zanella M. Context-sensitive diagnosis of discrete-event systems. Paper presented at: 22nd International Joint Conference on Artificial Intelligence (IJCAI); 2011; Barcelona, Spain.
49. Lamperti G, Zhao X. Diagnosis of active systems by semantic patterns. *IEEE Trans Syst Man Cybern Syst*. 2014;44(8):1028-1043. https://doi.org/10.1109/TSMC.2013.2296277
50. Carvalho L, Basilio J, Moreira M, Clavijo L. Diagnosability of intermittent sensor faults in discrete event systems. In: Proceedings of the American Control Conference; 2013; Washington, DC.
51. Rajaoarisoa L, Sayed-Mouchaweh M. Adaptive online fault diagnosis of manufacturing systems based on DEVS formalism. *IFAC-PapersOnLine*. 2017;50:6825-6830. https://doi.org/10.1016/j.ifacol.2017.08.1202
52. Rish I, Brodie M, Ma S, et al. Adaptive diagnosis in distributed systems. *IEEE Trans Neural Netw*. 2005;16(5):1088-1109. https://doi.org/10.1109/TNN.2005.853423
53. Zhao X, Ouyang D. Model-based diagnosis of discrete event systems with an incomplete system model. In: *ECAI 2008: 18th European Conference on Artificial Intelligence, July 21-25, 2008, Patras, Greece: Including Prestigious Applications of Intelligent Systems (PAIS 2008): Proceedings*. Amsterdam, The Netherlands: IOS Press; 2008.
54. Lamperti G, Quarenghi G. Intelligent monitoring of complex discrete-event systems. In: *Intelligent Decision Technologies 2016: Proceedings of the 8th KES International Conference on Intelligent Decision Technologies (KES-IDT 2016) - Part I*. Cham, Switzerland: Springer International Publishing; 2016:215-229.

55. Lamperti G, Zhao X. Diagnosis of complex active systems with uncertain temporal observations. In: *Availability, Reliability, and Security in Information Systems: IFIP WG 8.4, 8.9, TC 5 International Cross-Domain Conference, CD-ARES 2016, and Workshop on Privacy Aware Machine Learning for Health Data Science, PAML 2016, Salzburg, Austria, August 31 - September 2, 2016, Proceedings*. Cham, Switzerland: Springer; 2016:45-62.
56. Lamperti G, Zhao X. Viable diagnosis of complex active systems. Paper presented at: IEEE International Conference on Systems, Man, and Cybernetics (SMC); 2016; Budapest, Hungary.
57. Lamperti G, Zanella M, Zhao X. Knowledge compilation techniques for model-based diagnosis of complex active systems. In: Holzinger A, Kieseberg P, Tjoa AM, Weippl E, eds. *Machine Learning and Knowledge Extraction: Second IFIP TC 5, TC 8/WG 8.4, 8.9, TC 12/WG 12.9 International Cross-Domain Conference, CD-MAKE 2018, Hamburg, Germany, August 27-30, 2018, Proceedings*. Cham, Switzerland: Springer; 2018:43-64. *Lecture Notes in Computer Science*; vol. 11015.
58. Lamperti G, Zanella M, Zhao X. Abductive diagnosis of complex active systems with compiled knowledge. Paper presented at: 16th International Conference on Principles of Knowledge Representation and Reasoning; 2018; Tempe, AZ.

How to cite this article: Bertoglio N, Lamperti G, Zanella M, Zhao X. Twin-engined diagnosis of discrete-event systems. *Engineering Reports*. 2019;1:e12060. https://doi.org/10.1002/eng2.12060