Failure Diagnosis Using Discrete-Event Models

Meera Sampath, *Student Member, IEEE,* Raja Sengupta, Stéphane Lafortune, *Member, IEEE,* Kasim Sinnamohideen, *Member, IEEE,* and Demosthenis C. Teneketzis, *Member, IEEE*

Abstract— **Detection and isolation of failures in large, complex systems is a crucial and challenging task. The increasingly stringent requirements on performance and reliability of complex technological systems have necessitated the development of sophisticated and systematic methods for the timely and accurate diagnosis of system failures. We propose a discrete-event systems (DES) approach to the failure diagnosis problem. This approach is applicable to systems that fall naturally in the class of DES; moreover, for the purpose of diagnosis, continuous-variable dynamic systems can often be viewed as DES at a higher level of abstraction. We present a methodology for modeling physical systems in a DES framework and illustrate this method with examples. We discuss the notion of diagnosability, the construction procedure of the diagnoser, and necessary and sufficient conditions for diagnosability. Finally, we illustrate our approach using realistic models of two different heating, ventilation, and air conditioning (HVAC) systems, one diagnosable and the other not diagnosable. While the modeling methodology presented here has been developed for the purpose of failure diagnosis, its scope is not restricted to this problem; it can also be used to develop DES models for other purposes such as control. A detailed treatment of the theory underlying our approach can be found in a companion paper [27].**

I. Introduction

DETECTION and isolation of failures in large, complex systems is a crucial and challenging task. Most practical systems employ some means of fault detection, the simplest of such schemes involving threshold logic, alarms, and warning systems. The increasingly stringent requirements on performance and reliability of complex technological systems, however, have necessitated the development of sophisticated and systematic methods for the timely and accurate diagnosis of system failures. The problem of failure diagnosis has received considerable attention in the literature of reliability engineering, control, and computer science and a wide variety of schemes have been proposed. Failure diagnosis using fault trees has been studied in detail by reliability engineers [17], [16], [32], [8], [34]. Quantitative, analytical-model-based methods have been extensively studied by control systems researchers (see [10], [33] and [35] and references therein; also see [3] and [31]) while expert systems and model-based reasoning schemes for diagnosis have been proposed by computer scientists (see, e.g., [5], [20], [9], [7], [6], [22], [11], [23], and [26]). A detailed discussion of several of these methods has appeared in [24]. For a brief overview of the salient features of the aforementioned methods, see [28]. Recently, the problem of failure diagnosis has also been studied in the framework of discrete-event systems (DES) [4], [14], [18], [19], [29], [34]. In [18] and [19], the authors propose a state-based approach to diagnosability; they study the problems of off-line diagnosis and on-line diagnosis where the basic idea of the diagnostic procedure is to "test and observe." Extensions of the above work can be found in [4] where the authors study testability of DES. In [14], the authors present a template monitoring scheme based on timing and sequencing relationships of events for fault monitoring in manufacturing systems. In [34], the authors propose a Petri net based method for failure diagnosis of manufacturing systems which uses Petri net models for failure detection and fault trees for failure isolation.

We propose in this paper and in the companion paper [27] a DES approach to the failure diagnosis problem that expands on the work in [29] and is different from the DES-based approaches mentioned above. DES are characterized by a discrete-state space of logical values and event driven dynamics. Most large scale dynamic systems can be viewed as DES at some level of abstraction. Hence, the proposed method of fault diagnosis is applicable not only to systems that fall naturally in the class of DES (communication networks and computer systems, for instance), but also to systems traditionally treated as continuous variable dynamic systems and modeled by differential equations. One of the major advantages of the proposed method is that it does not require detailed in-depth modeling of the system to be diagnosed and hence is ideally suited for the diagnosis of large complex systems like heating, ventilation and air conditioning (HVAC) units, power plants, and semiconductor processes. Other application areas include automated manufacturing systems like automobile manufacturing where systematic diagnostic procedures are necessary to check equipment integrity before they leave the production line. Fig. 1 illustrates the overall system architecture which contains in it a DES-based diagnostic subsystem. We assume a two-level system architecture. At the lower level is the system itself with its set of controllers; these low-level controllers typically consist of equipment controllers and multivariable controllers. The upper level consists of the supervisor, which performs the tasks of control and coordination of the low-level controllers, failure diagnosis, failure recovery/system reconfiguration following failure identification, and coordination of all of these subsystem operations. The interface between the two layers conveys information on occurrences of observable

Manuscript received May 16, 1994. Recommended by Associate Editor, X. Cao. This work was supported in part by NSF Grants ECS-9057967, ECS-9312134, and ECS-9204419, with additional support from DEC and GE.

M. Sampath, R. Sengupta, S. Lafortune, and D. Teneketzis are with the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109-2122 USA.

K. Sinnamohideen is with Johnson Controls, Inc., Milwaukee, WI 53201 USA.

Publisher Item Identifier S 1063-6536(96)02070-2.

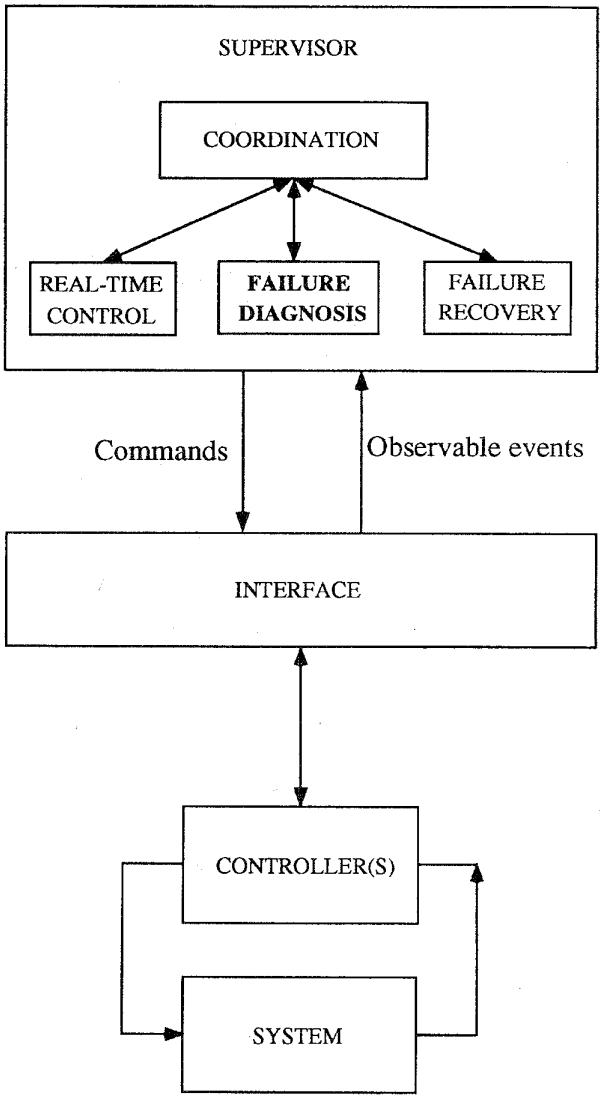


Fig. 1. The conceptual system architecture.

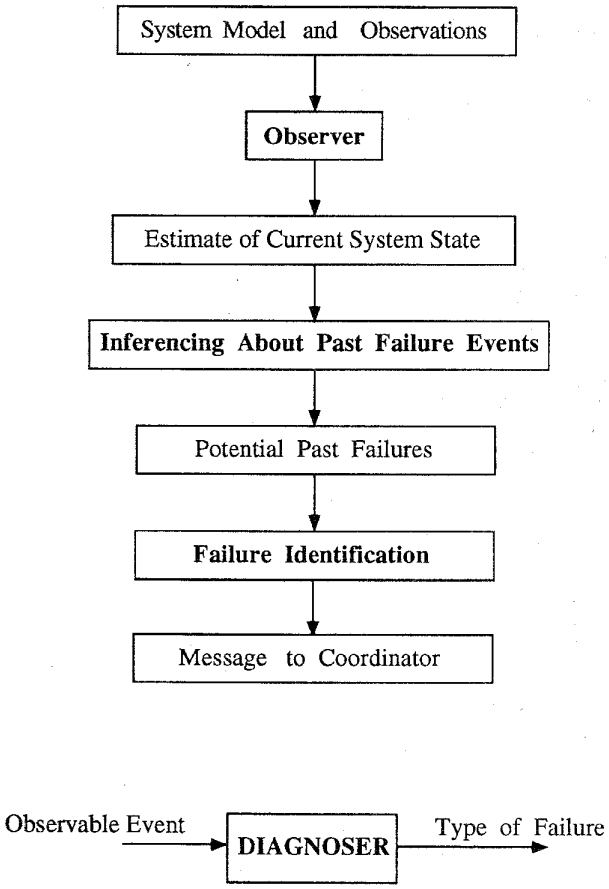


Fig. 2. The diagnostic process.

events in the system to the supervisor and communicates the commands issued by the supervisor to the system.

Our approach to failure diagnosis involves two major steps: developing a discrete-event model of the system to be diagnosed followed by construction of the diagnoser. The discrete-event model that we develop captures both the normal and the failed behavior of the system. The failures are modeled as unobservable events and the objective is to infer about past occurrences of these failures on the basis of the observed events. The diagnoser is a finite-state machine (FSM) built from the system model. This machine performs diagnosis when it observes on-line the behavior of the system. The diagnoser provides estimates of the state of the system after the occurrence of every observable event. In addition, states of the diagnoser carry failure information and occurrences of failures can be detected (with a finite delay) by inspecting these states. Fig. 2 illustrates the basic paradigm of our approach. The top part of this figure shows the various steps involved in failure diagnosis; all these steps are to be performed by the diagnoser, as shown in the bottom part of Fig. 2. This approach to diagnosis is appropriate for failures that involve significant changes in the status of system components but do not necessarily bring the system to a halt.

One of the main contributions of this paper is a precise methodology for modeling physical systems in a DES framework. The system is assumed to consist of several distinct physical components and equipped with a set of sensors. Starting from discrete-event models of the individual components and from the discrete-valued sensor maps, we present a systematic procedure for generating a composite model which captures the interaction among the components and also incorporates in it the sensor maps. This composite model is the DES on which we perform diagnostics. While this approach to modeling has been developed for the purpose of diagnostics, its scope is not restricted to this problem; the model building methodology presented here can be used to develop DES models of any real system for other purposes such as control.

Aside from the modeling methodology, the rest of the theoretical developments underlying our approach to failure diagnosis are presented in [27]. In [27] we introduce two related notions of diagnosability of a language generated by a DES. The first definition, referred to as diagnosability,

is more stringent than the second one, which we refer to as I-diagnosability. Roughly speaking, a system is said to be diagnosable if it is possible to detect, with finite delay, occurrences of certain specific unobservable events, namely, the failure events. In [27] we present a formal construction procedure of the diagnoser followed by necessary and sufficient conditions for diagnosability and I-diagnosability. These conditions are stated on the diagnoser or variations thereof. Thus, the diagnoser serves two purposes: 1) on-line detection and isolation of failures and 2) off-line verification of the diagnosability properties of the system.

In this paper, we restrict our attention to the notion of I-diagnosability introduced in [27]. Section II describes, with illustrative examples, model building for diagnosis. In Section III, we present some of the main results of [27]; we review the notion of I-diagnosability, the construction of diagnosers, and the necessary and sufficient conditions for I-diagnosability. Next, we illustrate our approach to failure diagnosis with two examples of HVAC systems. The DES models of these systems, the corresponding diagnosers and their analysis are presented in Section IV. In Section V, we provide a brief comparison of the proposed method with some of the other approaches to failure diagnosis mentioned earlier. Finally, in Section VI we summarize the main results of this paper.

II. Model Building for Diagnosis

Suppose that the system to be diagnosed has N individual components; typically, these components consist of equipment and controllers. We first build DES models for these components. Let

$$G_i = (X_i, \Sigma_i, \delta_i, x_{0i}) \tag{1}$$

refer to the FSM (see, e.g., [25]) model of the ith component; here X_i is the state space, Σ_i is the event set, δ_i is the transition function, and x_{0i} is the initial state of G_i. The states in X_i and the events in Σ_i reflect the normal and the failed behavior of the ith component. Some of the events in Σ_i are observable, i.e., their occurrence can be observed, while the rest are unobservable. Typically, the observable events include commands issued by the supervisor while the unobservable events include failure events.

Next, we compose these individual models using the standard synchronous composition operation on state machines (see, e.g., [15]). The synchronous composition procedure, recalled below, is used to model the joint operation of two or more DES given their individual FSM models. Consider two discrete-event systems $G_1 = (X_1, \Sigma_1, \delta_1, x_{01})$ and $G_2 = (X_2, \Sigma_2, \delta_2, x_{02})$. We denote by $e_i(x)$ the active event set of G_i at state x, i.e., the set of all transitions of G_i defined at state x. Let $G = (X, \Sigma, \delta, x_0)$ denote the synchronous composition of G_1 and G_2. Then

$$\Sigma = \Sigma_1 \cup \Sigma_2$$
$$X = X_1 \times X_2$$
$$x_0 = (x_{01}, x_{02})$$

$$\delta(\sigma, (x_1, x_2)) = \begin{cases} (\delta_1(\sigma, x_1), \delta_2(\sigma, x_2)) \\ \quad \text{if } \sigma \in e_1(x_1) \cap e_2(x_2) \\ (\delta_1(\sigma, x_1), x_2) \\ \quad \text{if } \sigma \in e_1(x_1) - \Sigma_2 \\ (x_1, \delta_2(\sigma, x_2)) \\ \quad \text{if } \sigma \in e_2(x_2) - \Sigma_1 \\ \text{undefined otherwise.} \end{cases}$$

Thus an event σ which is common to both G_1 and G_2 is possible at state (x_1, x_2) of G only if σ is in the active event set of G_1 at x_1 and in the active event set of G_2 at x_2. In this case, both systems G_1 and G_2 are assumed to execute σ. On the other hand, if σ is an event possible in G_1 (G_2) and it is not in Σ_2 (Σ_1), then only G_1 (G_2) executes the transition σ. It is not difficult to see that the synchronous composition procedure described above can be extended to model the joint operation of any number of DES.

Let

$$\tilde{G} = (\tilde{X}, \tilde{\Sigma}, \tilde{\delta}, \tilde{x}_0) \tag{2}$$

denote the synchronous composition of the component models G_i, $i = 1, \cdots, N$. Observe that we need only consider the accessible part of $\tilde{G}$. $\tilde{G}$ then models the joint operation of these components. Here

$$\tilde{X} \subseteq \prod_i X_i \quad \text{and} \quad \tilde{\Sigma} = \bigcup_i \Sigma_i. \tag{3}$$

Given the set of M sensors of the system of interest, we next identify the sensor maps $h_j : \tilde{X} \to Y_j$, $j = 1, \cdots, M$ where Y_j denotes the discrete set of possible outputs of the jth sensor. Define

$$Y = \prod_{j=1}^{M} Y_j \tag{4}$$

and let $h : \tilde{X} \to Y$ denote the global sensor map defined as follows:

$$h(x) = (h_1(x), h_2(x), \cdots, h_M(x)). \tag{5}$$

Finally, we transform $\tilde{G} = (\tilde{X}, \tilde{\Sigma}, \tilde{\delta}, \tilde{x}_0)$ to $G = (X, \Sigma, \delta, x_0)$ with $x_0 = \tilde{x}_0$ by redefining the transitions of $\tilde{G}$ as follows. Let $\delta(x, \sigma) = x'$ where $x, x' \in \tilde{X}$ and $\sigma \in \tilde{\Sigma}$.

1) If σ is observable (typically a command event), then rename σ in the transition as $\langle \sigma, h(x') \rangle$ and let $\delta(x, \langle \sigma, h(x') \rangle) = x'$. The new event $\langle \sigma, h(x') \rangle$ is observable in Σ.
2) If σ is unobservable and if $h(x) = h(x')$, then σ is left unchanged in G and $\tilde{\delta}(x, \sigma) = x'$. The event σ is treated as unobservable in Σ.
3) If σ is unobservable and if $h(x) \neq h(x')$, then replace the transition $\tilde{\delta}(x, \sigma) = x'$ by the following two transitions:
   a) $\delta(x, \sigma) = x_{\text{new}}$ and
   b) $\delta(x_{\text{new}}, \langle h(x) \to h(x') \rangle) = x'$

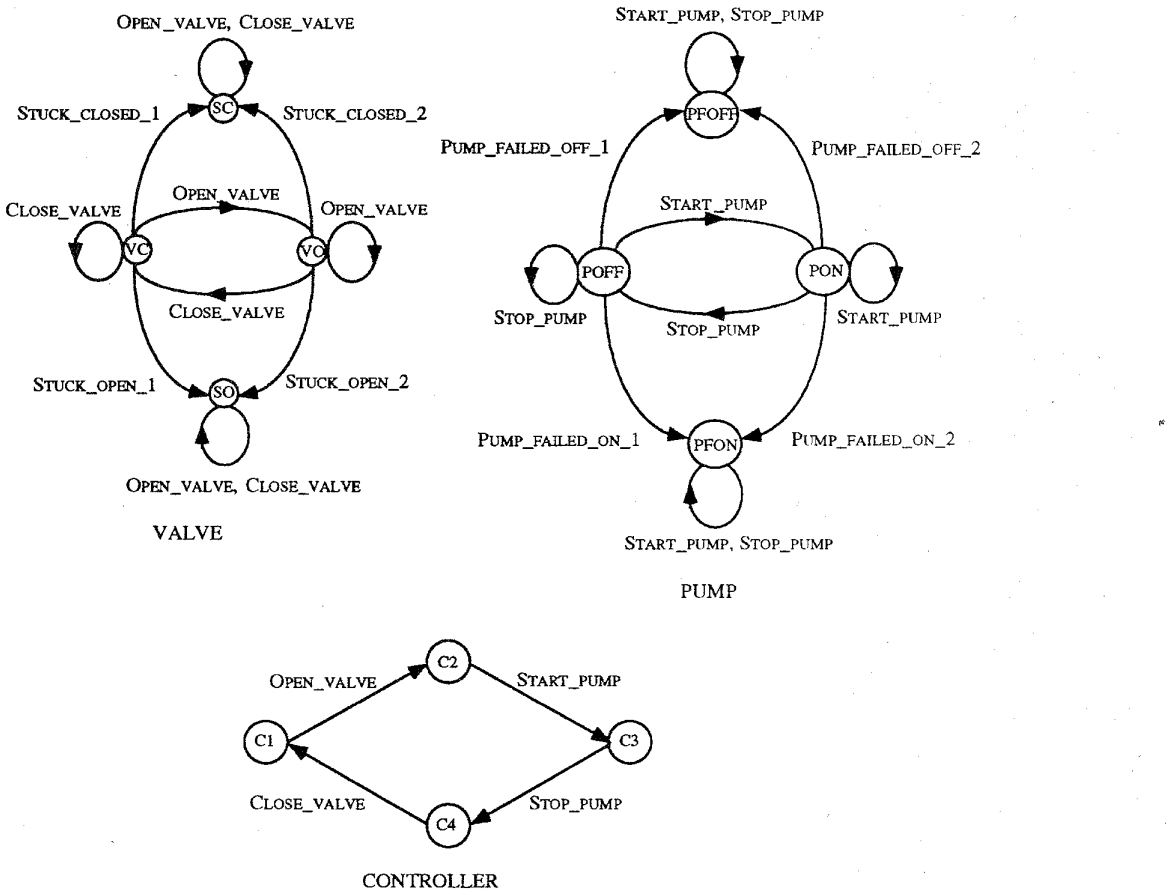


Fig. 3. Component models for Example 2.1.

where x_{new} denotes a newly introduced state and $\langle h(x) \rightarrow h(x') \rangle$ denotes the change in sensor readings corresponding to states x and x'. The first transition σ is unobservable in Σ while the second $\langle h(x) \rightarrow h(x') \rangle$ is observable.

For the purpose of clarity, we henceforth denote all events in the composite model G within braces, $\langle \cdots \rangle$. Therefore the event set Σ of G consists of composite events of the following three types:

1) $\langle \sigma, h(x') \rangle$: observable;
2) $\langle \sigma \rangle$: unobservable; and
3) $\langle h(x) \rightarrow h(x') \rangle$: observable.

Let X_{new} denote the set of all new states x_{new} introduced in Step 3) above. Then

$$X = \tilde{X} \cup X_{\text{new}}. \tag{6}$$

This completes the model building procedure for diagnosis. The system to be diagnosed is now represented by the discrete-event model

$$G = (X, \Sigma, \delta, x_0). \tag{7}$$

Note that the model G accounts for the normal and failed behavior of the system. The observable events in this system may be one of the following: commands issued by the supervisor and sensor readings immediately after the execution of the above commands, and changes of sensor readings. The unobservable events may be failure events or other events which cause changes in the system state not recorded by sensors.

We note at this point that the proposed approach to diagnosis is not limited to the case of equipment and controller failures. Sensor failures, too, can be handled in this framework by simply treating the sensor as an additional component of the system. In other words, we develop in addition to the equipment and controller models, explicit discrete-event models, which include both normal and failed states, for those sensors that can fail.

We now present two examples to illustrate the above modeling procedure. These examples also illustrate that in the proposed framework, the modeling can be done at different levels of granularity. In the first example, we model the dynamic behavior of a system over its entire range of operation including start-up and shutdown procedures. In the second example, we model deviations from the steady state of a system.

Example 2.1: Consider an elementary HVAC system consisting of a pump, a valve, and a controller. Fig. 3 depicts the individual component models $G_i, i = 1, 2, 3$, of the valve, pump, and controller, respectively. The valve has four failure events: STUCK_CLOSED_1,

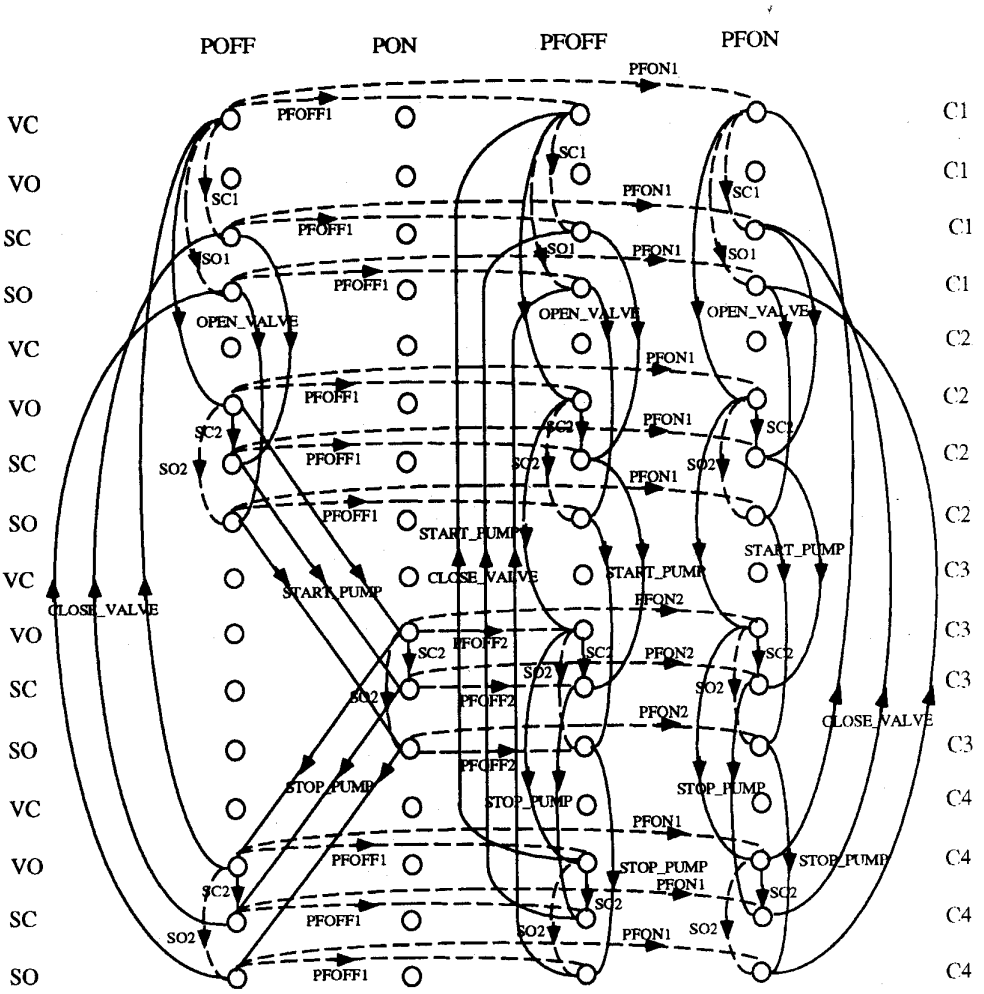


Fig. 4. Synchronous composition of the component models for Example 2.1.

STUCK_CLOSED_2, STUCK_OPEN_1, and STUCK_OPEN_2. The states SC and SO represent the stuck-closed and the stuck-open status of the valve, respectively, while the states VC and VO denote the closed-normal and open-normal status, respectively. Likewise, the pump has four failure events: PUMP_FAILED_OFF_1, PUMP_FAILED_OFF_2, PUMP_FAILED_ON_1, and PUMP_FAILED_ON_2. The states PFOFF and PFON represent the failed-off and failed-on status of the pump while the states PON and POFF represent the normally-on and off status. The only unobservable events in this system are the failure events of the pump and the valve.

The system $\tilde{G}$ in Fig. 4 is obtained by the synchronous composition of the valve, pump, and controller models of Fig. 3. Both the accessible and the inaccessible states of the system are shown in this figure. The inaccessible states are subsequently dropped. Dotted lines in this figure indicate unobservable events while solid lines indicate observable events. For the sake of clarity, some of the events in this figure are shown abbreviated. For instance, the event STUCK_CLOSED_1 is shown as SC1, the event PUMP_FAILED_ON_2 as PFON2, and so forth.

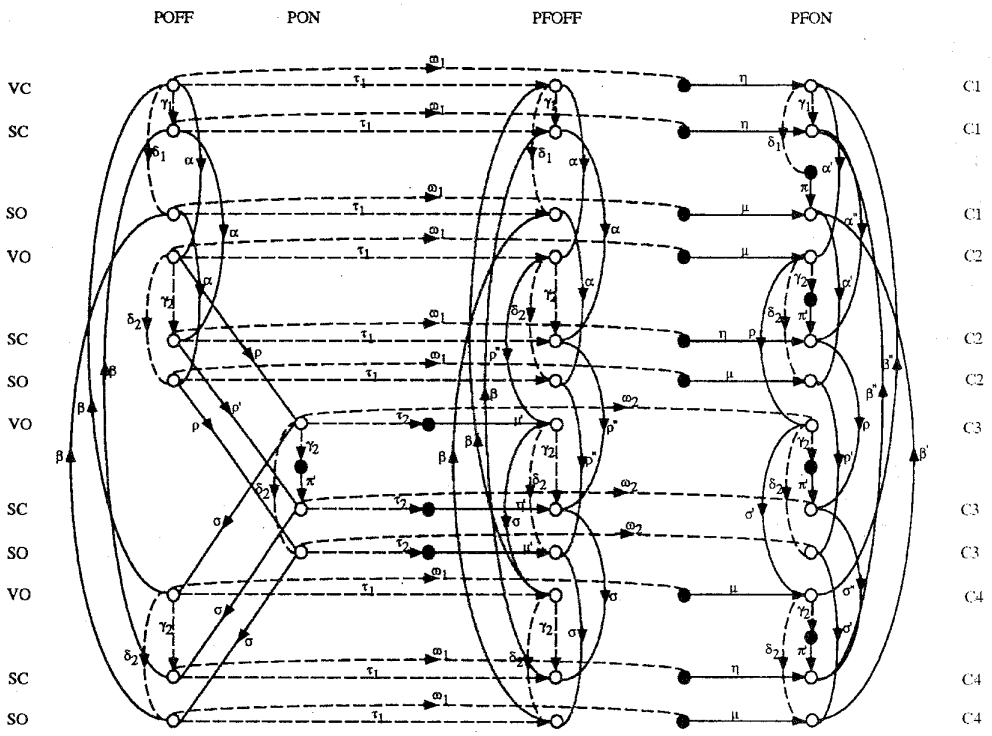
Next, assume that there are two sensors in the system, a pressure sensor on the pump and a valve flow sensor. Let

TABLE I
THE GLOBAL SENSOR MAP FOR EXAMPLE 2.1

h(POFF, VC, •)	=	NP, NF	h(PFOFF, VC, •)	=	NP, NF
h(POFF, VO, •)	=	NP, NF	h(PFOFF, VO, •)	=	NP, NF
h(POFF, SC, •)	=	NP, NF	h(PFOFF, SC, •)	=	NP, NF
h(POFF, SO, •)	=	NP, NF	h(PFOFF, SO, •)	=	NP, NF
			h(PFON, VC, •)	=	PP, NF
h(PON, VO, •)	=	PP, F	h(PFON, VO, •)	=	PP, F
h(PON, SC, •)	=	PP, NF	h(PFON, SC, •)	=	PP, NF
h(PON, SO, •)	=	PP, F	h(PFON, SO, •)	=	PP, F

$Y_1 = \{\text{NP, PP}\}$ and $Y_2 = \{\text{NF, F}\}$ denote the set of outputs of the pressure sensor and flow sensor, respectively. NP and PP denote no pressure and positive pressure, respectively, while NF and F denote no flow and flow, respectively. Table I lists the global sensor map h. Note that the map h is defined only for the accessible states of $\tilde{G}$ in Fig. 4. Also, h does not depend on the state of G_3, the controller, which is indicated in the table by the $\bullet$s.

The final composite model G is given in Fig. 5. The shaded circles in Fig. 5 denote the additional states x_{new}; as before, observable events are indicated by solid lines and unobservable events by dotted lines. The table in Fig. 5 lists the events in



α = <OPEN_VALVE, NP, NF >	γ_1 = <STUCK_CLOSED_1>
α' = <OPEN_VALVE, PP, F >	γ_2 = <STUCK_CLOSED_2>
α'' = <OPEN_VALVE, PP, NF >	δ_1 = <STUCK_OPEN_1>
β = <CLOSE_VALVE, NP, NF >	δ_2 = <STUCK_OPEN_2>
β' = <CLOSE_VALVE, PP, F >	π = <NF→F>
β'' = <CLOSE_VALVE, PP, NF >	π' = <F→NF>
ρ = <START_PUMP, PP, F >	τ_1 = <PUMP_FAILED_OFF_1>
ρ' = <START_PUMP, PP, NF >	τ_2 = <PUMP_FAILED_OFF_2>
ρ'' = <START_PUMP, NP, NF >	ω_1 = <PUMP_FAILED_ON_1>
σ = <STOP_PUMP, NP, NF >	ω_2 = <PUMP_FAILED_ON_2>
σ' = <STOP_PUMP, PP, F >	η = <NP→PP>
σ'' = <STOP_PUMP, PP, NF >	η' = <PP→NP>
	μ = <NP→PP, NF→F>
	μ' = <PP→NP, F→NF>

Fig. 5. The composite model for Example 2.1.

this composite system obtained as a result of the transition renaming procedure described above. ■

Example 2.2: Consider the nitric acid cooling system shown in Fig. 6 (cf. [17]). We now present discrete-event models for this system that capture deviations of its behavior about steady-state conditions. Steady state here refers to operating conditions when the temperature of the nitric acid leaving the system is maintained at the desired set-point value. The individual models of the various components in the system are shown in Fig. 7.

Note that, as in Example 2.1, the component models include both normal and failure events. Also note that Fig. 7 includes a "load" model. We assume that all minor disturbances which cause a deviation of the output temperature result in either of the two events, ABOVE_SP or BELOW_SP, depending on the direction of the deviation. If the temperature sensor is normal, then it reacts to the ABOVE_SP and BELOW_SP events by sending a SENSOR_HIGH and a SENSOR_LOW signal, respectively, to the controller. If the temperature controller is normal, then it reacts to the sensor signals SENSOR_HIGH and SENSOR_LOW, by issuing the commands, OPEN_VALVE and CLOSE_VALVE, respectively, to the cooling water valve. If, however, either of these two components is failed, then deviations of the output temperature cause no change in the status of the system components. The control valve and the cooling water valve are initially assumed to be in the states V1I and V2I, respectively. The event, HIGH_INLET_PR, which corresponds to an abnormal increase in the pressure at the inlet, causes the control valve to open completely, resulting in state V1CO from the initial state V1I. We assume that the system is equipped with a nitric acid shutdown system which stops the flow of incoming nitric acid whenever the event PUMP_SHUTDOWN occurs. This corresponds to the change of state of the control valve from the initial state V1I or the

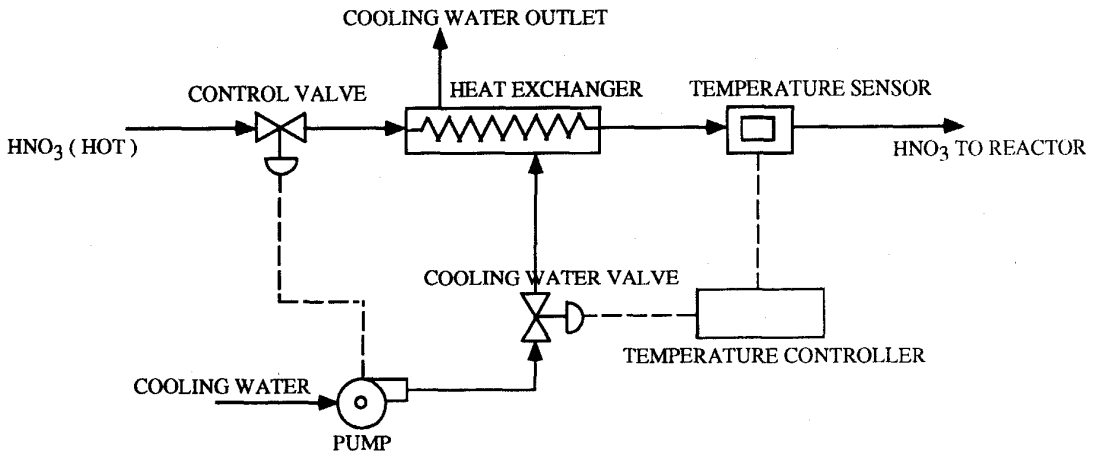


Fig. 6. A nitric acid cooling system.

open state V1CO to the completely closed state, V1CC. If the event NAS_FAILURE, which corresponds to failure of the nitric acid shutdown system, occurs, however, then the control valve remains open (in the failed states V1FI or V1FO) even after PUMP_SHUTDOWN occurs. The states V1FI and V1FO can be thought of as equivalent to the valve being stuck open. Likewise, the event LOW_AIR_PR, which corresponds to the major disturbance "low air pressure" at the cooling water valve, causes the cooling water valve to get into a completely closed state at which it gets stuck and responds to no further open or close valve commands. The event PUMP_SHUTDOWN causes transitions of the cooling water valve into its completely closed state V2CC. The major external disturbances, HIGH_INLET_PR and LOW_AIR_PR, cause a large deviation in output temperature; we assume that the controller cannot compensate for these large deviations. Note, finally, that we do not explicitly model the heat exchanger.

Suppose next that there are two sensors on the system: a valve stem-position-indicator mounted on the cooling water valve and a temperature sensor. We assume that the temperature sensor can fail and hence we develop an explicit model of this sensor. In Fig. 7, SI, SL, and SH correspond to normal states of the temperature sensor, and SF corresponds to its failed state. In addition to the outputs of the two sensors mentioned above, we assume that we have a third measurement available, namely the output of the controller.

As before, it is straightforward to obtain the synchronous composition of the component models in Fig. 7. This FSM has 368 states and 1486 transitions. (It is not shown here due to space limitations.) States of the synchronous composition are tuples of the form $(x_1, x_2, x_3, x_4, x_5, x_6)$ where x_1 is a state of the cooling water valve, x_2 is a state of the control valve, x_3 is a state of the pump, x_4 is a state of the load model, x_5 is a state of the temperature sensor, and x_6 is a state of the temperature controller.

The next step in the modeling procedure is to obtain the global sensor map for the system. Let Y_1 = {ZERO, UP, DOWN} denote the set of outputs of the valve stem-position-indicator. Here ZERO refers to the steady-state position of the valve, UP denotes that the valve is more open than at steady state, and DOWN denotes that the valve is less open compared to the steady-state position. The set of outputs of the temperature sensor is given by Y_2 = {ZERO, HIGH, LOW}, where ZERO denotes no deviation from the set point value of the temperature of the output nitric acid stream, and HIGH and LOW denote positive and negative deviations, respectively. We assume that when the sensor fails, its output remains at ZERO. The controller output is given by Y_3 = {ZERO, HIGH, LOW}, where ZERO, HIGH, and LOW denote zero, positive, and negative deviations of the controller output from the steady-state value. As in the case of the temperature sensor, the only possible output of the controller when it fails is ZERO. Table II illustrates the output maps for the two sensors and the controller. The global sensor map is simply the union of the individual sensor maps and can be obtained as in (5). The entries in Table II are to be interpreted as follows: Consider, for example, the entry h_1 (V2C, •, •, •, •, •) = DOWN. This means that the output of the valve stem-position-indicator is DOWN when the cooling water valve is in the state V2C, regardless of the states of the other components. The entry h_2 (V2CC, V1CC, PS, •, SI/SL/SH, •) = ZERO means that if the cooling water valve is in state V2CC, the control valve is in state V1CC, the pump is in state PS, and the temperature sensor is not failed, then the output of the temperature sensor is ZERO, and so on. Also, the entry V2I/V2O is to be interpreted as meaning that the control valve could be in state V2I or in state V2O, and so on.

The composite system model can be obtained following the procedure discussed before. This FSM has 646 states and 1764 transitions. Again, it is not shown here due to space limitations. ■

III. Diagnosability and Diagnosers

In this section, we review some of the main results in [27], namely the notion of I-diagnosability, the construction procedure of the diagnoser, and the necessary and sufficient conditions for I-diagnosability, and illustrate these ideas with the simple pump-valve-controller example introduced in the

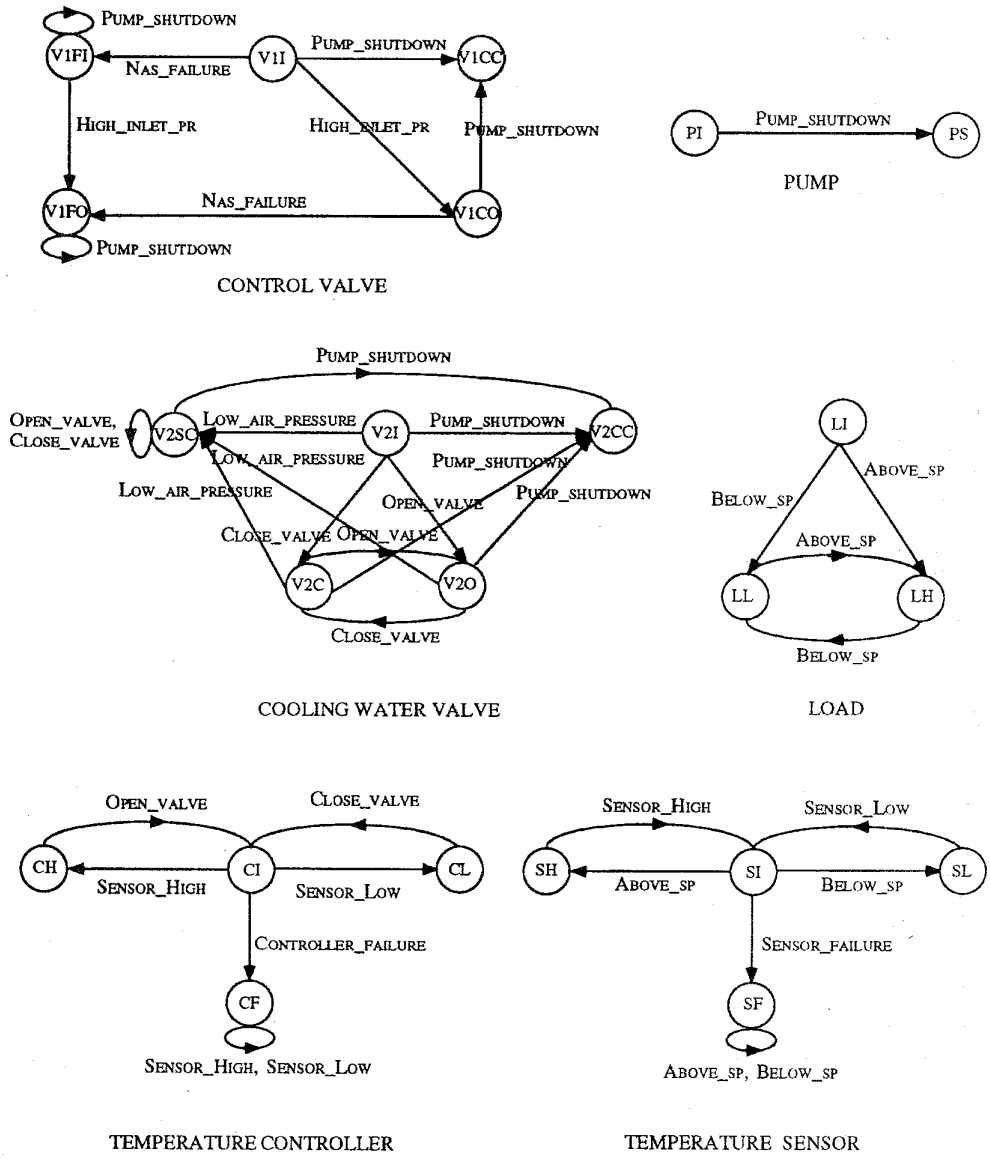


Fig. 7. Component models for the nitric acid cooling system.

preceding section. In Section IV, we will apply these results and the modeling methodology of Section II to two more complex HVAC systems. The reader is referred to [27] for a formal treatment of the ideas described in this section.

A. The Notion of I-Diagnosability

Let $G = (X, \Sigma, \delta, x_0)$ represent the discrete-event model of the system to be diagnosed. The behavior of this DES is described by the prefix-closed language $L(G)$ generated by G [25]. Henceforth, we shall denote $L(G)$ by L. We assume for simplicity that L is live, i.e., there is at least one transition defined at each state in X.

Some of the events in Σ are observable, i.e., their occurence can be observed by an external agent while the rest are unobservable. Thus the event set Σ is partitioned as $\Sigma = \Sigma_o \dot{\cup} \Sigma_{uo}$, where Σ_o represents the set of observable events and Σ_{uo} the set of unobservable events. Let $\Sigma_f \subseteq \Sigma$ denote the set of failure events which are to be diagnosed. We assume, without loss of generality, that $\Sigma_f \subseteq \Sigma_{uo}$, since an observable failure event can be trivially diagnosed. Our objective is to identify the occurrence, if any, of the failure events given partial event observations, i.e., when we observe only part of the traces generated by the system. Since detection of failure events is based on observable transitions of the system, we assume that G does not generate arbitrarily long sequences of unobservable events.

Roughly speaking, a system is diagnosable if it is possible to detect with a finite delay occurrences of failure events using the record of observed events. A system is said to be I-diagnosable if it is possible to diagnose failures, not always, but whenever the failure events are followed by

TABLE II
SENSOR MAPS FOR THE NITRIC ACID COOLING SYSTEM

h_1(V2I, •, •, •, •, •)	=	ZERO
h_1(V2O, •, •, •, •, •)	=	UP
h_1(V2C, •, •, •, •, •)	=	DOWN
h_1(V2CC, •, •, •, •, •)	=	DOWN
h_1(V2SC, •, •, •, •, •)	=	DOWN
h_2(V2I, V1I/V1FI, PI, LI, SI, CI/CF)	=	ZERO
h_2(V2I/V2O/V2C, V1I/V1FI, PI, LL, SL, CI/CF/CH)	=	LOW
h_2(V2I/V2O/V2C, V1I/V1FI, PI, LH, SH, CI/CF/CL)	=	HIGH
h_2(V2I/V2O/V2C, V1I/V1FI, PI, LL, SI, CL)	=	LOW
h_2(V2I/V2O/V2C, V1I/V1FI, PI, LH, SI, CH)	=	HIGH
h_2(V2C, V1I/V1FI, PI, LH, SI, CI/CF)	=	ZERO
h_2(V2O, V1I/V1FI, PI, LL, SI, CI/CF)	=	ZERO
h_2(V2I/V2O, V1I/V1FI, PI, LL, SI, CF)	=	LOW
h_2(V2I/V2C, V1I/V1FI, PI, LH, SI, CF)	=	HIGH
h_2(•, V1CO/V1FO, PI, • , SI/SL/SH, •)	=	HIGH
h_2(•, V1F/V1FO, PS, • , SI/SL/SH, •)	=	HIGH
h_2(V2SC, V1I/V1FI/V1CO/V1FO, PI, •, SI/SL/SH, •)	=	HIGH
h_2(V2CC, V1CC, PS, •, SI/SL/SH, •)	=	ZERO
h_2(•, •, •, •, SF, •)	=	ZERO
h_3(•, •, •, •, •, CI)	=	ZERO
h_3(•, •, •, •, •, CH)	=	HIGH
h_3(•, •, •, •, •, CL)	=	LOW
h_3(•, •, •, •, •, CF)	=	ZERO

certain observable indicator events that are associated with the failures. The notion of I-diagnosability is motivated by the following physical consideration. Consider, for example, an HVAC system with a controller unit. In the normal mode of operation, the controller responds by issuing the command "open valve" whenever it senses a heating load on the system. Likewise, it issues the command "close valve" when the load is removed. Assume that when the controller fails it does not sense the presence of any load on the system and hence does not issue any commands to the valve. Suppose that during operation, the controller does fail and suppose further that it is possible for the system to execute an arbitrarily long sequence of events, which does not involve any of the valve commands. Under such conditions, it is obvious that one cannot diagnose any failure of the valve. We do not want to classify this system as nondiagnosable under these conditions, however. We associate the indicator events "open valve" and "close valve," respectively, with the valve failure events "stuck-closed" and "stuck-open," and require the system to execute the "open valve" event or the "close valve" event before deciding on its diagnosability. The system is considered I-diagnosable if after the execution of the corresponding indicator events it is possible to detect valve failures, while it is termed not I-diagnosable if even after the indicator event is executed the corresponding valve failure remains undetectable. To reiterate, I-diagnosability requires detection of failures only after the occurrence of an indicator event corresponding to the failure.

Further, we may not be required to identify uniquely the occurrence of every failure event (irrespective of whether the failure is followed by an indicator event or not). We may simply be interested in knowing if one of a set of failure events has happened, as for example, when the effect of the set of failures on the system is the same. In this case we classify the set of failures into different classes corresponding to different failure types and require unique identification not of the failure event itself, but of the type of failure whenever such an event occurs in the system.

With the above requirements in mind, we associate to every failure event in Σ_f, one or more observable indicator events. Let $\Sigma_I \subseteq \Sigma_o$ denote the set of indicator events and let $I_f : \Sigma_f \rightarrow 2^{\Sigma_I}$ denote the mapping between failure events and indicator events. Next, we partition the set of failure events into m disjoint sets corresponding to different failure types

$$\Sigma_f = \Sigma_{f1} \dot{\cup} \cdots \dot{\cup} \Sigma_{fm} \tag{8}$$

such that, for each $i = 1, \cdots, m$

$$\sigma_{f1}, \sigma_{f2} \in \Sigma_{fi} \Rightarrow I_f(\sigma_{f1}) = I_f(\sigma_{f2})$$

and define

$$I(\Sigma_{fi}) = I_f(\sigma_f) \text{ for any } \sigma_f \in \Sigma_{fi}. \tag{9}$$

We now have a set of observable events $I(\Sigma_{fi})$ associated with every set Σ_{fi} of the partition. Let Π_f denote the partition on Σ_f. Hereafter, whenever we write that "a failure of type F_i has occurred" we will mean that some event from the set Σ_{fi} has occurred.

The notion of I-diagnosability can then be described more formally as follows. Let st_1 be any trace generated by the system that contains in it a failure event from the set Σ_{fi} followed by an indicator event from the set $I(\Sigma_{fi})$. Let t_2 be any sufficiently long continuation of st_1. I-diagnosability then requires that every trace belonging to the language that produces the same record of observable events as the trace st_1t_2 should contain in it a failure event from the set Σ_{fi}. This implies that along every continuation t_2 of st_1, one can detect the occurrence of a failure of the type F_i with a finite delay, specifically, in at most n_i transitions of the system after st_1. Hence diagnosability requires that every failure event leads to observations distinct enough to enable unique identification of the failure type with a finite delay. Alternately speaking, the observable event set should be rich enough to permit unique identification of each type of failure.

The following example illustrates the choice of indicator events and the partition Π_f for a physical system.

Example 3.1: Consider the HVAC system of Example 2.1. The set of failure events of this system is Σ_f = {⟨STUCK_CLOSED_1⟩, ⟨STUCK_CLOSED_2⟩, ⟨STUCK_OPEN_1⟩, ⟨STUCK_OPEN_2⟩, ⟨PUMP_FAILED_OFF_1⟩, ⟨PUMP_FAILED_OFF_2⟩, ⟨PUMP_FAILED_ON_1⟩, ⟨PUMP_FAILED_ON_2⟩}. The indicator events corresponding to these failure events are chosen as follows:

$$\begin{aligned} I_f(\langle \text{STUCK\_CLOSED\_1} \rangle) &= I_f(\langle \text{STUCK\_CLOSED\_2} \rangle) \\ &= \{\langle \text{OPEN\_VALVE}, \bullet, \bullet \rangle\}, \\ I_f(\langle \text{STUCK\_OPEN\_1} \rangle) &= I_f(\langle \text{STUCK\_OPEN\_2} \rangle) \\ &= \{\langle \text{CLOSE\_VALVE}, \bullet, \bullet \rangle\}, \\ I_f(\langle \text{PUMP\_FAILED\_OFF\_1} \rangle) &= I_f(\langle \text{PUMP\_FAILED\_OFF\_2} \rangle) \\ &= \{\langle \text{START\_PUMP}, \bullet, \bullet \rangle\}, \\ I_f(\langle \text{PUMP\_FAILED\_ON\_1} \rangle) &= I_f(\langle \text{PUMP\_FAILED\_ON\_2} \rangle) \\ &= \{\langle \text{STOP\_PUMP}, \bullet, \bullet \rangle\}. \end{aligned}$$

(Here ⟨OPEN_VALVE, •, •⟩ denotes all the composite events in Fig. 5 that contain OPEN_VALVE, namely α, α', and α'', etc.).

We choose the following partition on Σ_f:

$$\Sigma_{f1} = \{\langle \text{STUCK\_CLOSED\_1}\rangle, \langle \text{STUCK\_CLOSED\_2}\rangle\},$$
$$\Sigma_{f2} = \{\langle \text{STUCK\_OPEN\_1}\rangle, \langle \text{STUCK\_OPEN\_2}\rangle\},$$
$$\Sigma_{f3} = \{\langle \text{PUMP\_FAILED\_OFF\_1}\rangle, \langle \text{PUMP\_FAILED\_OFF\_2}\rangle\},$$
$$\Sigma_{f4} = \{\langle \text{PUMP\_FAILED\_ON\_1}\rangle, \langle \text{PUMP\_FAILED\_ON\_2}\rangle\}.$$

Therefore, we have

$$I(\Sigma_{f1}) = \{\langle \text{OPEN\_VALVE}, \bullet, \bullet\rangle\},$$
$$I(\Sigma_{f2}) = \{\langle \text{CLOSE\_VALVE}, \bullet, \bullet\rangle\},$$
$$I(\Sigma_{f3}) = \{\langle \text{START\_PUMP}, \bullet, \bullet\rangle\},$$
$$I(\Sigma_{f4}) = \{\langle \text{STOP\_PUMP}, \bullet, \bullet\rangle\}. \quad \blacksquare$$

We conclude this section by noting that when more than one failure of the same type occurs along a trace s of L we do not require that each of these occurrences be detected. It suffices to be able to conclude, within finitely many events, that along s, a failure of the type F_i happened.

B. The Diagnoser

Given an FSM model G of the system to be diagnosed, the diagnoser for the system is also an FSM denoted by G_d^I. A state q_d of G_d^I is of the form $q_d = \{(x_1, l_1), \cdots, (x_n, l_n)\}$ where x_i is a state of G and l_i is a label of the form

$$l_i = \{N\} \quad \text{or} \quad l_i = \{F_{i_1} F_{i_2}, \cdots, F_{i_k}\} \quad \text{or}$$
$$l_i = \{F_{i_1}, F_{i_2}, \cdots, F_{i_k}, I_{j_1}, I_{j_2}, \cdots, I_{j_l}\}$$

where $\{i_1, i_2, \cdots, i_k\} \subseteq \{1, 2, \cdots, m\}$, $m = |\Pi_f|$, and $\{j_1, j_2, \cdots, j_l\} \subseteq \{i_1, i_2, \cdots, i_k\}$. Here N is to be interpreted as meaning "normal," F_i, $i \in \{1, \cdots, m\}$ as meaning that a failure of the type F_i has occurred, and I_i, $i \in \{1, \cdots, m\}$ as meaning that an indicator event of the type I_i has occurred. The diagnoser G_d^I can be thought of as an extended observer for G which gives 1) an estimate of the current state of the system after the occurrence of every observable event and 2) information on potential past failure occurrences in the form of labels as mentioned above.

We now present a brief summary of the diagnoser construction procedure. We assume that the system is normal to start with and hence define the initial state of the diagnoser to be $(x_0, \{N\})$. Let the current state of the diagnoser (which is the set of estimates of the current state of G with their corresponding labels) be q_1 and let the next observed event be σ. The new state of the diagnoser q_2 is computed following a three step process:

1) For every state estimate x in q_1, compute the reach due to σ, defined to be $S(x, \sigma) = \{\delta(x, s\sigma)$ where $s \in \Sigma_{uo}^*\}$. The reach then gives the set of all possible states the system could be in after the occurrence of the event σ accounting for all possible unobservable events that could precede the occurrence of σ.
2) Let $x' \in S(x, \sigma)$ with $\delta(x, s\sigma) = x'$. Propagate the label l associated with x to the label l' associated with x' according to the following rules:
   a) If $l = \{N\}$ and s contains no failure events, then the label l' is also $\{N\}$.
   b) If $l = \{N\}$ and s contains failure events from Σ_{fi} and Σ_{fj}, then $l' = \{F_i, F_j\}$.
   c) If $l = \{N\}$, s contains failure events from Σ_{fi}, and $\sigma \in I(\Sigma_{fi})$, then $l' = \{F_i, I_i\}$.
   d) If $l = \{F_i\}$, s contains no failure events and $\sigma \in I(\Sigma_{fi})$, then $l' = \{F_i, I_i\}$.
   e) If $l = \{F_i, I_i\}$ and s contains no failure events then $l' = \{F_i, I_i\}$.
   f) If $l = \{F_i, F_j\}$, s contains failure events from Σ_{fk}, and $\sigma \in I(\Sigma_{fr})$, $r \in \{i, j, k\}$, then $l' = \{F_i, F_j, F_k, I_r\}$.
3) Let q_2 be the set of all (x', l') pairs computed following Steps 1) and 2) above, for each (x, l) in q_1. Replace by (x', l') all (x', l'), $(x', l'') \in q_2$ such that $F_i \in l' \Leftrightarrow F_i \in l'' \; \forall i \in \{1, \cdots, m\}$, $I_j \in l'$ and $I_j \notin l''$. That is, if the same state estimate x' appears more than once in q_2 with labels differing only in their I component, we eliminate from q_2 the pair that does not contain the I label.

We make the following observations on the diagnoser G_d^I:

1) The labels attached to the state estimates carry information on occurrences of failure events (F_i labels) and the corresponding indicator events (I_i labels).
2) We append the I_i label to any l only if an indicator event from $I(\Sigma_{fi})$ follows a failure event from Σ_{fi}. The set of I_i labels is always a subset of the set of F_i labels in any (x, l) pair in $q \in Q_d$.
3) The F_i labels as well as the I_i labels propagate from state to state.
4) When the diagnoser G_d^I is run in parallel with the system, failures can be detected on-line by inspecting the labels associated with the appropriate state of the diagnoser. If the diagnoser transitions into a state q such that every component of q contains the label F_i, then we conclude for sure that a failure of the type F_i has occurred regardless of what the current state of G is. We shall refer to such a state q as an F_i-certain state.
5) The I_i labels do not carry failure information, per se, but carry information on whether a particular failure event was followed by a corresponding indicator event or not. We shall find this information useful in checking for I-diagnosability, as we shall see in the next section.

The following example illustrates the construction of the diagnoser. In several of the illustrations that follow, we represent (x, l) pairs simply as xl for clarity.

Example 3.2: Consider the HVAC system of Example 2.1. For simplicity, we now assume that the pump has no failures, i.e., the only possible events and states of the pump are {START_PUMP, STOP_PUMP} and {POFF, PON}, respectively. The valve and the controller models are the same as before. We assume, as in Example 2.1, that there are two sensors: a pump pressure sensor and a valve flow sensor. The global sensor map for this system is given by the left column of Table I. Let the desired partition and indicator event set be as follows:

$$\Sigma_{f1} = \{\langle \text{STUCK\_CLOSED\_1}\rangle, \langle \text{STUCK\_CLOSED\_2}\rangle\},$$
$$\Sigma_{f2} = \{\langle \text{STUCK\_OPEN\_1}\rangle, \langle \text{STUCK\_OPEN\_2}\rangle\}$$

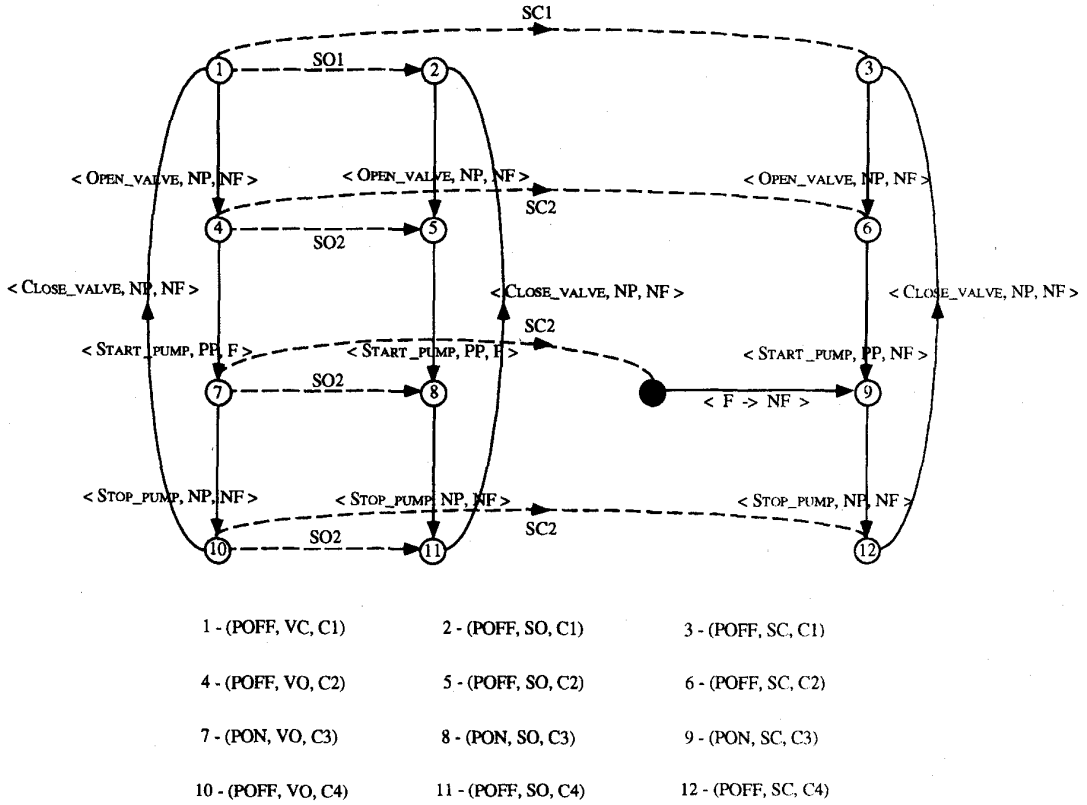


Fig. 8. The composite system for Example 3.2.

and

$$I(\Sigma_{f1}) = \{\langle \text{OPEN\_VALVE}, \bullet, \bullet \rangle\},$$
$$I(\Sigma_{f2}) = \{\langle \text{CLOSE\_VALVE}, \bullet, \bullet \rangle\}.$$

Fig. 8 represents the composite model for this system and Fig. 9 its diagnoser. The initial state of the system is chosen to be (POFF, VC, C1) denoted by state 1 in Fig. 8. ■

C. Necessary and Sufficient Conditions for I-Diagnosability

The diagnoser G_d^I discussed in the last section is used not only to perform on-line diagnosis but is used also to verify off-line the diagnostic properties of the system. In other words, the necessary and sufficient conditions for I-diagnosability of a given language can be stated as conditions on its diagnoser. The notion of an (F_i, I_i)-indeterminate cycle in G_d^I is the most crucial element in the development of the necessary and sufficient conditions for I-diagnosability and can be informally defined as follows. First, we define an (F_i, I_i)-uncertain state of G_d^I as a state that contains at least one component which carries the labels F_i and I_i, and at least one component which does not carry the label F_i. An (F_i, I_i)-indeterminate cycle is then a cycle composed exclusively of (F_i, I_i)-uncertain states such that there exist

1) a corresponding cycle (of observable events) in G involving only states that carry F_i and I_i in their labels in the cycle in G_d^I; and
2) a corresponding cycle (of observable events) in G involving only states that do not carry F_i in their labels in the cycle in G_d^I.

Example 3.3: Consider the system represented in Fig. 8. Inspection of the diagnoser G_d^I (shown in Fig. 9) for this system, reveals the presence of a cycle of (F_2, I_2)-uncertain states (bottom left of figure). Corresponding to this cycle, there are two cycles in the system: one involving states 1, 4, 7, and 10 which appear with an N label in the diagnoser and the other involving states 2, 5, 8, and 11, which appear with $\{F_2, I_2\}$ labels in the diagnoser. Hence this cycle qualifies as an (F_2, I_2)-indeterminate cycle. ■

An (F_i, I_i)-indeterminate cycle has the property that if the diagnoser enters this cycle, after the occurrence of a failure event, then it can continue to remain forever in this cycle. Since none of the states in the cycle is F_i-certain it is not possible to conclude whether a failure event of type F_i has occurred or not as long as the diagnoser remains in this cycle. Finally, since the state components that carry the F_i label also carry the I_i label, we know that the failure event was followed by a corresponding indicator event when the diagnoser entered the cycle. It is then intuitively obvious that such a cycle in the diagnoser G_d^I implies that the system is not I-diagnosable. We show in [27] that if the diagnoser G_d^I has no such cycles then, whenever a failure event occurs in the system, followed by an appropriate indicator event, the diagnoser will enter an F_i-certain state in a bounded number of transitions which implies that the failure event can be detected with a finite

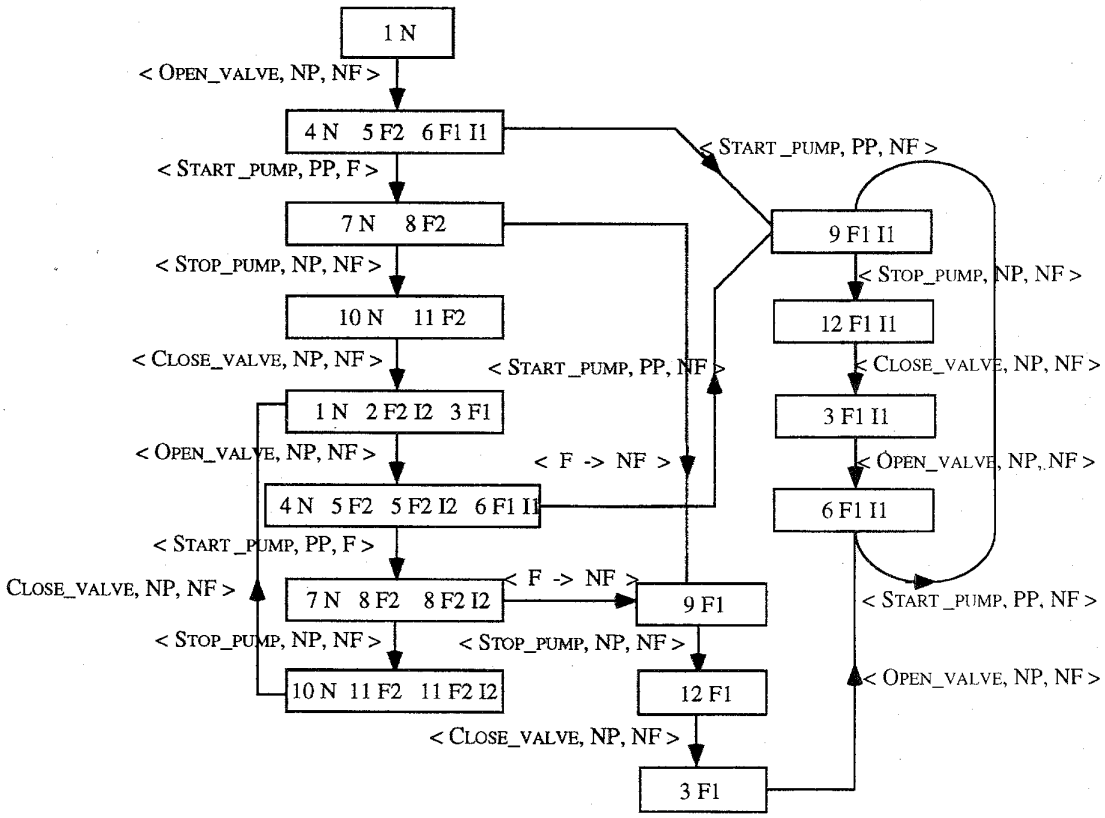


Fig. 9. Diagnoser G_d^I for Example 3.2.

delay. Thus the absence of (F_i, I_i)-indeterminate cycles in G_d^I is a necessary and sufficient condition for I-diagnosability. Since checking for I-diagnosability amounts to cycle detection in the diagnoser and in G, any of the standard cycle detection algorithms (which are of polynomial complexity) may be used.

Example 3.4: The system represented in Fig. 8 is not I-diagnosable since its diagnoser G_d^I contains an (F_2, I_2)-indeterminate cycle as discussed in Example 3.3. ■

In Section IV-B we present another example of a system that is not I-diagnosable and discuss its physical significance.

We conclude this section with a final remark on implementation. Recall our earlier comment that the I_i labels do not directly carry any failure information. They record the occurrence of indicator events following the failure events and are used solely to test for I-diagnosability. This then implies that once it is established that the system is I-diagnosable, one may then use a much simpler diagnoser that does not carry any I_i label for performing on-line diagnosis. This result is important from an implementation viewpoint, as the simplified diagnoser (denoted by G_d in [27]) will in general have far fewer states than G_d^I.

IV. Application to HVAC Systems

A complete HVAC system in a modern building (or a group of buildings) consists typically of a single set of boilers and chillers which serves as the primary air conditioning medium supplying a large number of air handling systems (AHS's) (see, e.g., [2], [30]). Each AHS, in turn, conditions the air supplied to several rooms, based on local thermostats in these rooms, whose settings govern the amount of air the AHS draws from the main supply. There are several factors favoring the design of accurate automated diagnostic mechanisms for HVAC systems: 1) Detecting failures in these systems is a complex task for a human operator controlling and monitoring the system from a central location since the operator has to respond to alarms coming from various parts of the system, spread over large physical spaces, and identify the root cause of the problem. 2) Quite often the number of sensors and thus the number of immediately observable failures is limited; thus it is necessary to make inferences based on the model of the system behavior and on the limited sensor information available. 3) Most HVAC system components are hard to access and one therefore needs to identify the exact fault location before attempting to take any corrective action that might involve component inspection or replacement.

Fig. 10 illustrates part of a complete HVAC system as described above. To be more precise, it depicts a single AHS with its associated primary conditioning system. We illustrate our approach to failure diagnosis by considering two different scenarios of this system: in System I, the valve and the controller are subject to failures and in System II, the pump and the valve are subject to failures.

A. System 1

Consider the HVAC system of Fig. 10 with the component models as depicted in Fig. 11. Here V1 and V2 represent the

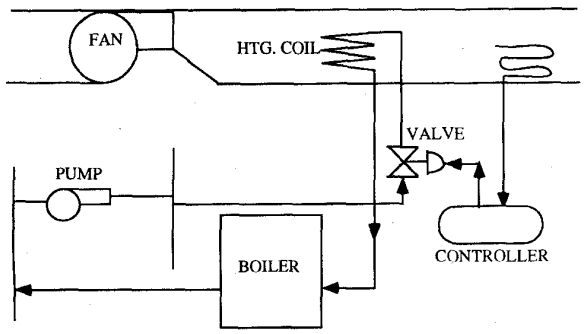


Fig. 10. (Part of) An HVAC system.

closed-normal and open-normal positions, respectively, of the valve while V3 and V4 represent the stuck-open and stuck-closed positions, respectively. F1, P1, and B1 represent the "off" positions of the fan, pump, and boiler, respectively, while F2, P2, and B2 represent their "on" positions. The fan-on state F2 is equivalent to "power-on" in the system; likewise, F1 denotes "power-off." The state L0 in the load model is to be interpreted as unknown load; the load is assumed to be unknown when the system is not powered on. The state L1 represents the absence of heating load on the system and L2 the presence of a heating load; the events SPI and SPD denote an increase in set point (which indicates a demand on the heating system) and a decrease of set point (no load on the system), respectively.

The controller is modeled based on the following assumptions:

- In normal mode of operation, the controller issues a sequence of commands, OV-PON-BON (open valve—pump on—boiler on) whenever it senses a load on the system. Likewise, it issues the commands CV-POFF-BOFF (close valve—pump off—boiler off) when the load disappears.
- When the controller fails off (i.e., when the event CFOFF occurs), it does not sense the presence of load on the system and hence does not issue any of the above commands. On the other hand, when it fails on (when CFON occurs), it always presumes a positive load and consequently, has the system running regardless of whether a load is actually present or not.
- The controller does not fail during operation, i.e., if it does fail, the failures occur at the start of operation.
- The system-power-off command, FOFF, is issued when a time-out event occurs, for instance, at the end of a working day, but only if the controller senses no load on the system.

The system is assumed to have only one sensor, a valve flow sensor, whose outputs are F (flow) and NF (no flow).

The first step in building the complete system model is to perform the synchronous composition of the individual component models. The second step is to obtain the composite system from the synchronous composition and the sensor map for the flow sensor. This system has 104 states; 90 of these states form the accessible part of the synchronous composition while the rest are the new states introduced during the transition renaming process. Due to space limitations we do not list here the transition table for the composite system. We refer the interested reader to [28] for a complete listing of the table.

The initial state of the system is chosen to be (C1.V1.F1.P1.B1.L0). The only unobservable events in this system are the six failure events partitioned as follows:

$$\Sigma_{f1} = \{\langle \text{SO1} \rangle, \langle \text{SO2} \rangle\},$$
$$\Sigma_{f2} = \{\langle \text{SC1} \rangle, \langle \text{SC2} \rangle\},$$
$$\Sigma_{f3} = \{\langle \text{CFON} \rangle\},$$
$$\Sigma_{f4} = \{$$

The indicator events associated with these failure types are

$$I(\Sigma_{f1}) = \{\langle \text{CV}, \bullet \rangle\},$$
$$I(\Sigma_{f2}) = \{\langle \text{OV}, \bullet \rangle\},$$
$$I(\Sigma_{f3}) = \{\langle \text{CV}, \bullet \rangle\},$$
$$I(\Sigma_{f4}) = \{\langle \text{OV}, \bullet \rangle\}.$$

The diagnoser G_d^I of this system has 158 states. Fig. 12 illustrates part of G_d^I. Inspection of Fig. 12 reveals five loops, marked A, B, C, D, and E. Loops A and B represent the portion of the diagnoser corresponding to normal system operation. Loops C, D, and E highlight some of the more interesting and significant features of the system and the diagnoser. We discuss these features in the following paragraphs.

Let the observed event sequence be ⟨FON, NF⟩ ⟨SPD, NF⟩ ⟨OV, NF⟩. The event ⟨SPD, •⟩ denotes that there is no load on the system. The controller issues the open valve command, however. Knowledge of the system model leads one to the immediate conclusion that the controller has failed on. This information can be obtained from the diagnoser by noting that the state of the diagnoser resulting from the above event sequence is F_3-certain. Suppose next that the event following the above sequence is ⟨PON, NF⟩. The diagnoser now transitions into an F_2, F_3-certain state indicating that the valve is stuck closed. No further failures are possible in the system and the diagnoser continues to remain in loop D of F_2, F_3-certain states.

Suppose, on the other hand, that the events ⟨PON, F⟩ and ⟨BON, F⟩ follow the sequence ⟨FON, NF⟩ ⟨SPD, NF⟩ ⟨OV, NF⟩. The diagnoser now enters into loop C which is a loop of F_1-uncertain states. A careful examination of this loop and of the system model G, (not listed here) reveals that loop C is F_1-indeterminate. It is obvious by inspection, however, that loop C is not (F_1, I_1)-indeterminate since the label I_1 does not appear in the states constituting this loop. This situation has the following physical interpretation. When the controller fails on, the valve, pump, and boiler are always enabled and as long as the valve is not stuck closed, the valve sensor always indicates flow; the actual status of the valve may either be open-normal or stuck open and it is not possible to distinguish between these two states from observed event sequences. Since the close valve command is never issued by the controller when it fails on, however, all traces generated by the system, in which the controller failed-on event is followed by the valve stuck-open event, do not violate the definition of

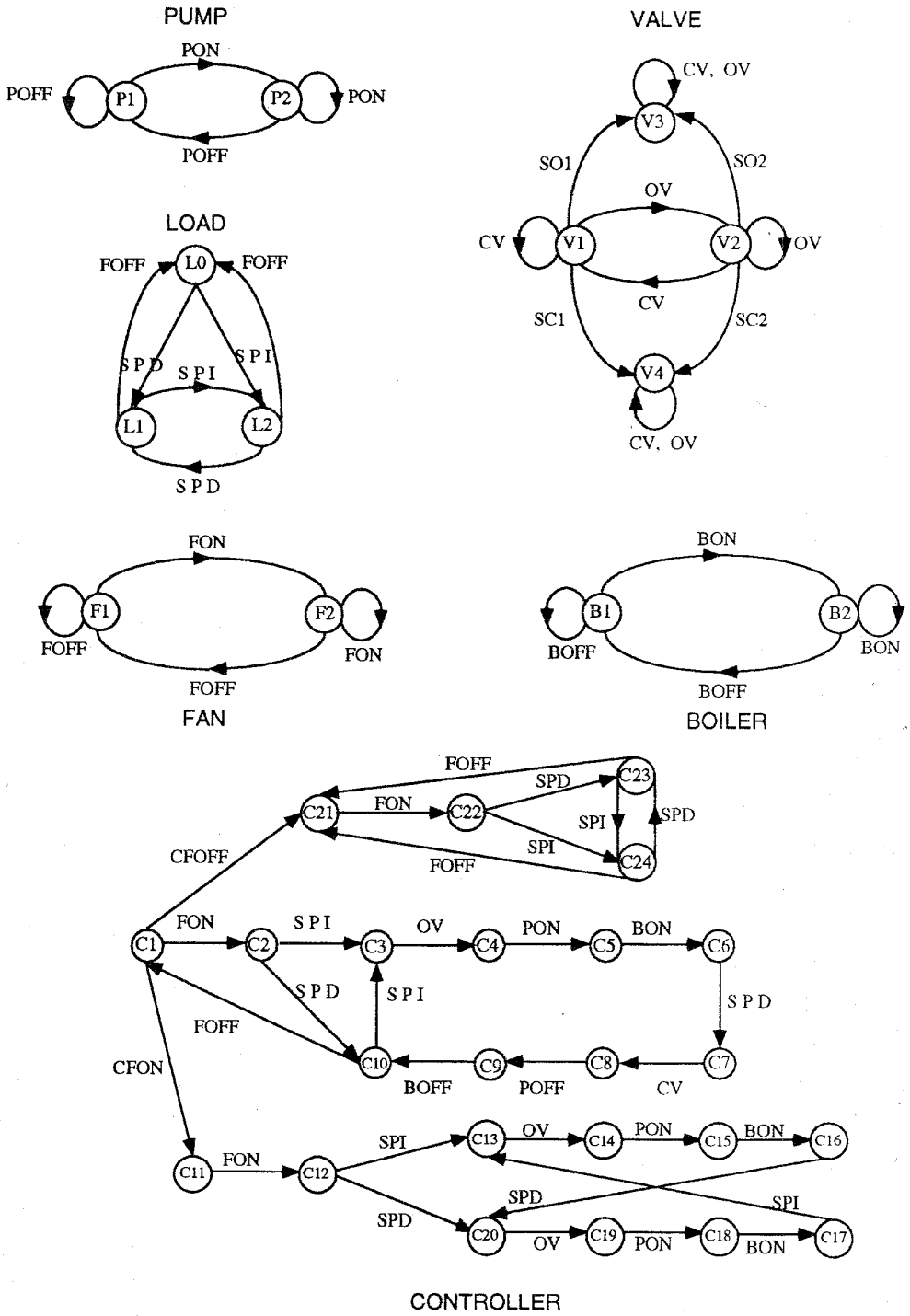


Fig. 11. Component models: HVAC System I.

I-diagnosability, because along these traces the corresponding indicator event does not follow the failure event. It is precisely this information that is provided by the diagnoser by the fact that loop C is not (F_1, I_1)-indeterminate.

Let us next consider the event sequence (⟨FON, NF⟩ ⟨SPD, NF⟩ ⟨FOF, NF⟩)*. The corresponding state sequence in the diagnoser (loop E) is a loop of F_i-uncertain states for $i = 1$, 2, and 4. Again, careful examination of the diagnoser and of the system model reveals that this is an F_i-indeterminate loop for $i = 1$, 2, and 4, which implies that it is not possible to determine if the controller has failed off or if any of the valve failures have occurred. This situation can be explained as follows. As long as the event sequence ⟨FON, NF⟩ ⟨SPD, NF⟩ ⟨FOFF, NF⟩ is observed, we know that there is no load on the system and hence the system is not activated. Hence, it is obvious that it would not be possible to diagnose occurrences

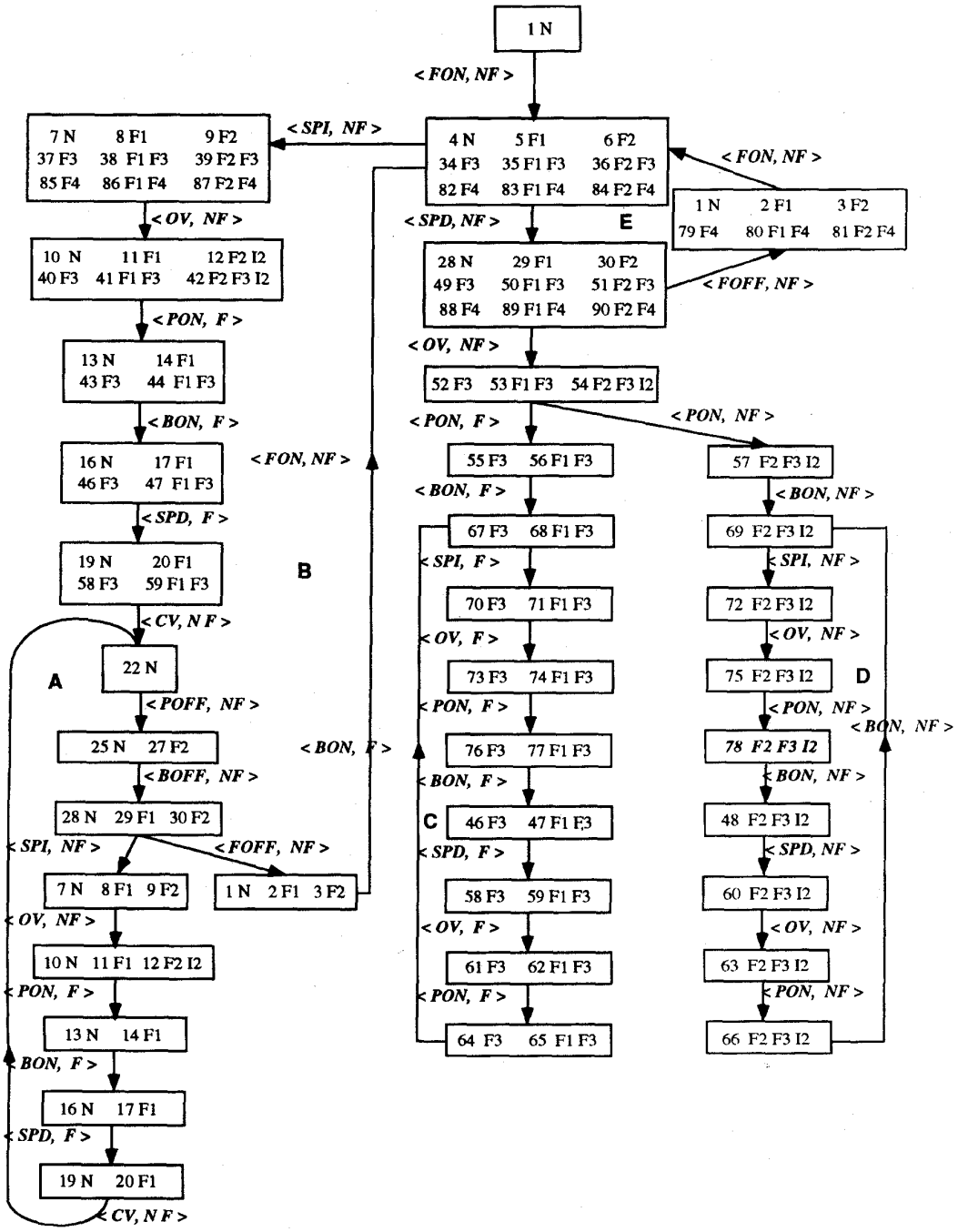


Fig. 12. Part of the diagnoser G_d^I for HVAC System I.

of the above mentioned failures. It is also clear that in this case, however, the definition of I-diagnosability is not violated since neither of the commands open valve and close valve that constitute the set of indicator events, is issued. This is reflected in the diagnoser by the fact that loop E is not (F_i, I_i)-indeterminate for any $i \in \{1, \cdots, 4\}$.

Examination of all other such cycles in the diagnoser (not shown in Fig. 12) reveals that there are no (F_i, I_i)-indeterminate cycles. We conclude therefore, by the results of Section III-C, that this system is I-diagnosable. With this knowledge, it would then suffice to implement the simpler G_d (and not G_d^I) for on-line diagnosis of the system. The diagnoser G_d has 97 states.

B. System II

Consider again the HVAC system of Fig. 10. We now assume that the valve and the pump can fail while the other

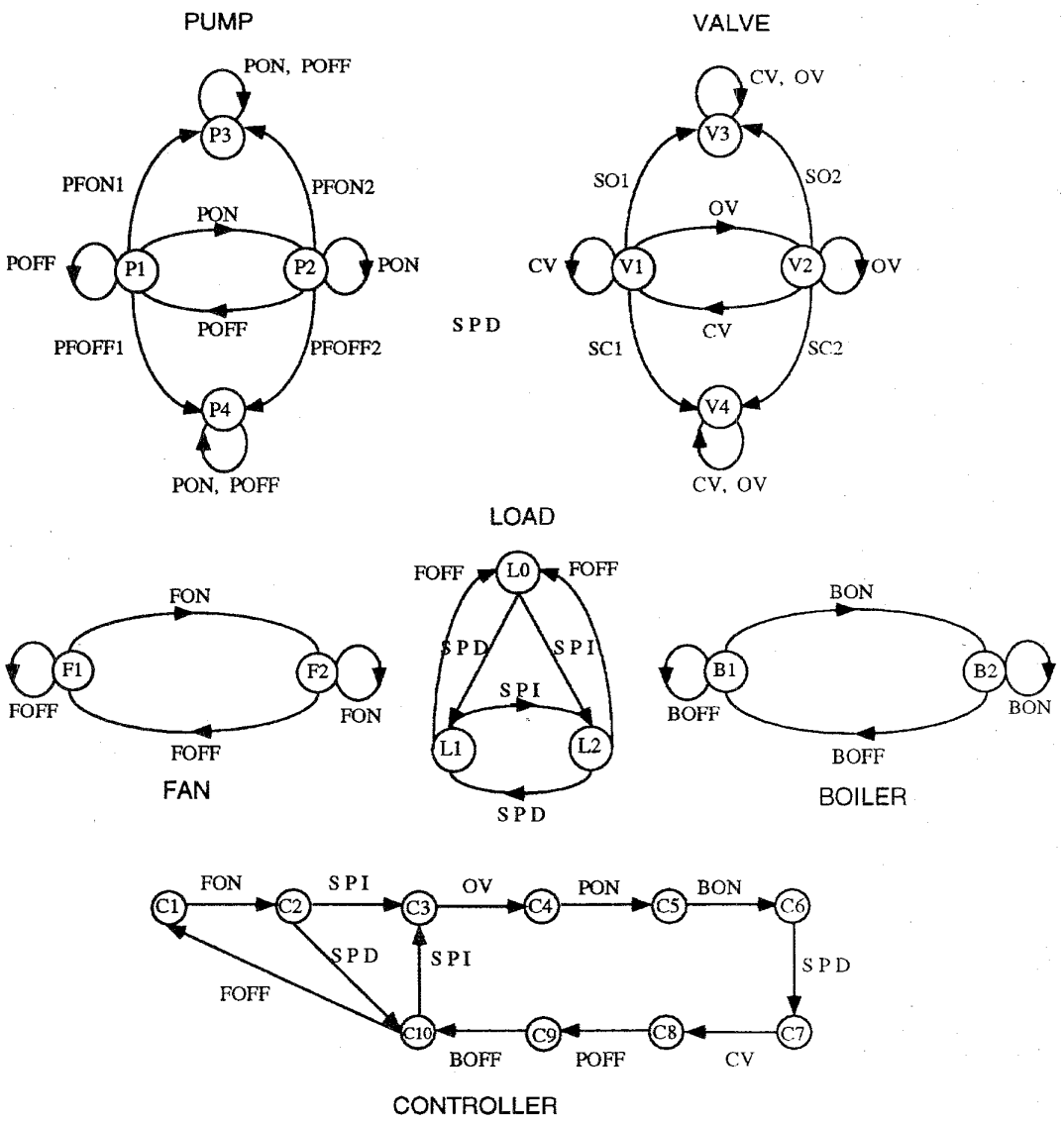


Fig. 13. Component models: HVAC System II.

components including the controller have no failure events. Fig. 13 illustrates the component models of this system. The models of the fan, valve, load, and the boiler are the same as before. The states P1 and P2 represent the normally off and on status of the pump while P3 and P4 denote the pump-failed-on and pump-failed-off status. The controller model is the same as before except for the fact that it now has no failure events.

The system is assumed to have two sensors: a pump pressure sensor, whose outputs are PPP (pump positive pressure) and PNP (pump no pressure), and a valve flow sensor, whose outputs are F (flow) and NF (no flow).

The composite system, obtained using the above component models and the sensor maps corresponding to the two sensors, has 130 states: 90 of these states are the accessible states of the synchronous composition of the component models shown in Fig. 13 and the remaining 40 are the new states introduced during the transition renaming process. Again, a complete listing of the transition table can be found in [28].

The initial state of the system, as before, is chosen to be (C1.V1.F1.P1.B1.L0). The only unobservable events in this system are the failure events of the pump and the valve. The partition Π_f and the corresponding indicator events are listed below

$$\Sigma_{f1} = \{\langle \text{SO1}\rangle, \langle \text{SO2}\rangle\},$$
$$\Sigma_{f2} = \{\langle \text{SC1}\rangle, \langle \text{SC2}\rangle\},$$
$$\Sigma_{f3} = \{\langle \text{PFON1}\rangle, \langle \text{PFON2}\rangle\},$$
$$\Sigma_{f4} = \{\langle \text{PFOFF1}\rangle, \langle \text{PFOFF2}\rangle\}$$

and

$$I(\Sigma_{f1}) = \{\langle \text{CV}, \bullet, \bullet\rangle\},$$
$$I(\Sigma_{f2}) = \{\langle \text{OV}, \bullet, \bullet\rangle\},$$
$$I(\Sigma_{f3}) = \{\langle \text{POFF}, \bullet, \bullet\rangle\},$$
$$I(\Sigma_{f4}) = \{\langle \text{PON}, \bullet, \bullet\rangle\}.$$

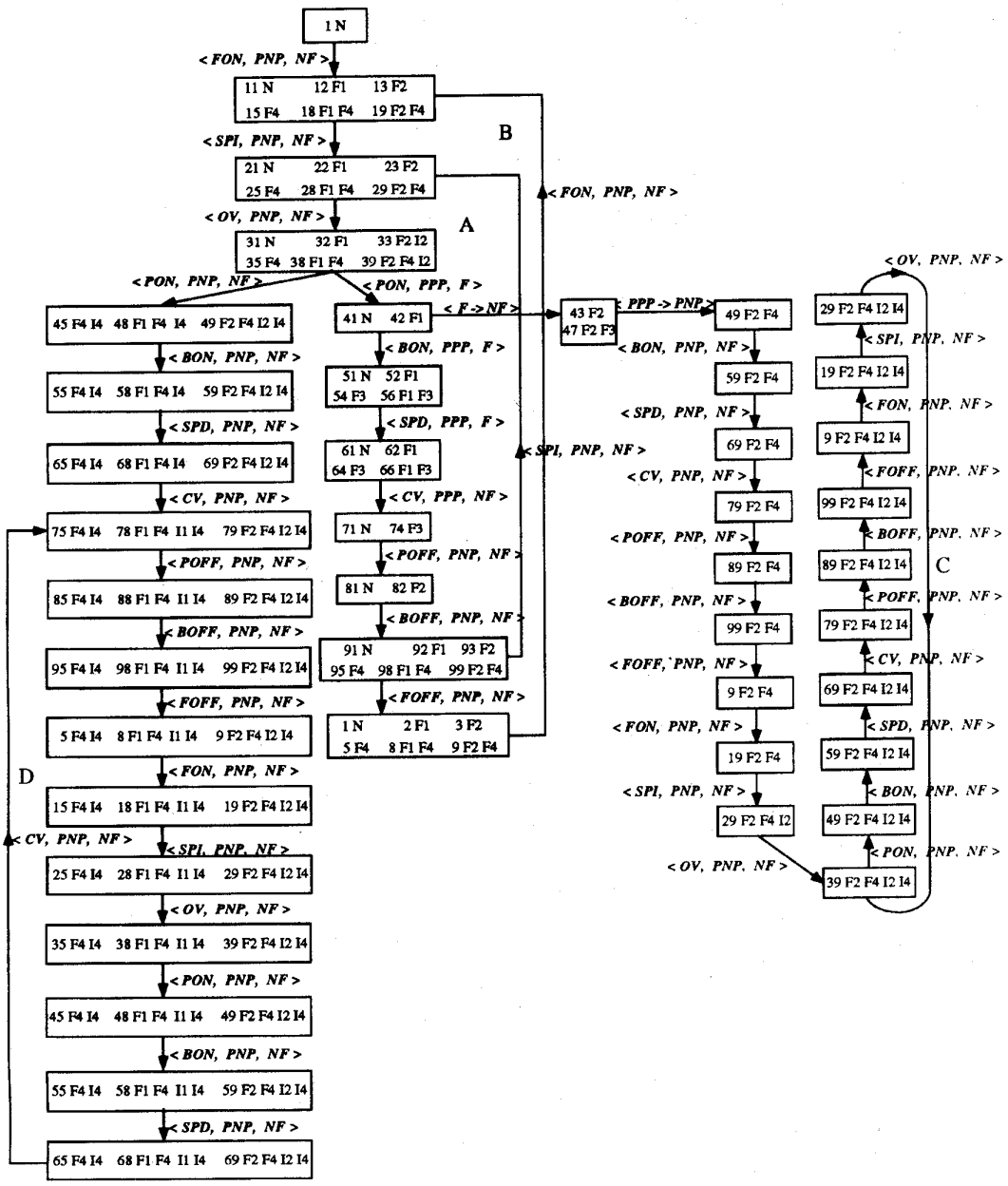


Fig. 14. Part of the diagnoser G_d^I for HVAC System II.

The diagnoser G_d^I for this system has 245 states. Fig. 14 illustrates a part of G_d^I. Loops A and B in Fig. 14 correspond to the normal operation of the system. It is interesting to note that these loops do not include any state such that all of the components of this state carry the N label corresponding to normal operation. We note, however, that all uncertainties regarding occurrences of failure events are resolved in a finite number of steps as one proceeds through these loops.

Suppose that the observed event sequence is ⟨FON, PNP, NF⟩ ⟨SPI, PNP, NF⟩ ⟨OV, PNP, NF⟩ ⟨PON, PPP, F⟩. This event sequence leads to the state $\{(41, \{N\}), (42, \{F1\})\}$ in the diagnoser. This implies that right after the above sequence of events is observed the system could be normal or the valve could be stuck open. If the next event observed is a change of the flow sensor reading from flow to no flow, the diagnoser immediately tells us that the valve is stuck closed; this is because the state of the diagnoser after the ⟨F → NF⟩ event is observed is F_2-certain. At this point, however, we do not know if the pump is normal or failed-on. If now we observe the event ⟨PPP → PNP⟩, the diagnoser hits a F_2, F_4-certain state and we know for sure that two failure events have occurred in the system: a valve-stuck-closed event and a pump-failed-off event. No further failures are possible after this and the diagnoser continues to remain in loop C of F_2, F_4-certain states.

Suppose next that the observed event sequence is ⟨FON, PNP, NF⟩ ⟨SPI, PNP, NF⟩ ⟨OV, PNP, NF⟩ ⟨PON,

PNP, NF⟩. A no-pressure reading of the pump sensor following the pump on command implies that the pump has failed off and correspondingly, the diagnoser transitions into a F_4-certain state. We note that the diagnoser then continues to remain in loop D which is both (F_1, I_1)-indeterminate and (F_2, I_2)-indeterminate, as revealed by a careful examination of the diagnoser and the system model. This implies that the system of Fig. 13 is not I-diagnosable. This result is intuitively obvious. When the pump fails off, irrespective of the commands issued to the valve, there will be no flow in the system and hence one can make no conclusions about the status of the valve. Therefore, the system in Fig. 13 with the pump pressure and valve flow sensors is not I-diagnosable.

Suppose now that the above HVAC system with pump and valve failures is equipped with the following set of sensors: a pressure sensor on the pump as before, and a valve stem position sensor. The latter has two possible outputs, UP and DOWN corresponding to the open (stuck-open) and closed (stuck-closed) positions of the valve. Construction of the diagnoser for this system reveals that it is I-diagnosable. This is because, irrespective of the status of the pump, it is now possible to detect the status of the valve; response of the stem position sensor to the valve commands, open valve and close valve, allows us to identify the status of the valve.

This example illustrates the following important fact: diagnosability of a system depends both on the partition Π_f and the projection P. Hence, for a given partition, it is possible to change the diagnosability properties of a system by altering the observable event set Σ_o which is equivalent to altering the set of sensors the system is equipped with.

Remark: The calculation described in Section IV regarding the construction of G, G_d, and G_d^I were performed using routines from the C library UMDES currently under development at the University of Michigan. The software package MEC [1] was also used for the synchronous composition operation.

V. Comparison with Other Approaches

In Section I, we listed several other approaches to failure diagnosis, namely, analytical model-based methods, fault trees, expert systems, and model-based reasoning methods. We now highlight the differences between these methods and our approach, focussing on the two main aspects of the failure diagnosis problem: 1) model building and 2) the diagnosis process. The differences between our approach and the analytical model-based methods as well as the expert systems approaches are quite evident. In the subsequent paragraphs, we concentrate on the comparison of our approach with fault-tree-based methods and the model-based reasoning methods for failure diagnosis.

On Model Building:

- Our approach is completely event-based and our modeling formalism is that of finite state machines, a formal class of models for DES that is amenable to composition (e.g., the synchronous composition) and analysis. In contrast, the models used for constructing fault-trees are variable based and they are typically represented in terms of digraphs, where nodes can represent either variables or certain failure events.
- Our approach separates local behavior from global behavior: local behavior is modeled in a completely modular manner by the individual component FSM models, which are then automatically composed by means of the synchronous composition; the sensor map however is global and captures component interactions. Thus, overall our approach builds the complete model in a "systematic" manner. In our opinion, the variable-based digraph models cannot be manipulated as easily and often lead to difficulties in going from local to global behavior (e.g., the problems of "negative feedback loops" [16], [22] and "apparent nonlocal causality" [22]).
- Our approach makes use of models of structure and behavior like the model-based reasoning methods. A wide variety of modeling formalisms have been proposed in the model-based reasoning literature, including FSM [23], [7], qualitative differential equations [9], [20], constraint equations [11], signed-digraphs [22], and so forth. In the work on model-based reasoning that we have seen, however, there is no systematic method to generate the complete system model that exploits modularity in the manner that we do.

On the Diagnostic Process:

- Our "diagnoser" is a formal dynamical model that is a "replica" of the system model G. Therefore, the state of the diagnoser can be easily updated recursively. Fault-trees are not really replicas of the underlying system models used to build them. This is also true for model-based reasoners.
- The entire process of failure-hypothesis generation, testing, and discrimination carried out by the model-based reasoners is "built-in" in the diagnoser. The hypothesis generation step of the reasoning process, which is based on comparing predictions from the model with actual observations, may be thought of as corresponding to the attachment of failure labels to the state estimates. The hypothesis generation, testing, and discrimination process may be thought of as transitions of the diagnoser from a normal state to an F_i-uncertain state and then to an F_i-certain state.
- Diagnosers are automatically synthesized from the system model. To the best of our knowledge, the automatic synthesis of fault-trees is not a completely resolved problem.
- The proposed approach deals with multiple failure occurrences just as well as single failure occurrences. This includes new failure occurrences during the diagnostic process. In the case of fault trees, it appears that multiple failures are more difficult to handle than single failures.

VI. Conclusion

We have considered the problem of failure diagnosis for large complex systems from the point of view of discrete-event systems. The underlying theory for our approach is developed in detail in [27]. The focus of the present paper has been on the modeling of systems for the purpose of diagnostics and on

the application of the theory in [27] to examples from HVAC units. We note here that the component models of the HVAC systems that we have used are generic to a lot of engineering systems. Further, the nature of the models discussed indicates the potential applicability of our approach to a wide class of systems.

There are two crucial issues regarding the applicability of our theory to HVAC units or other classes of systems: 1) building the system model and 2) dealing with the computational complexity of the diagnostics process. With regard to model building, we have proposed in this paper a systematic methodology for obtaining the complete system model from the (simpler) models of the individual components and from the information provided by the sensors. We emphasize that our methodology translates sensor information into events and thus all the information is captured in a pure event-based model. Building the individual component models, of course, may not be a trivial task and calls for knowledge of the application domain and engineering judgement in selecting the right level of abstraction. With regard to computational complexity, it is clear that since we are dealing with a partially observed system, in the worst case, the computational complexity of constructing diagnosers can be exponential in the size of the state space of the complete system model. This is an unavoidable feature of partial observation problems; however, three remarks are in order. First, our approach is scalable (up to computational limitations) because it is formal and systematic: given a library of individual component models that include normal and faulty behavior, everything else is solved algorithmically. Second, our experience so far, while limited in scope, tends to indicate that the system often has enough structure so that the worst case computational bounds may be rarely attained. Finally, for the task of on-line diagnosis of diagnosable systems, it is not necessary to store the complete diagnoser machine. It is sufficient to just remember its current state. Upon occurrence of an observable event, the new state of the diagnoser could be built on-line from its current state and the relevant part of the system model, with polynomial complexity at each stage.

ACKNOWLEDGMENT

K. Sinnamohideen would like to acknowledge Johnson Controls Inc. for their support, especially Dr. S. Bomba, Vice-President for Technology, for his encouragement and the opportunities he provided. The authors thank Profs. A. Arnold and A. Griffault of the LaBRI at Université Bordeaux I for providing a copy of MEC. Finally, the contributions of the members of the UMDES group, in particular, I. Siregar, Y.-L. Chen, and N. Ben Hadj-Alouane, for the development of the software programs used to generate the results in Section IV, are gratefully acknowledged.

REFERENCES

[1] A. Arnold, "MEC: A system for constructing and analyzing transition systems," in *Automatic Verification of Finite State Systems—Lecture Notes in Computer Science*, vol. 407, J. Sifakis, Ed. New York: Springer-Verlag, 1989, pp. 117–132.

[2] *The 1992 ASHRAE Handbook—Heating, Ventilating and Airconditioning Systems and Equipment*, Inch-Pound ed., Amer. Soc. Heating, Refrigerating, Airconditioning Eng., Inc., 1992.

[3] M. Basseville and I. Nikiforov, *Detection of Abrupt changes—Theory and Application.* Englewood Cliffs, NJ: Prentice-Hall, 1993.

[4] S. Bavishi and E. Chong, "Automated fault diagnosis using a discrete-event systems framework," in *Proc. 9th IEEE Int. Symp. Intell. Contr.*, 1994, pp. 213–218.

[5] P. Dague, P. Deves, P. Luciani, and P. Taillibert, "Analog systems diagnosis," in *Readings in Model–Based Diagnosis*, W. Hamscher, L. Console, and J. Kleer, Eds. San Mateo, CA: Morgan Kaufmann, 1992, pp. 229–234.

[6] R. Davis and W. Hamscher, "Model-based reasoning: Troubleshooting," in *Readings in Model–Based Diagnosis*, W. Hamscher, L. Console, and J. Kleer, Eds. San Mateo, CA: Morgan Kaufmann, 1992, pp. 3–24.

[7] D. Decoste, "Dynamic across-time measurement interpretation," in *Readings in Model–Based Diagnosis*, W. Hamscher, L. Console, and J. Kleer, Eds. San Mateo, CA: Morgan Kaufmann, 1992, pp. 255–261.

[8] R. De Vries, "An automated methodology for generating a fault tree," *IEEE Trans. Reliability*, vol. 39, pp. 76–86, 1990.

[9] D. Dvorak and B. Kuipers, "Model-based monitoring of dynamic systems," in *Readings in Model–Based Diagnosis*, W. Hamscher, L. Console, and J. Kleer, Eds. San Mateo, CA: Morgan Kaufmann, 1992, pp. 249–254.

[10] P. Frank, "Fault diagnosis in dynamic systems using analytical and knowledge based redundancy—A survey and some new results," *Automatica*, vol. 26, pp. 459–474, 1990.

[11] W. Hamscher, "Modeling digital circuits for troubleshooting," *Artificial Intell.*, vol. 51, no. 1-3, pp. 223–271, 1991.

[12] *Readings in Model-Based Diagnosis.* W. Hamscher, L. Console, and J. Kleer, Eds. San Mateo, CA: Morgan Kaufmann, 1992.

[13] D. Handelman and R. Stengel, "Combining expert systems and analytical redundancy concepts for fault tolerant flight control," *J. Guidance*, vol. 12, pp. 39–45, 1989.

[14] L. Holloway and S. Chand, "Time templates for discrete-event fault monitoring in manufacturing systems," in *Proc. 1994 Amer. Contr. Conf.*, pp. 701–706.

[15] S. Lafortune and E. Chen, "A relational algebraic approach to the representation and analysis of discrete-event systems," in *Proc. 1991 Amer. Contr. Conf.*, Boston, MA, pp. 2893–2898.

[16] S. Lapp and G. Powers, "Computer-aided synthesis of fault trees," *IEEE Trans. Reliability*, vol. 26, pp. 2–13, 1977.

[17] F. P. Lees, "Process computer alarm and disturbance analysis: Review of the state of the art," *Comput. Chemical Eng.*, vol. 7, no. 6, pp. 669–694, 1983.

[18] F. Lin, "Diagnosability of discrete-event systems and its applications," *J. DEDS*, vol. 4, no. 2, pp. 197–212, 1994.

[19] F. Lin, J. Markee, and B. Rado, "Design and test of mixed signal circuits: A discrete-event approach," in *Proc. 32nd Conf. Decision Contr.*, 1993, pp. 246–251.

[20] H. T. Ng, "Model-based, multiple fault diagnosis of time-varying, continuous physical devices," in *Readings in Model–Based Diagnosis*, W. Hamscher, L. Console, and J. Kleer, Eds. San Mateo, CA: Morgan Kaufmann, 1992, pp. 242–248.

[21] C. M. Özveren and A. S. Willsky, "Observability of discrete-event dynamic systems," *IEEE Trans. Automat. Contr.*, vol. 35, pp. 797–806, 1990.

[22] O. O. Oyeleye, F. E. Finch, and M. A. Kramer, "Qualitative modeling and fault diagnosis of dynamic processes by MIDAS," presented at the AIChE Spring Annu. Mtg., Houston, TX, 1989.

[23] J. Y. Pan, "Qualitative reasoning with deep-level mechanism models for diagnosis of mechanism failures," in *Proc. 1st IEEE Conf. AI Applicat.*, Denver, CO, 1984, pp. 295–301.

[24] A. D. Pouliezos and G. S. Stavrakakis, *Real-Time Fault Monitoring of Industrial Processes.* Norwell, MA: Kluwer, 1994.

[25] P. J. Ramadge and W. M. Wonham, "The control of discrete-event systems," *Proc. IEEE*, vol. 77, no. 1, pp. 81–98, 1989.

[26] S. Rich and V. Venkatasubramanian, "Model-based reasoning in diagnostic expert systems for chemical process plants," *Comput. Chem. Eng.*, vol. 11, pp. 111–122, 1987.

[27] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis, "Diagnosability of discrete event systems," *IEEE Trans. Automat. Contr.*, vol. 40, no. 9, pp. 1555–1575, 1995.

[28] ——, "Failure diagnosis using discrete-event models," Dep. EECS, Univ. Michigan, Tech. Rep. CGR 94-3, 1994.

[29] K. Sinnamohideen, "Discrete-event based diagnostic supervisory control system," presented at the AIChE Annu. Mtg., Los Angeles, CA, Nov. 17–22, 1991.

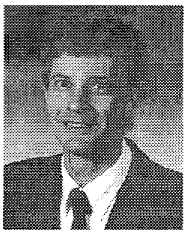
[30] W. F. Stoecker, *Design of Thermal Systems*, 3rd ed. New York: McGraw-Hill, 1989.
[31] L. Telksnys, Ed., *Detection of Changes in Random Processes*. New York: Optimization Software Inc., 1986.
[32] N. Ulerich and G. Powers, "On-line hazard aversion and fault diagnosis in chemical processes: The digraph + fault-tree method," *IEEE Trans. Reliability*, vol. 37, pp. 171–177, 1988.
[33] N. Viswanadham, "Control systems: Reliability" in *Systems and Control Encyclopedia: Theory, Technology, Applications*, M. G. Singh, Ed. New York: Pergamon, 1987.
[34] N. Viswanadham and T. L. Johnson, "Fault detection and diagnosis of automated manufacturing systems," in *Proc. 27th Conf. Decision Contr.*, Austin, TX, 1988, pp. 2301–2306.
[35] A. S. Willsky, "A survey of design methods for failure detection in dynamic systems," *Automatica*, vol. 12, pp. 601–611, 1976.

Meera Sampath (S'95) received the B.E. degree in electrical engineering from the College of Engineering, Guindy, Madras, India, and the M.Tech. degree in electrical engineering from the Indian Institute of Technology, Kharagpur, India, in 1988 and 1990, respectively. Since September 1991, she has been at the University of Michigan, Ann Arbor, where she recently received the doctoral degree in electrical engineering.

During the summer of 1995, she was an Intern at Johnson Controls, Inc., Milwaukee. Her current research interests include failure diagnosis, discrete-event systems, and intelligent transportation systems.

Raja Sengupta received the bachelor's degree in electrical engineering from Jadavpur University, Calcutta, India, in 1988 and the Ph.D. degree from the University of Michigan, Ann Arbor, in 1995.

He is currently working with the PATH program at the University of California at Berkeley. His research interests are in discrete-event systems, optimization, failure diagnosis, intelligent transportation systems, and automated highway systems.



Stéphane Lafortune (S'78–M'86) received the B.Eng. degree from the École Polytechnique de Montréal, Canada, in 1980, the M.Eng. degree from McGill University, in 1982, and the Ph.D. degree from the University of California at Berkeley in 1986, all in electrical engineering.

Since 1986, he has been with the Department of Electrical Engineering and Computer Science at the University of Michigan, Ann Arbor, where he is an Associate Professor. He was also Invited Professor at École Polytechnique de Montréal in 1993. His research interests are in discrete-event systems (modeling, supervisory control, failure diagnosis, and applications) and in intelligent transportation systems.

Dr. Lafortune received the Presidential Young Investigator Award from the National Science Foundation in 1990 and the George S. Axelby Outstanding Paper Award from the IEEE Control Systems Society in 1994.

Kasim Sinnamohideen (M'68) received the B.Sc. degree in physics from the University of Madras, India, in 1959 and the M.S. and Ph.D. degrees in mechanical engineering from Purdue University, West Lafayette, IN, in 1962 and 1968, respectively.

Since 1968, he has been with Johnson Controls Inc., Milwaukee, WI, where he has held various positions and participated in wide areas of research. He was a pioneer in 1972, in developing methods that are being practiced extensively in the industry today for modeling heating, ventilating, and air conditioning systems and applying them to conserve energy in buildings. He has developed several diagnostics and configuration type expert systems and knowledge based real-time monitoring systems. He has worked in the areas of performance modeling of computer systems and polymer processing. He was a Visiting Scholar in Electrical Engineering and Computer Science Department at the University of Michigan during 1992–93. His current interests are in modeling and analysis of manufacturing systems and monitoring theory.

Demosthenis C. Teneketzis (M'87) received the B.S. degree in eletrical engineering from the University of Patras, Greece, in 1974 and the M.S., E.E., and Ph.D. degrees in electrical engineering from the Massachusetts Institute of Technology, Cambridge, in 1976, 1977, and 1979, respectively.

From 1979 to 1980, he worked for Systems Control Inc., Palo Alto, CA, and from 1980 to 1984 he was with Alphatech Inc., Burlington, MA. Since September 1984, he has been with the University of Michigan, Ann Arbor, where he is a Professor of Electrical Engineering and Computer Science. During the winter and spring of 1992, he was a Visiting Professor at the Institute for Signal and Information Processing of the Swiss Federal Institute of Technology (ETH), Zurich, Switzerland. His current research interests include stochastic control, decentralized systems, queueing and communication networks, stochastic scheduling and resource allocation problems, and discrete-event systems.