Fault Diagnosis in Discrete-Event Systems: Incomplete Models and Learning

David L. Yeung
School of Computer Science
University of Waterloo
Waterloo, Ontario, Canada N2L 3G1
Email: dlyeung@uwaterloo.ca

Raymond H. Kwong
Electrical and Computer Engineering
University of Toronto
Toronto, Ontario, Canada M5S 3G4
Email: kwong@control.toronto.edu

Abstract— **Most state-based approaches to fault diagnosis of discrete-event systems require a complete and accurate model of the system to be diagnosed. In this paper, we address the problem of diagnosing faults given an incomplete model of the system. We introduce the learning diagnoser, which estimates the fault condition of the system and attempts to learn the missing information in the model using discrepancies between the actual and expected output of the system. We view the process of generating and evaluating hypotheses about the state of the system as an instance of the set covering problem, which we formalize by using parsimonious covering theory. We also explain through an example the steps in the construction of the learning diagnoser.**

I. Introduction

Fault detection is an important part of many engineering systems. A large variety of model-based methods for automated fault diagnosis have been developed [1], [2]. A major disadvantage of any model-based approach is that it is sometimes very difficult to create an accurate model of an electro-mechanical device or physical system (for example, see the various papers collected in [2]).

Recently, a state-based framework for fault diagnosis in discrete-event systems (DES) has been presented in [3]. In this framework, the diagnoser does not have to be initialized at the same time as the system, and no assumptions are made at the initiation of the diagnoser about the state of the system to be diagnosed or its failure status. However, construction of the diagnoser again requires accurate model of the DES.

In this paper, we describe a fault diagnosis framework for DES which may be used to diagnose faults even when a complete model of the system is unavailable. We introduce the notion of learning in fault diagnosis of DES and describe how a learning procedure can be combined with fault diagnosis. Machine learning in finite state machines has been extensively studied in the literature. The problem of inferring a finite state automaton from its input-output behaviour is discussed in [4], which also gives procedures for carrying out the inference. The complexity as well as the possibility of learning finite automata are discussed in [5], [6]. In addition, computational learning methods for finite automata, such as the use of genetic algorithms, have been applied to natural language processing applications [7]. However, the techniques described in much of this literature require the use of inputs for learning. For fault diagnosis in control systems, the use of test inputs is often infeasible. Our approach uses only the outputs for both learning through hypothesis generation and fault diagnosis through estimation of the system fault status.

The paper is organized as follows. In Section II, we review some notations and definitions. In Section III, we examine the fault diagnosis process when the model of the system is incomplete, leading to the development of the learning diagnoser in Section IV. We conclude in Section V with a discussion of more theoretical aspects of the problem.

II. Basic Notations and Definitions

We review some basic notations and definitions for the state-based approach to fault diagnosis of discrete-event systems (DES). Further details may be found in [3], [8], [9].

The system to be diagnosed (the plant along with continuous controllers and DES supervisors) is modeled as a non-deterministic finite-state Moore automaton (FSMA). This is represented as a 6-tuple structure

$$\mathbf{G} = (X, \Sigma, \delta, x_0, Y, \lambda)$$

where X is the finite *state set*, Σ is the finite (non-empty) *event set*, Y is the finite *output set*, x_0 is the *initial state*, $\delta : X \times \Sigma \to 2^X$ is the (partial) *transition function*, and $\lambda : X \to Y$ is the *output map*.

The event set Σ includes the set of *failure events* Σ_f. In addition to the normal mode N, there are p failure modes $F_1, \ldots, F_p$ that describe the operation of the system. We denote the *condition set* of the system by $\mathcal{K} := \{N, F_1, \ldots, F_p\}$, and we assume that it is possible to partition the state set X according to the condition of the system: $X = X_N \,\dot\cup\, X_{F_1} \,\dot\cup\, \ldots \,\dot\cup\, X_{F_p}$. (The symbol $\dot\cup$ denotes disjoint union.) The occurrence of a failure event brings the system into one of the sets X_{F_i}, corresponding to the failure mode F_i.

The *condition map*, $\kappa : X \to \mathcal{K}$, is defined as follows: for every $x \in X$, $\kappa(x) = N$ if $x \in X_N$ and $\kappa(x) = F_i$ if $x \in X_{F_i}$ (for some $i \in \{1, \ldots, p\}$). We extend (abusing notation) the definition of κ to subsets of X:

$$\forall z \subseteq X,\ \kappa(z) = \bigcup_{x \in z} \{\kappa(x)\}$$

Only changes in the output are assumed to be observable, so that in the output sequence, we have $y_k \neq y_{k+1}$ for $k \geq 1$.

For simplicity, we consider only the case $p = 1$ in this paper, and denote the single failure mode by F. Furthermore, faults are assumed to be permanent. The extension to $p > 1$ is conceptually the same but notationally more complex.

We use a running example to illustrate the ideas presented in this paper.

Example 1 Figure 1 shows a DES model of a plant under control. The dashed line between states 11 and 14 represents a failure event. In this example,

$$\begin{aligned} X &= \{1, 2, \ldots, 16\}, \\ Y &= \{\lambda, \alpha, \beta, \gamma, \mu, \zeta, \eta, \xi\}, \\ \mathcal{K} &= \{N, F\}, \text{ and} \\ \kappa(i) &= \begin{cases} N & 1 \leq i \leq 13 \\ F & 14 \leq i \leq 16. \end{cases} \end{aligned}$$

□

For any two states $x, x' \in X$, we say that x' is *output-adjacent* to x and write $x \Rightarrow x'$ if $\lambda(x) \neq \lambda(x')$ and $\exists l \geq 1$, states $x_1, \ldots, x_{l-1}$, and events $\sigma_1, \ldots, \sigma_l$ such that $x_{i+1} \in \delta(x_i, \sigma_{i+1})$ and $\lambda(x_i) = \lambda(x)$ for all $0 \leq i \leq l-1$, with $x_0 = x$ and $x_l = x'$.

A *diagnoser* $\mathbf{D}$ is a finite-state machine which takes the output sequence of the system $(y_1 y_2 \ldots y_k)$ as its input, and based on this sequence calculates a set $z_k \in 2^X - \{\emptyset\}$ to which x must belong at the time that y_k was generated. The estimate of the system's condition will be $\kappa(z_k)$. Formally,

$$\mathbf{D} = (Z \cup \{\underline{z}_0\}, Y, \zeta, \underline{z}_0, \hat{\mathcal{K}}, \kappa)$$

where $Z \cup \{\underline{z}_0\}$ is the state set, Y is the event set, $\zeta : Z \cup \{\underline{z}_0\} \times Y \rightarrow Z$ is the (partial) transition function, $\underline{z}_0 := (z_0, 0)$ is the initial state with $z_0 \in 2^X - \{\emptyset\}$, $\hat{\mathcal{K}} \subseteq 2^{\mathcal{K}} - \{\emptyset\}$ is the output set, and $\kappa : Z \cup \{\underline{z}_0\} \rightarrow \hat{\mathcal{K}}$ is the output map. We define $\Psi(z) = \{x \mid \exists x' \in z : x' \Rightarrow x\}$. The diagnoser state transition $z_{k+1} = \zeta(z_k, y_{k+1})$ is given by $z_1 = z_0 \cap \lambda^{-1}(\{y_1\})$, and for $k \geq 1$,

$$z_{k+1} = \Psi(z_k) \cap \lambda^{-1}(\{y_{k+1}\})$$

Example 2 Figure 2 shows the diagnoser for the DES model of Figure 1. Each state of the diagnoser, shown as a rounded box, is a set of states of the system. An output symbol and a failure condition is associated with each diagnoser state, except for the initial state.

Suppose that, subsequent to the initialization of the diagnoser, the output sequence $(\mu\zeta\eta)$ is observed. If the next output symbol is anything other than λ or ξ, then it is inconsistent with the model and the diagnoser cannot proceed. □

In the next section, we describe how to extend the diagnoser so that it can continue to perform fault diagnosis even with an incomplete model.

III. Diagnosis using an Incomplete Model

It is assumed in the diagnosis framework described above that a complete and accurate DES model of the system is available for the purpose of diagnosis. If the system develops an unforeseen failure mode, or if the properties of the system are not sufficiently known to provide an accurate model, then diagnosis could not be guaranteed in general using the diagnoser described above. In this section, we study what happens if the model is incomplete.

We assume that it is possible to accurately represent the system to be diagnosed as a DES model, and designate this model the *true model* of the system, $\mathbf{G_t}$. We call the model that we start with the *nominal model* of the system, $\mathbf{G_n}$. The nominal model is typically obtained from a simplification of the system, and may be incomplete in the sense that it may be missing a number of transitions which are found in the true model. If the true model of the system is $\mathbf{G_t} = (X, \Sigma_t, \delta_t, x_0, Y, \lambda)$, and the nominal model is $\mathbf{G_n} = (X, \Sigma_n, \delta_n, x_0, Y, \lambda)$, we assume that $\Sigma_n \subseteq \Sigma_t$ and $\delta_n \subseteq \delta_t$. We also assume that the number of missing transitions is $\leq N_{max}$, a fixed constant. In addition to performing fault diagnosis, we would like to learn the true model of the system. While our incompleteness model is somewhat specific, we believe it represents the first time that learning has been incorporated into fault diagnosis of discrete-event systems.

The upper bound to the number of missing transitions, N_{max}, is assumed in practice to be a small number. While the theoretical development only requires N_{max} to be finite, the computational effort increases rapidly with N_{max}. Assuming N_{max} to be small corresponds to assuming the nominal model is reasonably accurate. As model-based diagnosis is unlikely to be successful if we use a bad model to begin with, this assumption appears reasonable. A more detailed discussion on the computational complexity associated with learning can be found in [9].

We call the diagnoser $\mathbf{D_n}$ created from the nominal model the *nominal diagnoser*. Suppose that we have the nominal model $\mathbf{G_n}$ and the output sequence $(y_1 y_2 \ldots)$. Then $\mathbf{G_n}$ is *consistent* with the output sequence if $\Psi(z_k) \cap \lambda^{-1}(\{y_{k+1}\}) \neq \emptyset$ for all $k \geq 1$, where z_k is the state estimate of the $\mathbf{D_n}$. It is *inconsistent* otherwise. If $\mathbf{G_n}$ is inconsistent with the output sequence, then $z_{k+1} = \emptyset$ for some k, causing the diagnoser to fail.

Given an output sequence $(y_1 y_2 \ldots)$ and a nominal model $\mathbf{G_n}$ which may be inconsistent with the output sequence, we would like to accomplish the following goals: (1) to give a fault estimate for the system, and (2) to *revise* the nominal model by adding to it *hypothetical transitions* which we believe to be in the true model but are missing from the nominal model. A hypothetical transition $d = (x_{src}, x_{dst})$ is an ordered pair of states denoting a transition from the state $x_{src} \in X$ to the state $x_{dst} \in X$ which is not found in the transition graph of $\mathbf{G_n}$. For convenience, we will denote a hypothetical transition $d = (x_{src}, x_{dst})$ by d^{src}_{dst}.

A model $\mathbf{G}$ may be *extended* by a transition or a set of transitions by modifying the transition function δ. Given an output sequence, a *hypothesis* H is a non-empty set of hypothetical transitions, such that the extension of the

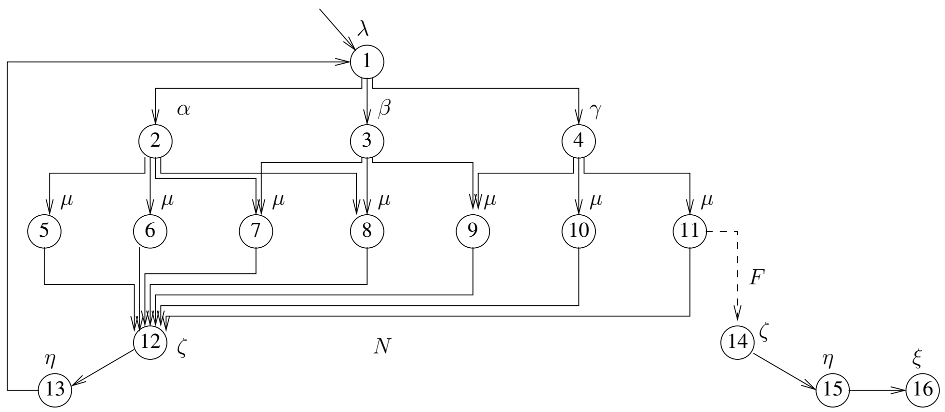

Fig. 1. A DES model of a plant under control.

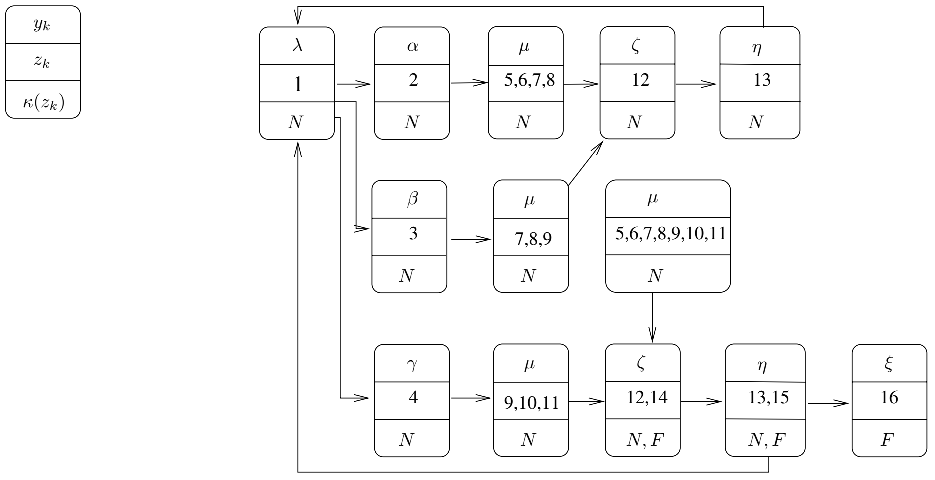

Fig. 2. The diagnoser for the DES model shown in Figure 1.

nominal model $\mathbf{G_n}$ by H is consistent with that output sequence. We call the hypothesis comprising transitions in $\mathbf{G_t}$ missing from $\mathbf{G_n}$ the *correct hypothesis*.

We motivate the main ideas behind hypothesis generation and evaluation with an example.

Example 3 Suppose that the two transitions d^5_{13} and d^9_{13} are missing from the nominal model $\mathbf{G_n}$ shown in Figure 1. We consider the output sequence $(\lambda\alpha\mu\eta\lambda\beta\mu\eta\lambda\gamma\mu\eta\lambda)$, which is possible under the true model but which is impossible according to the nominal model.

After the occurrence of $\lambda\alpha\mu$, the estimate of the nominal diagnoser is $z = \{5, 6, 7, 8\}$, which has a condition estimate of $\{N\}$. The next output symbol, η, is inconsistent with the nominal model since none of the states in $\{5, 6, 7, 8\}$ have a transition going to a state with the output η. The only states with the output η are 13 and 15. Thus, we guess that the true model contains (at least) one of the following transitions: $\{d^5_{13}, d^6_{13}, d^7_{13}, d^8_{13}, d^5_{15}, d^6_{15}, d^7_{15}, d^8_{15}\}$. We call this set of hypothetical transitions a *competing set*, since only one of them is necessary to explain the inconsistency. Furthermore, state 13 has the condition N while state 15 has the condition F, so that at this point, we cannot be sure whether the fault F has occurred.

The next output symbol is λ. The only state with this output is 1. We can infer that the previous state (with output η) was 13, because state 13 has a transition to state 1. It is impossible for the system to go from state 15 to state 1, since $\kappa(15) = F$ and $\kappa(1) = N$, and faults are assumed to be permanent. Thus, we can narrow the hypothetical transitions under consideration to $\{d^5_{13}, d^6_{13}, d^7_{13}, d^8_{13}\}$. We can also deduce that the condition of the system at this point is N, since we know that the system is in state 1.

By the above reasoning, following the output of a further $\beta\mu\eta\lambda\gamma\mu\eta\lambda$, we know that the missing transitions consist of (at least) one transition from $\{d^5_{13}, d^6_{13}, d^7_{13}, d^8_{13}\}$, (at least) one from from $\{d^7_{13}, d^8_{13}, d^9_{13}\}$, and (at least) one from $\{d^9_{13}, d^{10}_{13}, d^{11}_{13}\}$. When the nominal model is extended by a hypothesis of the form $H = \{d_1, d_2, d_3\}$ where $d_1 \in \{d^5_{13}, d^6_{13}, d^7_{13}, d^8_{13}\}$, $d_2 \in \{d^7_{13}, d^8_{13}, d^9_{13}\}$, and $d_3 \in \{d^9_{13}, d^{10}_{13}, d^{11}_{13}\}$, the resultant model will be consistent with the output sequence. □

In order to capture the relationships between hypothe-

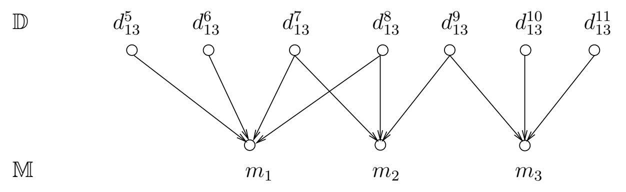

Fig. 3. Causal network for Example 3.

ses and their component transitions, we view the process of composing and evaluating hypotheses as a *set cover* problem, using the parsimonious covering theory (PCT) presented in [10].

Let $\mathbb{D}$ denote the set of all hypothetical transitions d_i of the nominal model, and let $\mathbb{M}$ denote the set of all competing sets containing the hypothetical transitions, i.e. for each $d_i \in \mathbb{D}$, $\exists m_j \in \mathbb{M}$ s.t. $d_i \in m_j$. We create a *causal network* from these two discrete finite sets. Figure 3 shows the causal network after the complete output sequence of Example 3. Letting $m_1 = \{d^5_{13}, d^6_{13}, d^7_{13}, d^8_{13}\}$, $m_2 = \{d^7_{13}, d^8_{13}, d^9_{13}\}$, and $m_3 = \{d^9_{13}, d^{10}_{13}, d^{11}_{13}\}$, the bipartite graph shown in the figure represents the relationship between the three competing sets. It may be seen that each cover of this bipartite graph is a hypothesis, such that the extension of $\mathbf{G_n}$ by the hypothesis is consistent with the output sequence.

Let $g_1, g_2, \ldots, g_n$ be non-empty pairwise-disjoint subsets of $\mathbb{D}$. Then $\mathbb{G}_I = \{g_1, g_2, \ldots, g_n\}$ is a *generator*. The *class generated by* $\mathbb{G}_I$, designated as $[\mathbb{G}_I]$, is defined to be $[\mathbb{G}_I] = \{\{d_1, d_2, \ldots, d_n\} \mid d_i \in g_i, 1 \le i \le n\}$. Notationally, we write $\mathbb{G}_I = (g_1, g_2, \ldots, g_n)$ (with rounded brackets) for readability. Let $\mathbb{G}_1, \mathbb{G}_2, \ldots, \mathbb{G}_N$ be generators and $[\mathbb{G}_I] \cap [\mathbb{G}_J] = \emptyset$ for $I \neq J$. Then $\mathcal{G} = \{\mathbb{G}_1, \mathbb{G}_2, \ldots, \mathbb{G}_N\}$ is a *generator-set*, and the *class generated by* $\mathcal{G}$ is $[\mathcal{G}] = \sum_{I=1}^{N} [\mathbb{G}_I]$.

We call the number of transitions in a hypothesis its *cardinality*. Not all hypotheses are equally preferable, since they have different cardinalities and some contain unnecessary transitions. A hypothesis is said to be a *minimum* cover of $\mathbb{M}$ if its cardinality is smallest among all covers of $\mathbb{M}$. It is said to be an *irredundant* cover if none of its proper subsets is also a cover of $\mathbb{M}$, and a *redundant* cover otherwise. Since the correct hypothesis may be a non-minimum cover, it is insufficient merely to find all the minimum covers. One can always obtain the set of all minimum covers from the set of all irredundant covers by simply picking out the covers of minimal cardinality. Note also that each redundant cover has a subset which is irredundant, and that both a redundant cover and its irredundant subset are hypotheses. While the correct hypothesis may indeed be a redundant cover, the redundant cover cannot be distinguished from its irredundant subset by the output sequence. The principle of parsimony says that, all other things being equal, the simpler hypothesis is preferable. Hence we need only focus on irredundant covers when we generate hypotheses. The PCT framework provides effective algorithms to find all irredundant covers for a set covering problem, thereby providing the tools for efficient generation and evaluation of hypotheses.

Example 4 In the causal network of Figure 3, there are eight hypotheses for the bi-partite graph. These may be compactly represented by the generator-set $\mathcal{G} = \{(\{d^7_{13}, d^8_{13}\}\{d^9_{13}, d^{10}_{13}, d^{11}_{13}\})(\{d^5_{13}, d^6_{13}\}\{d^9_{13}\})\}$. The class generated by $\mathcal{G}$ is $[\mathcal{G}] = \{\{d^7_{13}, d^9_{13}\}, \{d^7_{13}, d^{10}_{13}\}, \{d^7_{13}, d^{11}_{13}\}, \{d^8_{13}, d^9_{13}\}, \{d^8_{13}, d^{10}_{13}\}, \{d^8_{13}, d^{11}_{13}\}, \{d^5_{13}, d^9_{13}\}, \{d^6_{13}, d^9_{13}\}\}$. □

Generator-sets provide a compact way to represent multiple hypotheses so that their relationships to each other are evident. Furthermore, by defining certain operations on generator-sets, we can revise the hypotheses based on the observed output of the system.

IV. The Learning Diagnoser

In this section, we present a state-based fault diagnosis framework which is tolerant of missing information about the system to be diagnosed.

A *learning diagnoser* is a system that detects and isolates failures, and furthermore, attempts to learn the true model of the system if the nominal model is incomplete. It is a finite-state machine which takes the output sequence of the system $(y_1 y_2 \ldots y_k)$ as its input, and generates as its output hypotheses regarding the information that is missing from the nominal model, and an estimate of the system's condition for each hypothesis.

We define a *candidate* to be a pair $c = (\mathcal{G}, z)$, where z is an estimate of the state of the system, and $\mathcal{G}$ is a generator-set representing a set of hypotheses. A candidate is a state estimate along with information on the "history" of that estimate. Each hypothesis in the class generated by $\mathcal{G}$ is a set of hypothetical transitions which would result in the state estimate z had those transitions actually occurred. All the hypotheses within a candidate are indistinguishable from one another by the currently known output sequence, since they all produce the same state estimate. For a candidate $c = (\mathcal{G}, z)$, we say that $[\mathcal{G}]$ is the set of hypotheses under consideration. A candidate $c = (\mathcal{G}, z)$ is said to be a *match* for a state of the nominal diagnoser $\mathbf{D_n}$ if z is a state of the nominal diagnoser. Let C denote the set of all candidates.

The learning diagnoser is exactly analogous to the diagnoser defined in Section II, by substituting candidates for states of the system. The diagnoser determines a set of states of the system based on the current output, and updates that set of system states based on the subsequent output sequence. The learning diagnoser determines, in addition, sets of hypotheses (each set of which is associated with a state estimate), and these sets of hypotheses are updated according to the subsequent output sequence of the system. In other words, each candidate is a state estimate along with hypotheses which explain how the system arrived at that state.

Suppose that $\mathcal{G}$ is a generator-set, and $\mathbb{H}$ is a set of hypothetical transitions. Then the *revision of* $\mathcal{G}$ *due to* $\mathbb{H}$ is a new generator-set

$$\mathcal{G}' = revise(\mathcal{G}, \mathbb{H})$$

such that each hypothesis generated by $\mathcal{G}'$ contains (at least) one transition from $\mathbb{H}$. (In particular, note that $revise(\emptyset, \mathbb{H}) = \{(\mathbb{H})\}$.) The $revise$ operation involves several steps which we now outline.

Given a generator-set $\mathcal{G}$ and a set of hypothetical transitions $\mathbb{H}$, the $revise$ operation first creates a generator-set $\mathcal{F}$ called the *division* of $\mathcal{G}$ by $\mathbb{H}$ which generates all the hypotheses in $[\mathcal{G}]$ that already contain a transition from $\mathbb{H}$. It also creates a generator-set $\mathcal{F}'$ called the *residual* of the division of $\mathcal{G}$ by $\mathbb{H}$ which generates the hypotheses $[\mathcal{G}]\backslash[\mathcal{F}]$, that is, those hypotheses in $[\mathcal{G}]$ which do not contain a transition in $\mathbb{H}$. Next, it augments each hypothesis in $[\mathcal{F}']$ with each transition from $\mathbb{H}$ to form a new set of hypotheses having a generator-set $\mathcal{Q}$ called the *augmented residual* of $\mathcal{G}$ by $\mathbb{H}$. Finally, $\mathcal{G}' = revise(\mathcal{G}, \mathbb{H})$ is formed by taking the union of $\mathcal{F}$ with $\mathcal{Q}$, with duplicate hypotheses removed. The final result is a generator-set $\mathcal{G}'$ which explains both the old hypotheses as well as the new transitions, and therefore can be taken as the updated hypothesis generator.

In an analogous fashion to the definition of the diagnoser given in Section II, we define the learning diagnoser to be a finite-state machine

$$\mathbf{LD} = (W, Y, \rho, w_0, \hat{\mathcal{K}}, \kappa)$$

where W is the state set, Y is the event set, $\rho : W \cup \{\underline{w}_0\} \times Y \to W$ is the transition function, w_0 is the initial state, $\hat{\mathcal{K}} \subseteq 2^{\mathcal{K}} - \{\emptyset\}$ is the output set, and $\kappa : W \cup \{\underline{w}_0\} \to \hat{\mathcal{K}}$ is the output map.

Each state of the learning diagnoser, except for the initial state $\underline{w}_0$, is identified with a non-empty subset of C, that is, $W \subseteq 2^C - \{\emptyset\}$. If $w = \{(\mathcal{G}_1, z_1), \ldots, (\mathcal{G}_n, z_n)\}$, then the set of hypotheses under consideration in w is $[\mathcal{G}_1 \cup \ldots \cup \mathcal{G}_n]$.

The learning diagnoser state transition function $w_{k+1} = \rho(w_k, y_{k+1})$ is given by $w_1 = \{(\emptyset, z_0 \cap \lambda^{-1}(\{y_1\}))\}$, and for $k \geq 1$, $w_{k+1} = update(w_k, y)$. The interested reader may consult [9] for the definition of the $update$ operation. Conceptually, for a state of the learning diagnoser $w_k = \{c_1, \ldots, c_n\}$, the $update$ operation applies an update law involving the $revise$ operation described above to each of the candidates $c_1, \ldots, c_n$, but with the addition of some housekeeping such as the merging of candidates with the same state estimate.

We illustrate the use of the $revise$ operation and the process of updating candidates with the following example, which gives the learning diagnoser for the running example used throughout this paper. More technical details are given in [9].

Example 5 Figure 4 shows the learning diagnoser for Example 3 after the output sequence $(\lambda\alpha\mu\eta\lambda\beta\mu\eta\lambda\gamma\mu\eta\lambda)$. Transitions of the learning diagnoser where the set of hypotheses under consideration are changed are denoted by a 'z'-like break. For other transitions, the state estimate of the system has changed, but not the hypotheses regarding the missing transitions. Candidates $c = (\mathcal{G}, z)$ which are a match for a state of the nominal diagnoser $\mathbf{D_n}$ are shown in dotted lines.

When the learning diagnoser is initialized (step 0), no output has yet been observed, and the state estimate is X (the system may be in any state). No hypotheses have been generated, which we may represent by $\mathcal{G}_0 = \emptyset$. The known information about the system may be captured using the candidate $c_0 = (X, \mathcal{G}_0)$. After the output of $\lambda\alpha\mu$ (after step 3), the state estimate becomes $z_3 = \{5, 6, 7, 8\}$. Because the output sequence has thus far been consistent with the nominal model, no hypotheses have been generated, so $\mathcal{G}_3 = \emptyset$ and $c_3 = (z_3, \mathcal{G}_3)$.

The next output symbol, η (step 4), is inconsistent with the nominal model, and the only states with the output η are 13 and 15. Thus, either the system is in state 13 and the nominal model is missing one of the transitions $\{d^5_{13}, d^6_{13}, d^7_{13}, d^8_{13}\}$, or the system is in state 15 and the nominal model is missing one of the transitions $\{d^5_{15}, d^6_{15}, d^7_{15}, d^8_{15}\}$. We can capture this information using two candidates $c_{4a} = (\{13\}, \mathcal{G}_{4a})$ and $c_{4b} = (\{15\}, \mathcal{G}_{4b})$, where $\mathcal{G}_{4a} = revise(\mathcal{G}_3, \{d^5_{13}, d^6_{13}, d^7_{13}, d^8_{13}\}) = \{(\{d^5_{13}, d^6_{13}, d^7_{13}, d^8_{13}\})\}$ and $\mathcal{G}_{4b} = revise(\mathcal{G}_3, \{d^5_{15}, d^6_{15}, d^7_{15}, d^8_{15}\}) = \{(\{d^5_{15}, d^6_{15}, d^7_{15}, d^8_{15}\})\}$.

Since the next output symbol is λ (step 5), we know that the system is in state 1. Since faults are assumed to be permanent, it is impossible for the previous state to have been 15. Thus, the candidate c_{4b} leads to a dead end, and the known information about the system after the output of λ may be captured by a single candidate $c_5 = (\{1\}, \mathcal{G}_5)$, where $\mathcal{G}_5 = \mathcal{G}_{4a}$. Similarly, $c_6 = (\{3\}, \mathcal{G}_6)$ (step 6) and $c_7 = (\{7, 8, 9\}, \mathcal{G}_7)$ (step 7), with $\mathcal{G}_5 = \mathcal{G}_6 = \mathcal{G}_7$.

The next output symbol, η (step 8), is again inconsistent with the model. By the same reasoning as above, either the system is in state 13 and the nominal model is missing one of the transitions $\{d^7_{13}, d^8_{13}, d^9_{13}\}$, or the system is in state 15 and the nominal model is missing one of the transitions $\{d^7_{15}, d^8_{15}, d^9_{15}\}$. As above, there are two candidates, $c_{8a} = (\{13\}, \mathcal{G}_{8a})$ and $c_{8b} = (\{15\}, \mathcal{G}_{8b})$, where $\mathcal{G}_{8a} = revise(\mathcal{G}_7, \{d^7_{13}, d^8_{13}, d^9_{13}\}) = \{((\{d^7_{13}, d^8_{13}\})(\{d^5_{13}, d^6_{13}\}\{d^9_{13}\})\}$ and $\mathcal{G}_{8b} = revise(\mathcal{G}_7, \{d^7_{15}, d^8_{15}, d^9_{15}\}) = \{((\{d^5_{13}, d^6_{13}, d^7_{13}, d^8_{13}\}\{d^7_{15}, d^8_{15}, d^9_{15}\})\}$.

Note that while $\mathcal{G}_{8a} = \{((\{d^7_{13}, d^8_{13}\})(\{d^5_{13}, d^6_{13}\}\{d^9_{13}\})\}$ might look quite complicated, it generates precisely those hypotheses which contain one transition from $\{d^5_{13}, d^6_{13}, d^7_{13}, d^8_{13}\}$ and one transition from $\{d^7_{13}, d^8_{13}, d^9_{13}\}$.

It is easy to verify that the learning diagnoser of Figure 4 follows the "thought process" illustrated in Example 3. In particular, note that after the output of the final λ, the sole remaining candidate has the generator-set $\mathcal{G} = \{((\{d^7_{13}, d^8_{13}\}\{d^9_{13}, d^{10}_{13}, d^{11}_{13}\})(\{d^5_{13}, d^6_{13}\}\{d^9_{13}\}))\}$, and the class generated by $\mathcal{G}$ is precisely just the hypotheses

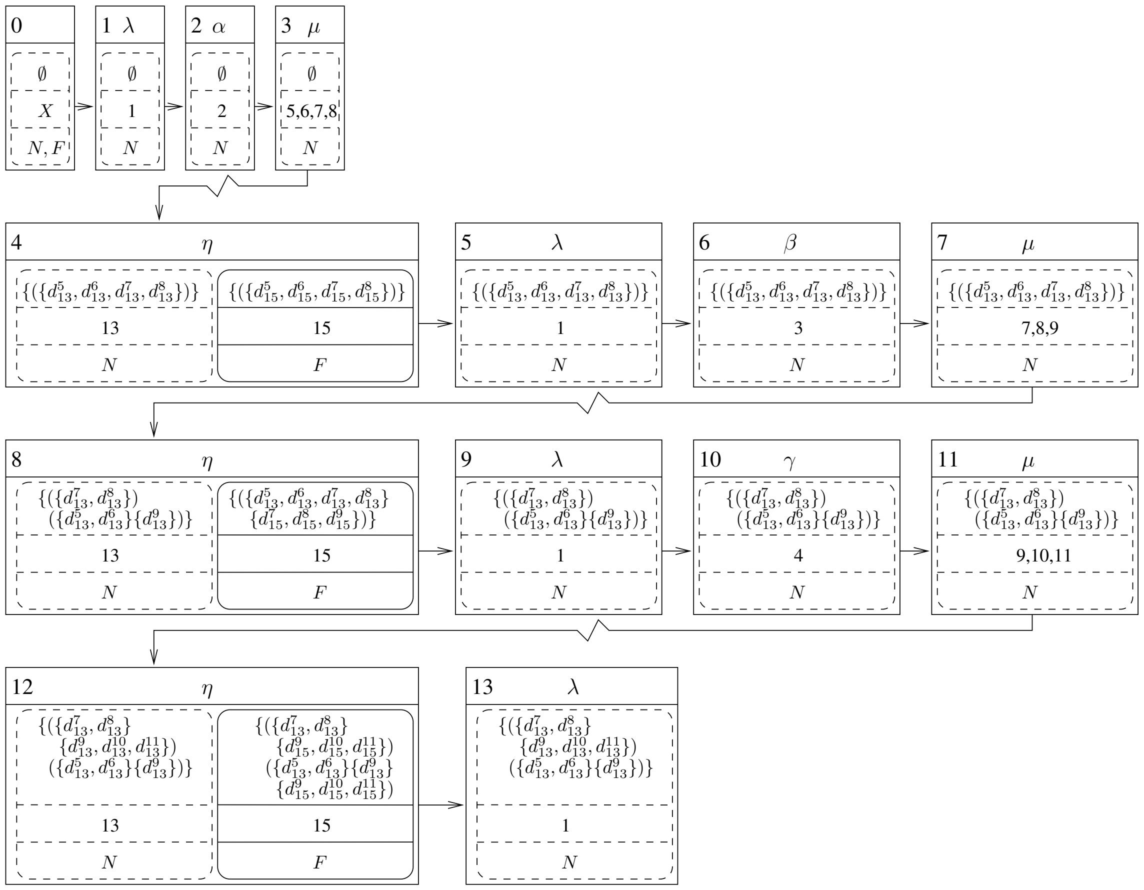

Fig. 4. The learning diagnoser for Example 3, after the output sequence $(\lambda\alpha\mu\eta\lambda\beta\mu\eta\lambda\gamma\mu\eta\lambda)$.

given in Example 4. □

The reader may consult [9] for further details on the learning diagnoser and on the computational complexity of the learning diagnoser.

V. Conclusions

In this paper, we have presented a framework for fault diagnosis in discrete-event systems which can carry out diagnosis even when the model of the system to be diagnosed is incomplete. We have illustrated the main concepts behind the learning diagnoser with some examples. The learning diagnoser generates hypotheses about the missing information in the model and estimates the state (and fault condition) of the system by using parsimonious covering theory. We have left out many technical details, which the interested reader may find in [9].

Owing to space limitations, there are a number of theoretical issues which we have not discussed in this paper. In [9], we have obtained conditions for the diagnosability of incomplete models and the learnability of the missing transitions. We will treat these issues as well as computational complexity and model reduction in future papers.

References

[1] Sampath, Meera, Sengupta, Raja, Lafortune, Stéphane, Sinnamohideen, Kasim, and Teneketzis, Demosthenis, "Diagnosability of Discrete-Event Systems," *IEEE Transactions on Automatic Control*, vol. 40, no. 9, pp. 1555–1575, 1995.

[2] Hamscher, W., Console, L., and de Kleer, J., *Readings in Model-based Diagnosis*. Morgan Kaufmann, 1992.

[3] Hashtrudi Zad, Shahin, Kwong, R.H., and Wonham, W.M., "Fault diagnosis in discrete-event systems: framework and model reduction," *IEEE Transactions on Automatic Control*, vol. 48, no. 7, pp. 1199–1212, 2003.

[4] Rivest, Ronald L. and Schapire, Robert E., "Diversity-based Inference of Finite Automata," *Journal of the ACM*, vol. 41, no. 3, pp. 555–589, 1994.

[5] Gold, E. M., "Complexity of Automaton Identification from Given Data," *Information and Control*, vol. 37, pp. 302–320, 1978.

[6] Li, Ming and Vazirani, Umesh, "On the Learnability of Finite Automata," in *Proceedings of the first annual workshop on Computational learning theory (COLT '88)*, 1988, pp. 359–370.

[7] Belz, Anja and Eskikaya, Berkan, "A Genetic Algorithm for Finite State Automata Induction with an Application to Phonotactics," in *Proceedings of the ESSLLI-98 Workshop on Automated Acquisition of Syntax and Parsing*, 1998.

[8] S. Hashtrudi Zad, "Fault Diagnosis in Discrete-Event and Hybrid Systems," Ph.D. dissertation, University of Toronto, 1999.

[9] Yeung, David L., "Fault Diagnosis and Learning in Discrete-Event Systems with Incomplete Information," Master's thesis, University of Toronto, 2003.

[10] Peng, Yun and Reggia, James A., *Abductive Inference Models for Diagnostic Problem-Solving*. Springer-Verlag, 1990.