Knowledge Compilation Techniques for Model-Based Diagnosis of Complex Active Systems

Gianfranco Lamperti[1(✉)], Marina Zanella[1], and Xiangfu Zhao[2]

[1] Department of Information Engineering, University of Brescia, Brescia, Italy
gianfranco.lamperti@unibs.it

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

Abstract. According to complexity science, the essence of a complex system is the emergence of unpredictable behavior from interaction among components. Loosely inspired by this idea, a diagnosis technique of a class of discrete-event systems, called complex active systems, is presented. A complex active system is a hierarchical graph, where each node is a network of communicating automata, called an active unit. Specific interaction patterns among automata within an active unit give rise to the occurrence of emergent events, which may affect the behavior of superior active units. This results in the stratification of the behavior of the complex active system, where each different stratum corresponds to a different abstraction level of the emergent behavior. As such, emergence is a peculiar property of a complex active system. To speed up the diagnosis task, model-based knowledge is compiled offline and exploited online by the diagnosis engine. The technique is sound and complete.

Keywords: Knowledge compilation · Complex system
Emergent behavior · Discrete-event system
Active system · Model-based diagnosis · Lazy techniques

1 Introduction

In an interview in January 2000, the physicist Stephen Hawking was asked the following question [32]: *Some say that while the 20th century was the century of physics, we are now entering the century of biology. What do you think of this?* Professor Hawking replied: *I think the next century will be the century of complexity.* Colloquially, we use the qualifier "complex" to indicate something that is complicated in nature. In this perspective, a *complex system* is complicated in structure and behavior, such as an aircraft or a nuclear power plant. However, according to complexity science [1,2,8,24], although it is likely to be complicated, a complex system does not equate to a complicated system. In this different perspective, a complex system is composed of several individual components that, once aggregated, assume collective characteristics that are not manifested, and cannot be predicted from the properties of the individual components. So, a human being is much more than the union of some 100 trillion cells that make up her body. Likewise, a cell of a human body is much more than the union of its molecules. What we think as a human being, in terms of personality and character, is in fact a collective manifestation of the different interactions among the neurons and synapses in the brain. On their turn, these are in continuous interaction with the other cells of the body, possibly with clusters of cells that constitute organs, which themselves are complex systems. Locally, each cell is characterized by its specific behavior and interaction rules, globally resulting in the collective manifestation of the human being.

Complexity science questions the traditional reductionist approach adopted in the natural sciences, namely the reduction of complex natural phenomena to several simple processes, and the application of a simple theory to each process. More specifically, the *principle of superimposition* is no longer accepted, namely that the comprehension of the whole phenomenon relies on the superimposition of its parts. Based on this principle, for instance, if you understand the theory of an elementary particle, then you will understand every natural phenomenon. Likewise, if you understand DNA, then you will comprehend all biological phenomena. By contrast, a significant aspect of complex systems is that a new collective level emerges through interactions between autonomous elements. Hence, in order to comprehend the complex system, additional knowledge is required beyond the comprehension of the single elements. More generally, a complex system is structured in a hierarchy of semi-autonomous subsystems, with the behavior of a subsystem at level i of the hierarchy being affected, although not completely determined, by the behavior of each subsystem at level $i-1$. As such, the behavior of a complex system is *stratified*.

Loosely inspired by complexity science, this paper presents a method to extract knowledge - above all, about emergent behavior - from the models of individual clusters of (component) systems and to exploit this knowledge for the lazy diagnosis of a class of discrete event systems (DESs) [5], called *complex active systems* (CASs). A sort of CASs was first introduced in [20,21]; however, the relevant model-based diagnosis task proved naive. Subsequently, a viable diagnosis technique was presented in [17,23] and extended in [22]. To the best of our knowledge, apart from the works cited above, no approach to diagnosis of DESs (much less to diagnosis of static systems) based on the complexity paradigm has been proposed so far. Moreover, none of the complexity-inspired approaches cited above is based on knowledge compilation. Still, several works can be related to this paper in varying degree, as discussed below.

An approach is described in [14], where the notion of a *supervision pattern* is introduced for flat DESs, which allows for a flexible specification of the diagnosis problem. However, the events associated with supervision patterns are not exploited for defining any emergent behavior. Diagnosis of hierarchical finite state machines (HFSMs), which are inspired by state-charts [9], provides a sort of structural complexity. Diagnosis of HFSMs has been considered in [13,26].

However, complexity is merely confined to structure, without emergent events or behavior stratification. An algorithm for computing minimal diagnoses of tree-structured systems is presented in [31]. Subdiagnoses are generated by traversing top-down the tree, which are eventually combined to yield the candidate diagnoses of the whole system. However, despite the fact that the considered systems are static and the diagnosis method is consistency-based, neither complexity nor emergent behavior is conceived, as the goal of the technique is efficiency of the diagnosis task by exploitation of the structure of the system. An approach to consistency-based diagnosis of static systems supported by structural abstraction (which aggregates components to represent the system at different levels of structural detail) is described in [6] as a remedy of computational complexity of model-based diagnosis. Evidence from experimental evaluation indicates that, on average, the technique performs favorably with the algorithm of Mozetič [25]. Still, no emergent behavior is conceived. A technique for consistency-based diagnosis of multiple faults in combinational circuits is presented in [30]. To scale the diagnosis to large circuits, a technique for hierarchical diagnosis is proposed. An implementation on top of the tool presented in [12], which assumes that the system model has been compiled into propositional formulas in decomposable negation normal form (DNNF), has demonstrated the effectiveness of the approach. However, neither emergent behavior nor behavior stratification is conceived, and the technique addresses static systems only. A scalable technique for diagnosability checking of DESs is proposed in [28]. In contrast with the method presented in [27], which suffers from exponential complexity, new algorithms with polynomial complexity were proposed in [15,33], called the *twin plant* method. However, the construction of a global twin plant, which corresponds to the synchronization based on observable events of two identical instances of the automaton representing the whole DES behavior, is often impractical. The method proposed in [28] exploits the distribution of a DES to build a local twin plant for each component. Then, the DES components (and their relevant local twin plants) are organized into a *jointree*, a classical tool adopted in various fields of Artificial Intelligence, including probabilistic reasoning [11,29] and constraint processing [7]. The transformation of the DES into a jointree is an artifice for reducing the complexity of the diagnosability analysis task. Neither emergent behavior nor behavior stratification is assumed for the DES, nor any knowledge extraction is performed. A variant of the decentralized/distributed approach to diagnosis of DESs is introduced in [16], with the aim of computing local diagnoses which are globally consistent. To this end, as in [28] but in the different perspective of diagnosis computation rather than diagnosability, a technique based on jointrees (called *junction trees* in the paper) is proposed. The goal is to mitigate the complexity of model-based diagnosis of DESs associated with abduction-based elicitation of system trajectories. However, the goal of [16] is the efficiency of the diagnosis task, which is performed online only, thereby without exploiting any compiled knowledge. The transformation of the DES into a junction tree is a technical stratagem for improving the decentralized/distributed approach to diagnosis of DESs. The DES is plain, with no emergent behavior.

The contribution of the present paper is threefold: (a) specification of a CAS based on active units, (b) proposal of a process for extracting knowledge from active units, and (c) specification of a diagnosis task for CASs exploiting compiled knowledge.

2 Complex Active Systems

A CAS can be defined bottom-up, starting from its atomic elements, the *active components* (ACs). The behavior of an AC, which is equipped with input/output *terminals*, is defined by a communicating automaton [3]. The transition function moves the AC from one state to another when a specific *input event* is *ready* at an input terminal. In so doing, the transition possibly generates a set of *output events* at several output terminals. If specified so, an AC can perform a transition without the need for a ready event; formally, the transition is triggered by the "ε" *empty* event.[1] ACs communicate with one another through *links*. A link is a connection between an output terminal o of an AC c and an input terminal i' of another AC c'. When an event e is generated at o by a transition in c, e becomes ready at terminal i' of c'. At most one link can be connected with a terminal.

When several ACs are connected by links, the resulting network is an *active system* (AS) [18,19]. A state of an AS is a pair (S, E), where S is the array of states of the components in the AS and E is the array of (possibly empty) events that are ready at the input terminals of the components. A state of the AS is *quiescent* iff all elements in E are ε (no event is ready). A *trajectory* T of an AS is a finite sequence of transitions of ACs in the AS, namely $T = [t_1(c_1), \ldots, t_q(c_q)]$, which moves the AS from an *initial* quiescent state to a *final* quiescent state. In an AS, a terminal that is not connected with any link is *dangling*.

An *active unit* (AU) is an AS extended by a set of input terminals, a set of output terminals, and a set of *emergent events*, where the input terminals are the dangling input terminals of the AS. Each emergent event is a pair $(e(o), r)$, where e is an event generated at the output terminal o of the AU and r is a regular expression whose alphabet is a subset of the transitions of the ACs involved in the AU. The event $e(o)$ *emerges* from a trajectory[2] of the AU when a sequence of transitions in the trajectory matches r. The state of recognition of an emergent event $(e(o), r)$ is maintained by a *matcher*, namely a deterministic finite automaton (DFA), derived from r, in which each final state corresponds to the occurrence of $e(o)$. Remarkably, the notion of a matcher of r differs from that of a recognizer of r, as illustrated below.

Example 1. Let $(e(o), r)$ be an emergent event in an AU $\mathcal{U}$, with $\Sigma = \{t_1, t_2, t_3\}$ being the alphabet of the regular expression

$$r = t_1\, t_2^+\, t_3 \,|\, t_2\, t_3^+\, t_1. \tag{1}$$

[1] Transitions triggered by empty events are typically used for modeling state changes caused by external events (e.g. a lightning may cause the reaction of a sensor in a protection system).

[2] The notion of a trajectory defined for an AS is also valid for the AU incorporating the AS.

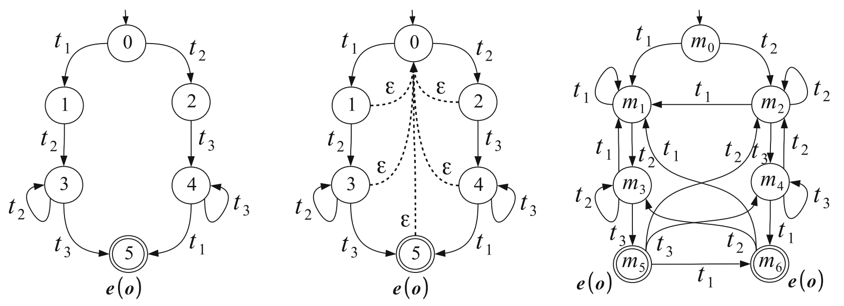

Fig. 1. The recognizer of $r = t_1\, t_2^+\, t_3 \mid t_2\, t_3^+\, t_1$, where $\Sigma = \{t_1, t_2, t_3\}$ (left); the NFA obtained by the insertion of ε-transitions (center); and the matcher $\mu(e(o), r)$ (right).

Depicted on the left side of Fig. 1 is the *recognizer* of r, a DFA with the final state being marked by the emergent event $e(o)$. When the final state 5 is reached, the emergent event e is generated at the output terminal o of $\mathcal{U}$. Assume the following trajectory T in $\mathcal{U}^*$,

$$T = [\, t_3, \overbrace{t_1, \rlap{$\underbrace{}_{T''}$}t_2, t_3}^{T'}, t_1, t_3 \,]. \tag{2}$$

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

Assume further to trace the occurrences of $e(o)$ based on the recognizer of r displayed on the left side of Fig. 1. The sequence of states of this recognizer is outlined in the second row of Table 1, where each state is reached by the corresponding transition of T listed in the first row of the table. As indicated in the third row, the emergent event $e(o)$ is correctly generated at the fourth transition in T, which is the last transition of the subtrajectory T'. At this point, since no transition exits the final state 5, the recognizer starts again from its initial state 0 in order to keep matching. It first changes state to 1 in correspondence of t_1, and with t_3 (mismatch) it returns to 0. The result is that, owing to the overlapping of T' and T'', the second $e(o)$ is not produced.

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

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

The final states of the minimized DFA are marked by the emergent event $e(o)$.

Table 1. Generation of emergent events with overlapping.

Transitions in T	t_3	t_1	t_2	t_3	t_1	t_3
States of the recognizer of r	0	1	3	5	1	0
Emergent events				$e(o)$		
States of the matcher $\mu(e(o), r)$	m_0	m_1	m_3	m_5	m_6	m_0
Emergent events				$e(o)$	$e(o)$	

Example 2. With reference to Example 1, consider the recognizer of the regular expression r displayed on the left side of Fig. 1. Outlined on the center is the NFA obtained by inserting five ε-transitions (dashed arrows) toward the initial state (step 1). The DFA resulting from the determinization of the NFA is displayed on the right side of the figure (step 2). Incidentally, this DFA is also minimal, thus minimization (step 3) is not applied. In conclusion, the DFA on the right side of Fig. 1 is the matcher $\mu(e(o), r)$, with the final states m_5 and m_6 being marked by $e(o)$. The dynamics of the matching performed by $\mu(e(o), r)$ on the trajectory T is outlined in the last two rows of Table 1. In sharp contrast with the matching performed by the recognizer of r, which produces only one $e(o)$, the matcher correctly generates the emergent event twice, based on the two overlapping subtrajectories of T, namely $T' = [\,t_1, t_2, t_3\,]$ and $T'' = [\,t_2, t_3, t_1\,]$. Unlike the recognizer of r, after reaching the state m_5 and generating $e(o)$, the next transition t_1 moves the matcher to the other final state m_6, thereby generating the second occurrence of $e(o)$ correctly.

A CAS $\mathcal{X}$ is a directed tree, where each node is an AU and each arc $(\mathcal{U},\mathcal{U}')$, with $\mathcal{U}$ being a child of $\mathcal{U}'$ in the tree, is a set of links connecting all the output terminals of $\mathcal{U}$ with some input terminals of $\mathcal{U}'$. A component/link is in $\mathcal{X}$ iff it is in an AU of $\mathcal{X}$.

Example 3. Outlined on the left side of Fig. 2 is a CAS called $\bar{\mathcal{X}}$, involving the AUs A, B, and C, where A has one input terminal, B has one input terminal and one output terminal, and C has one output terminal. Since each AU has at most one child, the hierarchy of $\bar{\mathcal{X}}$ is linear (no branches). To avoid irrelevant details, the internal structure of the AUs is omitted.

A state of $\mathcal{X}$ is a triple (S, E, M), where S is the array of states of the ACs in $\mathcal{X}$, E is the array of (possibly empty) events ready at the input terminals of the ACs in $\mathcal{X}$, and M is the array of states of the matchers of the events emerging from all AUs in $\mathcal{X}$. Like for ASs, a state of $\mathcal{X}$ is *quiescent* iff all elements in E are ε. A *trajectory* of $\mathcal{X}$ is a finite sequence of transitions of the ACs in $\mathcal{X}$,

$T = [t_1(c_1), \ldots, t_q(c_q)]$, which moves $\mathcal{X}$ from an *initial* quiescent state to a *final* quiescent state. Given an initial state x_0 of $\mathcal{X}$, the *space* of $\mathcal{X}$ is a DFA

$$\mathcal{X}^* = (\Sigma, X, \tau, x_0, X_\mathrm{f}), \tag{3}$$

where Σ is the alphabet, namely the set of transitions of the ACs in $\mathcal{X}$, X is the set of states, τ is the transition function, namely $\tau : X \times \Sigma \mapsto X$, and $X_\mathrm{f} \subseteq X$ is the set of final states, which are quiescent. In other words, the sequence of symbols in Σ (transitions of ACs) marking a path (sequence of transitions) in $\mathcal{X}^*$ from x_0 to a final state is a trajectory of $\mathcal{X}$. Hence, $\mathcal{X}^*$ is a finite representation of the (possibly infinite) set of trajectories of $\mathcal{X}$ starting at x_0.

Within a trajectory of a CAS $\mathcal{X}$, each transition is either *observable* or *unobservable.* The mode in which an observable transition is perceived is defined by the *viewer* $\mathcal{V}$ of $\mathcal{X}$, namely a set of pairs $(t(c), \ell)$, where $t(c)$ is a transition of an AC c and ℓ is an *observable label.* Given a transition $t(c)$, if $(t(c), \ell) \in \mathcal{V}$, then $t(c)$ is observable via label ℓ; otherwise, $t(c)$ is unobservable.

Assuming that $\mathcal{X}$ includes n AUs, namely $\mathcal{U}_1, \ldots, \mathcal{U}_n$, the *temporal observation* of a trajectory T of $\mathcal{X}$ based on a viewer $\mathcal{V}$, denoted $T_\mathcal{V}$, is an array $(O_1, \ldots, O_n)$ where, $\forall i \in [1 \mathinner{..} n]$, O_i is the *observation* of $\mathcal{U}_i$, defined as the sequence of observable labels associated with the observable transitions in T that are relevant to the ACs in $\mathcal{U}_i$ only:

$$O_i = [\, \ell \mid t(c) \in T, c \in \mathcal{U}_i, (t(c), \ell) \in \mathcal{V} \,]. \tag{4}$$

In other words, $T_\mathcal{V}$ is the result of grouping by AUs the observable labels associated with the observable transitions in T.

Not only each transition in a trajectory T is either observable or unobservable; it also is either *normal* or *faulty.* Faulty transitions are defined by the *ruler* $\mathcal{R}$ of $\mathcal{X}$, a set of pairs $(t(c), f)$, where $t(c)$ is a transition of an AC c and f is a *fault.* Given a transition $t(c)$, if $(t(c), f) \in \mathcal{R}$, then $t(c)$ is faulty via fault f; otherwise, $t(c)$ is normal. The *diagnosis* of a trajectory T of $\mathcal{X}$ based on $\mathcal{R}$, denoted $T_\mathcal{R}$, is defined as follows[3]:

$$T_\mathcal{R} = \{\, f \mid t(c) \in T, (t(c), f) \in \mathcal{R} \,\}. \tag{5}$$

If $T_\mathcal{R} \neq \emptyset$, then T is faulty; otherwise, T is normal. Even if $\mathcal{X}^*$ includes an infinite number of trajectories, the domain of the possible diagnoses is always finite, this being the powerset $2^\mathcal{F}$, where $\mathcal{F}$ is the domain of faults involved in $\mathcal{R}$.

3 Diagnosis Problem

When $\mathcal{X}$ performs an (unknown) trajectory T, what is observed is a temporal observation $\mathcal{O} = T_\mathcal{V}$, where $\mathcal{V}$ is the viewer of $\mathcal{X}$. However, owing to partial unobservability, several (possibly an infinite number of) trajectories may be compatible with $\mathcal{O}$. Hence, several (a finite number of) *candidate diagnoses* may be

[3] More generally, any set of faults is called a diagnosis.

compatible with $\mathcal{O}$. The goal is finding all these candidates. A *diagnosis problem* $\wp(\mathcal{X}, \mathcal{O})$ is an association between $\mathcal{X}$ and a temporal observation $\mathcal{O}$. Given the viewer $\mathcal{V}$ and the ruler $\mathcal{R}$ of $\mathcal{X}$, the *solution* to $\wp(\mathcal{X}, \mathcal{O})$ is the set of candidate diagnoses

$$\Delta(\wp(\mathcal{X}, \mathcal{O})) = \{ T_{\mathcal{R}} \mid T \in \mathcal{X}^*, T_{\mathcal{V}} = \mathcal{O} \}. \tag{6}$$

Example 4. Consider the CAS $\bar{\mathcal{X}}$ in Example 3. Assume that the observable labels involved in $\bar{\mathcal{V}}$ are a, b, c, d, e, and f; the faults involved in the ruler $\bar{\mathcal{R}}$ are p, q, v, w, x, y, and z; the temporal observation is $\bar{\mathcal{O}} = (\bar{O}_A, \bar{O}_B, \bar{O}_C)$, where $\bar{O}_A = [e, f]$, $\bar{O}_B = [c, d]$, and $\bar{O}_C = [a, b, a]$. We have that $\wp(\bar{\mathcal{X}}, \bar{\mathcal{O}})$ is a diagnosis problem.

It should be clear that Eq. (6) is only a definition formalizing what the solution to a diagnosis problem is. It makes no sense to enumerate the candidate diagnoses as suggested by this definition, as the size of $\mathcal{X}^*$ is exponential in the number of ACs and input terminals. The space $\mathcal{X}^*$ plays only a formal role and is never materialized. Even the reconstruction of the subspace of $\mathcal{X}^*$ that is compatible with $\mathcal{O}$ is out of the question because, in the worst case, it suffers from the same exponential complexity of $\mathcal{X}^*$. So, what to do? The short answer is: focusing on the AUs rather than on the whole of $\mathcal{X}$. The important points are soundness and completeness: the set of diagnoses determined must coincide with the set of candidate diagnoses defined in Eq. (6).

4 Knowledge Compilation

The diagnosis process involves several tasks, which are performed either *offline* or *online*, that is, *before* or *while* the CAS is being operated, respectively. The tasks performed offline are collectively called "knowledge compilation", and the resulting data structures, "compiled knowledge". Compiled knowledge is meant to speed up the task performed online by the diagnosis engine. In offline processing, the AUs are processed independently from one another. For each AU $\mathcal{U}$, three sorts of data structures are compiled in cascade: the *unit space* $\mathcal{U}^*$, the *unit labeling* $\mathcal{U}_\delta$, and the *unit knowledge* $\mathcal{U}_\Delta$, of which the latter is exploited online. Online processing depends on a diagnosis problem to be solved, specifically, on the observation, whereas offline processing does not.

Definition 1 (Unit space). *Let $\mathcal{U}$ be an AU involving an AS $\mathcal{A}$. The* space *of $\mathcal{U}$ is a DFA*

$$\mathcal{U}^* = (\Sigma, U, \tau, u_0, U_{\mathrm{f}}), \tag{7}$$

where: the alphabet Σ is the set of transitions of ACs in $\mathcal{U}$; U is the set of states (S, E, M), where (S, E) is a state of $\mathcal{A}$ and M is the array of states of the matchers of the emergent events of $\mathcal{U}$; $u_0 = (S_0, E_0, M_0)$ is the initial state, where (S_0, E_0) is a quiescent state of $\mathcal{A}$ and M_0 is the array of initial states of the matchers; $U_{\mathrm{f}} \subseteq U$ is the set of final states, where $(S, E, M) \in U_{\mathrm{f}}$ iff (S, E) is a quiescent state of $\mathcal{A}$; and $\tau : U \times \Sigma \mapsto U$ is the transition function,

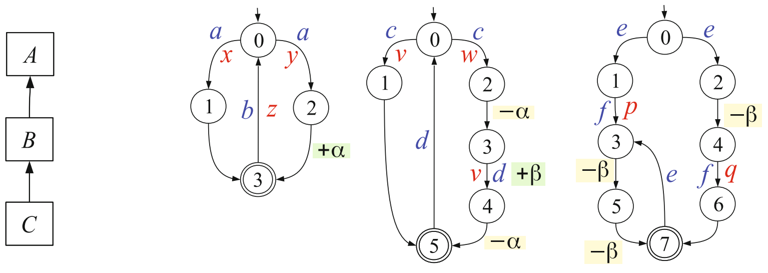

Fig. 2. CAS $\bar{\mathcal{X}}$ (left) and unit spaces C^*, B^*, and A^*. (Color figure online)

where $(S, E, M) \xrightarrow{t(c)} (S', E', M') \in \tau$ *iff* $(S, E) \xrightarrow{t(c)} (S', E')$ *occurs in* $\mathcal{A}$*,* M' *is the result of updating* M *based on* $t(c)$*, and* (S', E', M') *is connected to a final state*[4]*.*

Example 5. We assume that the space of each AU has been generated already, as shown next to the CAS $\bar{\mathcal{X}}$ in Fig. 2, namely C^* (left), B^* (center), and A^* (right). In each unit space, the states are identified by numbers, the final states are double circled, and the transitions are marked by relevant information only, namely: observable label (in blue), fault (in red), occurrence of an emergent event (prefixed by the "+" plus sign), and consumption of an event emerged from a child unit (prefixed by the "−" minus sign). For instance, in B^*, the transition from 3 to 4 is observable via the label d, is faulty via the fault v, and generates the emergent event β. The transition from 2 to 3, which is unobservable and normal, consumes the event α emerging from C, the child of B.

Since the space of an AU does not depend on other AUs, the occurrence of a transition depending on an event e emerging from a child AU in the CAS is not constrained by the actual readiness of e at one input terminal, as this information is outside the scope of the AU. Yet, the enforcement of this *interface* constraint is not neglected, but only deferred to the diagnosis engine operating online (cf. Sect. 5).

Definition 2 (Unit labeling). *Let* $\mathcal{U}^* = (\Sigma, U, \tau, u_0, U_{\rm f})$ *be the space of an AU* $\mathcal{U}$ *in a CAS* $\mathcal{X}$*, let* $\mathcal{R}$ *be the ruler of* $\mathcal{X}$*, and let* $\boldsymbol{\delta}$ *be the domain of diagnoses in* $\mathcal{U}$*. The* labeling *of* $\mathcal{U}$ *is a DFA*

$$\mathcal{U}_{\boldsymbol{\delta}} = (\Sigma, U', \tau', u'_0, U'_{\rm f}), \tag{8}$$

where: $U' \subseteq U \times \boldsymbol{\delta}$ *is the set of states;* $u'_0 = (u_0, \emptyset)$ *is the initial state;* $U'_{\rm f} \subseteq U'$ *is the set of final states, with* $(u, \delta) \in U'_{\rm f}$ *if* $u \in U_{\rm f}$*;* $\tau' : U' \times \Sigma \mapsto U'$ *is the transition function, with* $(u, \delta) \xrightarrow{t(c)} (u', \delta') \in \tau'$ *iff* $u \xrightarrow{t(c)} u' \in \tau$ *and*

$$\delta' = \begin{cases} \delta \cup \{f\} & \textit{if } (t(c), f) \in \mathcal{R} \\ \delta & \textit{otherwise.} \end{cases}$$

[4] The requirement on connection means that there is a contiguous sequence of transitions from (S', E', M') to a final state in $U_{\rm f}$.

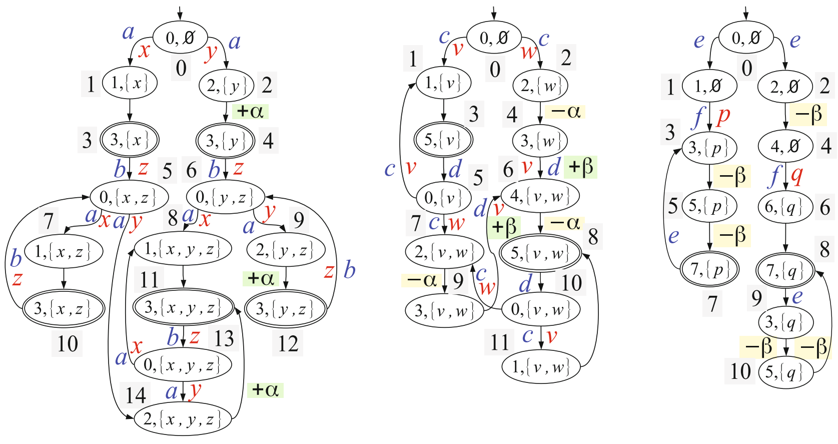

Fig. 3. Unit labelings C_δ (left), B_δ (center), and A_δ (right).

Example 6. Displayed on Fig. 3 are the unit labelings C_δ, B_δ, and A_δ derived from the unit spaces outlined in Fig. 2. To facilitate subsequent referencing, states are re-identified by numbers. Owing to the additional field δ (set of faults), the number of states in $\mathcal{U}_\delta$ is generally larger than the number of states in $\mathcal{U}^*$.

Definition 3 (Unit knowledge). *Let $\mathcal{U}$ be an AU in a CAS $\mathcal{X}$, let $\mathcal{U}_\delta$ be a labeling of $\mathcal{U}$, and let $\mathcal{R}$ be the ruler of $\mathcal{X}$. Let $\mathcal{U}'_\delta$ be the NFA obtained from $\mathcal{U}_\delta$ by substituting the alphabet symbols marking the transitions as follows. For each transition $(u, \delta) \xrightarrow{t(c)} (u', \delta')$ in $\mathcal{U}_\delta$, $t(c)$ is replaced with a triple (ℓ, e, E), where: if $(t(c), \ell') \in \mathcal{V}$, then $\ell = \ell'$, else $\ell = \varepsilon$; if $t(c)$ consumes an event e' emerging from a child unit, then $e = e'$, else $e = \varepsilon$; E is the (possibly empty) set of emergent events generated at $t(c)$ by $\mathcal{U}$. Eventually, all triples $(\varepsilon, \varepsilon, \emptyset)$ are replaced by ε. The* unit knowledge *$\mathcal{U}_\Delta$ is the DFA obtained by the determinization of $\mathcal{U}'_\delta$, where each final state s_f of $\mathcal{U}_\Delta$ is marked by the aggregation of the diagnosis sets associated with the final states of $\mathcal{U}'_\delta$ within s_f, denoted $\Delta(s_\mathrm{f})$*[5].

Example 7. Shown in Fig. 4 are the unit knowledge C_Δ, B_Δ, and A_Δ, derived from the unit labelings displayed in Fig. 3, where ε elements in triples (ℓ, α, β) are omitted and states are re-identified by numbers. For instance, consider the final state 4 of C_Δ (left of Fig. 4), which includes the states $7 = (1, \{x, z\})$, $10 = (3, \{x, z\})$, and $14 = (2, \{x, y, z\})$ of C_δ. Since the state $10 = (3, \{x, z\})$ in C_δ is final (it is final in C'_δ too), the state 4 of C_Δ is marked by $\{\{x, z\}\}$.

[5] Based on the algorithm *Subset Construction* [10], when an NFA is determinized into an equivalent DFA, each state in the DFA is identified by a subset of the states of the NFA.

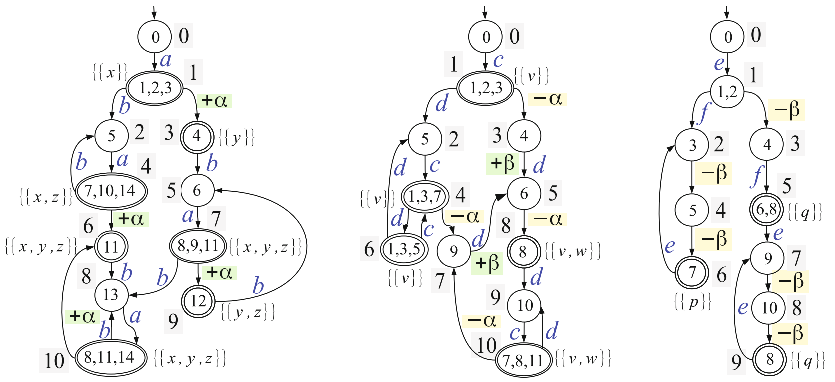

Fig. 4. Unit knowledge C_{Δ} (left), B_{Δ} (center), and A_{Δ} (right).

5 Diagnosis Engine

A diagnosis problem for $\mathcal{X}$ is solved online by the *diagnosis engine* by exploiting the knowledge of the AUs compiled offline. Each unit knowledge $\mathcal{U}_{\Delta}$ is constrained by the observation of $\mathcal{U}$ and by the events emerging from the child AUs of $\mathcal{U}$.

Definition 4 (Abduction of leaf AU). *Let $\mathcal{U}$ be an AU that is a leaf of the tree of a CAS, let $\mathcal{U}_{\Delta} = (\Sigma, S, \tau, s_0, S_{\mathrm{f}})$ be the knowledge of $\mathcal{U}$, and let $O = [\ell_1, \ldots, \ell_n]$ be an observation of $\mathcal{U}$. The* abduction *of $\mathcal{U}$ is a DFA*

$$\mathcal{U}_{\mathcal{O}} = (\Sigma, S', \tau', s'_0, S'_{\mathrm{f}}),$$

where: $S' \subseteq S \times [0\,..\,n]$ is the set of states[6]*; $s'_0 = (s_0, 0)$ is the initial state; $S'_{\mathrm{f}} \subseteq S'$ is the set of final states, where $s' \in S'_{\mathrm{f}}$ iff $s' = (s, n)$, $s \in S_{\mathrm{f}}$, with s' being marked by $\Delta(s') = \Delta(s)$; $\tau' : S' \times \Sigma \mapsto S'$ is the transition function, where $(s, j) \xrightarrow{(\ell, \varepsilon, E)} (s', j') \in \tau'$ iff (s', j') is connected to a final state, $s \xrightarrow{(\ell, \varepsilon, E)} s' \in \tau$, and*

$$j' = \begin{cases} j+1 & \textit{if } \ell \neq \varepsilon \textit{ and } \ell_{j+1} = \ell \\ j & \textit{if } \ell = \varepsilon. \end{cases} \tag{9}$$

Example 8. Consider the unit knowledge C_{Δ} displayed on the left side of Fig. 4. Let $\bar{O}_C = [\,a, b, a\,]$ be the observation of C. The unit abduction $C_{\mathcal{O}}$ is shown on the left side of Fig. 5.

To extend Definition 4 to nonleaf AUs, we introduce the notion of a unit interface in Definition 5 (generalized in Definition 8).

[6] Each natural number in $[0\,..\,n]$ is called an *index* of O.

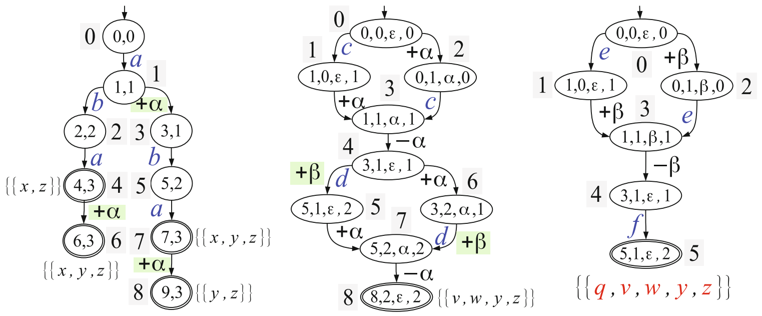

Fig. 5. Unit abductions $C_{\mathcal{O}}$ (left), $B_{\mathcal{O}}$ (center), and $A_{\mathcal{O}}$ (right).

Definition 5 (Interface of leaf AU). *Let $\mathcal{U}_{\mathcal{O}}$ be an abduction where $\mathcal{U}$ is a leaf AU. Let $\mathcal{U}'_{\mathcal{O}}$ be the NFA obtained from $\mathcal{U}_{\mathcal{O}}$ by substituting E for each transition symbol (ℓ, ε, E) such that $E \neq \emptyset$, and ε for all other symbols. The* interface *$\mathcal{U}_{\mathfrak{I}}$ of $\mathcal{U}$ is the DFA obtained by determinization of $\mathcal{U}'_{\mathcal{O}}$, where each final state s'_{f} is marked by the union of the sets of diagnoses marking the final states of $\mathcal{U}'_{\mathcal{O}}$ included in s'_{f}, denoted $\Delta(s'_{\mathrm{f}})$.*

Example 9. Considering the unit abduction $C_{\mathcal{O}}$ displayed on the left side of Fig. 5, the unit interface $C_{\mathfrak{I}}$ is shown on the left side of Fig. 6, where all three states are final.

To extend the notion of a unit abduction to nonleaf AUs (Definition 7), we make use of the join operator defined below.

Definition 6 (Join). *The* join *"$\otimes$" of two sets of diagnoses, Δ_1 and Δ_2, is the set of diagnoses defined as follows:*

$$\Delta_1 \otimes \Delta_2 = \{ \delta \mid \delta = \delta_1 \cup \delta_2, \delta_1 \in \Delta_1, \delta_2 \in \Delta_2 \}. \tag{10}$$

Definition 7 (Abduction of nonleaf AU). *Let $\mathcal{U}$ be a nonleaf AU in $\mathcal{X}$, let $\mathcal{U}_1, \ldots, \mathcal{U}_k$ be the child AUs of $\mathcal{U}$ in $\mathcal{X}$, let $\mathcal{U}_{\Delta} = (\Sigma, S, \tau, s_0, S_{\mathrm{f}})$ be the knowledge of $\mathcal{U}$, let $O = [\ell_1, \ldots, \ell_n]$ be an observation of $\mathcal{U}$, let $\mathbf{E}$ be the domain of tuples of (possibly empty) events emerging from child AUs ready at the input terminals of $\mathcal{U}$, let $\mathcal{U}_{i\mathfrak{I}} = (\Sigma_i, S_i, \tau_i, s_{0i}, S_{\mathrm{f}i})$ be the interface of $\mathcal{U}_i$, $i \in [1\,..\,k]$, and let $\mathbf{S} = S_1 \times \cdots \times S_k$. The* abduction *of $\mathcal{U}$ is a DFA*

$$\mathcal{U}_{\mathcal{O}} = (\Sigma', S', \tau', s'_0, S'_{\mathrm{f}}),$$

where: $\Sigma' = \Sigma \cup \Sigma_1 \cup \cdots \cup \Sigma_k$; $S' \subseteq S \times \mathbf{S} \times \mathbf{E} \times [0\,..\,n]$ is the set of states; $s'_0 = (s_0, (s_{01}, \ldots, s_{0k}), (\varepsilon, \ldots, \varepsilon), 0)$ is the initial state; $S'_{\mathrm{f}} \subseteq S'$ is the set of final states, where $s'_{\mathrm{f}} \in S'_{\mathrm{f}}$ iff $s'_{\mathrm{f}} = (s_{\mathrm{f}}, (s_{\mathrm{f}1}, \ldots, s_{\mathrm{f}k}), (\varepsilon, \ldots, \varepsilon), n)$, $s_{\mathrm{f}} \in S_{\mathrm{f}}$, $\forall i \in [1\,..\,k]$, $s_{\mathrm{f}i} \in S_{\mathrm{f}i}$, and s'_{f} is marked by the set of diagnoses

$$\Delta(s'_{\mathrm{f}}) = \Delta(s_{\mathrm{f}}) \otimes \Delta(s_{\mathrm{f}1}) \otimes \cdots \otimes \Delta(s_{\mathrm{f}k}); \tag{11}$$

$\tau' : \Sigma' \times S' \mapsto S'$ is the transition function, where

$$(s, (s_1, \ldots, s_k), E, j) \xrightarrow{\sigma} (s', (s'_1, \ldots, s'_k), E', j') \in \tau'$$

iff $(s', (s'_1, \ldots, s'_k), E', j')$ is connected to a final state and either of the following conditions holds:

(a) $s \xrightarrow{\sigma} s' \in \tau$, $\sigma = (\ell, e, \bar{E})$, either $e = \varepsilon$ or e is ready in E, E' equals E except that e is removed, and the index j' is defined as in Eq. (9);

(b) $s_i \xrightarrow{\sigma} s'_i \in \tau_i$, $\sigma = E_i$, all elements in E corresponding to the input emergent events in E_i are ε, E' equals E except that all events in E_i are inserted into E'.

Example 10. Consider the unit knowledge B_Δ displayed on the center of Fig. 4 and the unit interface $C_\Im$ displayed on the left side of Fig. 6. Let $\bar{O}_B = [\, c, d\,]$ be the observation of B. The unit abduction $B_\mathcal{O}$ is shown on the center of Fig. 5. Each state in $B_\mathcal{O}$ is marked by a quadruple (s_B, s_C, e, j), where s_B is a state of B_Δ, s_C is a state of $C_\Im$ (C is the unique child of B), e is the event emerging from C and possibly ready (if $e \neq \varepsilon$) at the input terminal of B (B includes one input terminal only), and j is an index of the observation $\bar{O}_B$. Let $\bar{\Delta}$ be the set of diagnoses marking the final state $8 = (8, 2, \varepsilon, 2)$. Based on Eq. (11), $\bar{\Delta}$ is generated via join $\Delta(8) \otimes \Delta(2)$, where $\Delta(8)$ is the set associated with the state 8 of the unit knowledge B_Δ (shown on the center of Fig. 4), namely $\{\{v, w\}\}$, and $\Delta(2)$ is the set associated with the state 2 of the unit interface $C_\Im$ (shown on the left side of Fig. 6), namely $\{\{y, z\}\}$. Hence, $\bar{\Delta} = \{\{v, w\}\} \otimes \{\{y, z\}\} = \{\{v, w, y, z\}\}$. The unit interface $\mathcal{B}_\Im$, drawn from $\mathcal{B}_\mathcal{O}$, is displayed on the right side of Fig. 6.

Definition 8 (Interface of AU). *Let $\mathcal{U}_\mathcal{O}$ be an abduction. Let $\mathcal{U}'_\mathcal{O}$ be the NFA obtained from $\mathcal{U}_\mathcal{O}$ by substituting E for each transition symbol (ℓ, e, E) such that $E \neq \emptyset$, and ε for all other symbols. The* interface *$\mathcal{U}_\Im$ of $\mathcal{U}$ is the DFA obtained by determinization of $\mathcal{U}'_\mathcal{O}$, where each final state s'_f is marked by the union of the sets of diagnoses marking the final states of $\mathcal{U}'_\mathcal{O}$ included in s'_f, denoted $\Delta(s'_\mathrm{f})$.*

Example 11. Considering the unit abduction $B_\mathcal{O}$ displayed on the center of Fig. 5, the unit interface $B_\Im$ is shown on the right side of Fig. 6.

6 Soundness and Completeness

Once the abduction process reaches the root $\mathcal{U}$ of the CAS $\mathcal{X}$, thereby generating $\mathcal{U}_\mathcal{O}$, the solution to the diagnosis problem $\wp(\mathcal{X}, \mathcal{O})$ can be determined by collecting the diagnoses marking the final states of $\mathcal{U}_\mathcal{O}$ (Proposition 1).

Proposition 1 (*Correctness*)**.** *Let $\wp(\mathcal{X}, \mathcal{O})$ be a diagnosis problem, let $\mathcal{U}$ be the AU at the root of $\mathcal{X}$, let $\mathcal{U}_\mathcal{O}$ be the abduction of $\mathcal{U}$, with S_f being the set of final states, and let*

$$\Delta(\mathcal{U}_\mathcal{O}) = \bigcup_{s_\mathrm{f} \in S_\mathrm{f}} \Delta(s_\mathrm{f}) \,. \tag{12}$$

We have $\Delta(\wp(\mathcal{X}, \mathcal{O})) = \Delta(\mathcal{U}_\mathcal{O})$.

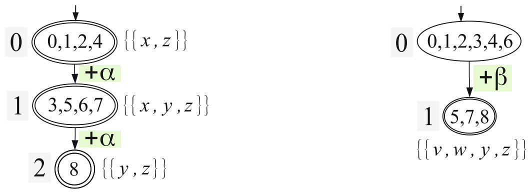

Fig. 6. Unit interfaces $C_{\Im}$ (left) and $B_{\Im}$ (right).

To prove Proposition 1, further terminology is required. Let $\bar{\mathcal{U}}$ be the CAS within $\mathcal{X}$ rooted in $\mathcal{U}$. The *restriction* of $\wp(\mathcal{X}, \mathcal{O})$ on $\bar{\mathcal{U}}$ is a diagnosis problem $\wp(\bar{\mathcal{U}}, \bar{\mathcal{O}})$ where the viewer $\bar{\mathcal{V}}$ of $\bar{\mathcal{U}}$ is the restriction of the viewer of $\mathcal{X}$ on pairs $(t(c), \ell)$ such that c is a component in $\bar{\mathcal{U}}$, the ruler $\bar{\mathcal{R}}$ of $\bar{\mathcal{U}}$ is the restriction of the ruler of $\mathcal{X}$ on pairs $(t(c), f)$ such that c is a component in $\bar{\mathcal{U}}$, the initial state of $\bar{\mathcal{U}}$ is the restriction of the initial state of $\mathcal{X}$ on the components and links in $\bar{\mathcal{U}}$, and $\bar{\mathcal{O}}$ is the restriction of $\mathcal{O}$ on observations of AUs in $\bar{\mathcal{U}}$. The notion of a trajectory is extended to any DFA involved in the diagnosis task, namely unit labeling, unit knowledge, unit abduction, and unit interface. A *trajectory* in any such DFA is a string of the regular language of the DFA, that is, the sequence of symbols in the alphabet which mark the sequence of transitions from the initial state to a final state of the DFA; such a final state is called the *accepting state* of the trajectory. The set of trajectories in a DFA $\mathcal{D}$ (the regular language of $\mathcal{D}$) is denoted by $\|\mathcal{D}\|$. For instance, $T \in \|\mathcal{U}_{\mathcal{O}}\|$ denotes a trajectory T in the abduction $\mathcal{U}_{\mathcal{O}}$.

A *path* p in a DFA $\mathcal{D}$ is the sequence of transitions yielding a trajectory $[\sigma_1, \ldots, \sigma_n]$ in $\mathcal{D}$, namely $p = s_0 \xrightarrow{\sigma_1} s_1 \xrightarrow{\sigma_2} \cdots \xrightarrow{\sigma_n} s_n$. A *semipath* in $\mathcal{D}$ is a prefix of a path in $\mathcal{D}$. A *subpath* p' in $\mathcal{D}$ is a contiguous sequence of transitions within a path, from state s_i to state s_j, namely $s_i \xrightarrow{\sigma_{i+1}} s_{1+1} \xrightarrow{\sigma_{i+2}} \cdots \xrightarrow{\sigma_{j-1}} s_{j-1} \xrightarrow{\sigma_j} s_j$, concisely denoted $s_i \overset{\boldsymbol{\sigma}}{\Rightarrow} s_j$, where $\boldsymbol{\sigma} = [\sigma_{i+1}, \ldots, \sigma_{j-1}, \sigma_j]$ is the *subtrajectory* generated by p'.

The notion of an interface is extended to any trajectory. The *interface* of a trajectory T, denoted $\Im(T)$, is the sequence of nonempty sets of emergent events occurring in T. Specifically, if T is a trajectory in $\mathcal{U}_{\Im}$, then $\Im(T) = T$. If T is a trajectory in either $\mathcal{U}_{\mathcal{O}}$ or $\mathcal{U}_{\Delta}$, then $\Im(T) = [\, E \mid (\ell, e, E) \in T, E \neq \emptyset \,]$. If T is a trajectory in either $\mathcal{U}^*$ or $\mathcal{U}_{\delta}$, then each set E in $\Im(T)$ is the nonempty set of events emerging at the occurrence of a component transition in T.

Let $\mathcal{G} = (N, A)$ be a directed acyclic graph (DAG), where N is the set of nodes and A is the set of arcs. Let n and n' be two nodes in N. We say that n precedes n', denoted $n \prec n'$, iff n' is reachable from n by a contiguous sequence of arcs. A *topological sort* $\mathcal{S}$ in $\mathcal{G}$ is a sequence of all nodes in N (with each node occurring once) such that, for each pair (n, n') of nodes in $\mathcal{S}$, if $n \prec n'$ in $\mathcal{G}$, then $n \prec n'$ in $\mathcal{S}$. The whole (finite) set of topological sorts in $\mathcal{G}$ is denoted $\|\mathcal{G}\|$.

Based on the terminology introduced above, a sketch of the proof of Proposition 1 can be given on the ground of Lemma 11, Lemma 12, Corollary 11, and Lemma 13.

Lemma 11 (*Mapping*). *Let $\mathcal{D} = (\Sigma, S, \tau, s_0, S_{\rm f})$ be a DFA. Let Σ' be an alphabet (possibly including ε) and let $\Sigma \mapsto \Sigma'$ be a surjective mapping from Σ to Σ'. Let $\mathcal{N} = (\Sigma', S, \tau', s_0, S_{\rm f})$ be the NFA derived from $\mathcal{D}$ by replacing each transition $s \xrightarrow{\sigma} \sigma \in \tau$ with $s \xrightarrow{\sigma'} \sigma \in \tau'$, where σ' is the symbol in Σ' mapping the symbol σ in Σ. Let $\mathcal{D}'$ be the DFA obtained by determinization of $\mathcal{N}$. If T' is a trajectory in $\|\mathcal{D}'\|$ with accepting state $s'_{\rm f}$, then, for each final state $s_{\rm f} \in s'_{\rm f}$, there is a trajectory T in $\|\mathcal{D}\|$ with accepting state $s_{\rm f}$ such that T' is the mapping of T based on $\Sigma \mapsto \Sigma'$.*

Proof. By induction on T'.

(*Basis*) According to the *Subset Construction* determinization algorithm, if $s \in s'_0$, where s'_0 is the initial state of $\mathcal{D}'$, then there is a path $s_0 \overset{\varepsilon^*}{\Rightarrow} s$ in $\mathcal{N}$ where ε^* is a (possibly empty) sequence of ε. Hence, there is a path $s_0 \overset{\boldsymbol{\sigma}}{\Rightarrow} s$ in $\mathcal{D}$ such that ε^* is the mapping of $\boldsymbol{\sigma}$ based on $\Sigma \mapsto \Sigma'$.

(*Induction*) Assume that there is a semipath $s'_0 \overset{\boldsymbol{\sigma}'}{\Rightarrow} s'$ in $\mathcal{D}'$ such that for each $s \in s'$ there is a semipath $s_0 \overset{\boldsymbol{\sigma}}{\Rightarrow} s$ in $\mathcal{D}$ where $\boldsymbol{\sigma}'$ is the mapping of $\boldsymbol{\sigma}$ based on $\Sigma \mapsto \Sigma'$. Consider the new semipath $s'_0 \overset{\boldsymbol{\sigma}'}{\Rightarrow} s' \xrightarrow{\bar{\sigma}} \bar{s}'$ in $\mathcal{D}'$. According to *Subset Construction*, for each state $\bar{s} \in \bar{s}'$ there is a state $s \in s'$ such that $s \overset{\bar{\boldsymbol{\sigma}}_n}{\Longrightarrow} \bar{s}$ is a subpath in $\mathcal{N}$ where $\bar{\boldsymbol{\sigma}}_n$ starts with $\bar{\sigma}$, followed by a (possibly empty) sequence of ε. Hence, there is a subpath $s \overset{\bar{\boldsymbol{\sigma}}_d}{\Longrightarrow} \bar{s}$ in $\mathcal{D}$ such that $\bar{\sigma}_d$ maps to $[\bar{\sigma}]$. Thus, by virtue of the induction hypothesis, $\boldsymbol{\sigma} \cup \bar{\boldsymbol{\sigma}}_d$ maps to $\boldsymbol{\sigma}' \cup [\bar{\sigma}]$ based on $\Sigma \mapsto \Sigma'$. □

Corollary 11. *With reference to the assumptions stated in Lemma 11, if T' is generated by a path $p' = s'_0 \xrightarrow{\sigma'_1} s'_1 \xrightarrow{\sigma'_2} s'_2 \xrightarrow{\sigma'_3} \cdots \xrightarrow{\sigma'_{n-1}} s'_{n-1} \xrightarrow{\sigma'_n} s'_n$ in $\mathcal{D}'$, then there is a path $p = s_0 \overset{\boldsymbol{\sigma}_1}{\Longrightarrow} s_1 \overset{\boldsymbol{\sigma}_2}{\Longrightarrow} s_2 \overset{\boldsymbol{\sigma}_3}{\Longrightarrow} \cdots \overset{\boldsymbol{\sigma}_{n-1}}{\Longrightarrow} s_{n-1} \overset{\boldsymbol{\sigma}_n}{\Longrightarrow} s_n$ in $\mathcal{D}$ such that, $\forall i \in [1 \,..\, n]$, $s_{i-1} \in s'_{i-1}$, $s_i \in s'_i$, and $[\sigma'_i]$ is the mapping of $\boldsymbol{\sigma}_i$ based on $\Sigma \mapsto \Sigma'$.*

Lemma 12 (*Soundness*). *If $s'_{\rm f}$ is a final state in $\mathcal{U}_{\mathcal{O}}$, $\delta \in \Delta(s'_{\rm f})$, and $T \in \|\mathcal{U}_{\mathcal{O}}\|$ with accepting state $s'_{\rm f}$, then $\delta \in \Delta(\wp(\bar{\mathcal{U}}, \bar{\mathcal{O}})$, where $\delta = \bar{T}_{\bar{\mathcal{R}}}$, $\bar{T} \in \|\bar{\mathcal{U}}^*\|$, and $\Im(\bar{T}) = \Im(T)$.*

Proof. By bottom-up induction on the tree of $\bar{\mathcal{U}}$.

(*Basis*) Assume that $\mathcal{U}$ is a leaf AU. Since $s'_{\rm f} = (s_{\rm f}, n)$ is a final state in $\mathcal{U}_{\mathcal{O}}$, $\delta \in \Delta(s'_{\rm f})$, and $T \in \|\mathcal{U}_{\mathcal{O}}\|$ with accepting state $s'_{\rm f}$, based on Definition 4 and Lemma 11, there is $T_\Delta \in \|\mathcal{U}_\Delta\|$ with accepting state $s_{\rm f}$ such that $\delta \in \Delta(s_{\rm f})$ and $\Im(T_\Delta) = \Im(T)$. Hence, based on Definition 3 and Lemma 11, there is $T_\delta \in \|\mathcal{U}_\delta\|$ with accepting state (s, δ) such that $\Im(T_\delta) = \Im(T_\Delta) = \Im(T)$. Therefore, based on Definition 2, there is $\bar{T} \in \|\mathcal{U}^*\|$ such that $\bar{T}_{\bar{\mathcal{V}}} = \bar{\mathcal{O}}$ and $\bar{T}_{\bar{\mathcal{R}}} = \delta$; in other words, according to Eq. (6), $\delta \in \Delta(\wp(\bar{\mathcal{U}}, \bar{\mathcal{O}})$. Also, $\Im(\bar{T}) = \Im(T_\delta) = \Im(T)$.

(*Induction*) Assume that $\mathcal{U}$ is the parent of the AUs $\mathcal{U}_1, \ldots, \mathcal{U}_k$, where $\forall i \in [1 \,..\, k]$, if $s'_{i{\rm f}}$ is a final state in $\mathcal{U}_{i\mathcal{O}}$, $\delta_i \in \Delta(s'_{i{\rm f}})$, and $T_i \in \|\mathcal{U}_{i\mathcal{O}}\|$ with accepting state $s'_{i{\rm f}}$, then $\delta_i \in \Delta(\wp(\bar{\mathcal{U}}_i, \bar{\mathcal{O}}_i)$, where $\delta_i = \bar{T}_{i\bar{\mathcal{R}}_i}$, $\bar{T}_i \in \|\bar{\mathcal{U}}_i^*\|$, and $\Im(\bar{T}_i) = \Im(T_i)$.

Since $s'_{\rm f} = (s_{\rm f}, (s_{1{\rm f}}, \ldots, s_{k{\rm f}}), (\varepsilon, \ldots, \varepsilon), n)$ is a final state in $\mathcal{U}_{\mathcal{O}}$, $\delta \in \Delta(s'_{\rm f})$, and $T \in \|\mathcal{U}_{\mathcal{O}}\|$ with accepting state $s'_{\rm f}$, according to Eq. (11), $\delta \in \Delta(s'_{\rm f}) = \Delta(s_{\rm f}) \otimes$

$\varDelta(s_{1\text{f}}) \otimes \cdots \otimes \varDelta(s_{k\text{f}})$, where, $\forall i \in [1 \mathinner{..} k]$, $s_{i\text{f}}$ is final in $\mathcal{U}_{i\Im}$. Hence, based on Definition 6, $\delta = \delta_0 \cup \delta_1 \cup \cdots \cup \delta_k$, where $\delta_0 \in \varDelta(s_\text{f})$ and, $\forall i \in [1 \mathinner{..} k]$, $\delta_i \in \varDelta(s_{i\text{f}})$. Also, based on Definition 8 and Lemma 11, $\forall i \in [1 \mathinner{..} k]$, there is $T_i \in \|\mathcal{U}_{i\mathcal{O}}\|$ with accepting state $s'_{i\text{f}} \in s_{i\text{f}}$ such that $\delta_i \in s'_{i\text{f}}$ and all emergent events in T_i are consumed in T. Thus, by virtue of the induction hypothesis, we have $\delta_i \in \varDelta(\wp(\bar{\mathcal{U}}_i, \bar{\mathcal{O}}_i)$, where $\delta_i = \bar{T}_{i\bar{\mathcal{R}}_i}$, $\bar{T}_i \in \|\bar{\mathcal{U}}^*_i\|$, and $\Im(\bar{T}_i) = \Im(T_i)$. According to Definition 7, the trajectory T is composed of triples (ℓ, e, E) interspersed by elements E_i, with each E_i being a nonempty set of events emerging from $\mathcal{U}_i$. Note that, for each E_i in T there is one transition in $\bar{T}_i$ generating E_i. Thus, each $\bar{T}_i$ is composed of transitions $t(c)$, where c is a component in $\bar{\mathcal{U}}_i$, in which there are some transitions generating sets E_i of emergent events. The sequence of E_i generated in $\bar{T}_i$ equals the subsequence of E_i in T, $i \in [1 \mathinner{..} k]$. So, there is a sequential correspondence between each E_i in T and each transition in $\bar{T}_i$ generating the set E_i of emergent events. According to Definition 7 and Corollary 11, if $s'_0 \xRightarrow{\boldsymbol{\sigma}'_1} s'_1 \xrightarrow{E_{i1}} s''_1 \xRightarrow{\boldsymbol{\sigma}'_2} s'_2 \xrightarrow{E_{i2}} s''_2 \xRightarrow{\boldsymbol{\sigma}'_3} \cdots \xRightarrow{\boldsymbol{\sigma}'_n} s'_n$ is the path in $\mathcal{U}_\mathcal{O}$ generating the trajectory $T = \boldsymbol{\sigma}'_1 \cup [E_{i1}] \cup \boldsymbol{\sigma}'_2 \cup [E_{i2}] \cup \boldsymbol{\sigma}'_3 \cup \cdots \cup \boldsymbol{\sigma}'_n$, then there is a path $s_0 \xRightarrow{\boldsymbol{\sigma_1}} s_1 \xRightarrow{\boldsymbol{\sigma_2}} s_2 \xRightarrow{\boldsymbol{\sigma_3}} \cdots \xRightarrow{\boldsymbol{\sigma_n}} s_n$ in $\mathcal{U}_\delta$ generating the trajectory $T^* = \boldsymbol{\sigma_1} \cup \boldsymbol{\sigma_2} \cup \cdots \cup \boldsymbol{\sigma_n}$, $T^* \in \|\mathcal{U}^*\|$, such that T^* generates the observation of $\mathcal{U}$ and the diagnosis δ_0. Now, consider the DAG $\mathcal{G}$ constructed by the following four steps, where each node is either a component transitions or a set E_i of emergent events, while arcs indicate precedence dependencies between these nodes. Step 1: in T, substitute $\boldsymbol{\sigma}_j$ for each $\boldsymbol{\sigma}'_j$, $j \in [1 \mathinner{..} n]$; this step substitutes sequences of transitions of components in $\mathcal{U}$ for corresponding sequences of triples (ℓ, e, E) in T; Step 2: transform each trajectory $\bar{T}_i = [t_1, t_2, \ldots, t_{n_i}]$, $i \in [1 \mathinner{..} k]$, into a connected sequence of nodes $t_1 \rightarrow t_2 \rightarrow \cdots \rightarrow t_{n_i}$, where arrows indicate arcs (precedence dependencies); Step 3: transform T into a connected sequence of nodes in the same way as in step 2; Step 4: for each transition t_i in $\bar{T}_i$ generating the set E_i of emergent events, insert an arc from t_i to the corresponding E_i in T. This results in a DAG, namely a *dependency graph* $\mathcal{G}$, where each E_i is entered by an arc from a transition in $\bar{T}_i$, $i \in [1 \mathinner{..} k]$. Let $\bar{T}$ be the sequence of transitions relevant to a topological sort of $\mathcal{G}$, where all E_i are eventually removed. As such, $\bar{T}$ is a sequence of transitions of components in $\bar{\mathcal{U}}$ fulfilling the precedences imposed by $\mathcal{G}$. Remarkably, $\bar{T}$ fulfills the following properties: (1) $\bar{T} \in \|\bar{\mathcal{U}}^*\|$, because the component transitions in each $\bar{T}_i$ are only constrained by the emptiness of the output terminals of $\mathcal{U}_i$, while the component transitions in T are only constrained by the availability of the events emerging from $\mathcal{U}_1, \ldots, \mathcal{U}_k$, which are checked by the mode in which the abduction $\mathcal{U}_\mathcal{O}$ is generated; (2) $\bar{T}_{\bar{\mathcal{V}}} = \bar{\mathcal{O}}$, because the sequence of observable labels generated by transitions in T equals the observation of $\mathcal{U}$ and, $\forall i \in [1 \mathinner{..} k]$, $\bar{T}_{i\bar{\mathcal{V}}_i} = \mathcal{O}_i$; and (3) the sequence of faults generated by the the transitions in T equals δ_0 and, $\forall i \in [1 \mathinner{..} k]$, $\bar{T}_{i\bar{\mathcal{R}}_i} = \delta_i$. In other words, $\bar{T}_{\bar{\mathcal{R}}} = \delta = \delta_0 \cup \delta_1 \cup \cdots \cup \delta_k$; hence, $\delta \in \varDelta(\wp(\bar{\mathcal{U}}, \bar{\mathcal{O}})$. Also, $\Im(\bar{T}) = \Im(T)$. $\square$

Lemma 13 (*Completeness*). *If $\delta \in \varDelta(\wp(\bar{\mathcal{U}}, \bar{\mathcal{O}})$, where $\delta = \bar{T}_{\bar{\mathcal{R}}}$ and $\bar{T} \in \|\bar{\mathcal{U}}^*\|$, then there is $T \in \|\mathcal{U}_\mathcal{O}\|$ ending in s'_f such that $\delta \in \varDelta(s'_\text{f})$ and $\Im(T) = \Im(\bar{T})$.*

Proof. By bottom-up induction on the tree of $\bar{\mathcal{U}}$.

(*Basis*) Assume that $\mathcal{U}$ is a leaf AU. Since $\delta \in \Delta(\wp(\bar{\mathcal{U}}, \bar{\mathcal{O}}))$, where $\delta = \bar{T}_{\bar{\mathcal{R}}}$ and $\bar{T} \in \|\bar{\mathcal{U}}^*\|$, based on Definition 2, there is $T_\delta \in \|\mathcal{U}_\delta\|$ with accepting state (s, δ) such that $\Im(T_\delta) = \Im(\bar{T})$. Hence, based on Definition 3, there is $T_\Delta \in \|\mathcal{U}_\Delta\|$ with accepting state s_f such that $\delta \in \Delta(s_\mathrm{f})$ and $\Im(T_\Delta) = \Im(T_\delta) = \Im(\bar{T})$. Thus, based on Definition 4, there is $T \in \|\mathcal{U}_\mathcal{O}\|$ ending in $s'_\mathrm{f} = (s_\mathrm{f}, n)$ such that $\delta \in \Delta(s'_\mathrm{f})$ and $\Im(T) = \Im(T_\Delta) = \Im(\bar{T})$.

(*Induction*) Assume that $\mathcal{U}$ is the parent of the AUs $\mathcal{U}_1, \ldots, \mathcal{U}_k$, where, $\forall i \in [1 \mathinner{..} k]$, if $\delta_i \in \Delta(\wp(\bar{\mathcal{U}}_i, \bar{\mathcal{O}}_i))$, where $\delta_i = \bar{T}_{i\bar{\mathcal{R}}_i}$ and $\bar{T}_i \in \|\bar{\mathcal{U}}_i^*\|$, then there is $T_i \in \|\mathcal{U}_{i\mathcal{O}}\|$ with accepting state $s'_{i\mathrm{f}}$ such that $\delta_i \in \Delta(s'_{i\mathrm{f}})$ and $\Im(T_i) = \Im(\bar{T}_i)$.

Since $\delta \in \Delta(\wp(\bar{\mathcal{U}}, \bar{\mathcal{O}})$, where $\delta = \bar{T}_{\bar{\mathcal{R}}}$ and $\bar{T} \in \|\bar{\mathcal{U}}^*\|$, we have $\delta = \delta_0 \cup \delta_1 \cup \cdots \cup \delta_k$, where δ_0 includes the faults of the components in the AU $\mathcal{U}$ and, $\forall i \in [1 \mathinner{..} k]$, δ_i includes the faults of the components in the CAS $\bar{\mathcal{U}}_i$. Since each δ_i is the diagnosis of the trajectory $\bar{T}_i$ corresponding to the restriction of $\bar{T}$ on the transitions of the components in $\bar{\mathcal{U}}_i$ such that $\bar{T}_i \in \|\bar{\mathcal{U}}_i^*\|$, $\bar{T}_{i\bar{\mathcal{V}}_i} = \mathcal{O}_i$, and $\bar{T}_{i\bar{\mathcal{R}}_i} = \delta_i$, based on Eq. (6), $\delta_i \in \wp(\bar{\mathcal{U}}_i, \bar{\mathcal{O}}_i)$. Hence, by virtue of the induction hypothesis, there is $T_i \in \|\mathcal{U}_{i\mathcal{O}}\|$ with accepting state $s'_{i\mathrm{f}}$ such that $\delta_i \in \Delta(s'_{i\mathrm{f}})$ and $\Im(T_i) = \Im(\bar{T}_i)$. Also, based on Definition 8, there is $T'_i \in \|\mathcal{U}_{i\Im}\|$ with accepting state $s_{i\mathrm{f}}$ such that $\delta_i \in \Delta(s_{i\mathrm{f}})$ and $\Im(T'_i) = \Im(T_i) = \Im(\bar{T}_i)$. Let T' be the subtrajectory of $\bar{T}$ including only the transitions of components in $\mathcal{U}$. As such, $T' \in \|\mathcal{U}^*\|$, $T'_\mathcal{V}$ equals the observation in $\mathcal{O}$ relevant to $\mathcal{U}$, and $T'_\mathcal{R} = \delta_0$. Hence, based on Definition 2, there is $T_\delta \in \|\mathcal{U}_\delta\|$ with accepting state (s, δ_0) such that $\Im(T_\delta) = \Im(T)$. Also, based on Definition 3, there is $T_\Delta \in \|\mathcal{U}_\Delta\|$ with accepting state s_f such that $\delta \in \Delta(s_\mathrm{f})$ and $\Im(T_\Delta) = \Im(T_\delta) = \Im(T)$. Thus, based on Definition 7, there is $T \in \|\mathcal{U}_\mathcal{O}\|$ with accepting state $s'_\mathrm{f} = (s_\mathrm{f}, (s_{1\mathrm{f}}, \ldots, s_{k\mathrm{f}}), (\varepsilon, \ldots, \varepsilon), n)$ such that $\delta_0 \in \Delta(s_\mathrm{f})$ and, $\forall i \in [1 \mathinner{..} k]$, $\delta_i \in \Delta(s_{i\mathrm{f}})$. Since $\delta = \delta_0 \cup \delta_1 \cup \cdots \cup \delta_k$, based on Definition 6 and according to Eq. (11), $\delta \in \Delta(s_\mathrm{f}) \otimes \Delta(s_{1\mathrm{f}}) \otimes \cdots \otimes \Delta(s_{k\mathrm{f}})$; that is, $\delta \in \Delta(s'_\mathrm{f})$. Also, $\Im(T) = \Im(\bar{T})$. □

Example 12. Consider the unit knowledge A_Δ displayed on the right side of Fig. 4 and the unit interface $B_\Im$ displayed on the right side of Fig. 6. Let $\bar{O}_A = [\, e, f \,]$ be the observation of A. The unit abduction $A_\mathcal{O}$ is shown on the right side of Fig. 5. Let $\bar{\Delta}$ be the set of diagnoses marking the final state $5 = (5, 1, \varepsilon, 2)$. Based on Eq. (11), $\bar{\Delta}$ is generated via join $\Delta(5) \otimes \Delta(1)$, where $\Delta(5)$ is the set associated with the state 5 of the unit knowledge A_Δ (on the right side of Fig. 4), namely $\{\{q\}\}$, and $\Delta(1)$ is the set associated with the state 1 of the unit interface $B_\Im$ (on the right side of Fig. 6), namely $\{\{v, w, y, z\}\}$. Hence, $\bar{\Delta} = \{\{q, v, w, y, z\}\}$. Based on Proposition 1, $\bar{\Delta}$ is the solution to the diagnosis problem $\wp(\bar{\mathcal{X}}, \bar{\mathcal{O}})$ defined in Example 4.

7 Computational Complexity

Had we adapted the diagnosis technique proposed for ASs [18,19] to the diagnosis of CASs, we would have inherited (and even exacerbated) its poor performance (see the experimental results in [17]). In contrast with diagnosis of ASs, the diagnosis engine described in Sect. 5 does not require the abduction of the whole system. Instead, it focuses on the abduction of each single AU based on the interface constraints coming from the child AUs. In doing so, it exploits the unit knowledge compiled offline.

We analyze the time complexity of solving a diagnosis problem $\wp(\mathcal{X}, \mathcal{O})$ based on the (not unreasonable) assumption that the processing time is proportional to the size (number of states) of the data structures generated by the diagnosis engine. Furthermore, we make the following *bounding assumptions*: $\mathcal{X}$ is composed of n AUs; each nonleaf AU has c child AUs, has h input terminals, and defines m emergent events; each unit knowledge (generated offline) includes k states; the length of each AU observation in $\wp(\mathcal{X}, \mathcal{O})$ is o. We also assume that the size of the DFA obtained by the determinization of an NFA compares to the size of the NFA[7]. We call this the *determinization assumption*. We first consider the (upper bound of the) complexity $\mathcal{C}$ of the abduction $\mathcal{U}_{\mathcal{O}}$ of a leaf AU (at level zero, that is, without children). Based on Definition 4, $\mathcal{C}_{\mathcal{U}_{\mathcal{O}}} = k \cdot o$. In order to estimate the complexity of each unit abduction $\mathcal{U}_{\mathcal{O}}$ at level one, where all children of $\mathcal{U}$ are leaf AUs, two steps have to be analyzed: (1) generation of c interfaces and (2) generation of $\mathcal{U}_{\mathcal{O}}$ based on Definition 7. As to step (1), on the ground of the determinization assumption, the number of states of the interfaces of the child AUs is $c \cdot k \cdot o$. Based on Definition 7, the (upper bound of the) number of states generated by step (2) for each unit abduction at level one is $k \cdot (k \cdot o)^c \cdot m^h \cdot o = (k \cdot o)^{c+1} \cdot m^h$. Owing to the factor $(k \cdot o)^{c+1}$, the contribution $c \cdot k \cdot o$ of step (1) can be neglected; hence,

$$\mathcal{C}_{\mathcal{U}_{\mathcal{O}}} = (k \cdot o)^{c+1} \cdot m^h \,. \tag{13}$$

The complexity of each unit abduction $\mathcal{U}_{\mathcal{O}}$ at the second level (where $\mathcal{U}$ is the grandparent of leaf AUs) can be computed as before, where the size of each interface equals $\mathcal{C}_{\mathcal{U}_{\mathcal{O}}}$ in Eq. (13), namely

$$\mathcal{C}_{\mathcal{U}_{\mathcal{O}}} = k \cdot ((k \cdot o)^{c+1} \cdot m^h)^c \cdot m^h \cdot o = (k \cdot o)^{c^2+c+1} \cdot m^{(c+1)\cdot h} \,. \tag{14}$$

At level d (depth of the tree) of the root AU, the complexity of the unit abduction is

$$\mathcal{C}_{\mathcal{U}_{\mathcal{O}}} = (k \cdot o)^{c^d + c^{d-1} + \cdots + c^2 + c + 1} \cdot m^{(c^{d-1} + \cdots + c^2 + c + 1)\cdot h} \,. \tag{15}$$

Since $1 + c + c^2 + \cdots + c^d = n$, with n being the number of AUs in the CAS, the dominant factor in Eq. (15) is $(k \cdot o)^n$. In other words, the complexity of the unit abduction is exponential in the number of AUs in the CAS.

Finally, we consider the space complexity of knowledge compilation, which is performed offline. We assume that each AU includes p components and l

[7] In the worst case, if n is the number of states of the NFA, then the number of states of the DFA is 2^n. However, this is an extremely improbable scenario since, in practice, the size of the DFA resulting from the determinization of an NFA generated randomly typically compares with the size of the NFA (cf. Fig. 4). If the NFA includes a considerable percentage of ε-transitions, then the size of the DFA is likely to be substantially smaller than the size of the NFA [4].

links, with each component having s states and generating q different events for each connected link. Each matcher (the DFA recognizing the occurrence of an emergent event) has μ states. The number of possible faults is f. The space complexity of the unit space $\mathcal{U}^*$ is $\mathcal{C}_{\mathcal{U}^*} = s^p \cdot (q+1)^\ell \cdot \mu^m$. The space complexity of the unit labeling $\mathcal{U}_\delta$ is

$$\mathcal{C}_{\mathcal{U}_\delta} = \mathcal{C}_{\mathcal{U}^*} \cdot 2^f = s^p \cdot (q+1)^\ell \cdot \mu^m \cdot 2^f \,. \tag{16}$$

According to the determinization assumption, the complexity of the unit knowledge equals the complexity of the unit labeling, namely $\mathcal{C}_{\mathcal{U}_\Delta} = \mathcal{C}_{\mathcal{U}_\delta}$.

8 Conclusion

The contributions of this paper are the specification of a class of DESs inspired by the complexity paradigm, called complex active systems, and a knowledge-compilation technique that speeds up online diagnosis for such systems. Most notably, the shift from ASs to CASs does not come with an additional cost in the diagnosis task; on the contrary, the diagnosis technique is not only sound and complete, but also viable compared to the diagnosis of ASs. In fact, since a state of the AS includes the states of *all* components and the states of *all* links, the complexity of the abduction of the whole AS in diagnosis of ASs is exponential both in the number of components *and* in the number of links.

This theoretical expectation is confirmed by the experimental results presented in [17], with the diagnosis engine being not supported by any compiled knowledge. Two diagnosis engines have been implemented, one *greedy* and one *lazy*. The greedy engine makes use of the same technique of behavior reconstruction adopted in diagnosis of ASs [19], while the lazy engine operates similarly to the technique proposed in this paper (although without any compiled knowledge). The results clearly show that the processing time of the lazy engine increases almost linearly with the size of the system, in contrast with the processing time of the greedy engine, which grows exponentially.

On the ground of these results, we expect that the technique proposed in this paper, which is essentially a lazy engine exploiting compiled knowledge, is still more efficient than the lazy engine in [17], since the low-level model-based reasoning, performed offline and incorporated in the compiled knowledge, is avoided online. Experiments to confirm this intuition will be carried out.

But, why should we model a real system as a CAS rather than a flat AS? In our opinion, the reason is twofold. First, real event-driven systems are typically organized hierarchically, at different abstraction levels. Modeling one such system as a (flat) AS may be awkward because of the mismatch between the hierarchical organization of the structure and behavior of the real system and the flat organization of the modeling AS. CASs provide a natural modeling support against such a mismatch, where emergent events are the means of communication between strata at different abstraction levels, thereby supporting behavior stratification. In short, the first benefit is ergonomics in the modeling task. Second, once the real system is modeled as a CAS, the diagnosis task provides the sound and complete solution to a diagnosis problem more efficiently than in a (flat) AS. Since diagnosis is potentially interesting to the degree that it is accurate and viable, the second benefit is correctness and viability of the diagnosis task.

As diagnosis of CASs is still in its infancy, several research paths can be envisaged. First, the tree-based topology of the CAS can be relaxed to a directed acyclic graph (DAG), where an AU can have several parent units. Moreover, each node of the DAG can be generalized to a (possibly cyclic) network of AUs. Second, the language of the patterns defining emergent events can be extended beyond regular expressions, based on more powerful grammars, possibly enriched by semantic rules. Third, the diagnosis task, which in this paper is assumed to be *a posteriori*, that is, performed at the reception of a complete temporal observation of the CAS, can be made *reactive*, where diagnosis is performed as soon as a piece of observation is received. Finally and more challengingly, an *adaptive* CAS can be envisaged, where the behavior of components and AUs can change based on specific patterns of events, so as to convert a nonconstructive or even disruptive behavior to a more constructive behavior.

Acknowledgments. This work was supported in part by the Zhejiang Provincial Natural Science Foundation of China (No. LY16F020004).

References

1. Atay, F., Jost, J.: On the emergence of complex systems on the basis of the coordination of complex behaviors of their elements: synchronization and complexity. Complexity **10**(1), 17–22 (2004)
2. Bossomaier, T., Green, D.: Complex Systems. Cambridge University Press, Cambridge (2007)
3. Brand, D., Zafiropulo, P.: On communicating finite-state machines. J. ACM **30**(2), 323–342 (1983)
4. Brognoli, S., Lamperti, G., Scandale, M.: Incremental determinization of expanding automata. Comput. J. **59**(12), 1872–1899 (2016)
5. Cassandras, C., Lafortune, S.: Introduction to Discrete Event Systems, 2nd edn. Springer, New York (2008). https://doi.org/10.1007/978-0-387-68612-7
6. Chittaro, L., Ranon, R.: Hierarchical model-based diagnosis based on structural abstraction. Artif. Intell. **155**(1–2), 147–182 (2004)
7. Dechter, R.: Constraint Processing. The Morgan Kaufmann Series in Artificial Intelligence. Morgan Kaufmann Publishers Inc., San Francisco (2003)
8. Goles, E., Martinez, S. (eds.): Complex Systems. Springer, Dordrecht (2001). https://doi.org/10.1007/978-94-010-0920-1
9. Harel, D.: Statecharts: a visual formalism for complex systems. Sci. Comput. Program. **8**(3), 231–274 (1987)
10. Hopcroft, J., Motwani, R., Ullman, J.: Introduction to Automata Theory, Languages, and Computation, 3rd edn. Addison-Wesley, Reading (2006)
11. Huang, C., Darwiche, A.: Inference in belief networks: A procedural guide. Int. J. Approx. Reason. **15**(3), 225–263 (1996)

12. Huang, J., Darwiche, A.: On compiling system models for faster and more scalable diagnosis. In: 20th National Conference on Artificial Intelligence (AAAI 2005), Pittsburgh, PA, pp. 300–306 (2005)
13. Idghamishi, A., Zad, S.: Fault diagnosis in hierarchical discrete-event systems. In: 43rd IEEE Conference on Decision and Control, Paradise Island, Bahamas, pp. 63–68 (2004)
14. Jéron, T., Marchand, H., Pinchinat, S., Cordier, M.: Supervision patterns in discrete event systems diagnosis. In: Seventeenth International Workshop on Principles of Diagnosis (DX 2006), Peñaranda de Duero, Spain, pp. 117–124 (2006)
15. Jiang, S., Huang, Z., Chandra, V., Kumar, R.: A polynomial algorithm for testing diagnosability of discrete event systems. IEEE Trans. Autom. Control **46**(8), 1318–1321 (2001)
16. Kan John, P., Grastien, A.: Local consistency and junction tree for diagnosis of discrete-event systems. In: Eighteenth European Conference on Artificial Intelligence (ECAI 2008), pp. 209–213. IOS Press, Amsterdam (2008)
17. Lamperti, G., Quarenghi, G.: Intelligent monitoring of complex discrete-event systems. In: Czarnowski, I., Caballero, A.M., Howlett, R.J., Jain, L.C. (eds.) Intelligent Decision Technologies 2016. SIST, vol. 56, pp. 215–229. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-39630-9_18
18. Lamperti, G., Zanella, M.: Diagnosis of Active Systems: Principles and Techniques. Springer International Series in Engineering and Computer Science, vol. 741. Springer, Dordrecht (2003). https://doi.org/10.1007/978-94-017-0257-7
19. Lamperti, G., Zanella, M., Zhao, X.: Introduction to Diagnosis of Active Systems. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-92733-6
20. Lamperti, G., Zhao, X.: Diagnosis of higher-order discrete-event systems. In: Cuzzocrea, A., Kittl, C., Simos, D.E., Weippl, E., Xu, L. (eds.) CD-ARES 2013. LNCS, vol. 8127, pp. 162–177. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40511-2_12
21. Lamperti, G., Zhao, X.: Specification and model-based diagnosis of higher-order discrete event systems. In: IEEE International Conference on Systems, Man, and Cybernetics (SMC 2013), Manchester, United Kingdom, pp. 2342–2347 (2013)
22. Lamperti, G., Zhao, X.: Diagnosis of complex active systems with uncertain temporal observations. In: Buccafurri, F., Holzinger, A., Kieseberg, P., Tjoa, A.M., Weippl, E. (eds.) CD-ARES 2016. LNCS, vol. 9817, pp. 45–62. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45507-5_4
23. Lamperti, G., Zhao, X.: Viable diagnosis of complex active systems. In: IEEE International Conference on Systems, Man, and Cybernetics (SMC 2016), Budapest, pp. 457–462 (2016)
24. Licata, I., Sakaji, A.: Physics of Emergence and Organization. World Scientific, Singapore (2008)
25. Mozetič, I.: Hierarchical diagnosis. In: Hamscher, W., Console, L., de Kleer, J. (eds.) Readings in Model-Based Diagnosis. Morgan Kaufmann (1992)
26. Paoli, A., Lafortune, S.: Diagnosability analysis of a class of hierarchical state machines. J. Discrete Event Dyn. Syst. Theory Appl. **18**(3), 385–413 (2008)
27. Sampath, M., Sengupta, R., Lafortune, S., Sinnamohideen, K., Teneketzis, D.: Diagnosability of discrete-event systems. IEEE Trans. Autom. Control **40**(9), 1555–1575 (1995)
28. Schumann, A., Huang, J.: A scalable jointree algorithm for diagnosability. In: Twenty-Third National Conference on Artificial Intelligence (AAAI 2008), Chicago, IL, pp. 535–540 (2008)

29. Shenoy, P., Shafer, G.: Propagating belief functions with local computations. IEEE Expert **1**(3), 43–52 (1986)
30. Siddiqi, S., Huang, J.: Hierarchical diagnosis of multiple faults. In: 20th International Joint Conference on Artificial Intelligence (IJCAI 2007), Hyderabad, India, pp. 581–586 (2007)
31. Stumptner, M., Wotawa, F.: Diagnosing tree-structured systems. Artif. Intell. **127**(1), 1–29 (2001)
32. West, G.: Scale: The Universal Laws of Growth, Innovation, Sustainability, and the Pace of Life in Organisms, Cities, Economies, and Companies. Penguin Press, New York (2017)
33. Yoo, T., Lafortune, S.: Polynomial-time verification of diagnosability of partially observed discrete-event systems. IEEE Trans. Autom. Control **47**(9), 1491–1495 (2002)