Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 292 (2026) 3522–3531

Procedia Computer Science

www.elsevier.com/locate/procedia

30th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems (KES 2026)

Agile Diagnosis of Active Systems by Condensed Abduction

Gianfranco Lamperti*

Department of Information Engineering, University of Brescia, 25123 Brescia, Italy

Abstract

Automation of diagnosis has been addressed since the dawn of AI, both in medical and industrial domains. This paper focuses on model-based diagnosis of a class of asynchronous discrete-event systems (DESs), called active systems, which can also be an abstract representation of continuous dynamical systems, such as automobiles, power networks, or nuclear plants. In that context, diagnosis is abduction-based, where the behavior of a DES that conforms with a given temporal observation needs to be constructed *ex ante* in order to eventually distill the candidate diagnoses. Since the abduction knowledge-structure produced may be huge, the diagnosis engine may slow down to a point that the monitoring of the system could be jeopardized, especially in critical domains that require recovery actions in real-time as a reaction to the diagnosis response. To make the diagnosis task more agile, a novel abduction technique is presented, which is based on two requirements: reduction of the size of the abduction and, more to the point, reduction of the computation time by avoiding duplicating low-level model-based reasoning. A worst-case performance analysis indicates that a CONDENSED ABDUCTION ENGINE may outperform abduction-based engines of active systems significantly.

© 2026 The Authors. Published by Elsevier B.V.
This is an open access article under the CC BY-NC-ND license (https://creativecommons.org/licenses/by-nc-nd/4.0)
Peer-review under responsibility of the scientific committee of the KES International Association

Keywords: intelligent systems; model-based reasoning; agile diagnosis; abduction; discrete-event systems; finite state machines; uncertainty

1. Introduction

The diagnosis task aims to find out the causes of given symptoms, traditionally in the medical domain. However, diagnosis can be equally applied in the industrial sector, where any anomalous behavior of a physical system can manifest itself as a symptom expressed by relevant observations. In contrast with *rule-based diagnosis*, which incorporates the diagnostic knowledge collected from human expertise, *model-based diagnosis* [8] is a general-purpose technique that only requires a description (model) of the system. Model-based diagnosis can be either *consistency-based* or *abduction-based*. In consistency-based diagnosis, only the normal (correct) behavior of the system is modeled in a logical formalism [17, 10]. Given some observations of the system, a collection of conflict sets is generated, with each set involving some components of the system where at least one component is necessarily faulty in order to preserve

* Gianfranco Lamperti. Tel.: +39-030-371-5491 ; fax: +39-030-380-014.
E-mail address: gianfranco.lamperti@unibs.it

1877-0509 © 2026 The Authors. Published by Elsevier B.V.
This is an open access article under the CC BY-NC-ND license (https://creativecommons.org/licenses/by-nc-nd/4.0)
Peer-review under responsibility of the scientific committee of the KES International Association

logical consistency between the observations and the system model; candidate diagnoses are then enumerated based on conflict sets. In abduction-based diagnosis [15], instead, a complete model of the system is required, including both normal and abnormal (faulty) behavior: the behavior of the system that conforms with a collection of observations is reconstructed (abduced), from which relevant diagnoses are eventually distilled. In this paper, we focus on model-based diagnosis of a class of discrete-event systems (DESs) [6] called *active systems* [14]. The main contribution is the proposal of a novel abduction technique for model-based diagnosis of active systems, that allows for both a compact abduction structure and an agile diagnosis engine where computational redundancy is avoided.

2. Background

A DES can be modeled in various ways, including a Petri net, like in [9, 5, 2, 7], but most typically as a network of communicating finite state machines [4], like in [18, 1, 16, 11]. Among these references, in the context of abduction-based diagnosis of DESs, the seminal work [18] coins the notion of *diagnosability* of a DES and exploits a *diagnoser* to carry out the diagnosis task efficiently. In [1], a technique is presented for the incremental diagnosis of an asynchronous DES irrespective of its diagnosability and without the need for a diagnoser. The work in [16] aims to integrate the techniques presented in [18] and [1] into a single framework, with an application to telecommunication networks. In [11], a diagnosis technique for DESs with incomplete models is proposed, along with a learning diagnoser that estimates the fault condition of the DES and attempts to learn the missing information in the model using discrepancies between the actual and expected output of the DES.

Diagnosis of DESs may be *interpreter-based*, like in [1], or *compiler-based*, like in [18]. The interpreted technique is typically adopted in *a posteriori* diagnosis, where the diagnosis task starts after the complete reception of a sequence of observations (a *temporal observation* of the DES). The compiled technique, instead, is conceived mainly for *monitoring-based* diagnosis, where a chunk of candidate diagnoses is generated at the reception of each single observation of the DES. In interpreter-based diagnosis, the behavior of a DES that conforms with a temporal observation is reconstructed at each diagnosis session. In compiler-based diagnosis, instead, a *diagnoser* (or *dictionary*) for a specific DES is compiled off-line once and for all, so that it can be exploited efficiently in each diagnosis session of the DES. Albeit theoretically attractive, the compiled approach supposes the generation of the behavior of the DES as a whole, which is impractical for real DESs, as the number of system states (where a system state is represented as the combination of the states of all components) is exponential with the number of the components involved. For example, with 20 components in the DES, each component involving 3 states, we may come up with 3^{20} (more than three billion) system states. On the other hand, in contrast with the compiled approach, which provides candidate diagnoses in linear time, the response time of the interpreted approach becomes increasingly larger with the length of the temporal observation. A good compromise for monitoring-based diagnosis is the so-called *hybrid* technique, which combines the interpreted and compiled approaches, like in [3, 12, 13]. In the hybrid technique, the dictionary of a DES is generated incrementally based on the observations actually received: at the reception of each new observation, only the fragment of dictionary relevant to that observation is materialized, so that it can be exploited efficiently for subsequent observations, thereby avoiding generating the (typically huge) portion of the dictionary that is irrelevant to the actual temporal observation.

A problem that is common to all these approaches to diagnosis of active systems, however, is that the set of states of the abduction knowledge-structure (which is itself a finite state machine involving diagnoses in its states) is significantly larger than necessary. Even worse, when generating states and transitions of the abduction structure, the same low-level model-based reasoning is bound to be uselessly repeated many times, thereby causing a significant delay in the response time of the diagnosis engine, which is bound to jeopardize the monitoring of the system, more significantly so in critical domains that require an (artificial) agent to perform timely recovery actions as a reaction to the diagnosis response.

3. Preliminaries

Roughly, an active system (AS) is a graph where nodes are *components* and arcs are *links*. A component is equipped with some input and output *pins*, through which *events* are either received from or sent to other components, respectively. A link is a connection from an output pin of a component to an input pin of another component. The behavior

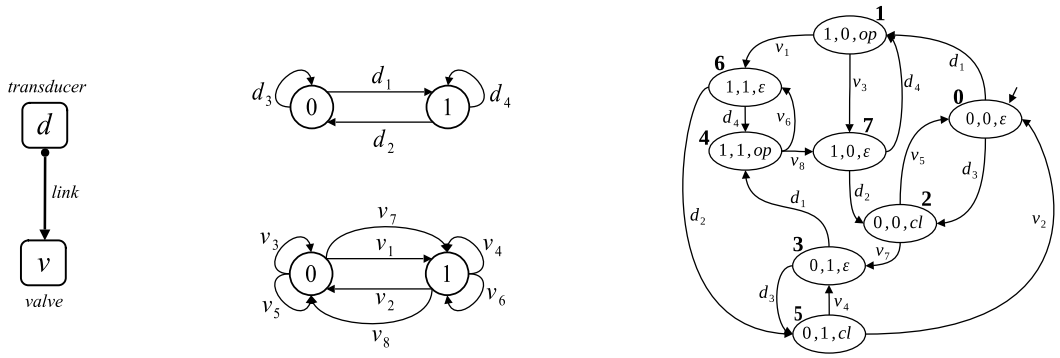


Fig. 1: From left to right: AS $\mathcal{S}$, models of transducer d (top) and valve v (bottom), and $Space(\mathcal{S})$.

of a component is modeled as a communicating finite state machine, which is characterized by a set of *states* and a set of *transitions*. When a component is in a state and an event is present on an input pin, the component may change state by consuming the input event and possibly generating new events at some output pins: the component may perform a state transition. An event generated on an output pin of a component, that is connected via a link with an input pin of another component, becomes present on that input pin. However, a transition may be *spontaneous*, when occurring without the need for any input event: formally, the transition is said to be triggered by an *empty* event, denoted ε.

Example 1. With reference to Figure 1, outlined on the left side is an AS $\mathcal{S}$ monitoring a cooling system. The AS involves two components: a transducer d, which is sensible to temperature, and a valve v, which can be either closed or opened by the transducer by means of specific commands (events) sent through a link from d to v. When the temperature becomes high, the transducer commands the valve to open for the cooling fluid to flow; when the temperature returns to normal, the transducer commands the valve to close. The behavioral models of d and v are depicted on the center of the figure. In particular, the model of d (top) involves two states (circles) and four transitions (arcs), while the model of v (bottom) involves two states and eight transitions. A transition from a state s to a state s', that is triggered by an input event e and generates a set of output events E, is formally represented as an angular triple $\langle s, (e, E), s' \rangle$. Component transitions are described in the first two columns of Table 1.

It should be noticed that the description of an AS is only given indirectly in terms of its components and links. Since in ASs component transitions occur one at a time, a question arises: *how does the AS behave as a whole?* Or, stated another way: *which is the space in which the AS can move?* The *space* of an AS $\mathcal{X}$ turns out to be itself a finite state machine, $Space(\mathcal{X}) = (\mathbf{T}, \mathbf{X}, \tau, x_0)$, where the alphabet $\mathbf{T}$ is the whole set of component transitions, $\mathbf{X}$ is the set of states, specifically pairs (C, L), where C is a vector of component states (one for each component) and E is a vector of events, one (possibly empty) event present on each input pin, τ is the transition function mapping an $\mathcal{X}$ state

Table 1: Description of transition actions for transducer d and valve v involved in AS $\mathcal{S}$, along with observations and faults in $Map(\mathcal{S})$.

Component transition	*Action performed*	*Obs*	*Fault*
$d_1 = \langle 0, (\varepsilon, \{op\}), 1 \rangle$	The transducer reacts normally to high temperature by generating the open event	od	ε
$d_2 = \langle 1, (\varepsilon, \{cl\}), 0 \rangle$	The transducer reacts normally to normal temperature by generating the close event	od	ε
$d_3 = \langle 0, (\varepsilon, \{cl\}), 0 \rangle$	The transducer reacts abnormally to high temperature by generating the close event	ε	dfo
$d_4 = \langle 1, (\varepsilon, \{op\}), 1 \rangle$	The transducer reacts abnormally to normal temperature by generating the open event	ε	dfc
$v_1 = \langle 0, (op, \emptyset), 1 \rangle$	The valve reacts normally to the open event by opening	ov	ε
$v_2 = \langle 1, (cl, \emptyset), 0 \rangle$	The valve reacts normally to the close event by closing	ov	ε
$v_3 = \langle 0, (op, \emptyset), 0 \rangle$	The valve reacts abnormally to the open event by remaining closed	ε	vfo
$v_4 = \langle 1, (cl, \emptyset), 1 \rangle$	The valve reacts abnormally to the close event by remaining open	ε	vfc
$v_5 = \langle 0, (cl, \emptyset), 0 \rangle$	The valve reacts normally to the close event by remaining closed	ov	ε
$v_6 = \langle 1, (op, \emptyset), 1 \rangle$	The valve reacts normally to the open event by remaining open	ov	ε
$v_7 = \langle 0, (cl, \emptyset), 1 \rangle$	The valve reacts abnormally to the close event by opening	ov	vop
$v_8 = \langle 1, (op, \emptyset), 0 \rangle$	The valve reacts abnormally to the open event by closing	ov	vcl

and a component transition to another $\mathcal{X}$ state, namely $\tau : \mathbf{X} \times \mathbf{T} \mapsto \mathbf{X}$, while x_0 is the initial state, where all links are assumed to be empty. Starting from x_0, a *trajectory* of $\mathcal{X}$ is a sequence $T = [t_1, \ldots, t_n]$ of component transitions that can occur one after another in $Space(\mathcal{X})$. In other words, T is a string in the regular language of $Space(\mathcal{X})$, that is, a possible mode in which $\mathcal{X}$ may move from one state to another. Since the space of a real AS may be huge, however, we assume that it cannot be materialized; hence, in this paper it is only referenced as a theoretical entity.

Example 2. With reference to AS $\mathcal{S}$ introduced in Example 1, shown on the right side of Figure 1 is $Space(\mathcal{S})$, where each state is represented as an ellipsis marked with a triple (s_d, s_v, e), in which s_d is a state of the transducer, s_v is a state of the valve, and e is an event present on the input pin of v, namely a command sent by d to v via the link. To facilitate referencing, states are marked with numbers $\mathbf{0}..\mathbf{7}$, where $\mathbf{0}$ is the initial state. Each arc (x, t, x'), which exits a state x, is marked with a component transition t, and enters a state x', represents an AS transition $\tau(x, t) = x'$. For instance, $(\mathbf{1}, v_1, \mathbf{6})$ is a transition of $\mathcal{S}$ from state $\mathbf{1}$ to state $\mathbf{6}$ that is triggered by transition v_1 of the valve, which, according to Table 1, opens by consuming event *op* (the opening command sent by the transducer). A trajectory of $\mathcal{S}$ is $T = [d_3, v_5, d_1, v_3, d_4, v_3, d_2, v_5]$, which moves $\mathcal{S}$ through states $\mathbf{2}, \mathbf{0}, \mathbf{1}, \mathbf{7}, \mathbf{1}, \mathbf{7}, \mathbf{2}, \mathbf{0}$ of its space sequentially.

Even though the specification of an AS given above is complete as to its behavior, no information is provided about a distinction between normal and abnormal behavior of the AS, which is essential for the diagnosis task. Moreover, no observability properties are associated with the AS in order to have a clue on its (possibly faulty) behavior when moving within its space. We assume that the AS performs a trajectory that is only partially observable: each component transition can be either *observable* or *unobservable*. We also assume that each component transition is either *normal* or *abnormal*. An observable transition is perceived by the observer of the AS as a label called an *observation*. Also, an abnormal transition is associated with a label called a *fault*. The specification of observability and abnormality properties of an AS $\mathcal{X}$ is given in a *mapping table*, namely $Map(\mathcal{X})$, which associates with each component transition both a (possibly empty) observation and a (possibly empty) fault. An empty observation or fault is denoted by ε. More accurately said, each row of the table is a triple (t, o, f) where t is a component transition, o is an observation, and f is a fault. If o is empty then t is unobservable else t is observable. If f is empty then t is normal else t is abnormal. Hence, a trajectory $T = [t_1, \ldots, t_n]$ of an AS $\mathcal{X}$ is associated with both a *temporal observation*, $\mathcal{O} = [o \mid t \in T, (t, o, f) \in Map(\mathcal{X}), o \neq \varepsilon]$, and a *diagnosis*, $\delta = \{f \mid t \in T, (t, o, f) \in Map(\mathcal{X}), f \neq \varepsilon\}$. It should be noticed that $\mathcal{O}$ is a sequence, while δ is a set (where duplicate faults are removed).

Example 3. With reference to AS $\mathcal{S}$, shaded on the last two columns of Table 1 are observations and faults of component transitions. Consequently, with trajectory $T = [d_3, v_5, d_1, v_3, d_4, v_3, d_2, v_5]$ defined in Example 2, we have $\mathcal{O} = [ov, od, od, ov]$ and $\delta = \{dfo, dfc, vfo\}$.

Given a temporal observation $\mathcal{O}$ of an AS $\mathcal{X}$, a *diagnosis problem* consists in generating the set of diagnoses of the trajectories whose temporal observation is $\mathcal{O}$, called the *candidate set* of $\mathcal{O}$, namely $\Delta(\mathcal{O})$. The qualification 'candidate' stems from the fact that $\mathcal{O}$ is like a symptom that can be associated with different diseases. Similarly, since $\mathcal{O}$ can be generated by several trajectories, several diagnoses can be associated with the 'symptom' $\mathcal{O}$ of $\mathcal{X}$. Model-based diagnosis of ASs is *abductive* in nature: based on $\mathcal{O}$, the diagnosis engine aims to reconstruct the behavior (trajectories) of the AS conforming with $\mathcal{O}$, thereby generating the diagnoses of these trajectories. Even though the set of trajectories conforming with $\mathcal{O}$ may be unbounded owing to unobservable cycles in the space of $\mathcal{X}$, the candidate set of $\mathcal{O}$ is always bounded by the power set of the set $\mathbf{F}$ of faults defined in the mapping table, namely $2^{\mathbf{F}}$.

Example 4. Considering trajectory $T = [d_3, v_5, d_1, v_3, d_4, v_3, d_2, v_5]$ of AS $\mathcal{S}$ along with temporal observation $\mathcal{O} = [ov, od, od, ov]$, based on $Space(\mathcal{X})$ in Figure 1 and $Map(\mathcal{X})$ in Table 1, we can find out that the candidate set $\Delta(\mathcal{O})$ includes six diagnoses, namely: $\{dfo, vfo\}$, $\{dfo, dfc, vfo\}$, $\{dfo, vfo, vop\}$, $\{dfo, dfc, vfo, vop\}$, $\{dfo, vfo, vfc, vop\}$, and $\{dfo, dfc, vfo, vfc, vop\}$.

A systematic method for generating the candidate set is based on the notion of *abduction* of a temporal observation $\mathcal{O}$ of an AS $\mathcal{X}$, which extends each state in the subspace of $Space(\mathcal{X})$ that is involved in a trajectory conforming with $\mathcal{O}$ by both an index i of $\mathcal{O}$, indicating a prefix of $\mathcal{O}$, and a set δ of faults, as formally specified in Definition 1.

Definition 1 (Abduction)**.** *Let* $\mathcal{O} = [o_1, \ldots, o_k]$ *be a temporal observation of an AS* $\mathcal{X}$ *with* $Space(\mathcal{X}) = (\mathbf{T}, \mathbf{X}, \tau, x_0)$*. The* abduction *of* $\mathcal{O}$ *is a finite state machine,* $Abd(\mathcal{O}) = (\mathbf{T}, \mathbf{A}, \tau', a_0, \mathbf{A}_f)$*, where* $\mathbf{A}$ *is the set of states*

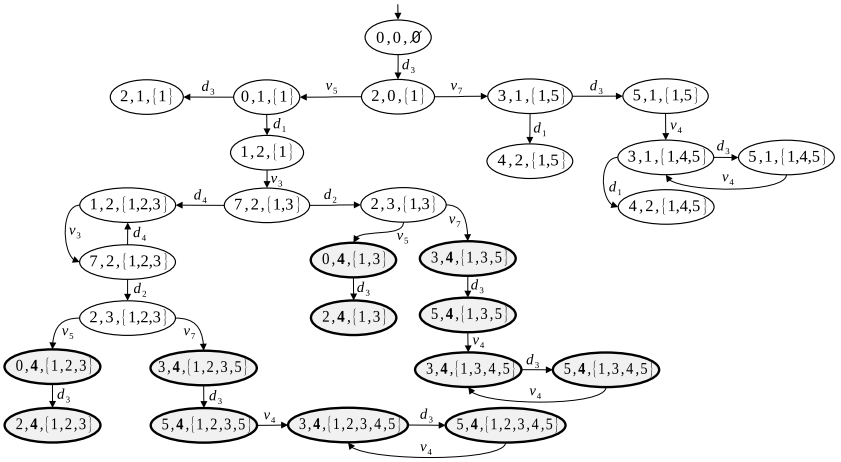


Fig. 2: Abduction $Abd(\mathcal{O})$ for AS $\mathcal{S}$, where $\mathcal{O} = [ov, od, od, ov]$ (cf. $Space(\mathcal{S})$ in Figure 1).

(x, i, δ), *where* $x \in \mathbf{X}$, $i \in [0\,..\,k]$ *is an index of* $\mathcal{O}$, *and* δ *is a set of faults of* $\mathcal{X}$; $a_0 = (x_0, 0, \emptyset)$ *is the initial state;* $\mathbf{A}_f \subseteq \mathbf{A}$ *is the set of final states* (x, k, δ); $\tau' : \mathbf{A} \times \mathbf{T} \mapsto \mathbf{A}$ *is the transition function, where* $\tau'((x, i, \delta), t) = (x', i', \delta')$ *iff* $(t, o, f) \in Map(\mathcal{X})$, *either* o *is empty or* $o = \mathcal{O}[i + 1]$ *where* $i < k$, *if* $o = \varepsilon$ *then* $i' = i$ *else* $i' = i + 1$, *and* $\delta' = \delta \uplus f$, *where the set-extension operator* $\uplus$ *is defined as:*

$$\delta \uplus f = \begin{cases} \delta & \text{if } f = \varepsilon \\ \delta \cup \{f\} & \text{otherwise.} \end{cases}$$

Given $Abd(\mathcal{O})$ having set $\mathbf{A}_f$ of final states, with $\mathcal{O}$ including k observations, the candidate set of $\mathcal{O}$ can be easily derived from $\mathbf{A}_f$ as follows: $\Delta(\mathcal{O}) = \{\delta \mid (x, k, \delta) \in \mathbf{A}_f\}$.

Example 5. Shown in Figure 2 is the abduction of $\mathcal{O} = [ov, od, od, ov]$ for AS $\mathcal{S}$. Each state is depicted as an ellipsis marked with a triple (s, i, δ), where s is a state of $\mathcal{S}$, i is an index of $\mathcal{O}$, and δ is a set of faults of $\mathcal{X}$, with each fault being represented as an integer in range $[1\,..\,6]$ that corresponds to the relative position of the (non empty) fault in the mapping table (cf. last column in Table 1); for instance, number 3 denotes the fault in third relative position, namely *vfo*. Final states, which involve $i = 4$, are shaded. If we collect all the diagnoses involved in the final states, remove duplicates, and decode numbers to fault names, then we come up with the candidate set foreseen in Example 4.

Albeit being effective in providing the correct candidate set, the actual construction of $Abd(\mathcal{O})$ for an AS $\mathcal{X}$ is bound to suffer from performance issues. Specifically, given a state $a = (x, i, \delta)$, the diagnosis engine is required to perform model-based reasoning in order to generate the component transitions exiting state x in $Space(\mathcal{X})$. Then, each of these transitions needs to be checked against $\mathcal{O}$: if a transition t is consistent with $\mathcal{O}$, then a (possibly new) abduction state $a' = (x', i', \delta')$ is created, as well as a new transition $\langle a, t, a' \rangle$ in the abduction. The problem lies in that the same computation needs to be repeated every time a new abduction state (x, i'', δ'') is generated, which shares with a the same AS state x. For example, with reference to Figure 2, there are six abduction states involving $x = 3$, which requires the diagnosis engine to repeat the computation of the transition function of state 3 in $Space(\mathcal{S})$ six times. Inefficiency is not only relevant to computation time (because of repetition of the same model-based reasoning for transition-function generation), but also to memory allocation. With reference to Figure 2, in order to save space, the six abduction states relevant to state $x = 3$, namely $(3, 1, \{1, 5\})$, $(3, 1, \{1, 4, 5\})$, $(3, 4, \{1, 3, 5\})$, $(3, 4, \{1, 2, 3, 5\})$, $(3, 4, \{1, 3, 4, 5\})$, and $(3, 4, \{1, 2, 3, 4, 5\})$, might be condensed into a newly-structured state $(3, \{(1, \{\{1, 5\}, \{1.4.5\}\}), (4, \{\{1, 3, 5\}, \{1, 2, 3, 5\}\{1, 3, 4, 5\}, \{1, 2, 3, 4, 5\}\})\})$ by first grouping the diagnoses δ for the same index i into a set $\mathcal{D}$ and, then, by collecting all the pairs $(i, \mathcal{D})$ into a set $\Im$, thereby, generally speaking, resulting in a single pair $(x, \Im)$, where $\Im = \{(i_1, \mathcal{D}_1), \ldots, (i_n, \mathcal{D}_n)\}$. The idea of such a *condensed abduction* is presented in the next section.

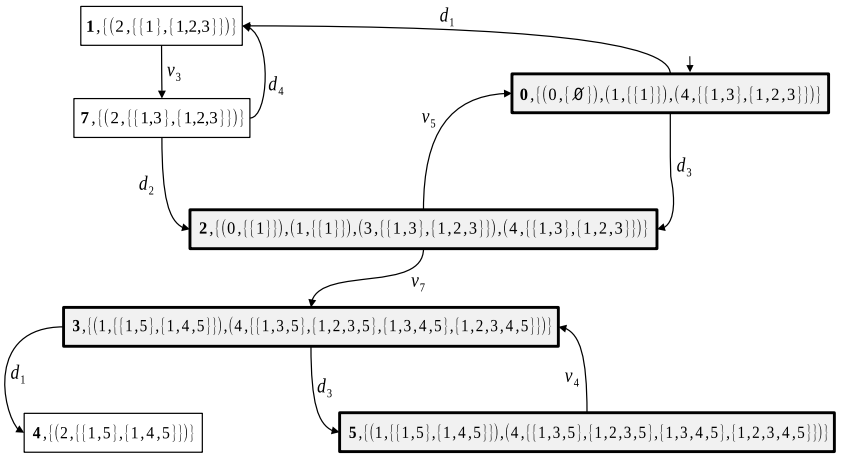


Fig. 3: Condensed abduction $Cab(\mathcal{O})$ for AS $\mathcal{S}$, where $\mathcal{O} = [ov, od, od, ov]$ (cf. $Abd(\mathcal{O})$ in Figure 2).

4. Diagnosis by Condensed Abduction

As pointed out, a major source of computational inefficiency in building the abduction knowledge-structure is the multiple generation of abduction states (x, i, δ) sharing the same AS state x, which causes the diagnosis engine to repeat the same low-level model-based reasoning to generate the transition function of x within the AS space (which is assumed not to be materialized). Ideally, the abduction should be isomorphic to the fragment (states and transitions) of the AS space that is touched by the trajectories conforming with the temporal observation. To this end, we need a new notion of abduction in which states (x, i, δ) in $Abd(\mathcal{O})$ are grouped into a single state $(x, \{(i_1, \mathcal{D}_1), \ldots, (i_n, \mathcal{D}_n)\})$, where each pair $(i_j, \mathcal{D}_j)$ relates an index i_j of $\mathcal{O}$ with the whole set $\mathcal{D}_j$ of diagnoses associated with i_j in the states of $Abd(\mathcal{O})$, thereby avoiding replicating abduction states with same system state x. Obviously, there is no point in performing that sort of abduction 'condensation' *after* the generation of $Abd(\mathcal{O})$. Hence, we introduce a new sort of abduction in Definition 2 and, then, present a new algorithm to generate it efficiently.

Definition 2 (Condensed Abduction)**.** *Let $Abd(\mathcal{O})$ be an abduction for an AS $\mathcal{X}$ having initial state x_0, with* **O** *and* **F** *being the set of observations and the set of faults in $Map(\mathcal{X})$. The* condensed abduction *of $\mathcal{O}$ is a finite state machine,*

$$Cab(\mathcal{O}) = (\mathbf{T}, \Xi, \tau, \xi_0, \Xi_f) \tag{1}$$

where, as in $Space(\mathcal{X})$, the alphabet **T** *is the set of component transitions; Ξ is the set of states $(x, \Im)$, where x is a state of $\mathcal{X}$ and $\Im$ is a set of pairs $(i, \mathcal{D})$, where i is an index of $\mathcal{O}$ and $\mathcal{D}$ is a set of diagnoses; $\tau : \Xi \times \mathbf{T} \mapsto \Xi$ is the transition function; ξ_0 is the initial state; and $\Xi_f \subseteq \Xi$ is the set of final states. Specifically, $Cab(\mathcal{O})$ may be derived from $Abd(\mathcal{O})$ by applying the following construction rules inductively:*

- *If $\mathbb{A}$ is the set of states (x, i, δ) in $Abd(\mathcal{O})$ sharing a state x of $\mathcal{X}$, where $\mathbb{I} = \{i \mid (x, i, \delta) \in \mathbb{A}\}$ is the relevant set of indexes, then Ξ includes a state $(x, \Im)$ where $\Im = \{(i, \mathcal{D}) \mid i \in \mathbb{I}, \mathcal{D} = \{\delta \mid (x, i, \delta) \in \mathbb{A}\}\}$.*
- *If $\langle (x, i, \delta), t, (x', i', \delta') \rangle$ is a transition in $Abd(\mathcal{O})$, then τ includes a transition $\langle (x, \Im), t, (x', \Im') \rangle$, where $(i, \mathcal{D}) \in \Im$ and $(i', \mathcal{D}') \in \Im'$.*
- *$\xi_0 = (x_0, \Im_0)$, where $(0, \{\emptyset\}) \in \Im_0$.*
- *$\Xi_f = \{(x, \Im) \mid (x, \Im) \in \Xi, (k, \mathcal{D}) \in \Im\}$, where k is the length (number of observations) of $\mathcal{O}$.*

Example 6. Shown in Figure 3 is the condensed abduction $Cab(\mathcal{O})$ for AS $\mathcal{S}$ (with final states being shaded), which is derived from the abduction $Abd(\mathcal{O})$ displayed in Figure 2 based on Definition 2. As expected, the topology of $Cab(\mathcal{O})$ matches the space of $\mathcal{S}$ outlined on the right side of Figure 1, where each state $(x, \Im)$ in $Cab(\mathcal{O})$ matches a state x in

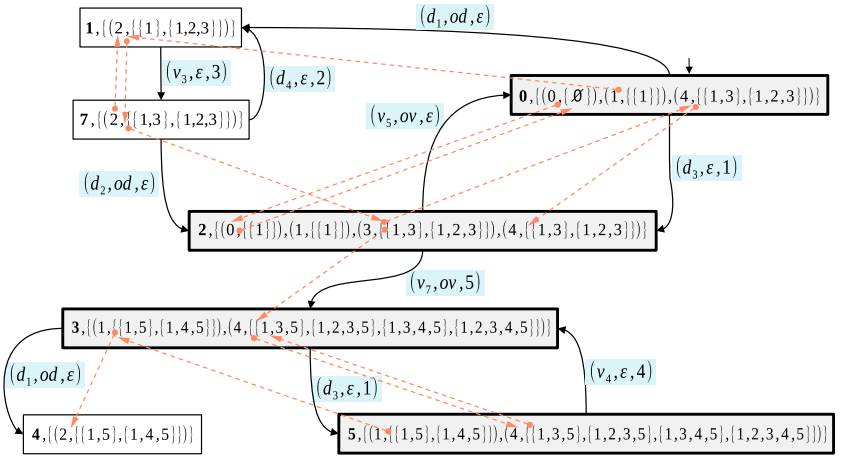


Fig. 4: Improved condensed abduction $Cab^*(\mathcal{O})$ for $\mathcal{S}$ generated by the CONDENSED ABDUCTION ENGINE algorithm (cf. $Cab(O)$ in Figure 3).

$Space(\mathcal{S})$, and each transition $\langle(x,\Im)),t,(x',\Im')\rangle$ in $Cab(\mathcal{O})$ matches a transition $\langle x,t,x'\rangle$ in $Space(\mathcal{S})$. Moreover, for each state $(x,\Im)$, $\Im$ is the whole set of $\mathcal{O}$-indexes and relevant set of diagnoses collected from all abduction states (x,i,δ). For instance, in the condensed-abduction state $(0,\Im)$, $\Im=\{(0,\{\emptyset\}),(1,\{\{1\}\}),(4,\{\{1,3\},\{1,2,3\}\})\}$ is in fact the condensation of four abduction states, namely $(0,0,\emptyset)$, $(0,1,\{1\})$, $(0,4,\{1,3\})$, and $(0,4,\{1,2,3\})$. On the other hand, not all states in $Space(\mathcal{S})$ are matched by a state in $Cab(\mathcal{O})$. For instance, space state 6 is not matched by a corresponding state $(6,\Im)$ in $Cab(\mathcal{O})$. This is also true for transitions: space transition $\langle 4,v_8,7\rangle$ is unmatched in $Cab(\mathcal{O})$. The reason is that, similarly to $Abd(\mathcal{O})$, $Cab(\mathcal{O})$ matches only the trajectories that conform with $\mathcal{O}$, while $Space(\mathcal{S})$ embodies all possible trajectories of $\mathcal{S}$.

Proposition 1. *Let $Cab(\mathcal{O})=(\mathbf{T},\Xi,\tau,\xi_0,\Xi_f)$, with k being the length of $\mathcal{O}$. The candidate set of $\mathcal{O}$ is*

$$\Delta(\mathcal{O})=\bigcup_{(x,\Im)\in\Xi_f,(k,\mathcal{D})\in\Im}\mathcal{D}. \tag{2}$$

Example 7. Based on $Cab(\mathcal{O})$ shown in Figure 3 (where final states are shaded), according to eqn. (2), we have $\Delta(\mathcal{O})=\{\{1,3\},\{1,2,3\},\{1,3,5\},\{1,2,3,5\},\{1,3,4,5\},\{1,2,3,4,5\}\}$, which equals the set of candidates distilled from $Abd(\mathcal{O})$ in Example 5.

5. Condensed Abduction Engine

For the diagnosis engine to generate a condensed abduction efficiently, an extension of $Cab(\mathcal{O})$ is in order, as specified in Definition 3.

Definition 3 (Improved Condensed Abduction)**.** *Let $Cab(\mathcal{O})$ be a condensed abduction relevant to an AS $\mathcal{X}$. The* improved condensed abduction *of $\mathcal{O}$, denoted $Cab^*(\mathcal{O})$, is a graph obtained by extending $Cab(\mathcal{O})$ as follows:*

- *Each transition $\langle\xi,t,\xi'\rangle$ in $Cab(\mathcal{O})$ is replaced by transition $\langle\xi,(t,o,f),\xi'\rangle$, where $(t,o,f)\in Map(\mathcal{X})$.*
- *If there is a transition $\langle(x,i,\delta),t,(x',i',\delta')\rangle$ in $Abd(\mathcal{O})$, then there is an arc from $(i,\mathcal{D})$ to $(i',\mathcal{D}')$ in $Cab^*(\mathcal{O})$, where $\langle(x,\Im),(t,o,f),(x',\Im')\rangle$ is a transition in $Cab^*(\mathcal{O})$, $(i,\mathcal{D})\in\Im$, $\delta\in\mathcal{D}$, $(i',\mathcal{D}')\in\Im'$, and $\delta'\in\mathcal{D}'$.*

Example 8. Displayed in Figure 4 is the improved condensed abduction $Cab^*(\mathcal{O})$ obtained from the condensed abduction $Cab(\mathcal{O})$ (shown in Figure 3) based on Definition 3, where arcs are depicted as dashed arrows.

The CONDENSED ABDUCTION ENGINE algorithm listed below (lines 1–26) takes as input a temporal observation $\mathcal{O}$ of an AS $\mathcal{X}$, and generates the improved condensed abduction $Cab^*(\mathcal{O})$. To this end, AS states x, indexes i,

Algorithm 1: CONDENSED ABDUCTION ENGINE

input : $\mathcal{O} = [o_1, \ldots, o_k]$, a temporal observation of an AS $\mathcal{X}$ having initial state x_0
output: $Cab^*(\mathcal{O})$, the improved condensed abduction of $\mathcal{O}$

1 $\xi_0 \leftarrow (x_0^\circ, \Im_0)$, where ξ_0 is the initial state, with $\Im_0 = \{(0^\circ, \{\emptyset^\circ\})\}$
2 **while** *there is a state* $\xi = (x^\circ, \Im)$ *where* $(i^\circ, \mathcal{D}) \in \Im$ **do**
3 **foreach** *transition* $\langle x, t, x' \rangle$ *in Space*$(\mathcal{X})$*, with* $(t, o, f) \in Map(\mathcal{X})$**, do**
4 **if** *there is not a state* $\xi' = (x', \Im')$ **then**
5 Generate a state $\xi' = (x'^\circ, \emptyset)$
6 Insert a transition $\langle \xi, (t, o, f), \xi' \rangle$, where ξ' is the state involving the space state x' of $\mathcal{X}$
7 Mark x° as $x^\bullet$
8 **while** *there is a state* $\xi = (x^\bullet, \Im)$ *where* $(i^\circ, \mathcal{D}) \in \Im$ **do**
9 **foreach** $(i^\circ, \mathcal{D}) \in \Im$ **do**
10 **foreach** *transition* $\langle \xi, (t, o, f), (x', \Im') \rangle$ **do**
11 **if** $o = \varepsilon$ **or** $(i < k$ **and** $\mathcal{O}[i+1] = o)$ **then**
12 $i' \leftarrow$ **if** $o = \varepsilon$ **then** i **else** $i + 1$
13 **if** $(i', \mathcal{D}') \notin \Im'$ **then**
14 Insert $(i'^\circ, \mathcal{D}')$ into $\Im'$, where $\mathcal{D}' = \emptyset$
15 Insert an arc from $(i, \mathcal{D})$ to $(i', \mathcal{D}')$
16 Mark i° as $i^\bullet$
17 **while** *there is a state* $\xi = (x^\bullet, \Im)$ *where* $(i^\bullet, \mathcal{D}) \in \Im, \delta^\circ \in \mathcal{D}$ **do**
18 **foreach** $\delta^\circ \in \mathcal{D}$*, where* $(i^\bullet, \mathcal{D}) \in \Im$ **do**
19 **foreach** *transition* $\langle \xi, (t, o, f), (x', \Im') \rangle$ **do**
20 $\delta' \leftarrow \delta \uplus f$
21 **foreach** *arc from* $(i, \mathcal{D})$ *to* $(i', \mathcal{D}')$ **do**
22 **if** $\delta' \notin \mathcal{D}'$ **then**
23 Insert δ'° into $\mathcal{D}'$
24 Mark δ° as $\delta^\bullet$
25 Remove states $(x, \emptyset)$ along with their entering transitions
26 Remove the transitions $\langle (x, \Im), (t, o, f), (x', \Im') \rangle$ where there are no arcs from any pair in $\Im$ to any pair in $\Im'$.

and diagnoses δ within states $(x, \Im)$ are possibly marked during the computation. To avoid confusion, a blank circle indicates no marking, as in x°, while a bullet indicates a marking, as in $\delta^\bullet$. After the (partial) generation of the initial state ξ_0 (line 1), a loop is performed (lines 2–24) as long as there is a state $\xi = (x^\circ, \Im)$ where $\Im$ includes a pair $(i^\circ, \mathcal{D})$, in other words, while there is a state ξ whose transition function is not yet materialized, provided that there is an index i° involved in $\Im$ that has not yet been processed. The body of the loop is composed of three sections.

In the first section (lines 3–7), the transitions exiting ξ are generated, possibly by creating new target states $\xi' = (x'^\circ, \emptyset)$, where the set $\Im$ of pairs $(i, \mathcal{D})$ associated with x'° is empty. Eventually, the unmarked state x° is marked. In the second section (lines 8–16), all unmarked indexes i° within pairs in $\Im$ involved in states $\xi = (x^\bullet, \Im)$ are processed based on a triple (t, o, f) marking a transition exiting ξ, thereby generating a (possibly incremented) index i'. If a pair $(i', \mathcal{D}')$ does not exist in ξ' (the target state of the transition exiting ξ), then it is created with an empty set $\mathcal{D}'$. In either case, an arc from $(i, \mathcal{D})$ to $(i', \mathcal{D}')$ is inserted, to be possibly exploited later. Eventually, the unmarked index i° is marked. In the third section (lines 17–24), in a way similar to the previous section, all unmarked diagnoses δ° in a set $\mathcal{D}$ associated with a marked index $i^\bullet$ involved in a set $\Im$ associated with a marked state $x^\bullet$ are processed based on a triple (t, o, f) marking a transition exiting ξ, thereby generating a (possibly extended) diagnosis δ'. It is at this point that an arc from $(i, \mathcal{D})$ to $(i', \mathcal{D}')$ comes into play: $\mathcal{D}'$ is possibly extended with the (unmarked) target diagnosis δ'°. Eventually, the unmarked diagnosis δ° is marked. When the loop condition in line 2 becomes false, it means that all

states, transitions, and arcs in $Cab^*(\mathcal{O})$ are complete. It only remains to remove the states with empty $\Im$ and relevant entering transitions (line 25), as well as the transitions $\langle(x, \Im), (t, o, f), (x', \Im')\rangle$ where no arc exists from any pair in $\Im$ to any pair in $\Im'$ (line 26). In fact, in the first case, the system state x is not involved in $Abd(\mathcal{O})$; in the second case, no transition exiting a system state x and marked with component transition t is involved in the abduction.

Table 2: Performance comparison between $Abd(\mathcal{O})$ and $Cab^*(\mathcal{O})$.

Parameter	$Abd(\mathcal{O})$	$Cab^*(\mathcal{O})$	*Saving*	*Limit of saving for* $\omega \to \infty$
States	$\sigma(\omega+1)\Delta$	σ	$1-1/((\omega+1)\Delta)$	1
Transitions	$\sigma(\omega+1)\Delta\beta$	$\sigma\beta$	$1-1/((\omega+1)\Delta)$	1
Arrows	$\sigma(\omega+1)\Delta\beta$	$\sigma\beta+\sigma(\omega+1)$	$1-(\beta+\omega+1)/((\omega+1)\Delta\beta)$	$1-1/(\Delta\beta)$
Indexes	$\sigma(\omega+1)\Delta$	$\sigma(\omega+1)$	$1-1/\Delta$	$1-1/\Delta$
Diagnoses	$\sigma(\omega+1)\Delta$	$\sigma(\omega+1)\Delta$	0	0

6. Performance Analysis

A worst-case (theoretical) performance analysis is reported in Table 2, where $Abd(\mathcal{O})$ and $Cab^*(\mathcal{O})$ are compared based on five parameters: *states*, *transitions*, *arrows* (transitions plus arcs where applicable), *indexes*, and *diagnoses*. For each parameter, four expressions are given, which involve four variables: σ, the number of system states, ω, the length of $\mathcal{O}$, $\Delta = 2^{|\mathbf{F}|}$, the number of diagnoses where $\mathbf{F}$ is the set of fault labels, and β (branching), the number of component transitions exiting a system state in its space. The expressions in second and third columns indicate the number of instances of the relevant parameter that are generated in $Abd(\mathcal{O})$ and $Cab^*(\mathcal{O})$, respectively. The value of *Saving* is the percentage, expressed in range $[0\,..\,1]$, of the number of parameter instances saved in $Cab^*(\mathcal{O})$ with respect to $Abd(\mathcal{O})$, which is defined as $(\wp_a - \wp_c)/\wp_a = 1 - \wp_c/\wp_a$, where $\wp_a$ and $\wp_c$ are the number of instances of $\wp$ generated in $Abd(\mathcal{O})$ and $Cab^*(\mathcal{O})$, respectively. For example, in the worst case, the number of states (x, i, δ) in $Abd(\mathcal{O})$ is the product $\sigma(\omega+1)\Delta$, while, owing to condensation, this number equals σ in $Cab^*(\mathcal{O})$. Hence, the saving is $1-\sigma/(\sigma(\omega+1)\Delta) = 1-1/((\omega+1)\Delta)$. The last column in the table indicates the theoretical limit of the saving when ω becomes increasingly larger. Note how the whole set of diagnoses stored in abduction states is the same as those in $Cab^*(\mathcal{O})$, namely $\sigma(\omega+1)\Delta$, thereby leading to a zero saving.

To grasp the magnitude of expressions in Table 2, displayed in Figure 5 is a chart that shows the saving for states, arrows, and indexes, by assuming $\Delta = 4$ (two faults), $\beta = 2$ (two transitions exiting each system state), and varying length ω of $\mathcal{O}$. For states, the saving quickly tends to 1 (for instance, with $\omega = 9$ the saving is 97.5%), for arrows it tends to $1 - 1/8 = 87.5\%$ (with $\omega = 9$ the saving is 85%), while for indexes it remains fixed to $1 - 1/4 = 75\%$. Although encouraging, these are only theoretical figures in a worst-case scenario, which need to be corroborated by real experimental results in future work. For the time being, it suffices to compare Table 2 with the numbers relevant to our reference example for AS $\mathcal{S}$, which are derived form $Space(\mathcal{S})$ in Figure 1, $Abd(\mathcal{O})$ in Figure 2, and $Cab^*(\mathcal{O})$ in Figure 4. Here, the actual parameters are: $\sigma = 8$, $\Delta = 2^6 = 64$, $\beta = 2$, and $\omega = 4$. Doing the calculations, it turns out that the theoretical saving for states, transitions, and arrows is 99%, while for indexes is 98%. Instead, the real saving is 75% for states, 67% for transitions, 25% for arrows, and 53% for indexes (not so bad, though).

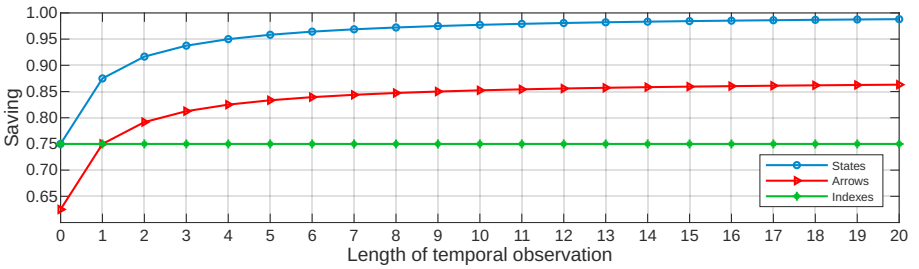


Fig. 5: Saving in generating $Cab^*(\mathcal{O})$ rather than $Abd(\mathcal{O})$, where $\Delta = 4$, $\beta = 2$, and varying length of $\mathcal{O}$ (cf. Table 2).

7. Conclusion

The motivation for this paper lies in that a critical problem in model-based diagnosis of DESs is performance, which is a consequence of the explosive number of DES states to be generated by the diagnosis engine before the distillation of candidate diagnoses. In abduction-based diagnosis of ASs, this translates into a huge number of abduction states. In Section 3, we outlined the current technique for generating the abduction of an AS temporal observation. Then, with the help of a reference example, we pointed out the intrinsic redundancy of the abduction structure, in both memory allocation and model-based reasoning. The actual contribution of the paper starts in Section 4, where the notion of a condensed abduction was introduced. To generate a condensed abduction (improved by additional information), the pseudo-code of a CONDENSED ABDUCTION ENGINE was formalized in Section 5. Eventually, a worst-case performance analysis was presented in Section 6 where, based on several parameters, the (improved) condensed abduction produced by the new diagnosis engine was compared with a classical (redundant) abduction. Although the theoretical results of the performance analysis are more than encouraging, the implementation and experimentation of the new engine is in order for the validation of such results. As to future work, besides the implementation and experimentation of the CONDENSED ABDUCTION ENGINE, we expect to extend the technique presented in this paper to other approaches to diagnoses of ASs, including hybrid diagnosis [3], sequence-oriented diagnosis [12], and minimal diagnosis [13].

References

[1] Baroni, P., Lamperti, G., Pogliano, P., Zanella, M., 1999. Diagnosis of large active systems. Artificial Intelligence 110, 135–183. doi:10.1016/S0004-3702(99)00019-3.

[2] Basile, F., 2014. Overview of fault diagnosis methods based on Petri net models, in: Proceedings of the 2014 European Control Conference, ECC 2014, pp. 2636–2642. doi:10.1109/ECC.2014.6862631.

[3] Bertoglio, N., Lamperti, G., Zanella, M., Zhao, X., 2020. Explanatory diagnosis of discrete-event systems with temporal information and smart knowledge-compilation, in: Calvanese, D., Erdem, E., Thielsher, M. (Eds.), Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning (KR 2020). IJCAI Organization, pp. 130–140. doi:10.24963/kr.2020/14.

[4] Brand, D., Zafiropulo, P., 1983. On communicating finite-state machines. Journal of the ACM 30, 323–342. doi:10.1145/322374.322380.

[5] Cabasino, M.P., Giua, A., Seatzu, C., 2010. Fault detection for discrete event systems using Petri nets with unobservable transitions. Automatica 46, 1531–1539.

[6] Cassandras, C., Lafortune, S., 2008. Introduction to Discrete Event Systems. second ed., Springer, New York.

[7] Cong, X., Fanti, M., Mangini, A., Li, Z., 2018. Decentralized diagnosis by Petri nets and integer linear programming. IEEE Transactions on Systems, Man, and Cybernetics: Systems 48, 1689–1700. doi:10.1109/TSMC.2017.2726108.

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

[9] Jiroveanu, G., Boel, R., Bordbar, B., 2008. On-line monitoring of large Petri net models under partial observation. Journal of Discrete Event Dynamic Systems 18, 323–354.

[10] de Kleer, J., Williams, B., 1987. Diagnosing multiple faults. Artificial Intelligence 32, 97–130. doi:10.1016/0004-3702(87)90063-4.

[11] Kwong, R., Yonge-Mallo, D., 2011. Fault diagnosis in discrete-event systems: incomplete models and learning. IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics 41, 118–130.

[12] Lamperti, G., Trerotola, S., Zanella, M., Zhao, X., 2023. Sequence-oriented diagnosis of discrete-event systems. Journal of Artificial Intelligence Research 78, 69–141. doi:10.1613/jair.1.14630.

[13] Lamperti, G., Zanella, M., 2024. Minimalist Diagnosis of Discrete-Event Systems, in: Pill, I., Natan, A., Wotawa, F. (Eds.), 35th International Conference on Principles of Diagnosis and Resilient Systems (DX 2024), Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany. pp. 12:1–12:20. URL: https://drops.dagstuhl.de/entities/document/10.4230/OASIcs.DX.2024.12, doi:10.4230/OASIcs.DX.2024.12.

[14] Lamperti, G., Zanella, M., Zhao, X., 2018. Introduction to Diagnosis of Active Systems. Springer, Cham. doi:10.1007/978-3-319-92733-6.

[15] McIlraith, S., 1997. Explanatory diagnosis: conjecturing actions to explain observations, in: Eighth International Workshop on Principles of Diagnosis (DX 1997), Mont St. Michel, France.

[16] Pencolé, Y., Cordier, M., 2005. A formal framework for the decentralized diagnosis of large scale discrete event systems and its application to telecommunication networks. Artificial Intelligence 164, 121–170.

[17] Reiter, R., 1987. A theory of diagnosis from first principles. Artificial Intelligence 32, 57–95. doi:10.1016/0004-3702(87)90062-2.

[18] Sampath, M., Sengupta, R., Lafortune, S., Sinnamohideen, K., Teneketzis, D., 1996. Failure diagnosis using discrete-event models. IEEE Transactions on Control Systems Technology 4, 105–124.