

Diagnosis of Discrete-Event Systems with Stratified Behavior

Gianfranco Lamperti¹ and Xiangfu Zhao²

¹Department of Information Engineering, University of Brescia, Italy

e-mail: lamperti@ing.unibs.it

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

e-mail: xiangfuzhao@zjnu.cn

Abstract

A gap still exists between the complexity of real discrete-event systems (DESs) and the effectiveness of state-of-the-art diagnosis techniques. To deal with this gap, a novel class of discrete-event systems, called higher-order DESs (HDESs) is introduced, along with a relevant diagnosis technique. The behavior of a HDES is stratified, resulting in a hierarchy of cohabiting sub-DESs, each one living its own life. The communication between subsystems at different levels relies on complex events, occurring when specific patterns of transitions are matched. Diagnosis of HDESs is context-sensitive and scalable.

1 Introduction

Diagnosis of DESs [Cassandras and Lafortune, 1999] has been attracting attention of the scientific community since the seminal work of [Sampath *et al.*, 1996]. However, a gap still exists between the complexity of real DESs and the effectiveness of state-of-the-art diagnosis techniques. A complex DES is not necessarily large (even though a large DES is likely to be complex). In our meaning, complexity refers to the mode in which the DES is organized, at different levels of abstraction, with each level being characterized by its proper behavior, which depends on the behaviors of lower-level layers, yet differs from just the composition of them. This property is called *behavior stratification*. In the literature, DESs are typically modeled as networks of interacting components, where the behavior of each component is described by a communicating automaton [Brand and Zafirovulo, 1983]. However, complexity of the DES has become a research issue only recently. Previous research has mainly focused on relevant yet different aspects, including incrementality [Baroni *et al.*, 1999; Grastien *et al.*, 2005], distribution/decentralization [Pencol , 2000; Debouk *et al.*, 2000; Pencol  *et al.*, 2001; Debouk *et al.*, 2003; Grastien *et al.*, 2004; Pencol  and Cordier, 2005; Qiu and Kumar, 2006], and uncertainty/incompleteness [Lamperti and Zanella, 2002; Zhao and Ouyang, 2008; Lamperti and Zanella, 2011b; Kwong and Yonge-Mallo, 2011; Zhao *et al.*, 2012]. The notion of context-sensitive diagnosis has been introduced for DESs that are organized within abstraction hierarchies, so that candidate diagnoses can be generated at different abstraction levels [Lamperti and Zanella, 2011a]. Even in that work, albeit the diagnosis depends on the context, the DES is assumed to be a network of components with no behavior stratification. When behavior stratification occurs, the DES is called a *higher-order DES* (HDES).

2 Higher-Order Discrete-Event System

A higher-order DES, namely \mathcal{H} , is a tree where nodes are *components*. Leaf nodes are *basic components*, while internal nodes are *complex components*. The set of child components of a complex component X is indicated by $\mathcal{C}(X)$. Each (either basic or complex) component in \mathcal{H} is defined in terms of a *topological model* and a *behavioral model*. The topological model consists of a set of *input terminals* and a set of *output terminals*. Components in $\mathcal{C}(X)$ are connected to one another through *links*, with each link exiting the output terminal of one component and entering the input terminal of another component. These connections form a network $\mathcal{N}(X)$. Let \mathbf{I} and \mathbf{O} denote the input and output terminals, respectively, of a component C . The behavioral model of C is a communicating automaton $(\mathcal{S}, \mathcal{I}, \mathcal{O}, \mathcal{T})$, where \mathcal{S} is the set of states, \mathcal{I} the set of input events, \mathcal{O} the set of output events, and $\mathcal{T} : \mathcal{S} \times (\mathcal{I} \times \mathbf{I}) \times 2^{(\mathcal{O} \times \mathbf{O})} \mapsto 2^{\mathcal{S}}$ the (nondeterministic) transition function. As such, a transition is triggered by an input event and generates a (possibly empty) set of output events. The latter are thus made available as input events at the corresponding component terminals, while the input (triggering) event is consumed. A transition can be triggered only if all links, towards which output events are generated, are empty (no event is in the link). Each complex component X is endowed with a *Cot* additional (virtual) input terminal, which is sensitive to *complex events*. A complex event is a set of *pattern events*. A pattern event occurs when the network $\mathcal{N}(X)$ undergoes a string of transitions matching a given regular expression. The alphabet of such a regular expression is the whole set of transitions of components in $\mathcal{C}(X)$. In general, for each complex component X , a set $\mathcal{P}(X)$ of *patterns* is defined, with each pattern being a pair (p, r) , where p is the name of a pattern event and r a regular expression on transitions of $\mathcal{C}(X)$. Several pattern events may occur simultaneously. In fact, given a string \mathcal{T} of transitions of components in $\mathcal{C}(X)$, each suffix of \mathcal{T} matching a regular expression in $\mathcal{P}(X)$ gives rise to a pattern event. The whole set of these pattern events forms a complex event.

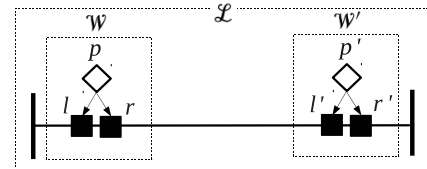


Figure 1: HDES: protected power transmission line.

Example 1. Shown in Fig. 1 is a HDES representing a power transmission line \mathcal{L} . On both sides, the line is pro-

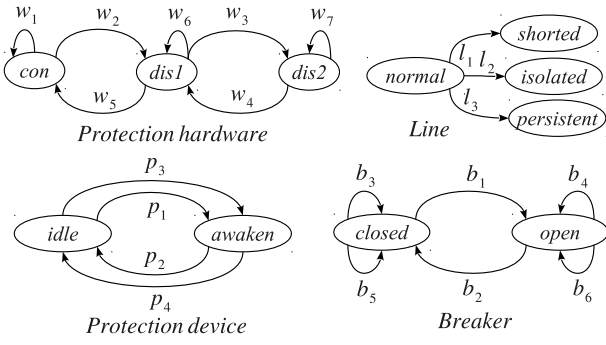


Figure 2: Behavioral models.

tected from short circuits by a protection hardware, \mathcal{W} and \mathcal{W}' , each composed of a protection device, p and p' , respectively, and two breakers, l and r , and l' and r' , respectively. Boxes denote complex components \mathcal{L} , \mathcal{W} , and \mathcal{W}' . We assume that the output terminal of the protection device is excited by two links directed to the input terminals of the two breakers. The protection device is sensitive to short circuits on the line, detected as lowering of voltage, in which case it commands the breakers to open in order to isolate the line (just one open breaker on both sides is sufficient for isolation). Once the line is isolated, the short circuit is expected to vanish. If so, the protection device commands both breakers to close in order to reconnect the line (all breakers need to be closed). However, faulty behavior may occur:

- The protection device sends the wrong command;
- The breaker does not react to the command of protection device (thereby remaining in its state);
- The protection hardware fails to either disconnect or connect the line (if just one breaker does not open, the protection hardware is normal, as disconnection occurs; for the connection, both breakers must close);
- The line is not isolated, or once the short circuit has vanished, the line is not reconnected, or after reconnection the short circuit still persists (e.g. a tree fallen on the line).

Table 1: Details for transitions of models in Fig. 2.

T	Action performed by component transition T
p_1	Detects low voltage and outputs <i>op</i> (open) event
p_2	Detects normal voltage and outputs <i>cl</i> (close) event
p_3	Detects low voltage, yet outputs <i>cl</i> event
p_4	Detects normal voltage, yet outputs <i>op</i> event
b_1	Consumes <i>op</i> event and opens
b_2	Consumes <i>cl</i> event and closes
b_3	Consumes <i>op</i> event, yet keeps closed
b_4	Consumes <i>cl</i> event, yet keeps open
b_5	Consumes <i>cl</i> event
b_6	Consumes <i>op</i> event
w_1	Consumes nd (not disconnected) complex event
w_2, w_3	Consumes di (disconnected) complex event
w_4, w_5	Consumes co (connected) complex event
w_6, w_7	Consumes nc (not connected) complex event
l_1	Consumes ni (not isolated) complex event
l_2	Consumes nr (not reconnected) complex event
l_3	Consumes ps (persistent short) complex event

Outlined in Fig. 2 are the behavioral models. Details on component transitions are provided in Table 1. Patterns relevant to complex events in Table 1 are defined in Table 2. For instance, consider pattern event *ps* (persistent short circuit) and the corresponding regular expression, where $*$ means

repetition zero or more times, $+$ means repetition one or more times, and \neg (negation) means any transition different from its argument. Event *ps* occurs when either \mathcal{W} or \mathcal{W}' repeats one or more times the following sequence of transitions: it closes (w_5) and then, after zero or more occurrences of w_1 , it opens again (w_2), followed by zero or more transitions other than w_5 . Notice that *ps* is a single pattern event, while **ps** is the singleton $\{ps\}$ (complex event).¹ \diamond

2.1 Pattern Space

In order to detect complex events, the state of the matching of patterns is to be maintained somewhere. To this end:

- For each pattern (p, r) , a deterministic *pattern automaton* A equivalent to regular expression r is generated, where final states are marked by pattern event p .
- For each complex component X for which the set $\{A_1, \dots, A_k\}$ of pattern automata were generated, a *pattern space*, written $Pts(X)$, is created as follows:
 1. A nondeterministic automaton N is created by generating initial state S_0 and one empty transition from S_0 to each initial state of A_i , $i \in [1..k]$;
 2. In each A_i , $i \in [1..k]$, an empty transition from each non-initial state to S_0 is inserted;²
 3. N is determinized into $Pts(X)$, where each final state S is marked by the union **p** of the pattern events that are associated with the states in S that are final in the corresponding pattern automaton.³

Table 2: Specification of patterns by regular expressions.

Pattern event	Regular expression
<i>di</i>	$b_1(l) \mid b_1(r)$
<i>co</i>	$b_2(l) \mid b_2(r)$
<i>nd</i>	$p_3(p) \mid p_1(p)((b_3(l)b_3(r)) \mid (b_3(r)b_3(l)))$
<i>nc</i>	$p_4(p) \mid p_2(p)(b_5(r)? b_4(l) \mid b_5(l)? b_4(r))$
<i>ni</i>	$w_1(\mathcal{W}) \mid w_1(\mathcal{W}')$
<i>nr</i>	$w_6(\mathcal{W}) \mid w_7(\mathcal{W}) \mid w_6(\mathcal{W}') \mid w_7(\mathcal{W}')$
<i>ps</i>	$(w_5(\mathcal{W})w_1(\mathcal{W})^*w_2(\mathcal{W})(\neg w_5(\mathcal{W}))^+ \mid (w_5(\mathcal{W}')w_1(\mathcal{W}')^*w_2(\mathcal{W}')(\neg w_5(\mathcal{W}'))^+)^+$

Example 2. Based on Example 1, consider complex component \mathcal{W} , whose patterns are defined in Table 2 (top). Following the steps specified above, $Pts(\mathcal{W})$ is generated as detailed in Table 3. The main part of the table represents the transition function, where for each component transition $T \in \{p_1(p), \dots, b_5(r)\}$ (listed in the first column), and for each state \mathcal{P}_i , $i \in [0..10]$ (listed in the first row), the reached state is indicated in the cell (T, \mathcal{P}_i) . Moreover, highlighted states are final, namely $\mathcal{P}_1, \mathcal{P}_4, \mathcal{P}_7, \mathcal{P}_8$. Complex events associated with final states are listed in the last row, namely **di**, **co**, **nc**, and **nd** (details are in Table 1). These are singletons of the homonymous pattern event. \diamond

Proposition 1. The set **p** marking a final state S_f of $Pts(X)$ is composed of the pattern events p such that $(p, r) \in \mathcal{P}(X)$, \mathcal{T} is a string in the language of $Pts(X)$ ending at S_f , \mathcal{T}' is a string matching regular expression r , and \mathcal{T}' is a suffix of \mathcal{T} .

Proof. (Sketch) In creating $Pts(X)$, if we omit step 2 then the language of $Pts(X)$ will be the union of the languages of regular expressions involved in $\mathcal{P}(X)$, where each string

¹Although a complex event is a set of pattern events, incidentally in our example all complex events are singletons.

²This allows for pattern-matching of overlapping strings.

³Each state S of the deterministic automaton is identified by a subset of the states of the equivalent nondeterministic automaton.

Table 3: Tabular specification of pattern space $Pts(\mathcal{W})$.

$T \setminus P_i$	P_0	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}
$p_1(p)$	P_2	P_2	P_2	P_2	P_2	P_2	P_2	P_2	P_2	P_2	P_2
$p_2(p)$	P_3	P_3	P_3	P_3	P_3	P_3	P_3	P_3	P_3	P_3	P_3
$p_3(p)$	P_8	P_8	P_8	P_8	P_8	P_8	P_8	P_8	P_8	P_8	P_8
$p_4(p)$	P_7	P_7	P_7	P_7	P_7	P_7	P_7	P_7	P_7	P_7	P_7
$b_1(l)$	P_1	P_1	P_1	P_1	P_1	P_1	P_1	P_1	P_1	P_1	P_1
$b_1(r)$	P_1	P_1	P_1	P_1	P_1	P_1	P_1	P_1	P_1	P_1	P_1
$b_2(l)$	P_4	P_4	P_4	P_4	P_4	P_4	P_4	P_4	P_4	P_4	P_4
$b_2(r)$	P_4	P_4	P_4	P_4	P_4	P_4	P_4	P_4	P_4	P_4	P_4
$b_3(l)$	—	—	P_5	—	—	—	P_8	—	—	—	—
$b_3(r)$	—	—	P_6	—	—	P_8	—	—	—	—	—
$b_4(l)$	—	—	—	P_7	—	—	—	—	—	—	P_7
$b_4(r)$	—	—	—	P_7	—	—	—	—	—	P_7	—
$b_5(l)$	—	—	—	P_9	—	—	—	—	—	—	—
$b_5(r)$	—	—	—	P_{10}	—	—	—	—	—	—	—
	di			co			nc		nd		

ending at S_f matches the regular expression associated with a pattern event in \mathbf{p} . Consequently, the statement of the theorem should be restricted in the last condition by $\mathcal{T}' = \mathcal{T}$. The more relaxed condition, namely \mathcal{T}' being a suffix of \mathcal{T} , comes from step 2 of the construction: in general, because of additional empty transitions, each string \mathcal{T} ending at S_f matches regular expressions only in its suffixes. \square

2.2 Behavior Space

Starting from its initial state \mathcal{H}_0 , HDES \mathcal{H} may perform a sequence of component transitions within its *behavior space*, written $Bsp(\mathcal{H}, \mathcal{H}_0)$, which is a finite automaton

$$Bsp(\mathcal{H}, \mathcal{H}_0) = (\mathbf{S}, \mathbf{T}, S_0).$$

\mathbf{S} is the set of states $(\mathcal{S}, \mathcal{E}, \mathcal{P})$, with $\mathcal{S} = (s_1, \dots, s_n)$ being the tuple of states of (both basic and complex) components in \mathcal{H} . $\mathcal{E} = (e_1, \dots, e_m)$ is the tuple of events at input terminals of components in \mathcal{H} (ϵ indicates no event), and $\mathcal{P} = (P_1, \dots, P_k)$ the tuple of pattern-space states. $S_0 = (\mathcal{H}_0, \mathcal{E}_0, \mathcal{P}_0)$ is the initial state, where $\mathcal{E}_0 = (\epsilon, \dots, \epsilon)$ and $\mathcal{P}_0 = (P_{10}, \dots, P_{k0})$ the tuple of the initial states of pattern spaces $Pts(X_1), \dots, Pts(X_k)$, respectively. \mathbf{T} is the transition function, where

$$(\mathcal{S}, \mathcal{E}, \mathcal{P}) \xrightarrow{T} (\mathcal{S}', \mathcal{E}', \mathcal{P}') \in \mathbf{T} \text{ if and only if:}$$

1. $T = s \xrightarrow{(e, I) | E_{out}} s'$ (where e is the input event and E_{out} the output events with relevant terminals) is a transition of a (either basic or complex) component C such that s equals one element of \mathcal{S} , I is an input terminal of C , and $\mathcal{E}(I) = e$;
2. \mathcal{S}' differs from \mathcal{S} only in s' replacing s ;
3. If $C \in \mathcal{C}(X_i)$, $i \in [1..k]$, then \mathcal{P}' differs from \mathcal{P} only in the i -th element as follows:

$$\mathcal{P}'(P_i) = \begin{cases} \bar{P} & \text{if } \mathcal{P}(P_i) \xrightarrow{T} \bar{P} \in Pts(X_i) \\ P_{i0} & \text{otherwise;} \end{cases}$$

4. \mathcal{E}' differs from \mathcal{E} based on these conditions:
 - (a) $\mathcal{E}'(I) = \epsilon$ (event e is consumed);
 - (b) $\forall (o, O) \in E_{out}$, denoting with I' the terminal entered by the link exiting O , we have: $\mathcal{E}(I') = \epsilon$ (no event is initially present at terminal I') and $\mathcal{E}'(I') = o$ (o is then present at terminal I');
 - (c) If $\mathcal{P}'(P_i) \neq \mathcal{P}(P_i)$, $i \in [1..k]$, $\mathcal{P}'(P_i)$ is final in $Pts(X_i)$ and marked by complex event \mathbf{p} , then $\mathcal{E}(Cot) = \epsilon$, $\mathcal{E}'(Cot) = \mathbf{p}$.

As such, transitions in $Bsp(\mathcal{H}, \mathcal{H}_0)$ are marked by transitions of components in \mathcal{H} . The new state not only reflects

the consumption of input event e of the component transition T and the generation of the output events in E_{out} : it also accounts for the possible occurrence of a complex event \mathbf{p} .

A string in the language of $Bsp(\mathcal{H}, \mathcal{H}_0)$ is a *history* of \mathcal{H} . The behavior space is defined for formal reasons only, as its actual materialization is impractical in real HDESs.

3 Diagnosis Problem

Diagnosing a HDES means finding the faults in its history. A history can be observed only in its observable transitions, as a sequence of *observation labels*, called the *trace* of the history, with each label being associated with an observable transition. The diagnosis process is complicated by two facts. First, several histories may generate the same trace. Second, because of noise and distribution of the channels conveying labels from the HDES, rather than a sequence of labels, the trace is perceived as a DAG, called *temporal observation*, where each node contains a set of observation labels and each arc represents partial (rather than total) temporal ordering between observation labels. Consequently, several *candidate traces* are observed, each one made up by choosing a label in each node of the DAG fulfilling the partial ordering imposed by arcs. Furthermore, since several (even infinite) histories may be consistent with the same trace, the diagnosis output is a set of *candidate diagnoses*, with each candidate corresponding to a subset of the possible histories. However, despite the possible infinite set of histories consistent with the temporal observation, the set of candidate diagnoses is always finite (being it bounded from above by the powerset of component transitions).

A *diagnosis problem* for a HDES \mathcal{H} is a quadruple

$$\wp(\mathcal{H}) = (\mathcal{H}_0, \mathcal{V}, \mathcal{O}, \mathcal{R}), \text{ where}$$

- \mathcal{H}_0 is the initial state of \mathcal{H} ;
- \mathcal{V} is the *viewer* of \mathcal{H} , a set of pairs (T, ℓ) , where T is a component transition and ℓ an observation label, with $h_{[\mathcal{V}]}$ denoting the trace of history h based on \mathcal{V} ;
- \mathcal{O} is the *temporal observation* of \mathcal{H} , with $\|\mathcal{O}\|$ denoting the set of candidate traces in \mathcal{O} ;
- \mathcal{R} is the *ruler* of \mathcal{H} , a set of pairs (T, f) , where T is a component transition and f a fault label, with $h_{[\mathcal{R}]}$ denoting the *diagnosis* of history h based on \mathcal{R} , defined as:

$$h_{[\mathcal{R}]} = \{f \mid T \in h, (T, f) \in \mathcal{R}\}.$$

If a transition T is included in \mathcal{V} , then it is *observable*, otherwise it is *unobservable*. If T is included in \mathcal{R} , then it is *faulty*, otherwise it is *normal*.

The *solution* Δ of $\wp(\mathcal{H})$ is the set of candidate diagnoses:

$$\Delta(\wp(\mathcal{H})) = \{\delta \mid h \in Bsp(\mathcal{H}, \mathcal{H}_0), h_{[\mathcal{V}]} \in \|\mathcal{O}\|, \delta = h_{[\mathcal{R}]}\}.$$

Each candidate diagnosis is the set of faulty transitions of a history that is consistent with the temporal observation.

For practical reasons, instead of processing observation \mathcal{O} , the *index space* of \mathcal{O} is generated [Lamperti and Zanella, 2002], namely $Isp(\mathcal{O})$. This is a deterministic automaton whose language equals $\|\mathcal{O}\|$ (the set of candidate traces).

Example 3. With reference to Example 1, we define the diagnostic problem for the left-hand side protection-hardware as $\wp(\mathcal{W}) = (\mathcal{W}_0, \mathcal{V}, \mathcal{O}, \mathcal{R})$, where:

- In \mathcal{W}_0 both breakers are *closed*, protection device is *idle*, and protection hardware is *con* (see Fig. 2);
- $\mathcal{V} = \{(b_1(l), opl), (b_1(r), opr), (b_2(l), cll), (b_2(r), clr), (p_1, awk), (p_2, ide), (p_3, awk), (p_4, ide)\}$;
- \mathcal{O} is the temporal observation displayed in Fig. 3 (left);

- $\mathcal{R} = \{(b_3(l), nol), (b_3(r), nor), (b_4(l), ncl), (b_4(r), ncr), (p_3, fop), (p_4, fcp), (w_1, fdw), (w_6, fcw), (w_7, fcw)\}, (l_1, fil), (l_2, frl), (l_3, psl)\}$.

Temporal observation \mathcal{O} (Fig. 3, left) includes four nodes, with ω_1 and ω_4 containing two observation labels (ϵ is the empty label). As such, \mathcal{O} is uncertain. Because of this uncertainty and partial temporal ordering, \mathcal{O} embodies six candidate traces, which are the strings of the language of the index space $Isp(\mathcal{O})$ displayed on the right of Fig. 3 (where \mathfrak{S}_3 and \mathfrak{S}_5 are final).

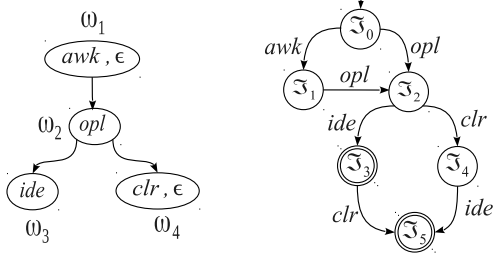


Figure 3: Temporal observation \mathcal{O} (left) and $Isp(\mathcal{O})$ (right).

4 Diagnosis Computation

The definition of diagnosis-problem solution is not operational in nature: it refers to the behavior space, which is assumed not to be available in practice. The diagnosis engine is expected to be sound and complete in generating the solution of the problem, without the availability of the behavior space. To this end, it reconstructs only the subpart of the behavior space that is consistent with the temporal observation. In doing so, the reconstruction needs to keep four sorts of information: the state of components, the state of input terminals, the state of the matching of pattern events, and the state of the matching of the temporal observation. Specifically, the solution of a diagnostic problem $\wp(\mathcal{H}) = (\mathcal{H}_0, \mathcal{V}, \mathcal{O}, \mathcal{R})$ is computed in three steps:

- Generating the *index space* of temporal observation \mathcal{O} ;
- Generating the subspace of $Bsp(\mathcal{H}, \mathcal{H}_0)$ that is consistent with temporal observation \mathcal{O} , based on viewer \mathcal{V} , called the *behavior* of $\wp(\mathcal{H})$, written $Bhv(\wp(\mathcal{H}))$;
- Decorating the states of behavior $Bhv(\wp(\mathcal{H}))$ by the associated set of candidate diagnoses.

The actual solution $\Delta(\wp(\mathcal{H}))$ is the union of the decorations associated with final states of $Bhv(\wp(\mathcal{H}))$ (see Theorem 1).

Formally, $Bhv(\wp(\mathcal{H}))$ is defined as follows. Let \mathbf{S} be the domain of tuples (s_1, \dots, s_n) of states of components in $\mathcal{C}(\mathcal{H})$. Let \mathbf{E} be the domain of tuples (e_1, \dots, e_m) of events at input terminals (other than *Ext*) of components in $\mathcal{C}(\mathcal{H})$. Let \mathfrak{S} be the domain of states in $Isp(\mathcal{O})$. Let \mathcal{P} be the domain of tuples (P_1, \dots, P_k) of pattern-space states. The behavior of $\wp(\mathcal{H})$ is a deterministic automaton:

$$Bhv(\wp(\mathcal{H})) = (\mathcal{S}, \mathcal{T}, S_0, \mathcal{S}_f), \text{ where}$$

- $\mathcal{S} \subseteq \mathbf{S} \times \mathbf{E} \times \mathcal{P} \times \mathfrak{S}$ is the set of states;
- $S_0 = (\mathcal{H}_0, \mathcal{E}_0, \mathcal{P}_0, \mathfrak{S}_0)$ is the initial state, where $\mathcal{E}_0 = (\epsilon, \dots, \epsilon)$, $\mathcal{P}_0 = (P_{10}, \dots, P_{k0})$ the tuple of the initial states of pattern spaces $Pts(X_1), \dots, Pts(X_k)$, respectively, and \mathfrak{S}_0 the initial state of $Isp(\mathcal{O})$;
- $\mathcal{S}_f = \{(\mathcal{S}, \mathcal{E}, \mathcal{P}, \mathfrak{S}) \mid \mathcal{E} = (\epsilon, \dots, \epsilon), \mathfrak{S} \text{ is final}\}$ is the set of final states;
- \mathcal{T} is the transition function, where

$$(\mathcal{S}, \mathcal{E}, \mathcal{P}, \mathfrak{S}) \xrightarrow{\mathcal{T}} (\mathcal{S}', \mathcal{E}', \mathcal{P}', \mathfrak{S}') \in \mathcal{T} \text{ if and only if:}$$

1. Conditions 1–4 on \mathcal{S}' , \mathcal{E}' , and \mathcal{P}' , for the transition function of $Bsp(\mathcal{H}, \mathcal{H}_0)$, hold;
2. $\mathfrak{S}' = \begin{cases} \tilde{\mathfrak{S}} & \text{if } (T, o) \in \mathcal{V}, \mathfrak{S} \xrightarrow{o} \tilde{\mathfrak{S}} \in Isp(\mathcal{O}) \\ \mathfrak{S} & \text{otherwise.} \end{cases}$

The actual algorithm that builds $Bhv(\wp(\mathcal{H}))$ starts from the initial state S_0 and generates all possible transitions based on the conditions above. Eventually, it removes all spurious states and transitions that are not in a path from the initial state to a final state.

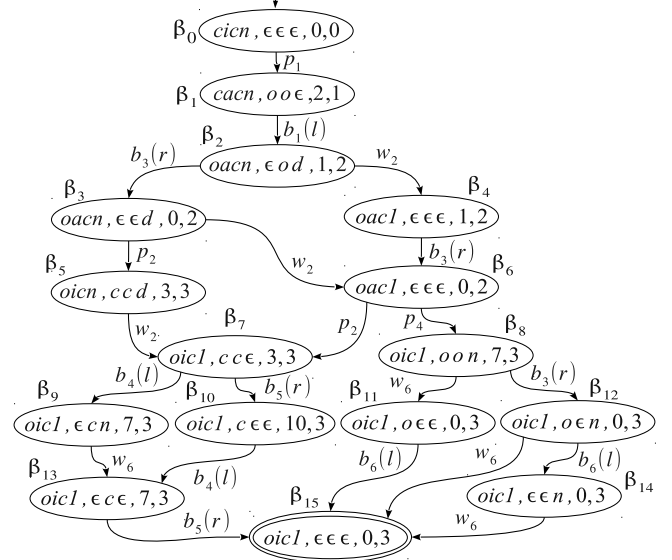


Figure 4: Behavior $Bhv(\wp(\mathcal{W}))$.

Example 4. With reference to $\wp(\mathcal{W})$ in Example 3, shown in Fig. 4 is $Bhv(\wp(\mathcal{W}))$, including states $\beta_0, \dots, \beta_{15}$, with β_{15} final. Each state $(\mathcal{S}, \mathcal{E}, \mathcal{P}, \mathfrak{S})$ is such that \mathcal{S} is the quadruple of states for l, p, r , and \mathcal{W} , where *closed*, *open*, *idle*, *awaken*, *con*, and *disl* are written *c*, *o*, *i*, *a*, *n*, and *l*, respectively, \mathcal{E} is the triple of events at input terminals of l, r , and \mathcal{W} , respectively, where *op*, *cl*, *di*, and *nc* are written *o*, *c*, *d*, and *n*, respectively, while \mathcal{P} and \mathfrak{S} are the indices of states in $Pts(w)$ and $Isp(\mathcal{O})$, respectively. For instance, $\beta_2 = (oacn, \epsilon od, 1, 2)$ stands for $\mathcal{S} = (open, awaken, closed, con)$, $\mathcal{E} = (\epsilon, op, di)$, $\mathcal{P} = \mathcal{P}_1$, and $\mathfrak{S} = \mathfrak{S}_2$. \diamond

Once the behavior has been constructed, each state S of $Bhv(\wp(\mathcal{H}))$ is decorated by the minimal set of candidate diagnoses $\Delta(S)$ fulfilling the following two inductive rules:

- (1) For the initial state: $\Delta(S_0) = \{\emptyset\}$.
- (2) For each transition $S \xrightarrow{\mathcal{T}} S'$ in $Bhv(\wp(\mathcal{H}))$:
If T is normal then $\delta \in \Delta(S) \Rightarrow \delta \in \Delta(S')$;
If $(T, f) \in \mathcal{R}$ then $\delta \in \Delta(S) \Rightarrow (\delta \cup \{f\}) \in \Delta(S')$.

The algorithm that decorates $Bhv(\wp(\mathcal{H}))$ starts by applying the first rule, marking the initial state with the singleton of the empty diagnosis. Then, based on the decoration of the initial state, it continuously applies the second rule for each transition exiting a state S whose decoration has changed. If $(T, f) \in \mathcal{R}$ then T is faulty, with f being the relevant fault. If so, each candidate diagnosis in $\Delta(S)$ extended by fault f is also a candidate diagnosis in $\Delta(S')$. Instead, if T is normal, all candidate diagnoses in $\Delta(S)$ are candidate diagnoses in $\Delta(S')$ too. As such, $\Delta(S)$ is constructed as the set of diagnoses relevant to histories ending at state S . The algorithm terminates when the application of the second rule does not cause any change in any decoration.

Table 4: Decoration of $Bhv(\wp(\mathcal{W}))$.

States	Decoration
$\beta_0, \beta_1, \beta_2, \beta_4$	$\{\emptyset\}$
$\beta_3, \beta_5, \beta_6, \beta_7, \beta_{10}$	$\{\{nor\}\}$
β_9	$\{\{nor, ncl\}\}$
β_{13}	$\{\{nor, ncl, fcw\}\}$
$\beta_8, \beta_{12}, \beta_{14}$	$\{\{nor, fcp\}\}$
β_{11}	$\{\{nor, fcp, fcw\}\}$
β_{15}	$\{\{nor, ncl, fcw\}, \{nor, fcp, fcw\}\}$

Example 5. Based on ruler \mathcal{R} (Example 3), the behavior in Fig. 4 will be decorated as specified in Table 4. Therefore, two candidate diagnoses are associated with final state β_{15} , namely $\delta_1 = \{nor, ncl, fcw\}$, and $\delta_2 = \{nor, fcp, fcw\}$, corresponding to these two scenarios:

δ_1 : Breaker r fails to open, breaker l fails to close, and protection hardware \mathcal{W} fails to connect;

δ_2 : Breaker r fails to open, protection device trips breakers to open rather than to close, and \mathcal{W} fails to connect.

Based on Theorem 1, $\{\delta_1, \delta_2\}$ is the solution of $\wp(\mathcal{W})$. Albeit we have two candidates, since $\delta_1 \cap \delta_2 = \{nor, fcw\}$, certainly r failed to open and \mathcal{W} failed to connect. \diamond

Theorem 1. Let $\wp(\mathcal{H}) = (\mathcal{H}_0, \mathcal{V}, \mathcal{O}, \mathcal{R})$. Let \mathcal{B}^s and \mathcal{B}^v denote $Bsp(\mathcal{H}, \mathcal{H}_0)$ and $Bhv(\wp(\mathcal{H}))$, respectively. Let $\Delta(\mathcal{B}^v)$ denote the union of the sets of diagnoses decorating the final states of \mathcal{B}^v . Then, $\Delta(\mathcal{B}^s) = \Delta(\mathcal{B}^v)$.

Proof. (Sketch) Grounded on Lemmas 1.1–1.5.

Lemma 1.1. If history $h \in \mathcal{B}^v$ then $h \in \mathcal{B}^s$.

This derives from the fact \mathcal{B}^v differs from \mathcal{B}^s in field \mathfrak{S} , which is irrelevant for conditions on \mathcal{S}' , \mathcal{E}' , and \mathcal{P}' . By induction on h , starting from the initial state, each new transition applicable in \mathcal{B}^v is applicable in \mathcal{B}^s too.

Lemma 1.2. If history $h \in \mathcal{B}^v$ then $h_{[\mathcal{V}]} \in \|\mathcal{O}\|$.

Recall that $h_{[\mathcal{V}]}$ is the sequence of observable labels associated with visible transitions in viewer \mathcal{V} . Based on the definition of \mathcal{B}^v , $h_{[\mathcal{V}]}$ belongs to the language of $Isp(\mathcal{O})$, which equals $\|\mathcal{O}\|$. Thus, $h_{[\mathcal{V}]} \in \|\mathcal{O}\|$.

Lemma 1.3. If history $h \in \mathcal{B}^s$, $h_{[\mathcal{V}]} \in \|\mathcal{O}\|$, then $h \in \mathcal{B}^v$.

By induction on h , starting from the initial state, each new transition T applicable in \mathcal{B}^s is applicable in \mathcal{B}^v too. In fact, if T is invisible, no further condition is required. If T is visible, based on the assumption $h_{[\mathcal{V}]} \in \|\mathcal{O}\|$ and on the fact that the language of $Isp(\mathcal{O})$ equals $\|\mathcal{O}\|$, the label associated with T in viewer \mathcal{V} matches a transition in $Isp(\mathcal{O})$.

Lemma 1.4. If history $h \in \mathcal{B}^v$ ends at final state S_f then $\Delta(S_f)$ includes a candidate diagnosis $h_{[\mathcal{R}]}$.

Based on the two rules for decoration of \mathcal{B}^v , by induction on h , starting from the initial state (h empty) and the empty diagnosis δ , the addition of a new transition T in h extends δ (within the decoration of the new state) by either nothing (T normal) or a fault label (T faulty). Upon the last transition of h , δ includes all fault labels associated with faulty transition in ruler \mathcal{R} , in other words $\delta = h_{[\mathcal{R}]}$.

Lemma 1.5. If S_f is a final state in \mathcal{B}^v and $\delta \in \Delta(S_f)$ then \exists history $h \in \mathcal{B}^v$ ending at S_f such that $h_{[\mathcal{R}]} = \delta$.

Based on the decoration rules for \mathcal{B}^v , δ is incrementally generated starting from the empty diagnosis initially associated with S_0 , by inserting each faulty label associated with each faulty transition encountered in a path from S_0 to S_f . This path is a history provided that it is finite. In fact, cycles

in \mathcal{B}^v allow for an infinite number of applications of the second decoration rule. However, since δ is a set, once a cycle has been covered, all associated fault labels are inserted into δ . Successive iterations of the cycle do not extend δ because of duplicate removals. Thus, δ can always be generated by a finite history h , in other words, $\delta = h_{[\mathcal{R}]}$.

To prove Theorem 1, we show $\delta \in \Delta(\mathcal{B}^v) \Leftrightarrow \delta \in \Delta(\mathcal{B}^s)$. On the one hand, if $\delta \in \Delta(\mathcal{B}^v)$ then, based on Lemmas 1.1, 1.2, and 1.5, there exists a history $h \in \mathcal{B}^s$ such that $h_{[\mathcal{V}]} \in \|\mathcal{O}\|$ and $h_{[\mathcal{R}]} = \delta$, that is, $\delta \in \Delta(\mathcal{B}^s)$. On the other, if $\delta \in \Delta(\mathcal{B}^s)$ then, based on Lemmas 1.3 and 1.4, there exists a history $h \in \mathcal{B}^v$ ending at final state S_f such that $\delta = h_{[\mathcal{R}]}$ and $\delta \in \Delta(S_f)$, that is, $\delta \in \Delta(\mathcal{B}^v)$. \square

5 Discussion

HDESs are a means of modeling complex DESs, where behavior is stratified and events can be generated by patterns of transitions. In spite of being influenced by other components, each internal node X of the hierarchy is a complex component living its own life. This means that X has its own behavioral model, which does not coincide with the composition of the behavioral models of its components. This results in a hierarchical system (HDES) made up of several cohabiting subsystems accommodated at different abstraction levels. The diagnosis technique defined for HDESs is model-based in nature: diagnosis is output based on the model of the system and the temporal observation. Only the portion of the behavior space consistent with the observation is reconstructed and eventually decorated by candidate diagnoses. Not only does separation of concerns apply to the modeling, it also applies to the diagnosis task. Since each (complex) component is provided with its own behavioral model, diagnosis is context-sensitive [Lamperti and Zanella, 2011a]. Moreover, depending on the degree of constraints on computational resources and time response of the diagnosis engine, model-based reasoning can be scaled to a convenient level of abstraction. This means restricting the HDES \mathcal{H} to a portion \mathcal{H}' (for instance, a complex component along with its children) and projecting the temporal observation \mathcal{O} on \mathcal{O}' , resulting from the removal of irrelevant labels, nodes, and arcs. This way, within the context of \mathcal{H}' , the diagnosis output is complete, even if not sound (due to the removal of the behavioral constraints imposed by $\mathcal{H} - \mathcal{H}'$ and the observation constraint imposed by $\mathcal{O} - \mathcal{O}'$). To refine diagnosis, both \mathcal{H}' and \mathcal{O}' may be then enlarged to a suitable extent.

6 Related Work

This paper substantially extends the idea of context-sensitive diagnosis [Lamperti and Zanella, 2011a] in three directions. First, in [Lamperti and Zanella, 2011a] pattern stratification is only apparent, as, after macro-substitution, the regular expression is invariably defined on (basic) component transitions. In this paper, pattern stratification is real, since regular expressions are defined on the transitions of possibly complex components. Second, in [Lamperti and Zanella, 2011a] faults are associated with pattern matching. In this paper, faults are associated with transitions of components: pattern matching generates pattern events and, by union, complex events, to which complex components are sensitive. More importantly, in [Lamperti and Zanella, 2011a] context-sensitivity is defined on active systems, while this paper deals with HDESs, which provide behavioral stratification: separation of concerns holds not only for diagnosis but also for behavior. This paper also differs from [Jéron et al., 2006], where the notion of supervision pattern is introduced, mainly because neither a hierarchical

structure for the system is conceived nor behavioral stratification is applicable. HDEs are not HFSMs (*Hierarchical Finite State Machines*). The notion of HFSM was inspired by statecharts [Harel, 1987], a visual formalism for complex systems. The most important feature of a HFSM is hierarchical state-nesting: if a system is in a nested state (sub-state), it is also in all its surrounding states (superstates). Moreover, transitions are defined at each level of the hierarchy. A simplified version of statechart, namely HFSM, was considered for solving a class of control problems in [Brave and Heymann, 1993]. Recently, diagnosis of HFSMs has been considered in [Idghamishi and Zad, 2004; Paoli and Lafortune, 2008]. However, no patterns are involved, events are simple, and diagnosis is context-free.

7 Conclusion

HDEs are a means to formalize complex DESs with behavior stratification. This allows for the modeling of a hierarchy wherein different, yet integrated, subsystems coexist, each one living its own life. This also allows for context-sensitivity and scalability of diagnosis.

Acknowledgment

This work was supported in part by NSFC under Grant No. 61003101, and Zhejiang Provincial Natural Science Foundation under Grant No. Y1100191.

References

- [Baroni *et al.*, 1999] P. Baroni, G. Lamperti, P. Pogliano, and M. Zanella. Diagnosis of large active systems. *Artificial Intelligence*, 110(1):135–183, 1999.
- [Brand and Zafiropulo, 1983] D. Brand and P. Zafiropulo. On communicating finite-state machines. *Journal of ACM*, 30(2):323–342, 1983.
- [Brave and Heymann, 1993] H. Brave and M. Heymann. Control of discrete event systems modeled as hierarchical state machines. *IEEE Transactions on Automatic Control*, 38(12):1803–1819, 1993.
- [Cassandras and Lafortune, 1999] C.G. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems*, volume 11 of *The Kluwer International Series in Discrete Event Dynamic Systems*. Kluwer Academic Publishers, Boston, MA, 1999.
- [Debouk *et al.*, 2000] R. Debouk, S. Lafortune, and D. Teneketzis. Coordinated decentralized protocols for failure diagnosis of discrete-event systems. *Journal of Discrete Event Dynamic Systems: Theory and Applications*, 10:33–86, 2000.
- [Debouk *et al.*, 2003] R. Debouk, S. Lafortune, and D. Teneketzis. On the effect of communication delays in failure diagnosis of decentralized discrete event systems. *Journal of Discrete Event Dynamic Systems: Theory and Applications*, 13:263–289, 2003.
- [Grastien *et al.*, 2004] A. Grastien, M.O. Cordier, and C. Largouët. Extending decentralized discrete-event modelling to diagnose reconfigurable systems. In *Fifteenth International Workshop on Principles of Diagnosis –DX’04*, pages 75–80, Carcassonne, F, 2004.
- [Grastien *et al.*, 2005] A. Grastien, M.O. Cordier, and C. Largouët. Incremental diagnosis of discrete-event systems. In *Sixteenth International Workshop on Principles of Diagnosis DX’05*, pages 119–124, Monterey, CA, 2005.
- [Harel, 1987] D. Harel. Statecharts: a visual formalism for complex systems. *Science of Computer Programming*, 8:231–274, 1987.
- [Idghamishi and Zad, 2004] A.M. Idghamishi and S.H. Zad. Fault diagnosis in hierarchical discrete-event systems. In *43rd IEEE Conference on Decision and Control*, pages 63–68, Paradise Island, BSH, 2004.
- [Jéron *et al.*, 2006] T. Jérón, H. Marchand, S. Pinchinat, and M.O. Cordier. Supervision patterns in discrete event systems diagnosis. In *Seventeenth International Workshop on Principles of Diagnosis DX’06*, pages 117–124, Peñaranda de Duero, E, 2006.
- [Kwong and Yonge-Mallo, 2011] R.H. Kwong and D.L. Yonge-Mallo. Fault diagnosis in discrete-event systems: incomplete models and learning. *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics*, 41(1):118–130, 2011.
- [Lamperti and Zanella, 2002] G. Lamperti and M. Zanella. Diagnosis of discrete-event systems from uncertain temporal observations. *Artificial Intelligence*, 137(1–2):91–163, 2002.
- [Lamperti and Zanella, 2011a] G. Lamperti and M. Zanella. Context-sensitive diagnosis of discrete-event systems. In T. Walsh, editor, *Twenty-Second International Joint Conference on Artificial Intelligence IJCAI’11*, volume 2, pages 969–975, Barcelona, E, 2011. AAAI Press.
- [Lamperti and Zanella, 2011b] G. Lamperti and M. Zanella. Monitoring of active systems with stratified uncertain observations. *IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans*, 41(2):356–369, 2011.
- [Paoli and Lafortune, 2008] A. Paoli and S. Lafortune. Diagnosability analysis of a class of hierarchical state machines. *Journal of Discrete Event Dynamic Systems: Theory and Applications*, 18(3):385–413, 2008.
- [Pencolé and Cordier, 2005] Y. Pencolé and M.O. Cordier. A formal framework for the decentralized diagnosis of large scale discrete event systems and its application to telecommunication networks. *Artificial Intelligence*, 164:121–170, 2005.
- [Pencolé *et al.*, 2001] Y. Pencolé, M.O. Cordier, and L. Rozé. Incremental decentralized diagnosis approach for the supervision of a telecommunication network. In *Twelfth International Workshop on Principles of Diagnosis DX’01*, pages 151–158, San Sicario, I, 2001.
- [Pencolé, 2000] Y. Pencolé. Decentralized diagnoser approach: application to telecommunication networks. In *Eleventh International Workshop on Principles of Diagnosis DX’00*, pages 185–192, Morelia, MX, 2000.
- [Qiu and Kumar, 2006] W. Qiu and R. Kumar. Decentralized failure diagnosis of discrete event systems. *IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans*, 36(2):384–395, 2006.
- [Sampath *et al.*, 1996] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D.C. Teneketzis. Failure diagnosis using discrete-event models. *IEEE Transactions on Control Systems Technology*, 4(2):105–124, 1996.
- [Zhao and Ouyang, 2008] X. Zhao and D. Ouyang. Model-based diagnosis of discrete event systems with an incomplete system model. In *Eigteenth European Conference on Artificial Intelligence ECAI’2008*, pages 189–193. IOS Press, Amsterdam, NL, 2008.
- [Zhao *et al.*, 2012] X. Zhao, D. Ouyang, L. Zhang, X. Wang, and Y. Mo. Reasoning on partially-ordered observations in online diagnosis of DESs. *AI Communications*, 25(4):285–294, 2012.