A Discrete Event Systems Approach to Failure Diagnosis

by

Meera Sampath

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Electrical Engineering:Systems)
in The University of Michigan
1995

Doctoral Committee:

Associate Professor Stéphane Lafortune, Co-Chair
Professor Demosthenis Teneketzis, Co-Chair
Professor Pramod Khargonekar
Associate Professor Feng Lin, Wayne State University
Dr. Kasim Sinnamohideen, Johnson Controls, Inc.
Assistant Professor Mike Wellman

Dedicated to

my

parents

ACKNOWLEDGEMENTS

My sincere thanks and gratitude are due to Profs. Lafortune and Teneketzis for their valuable guidance, support, and encouragement. Working with them on this thesis has been a very rewarding and pleasurable experience that has greatly benefited my education. My special thanks are due to Prof. Lafortune for providing me with financial support throughout my graduate studies at the University of Michigan. I wish to express my thanks to Prof. Khargonekar, Prof. Lin, and Prof. Wellman, and Dr. Sinnamohideen for their consent to be on my committee and for their interest in my work. I acknowledge with gratitude the several invaluable discussions that I have had with Dr. Sinnamohideen whose ingenuity and insight led to the conception of this thesis. I also wish to express my thanks to Dr. Raja Sengupta for his active participation and involvement in this work. My sincere thanks to Irwan Siregar, Kin-Yip Sit, Yi-Liang Chen, and Dr. Nejib Ben Hadj-Alouane, the members of the discrete event systems software group, for their effort and help in developing the UMDES-LIB software library without which most of the examples in this thesis and my work at Johnson Controls, Inc. would not have been possible.

I recall and remember with affection and gratitude my association with Prof. Rajagopalan at IIT, Kharagpur who was instrumental in inspiring me to pursue my graduate studies in the U.S. I am indebted to Prof. Kan Chen for providing me with the initial financial support to begin my studies at the University of Michigan. I sincerely acknowledge all the help that Laura, Chuck, Don, and Hugh of the DCO, Ann, Beth, Carol, and Linda of the Systems Division, and Cathy at the ITS Center of Excellence, have given me during the course of my graduate studies. I am thankful to Prof. Freudenberg for giving me the opportunity to be his teaching assistant, an experience I consider a valuable part of my graduate education. I also wish to express my thanks to Ms. Carol Lomonaco for providing me with the opportunity to work as an intern at Johnson Controls, Inc which helped me broaden my experience.

My very special thanks to my friends, Anna, Srivathsan, Natraj, Devina, Ravi, Paul, and Raja, whose friendship has made my stay at Ann Arbor an unforgettable experience and one of the best periods of my life. My deepest gratitude and affection are due to my mother, without whose support and encouragement I could not have done most of the things I have enjoyed doing. I remember with fondness my father, who I hope would have been proud to see this thesis. I extend my special thanks to my brothers, Rajan and Balaji, and to my sister, Priya, for their affection and support. Finally, I wish to thank my husband, Siva, who has been a constant source of inspiration and encouragement.

TABLE OF CONTENTS

LIST OF TABLES

Table

LIST OF FIGURES

Figure

LIST OF APPENDICES

APPENDIX

CHAPTER 1

Introduction

Law: If anything can go wrong, it will.
Corollary: Left to themselves, things tend to go from bad to worse.

... The legendary Murphy

1.1 The Problem of Failure Diagnosis

Failure detection and isolation in industrial systems is a subject that has received a great deal of attention in the past few decades. A failure[1] is defined to be any deviation of a system from its normal or intended behavior; diagnosis is the process of detecting an abnormality in the system behavior and isolating the cause or the source of this abnormality. Failures in industrial systems could arise from several sources such as design errors, equipment malfunctions, operator mistakes, and so on. There are three major factors that motivate the study of the failure diagnosis problem:

1. failures are inevitable;
2. failure diagnosis is important (if not crucial); and
3. failure diagnosis is difficult.

In the paragraphs that follow we will examine briefly each of the above three factors.

Failures are inevitable in today's complex industrial environment. As technology advances, as we continue to build systems of increasing size and functionality, and as we continue to place increasing demands on the performance of these systems, then so do we increase the complexity of these systems. Consequently (and unfortunately), we enhance the potential for systems to fail, and no matter how safe our designs are, how improved our quality control techniques are, and how better trained the operators are, system failures become unavoidable. Indeed, given the complex interactions between components,

[1] Often, the term *failure* is used to denote a complete operational breakdown, whereas the term *fault* is used to denote any abnormal change in behavior; in this thesis we will use the two terms synonymously.

sub-systems, and proccesses, a system failure can well be considered to be a "normal" occurrence [35], or, an inherent characteristic of most industrial systems.

Given the fact that failures are inevitable, the need for effective means of detecting them is quite apparent if we consider their consequences and impacts not just on the systems involved but on the society as a whole. Several of the major industrial disasters and accidents in the past, such as the Three Mile Island accident, the explosion at the Texas petrochemical plant, the major blackout of New York city, and the Appollo 13 incident have had their origin in a leaky valve, a failed relay, or a burnt-out switch [35]. Timely and accurate detection of these failures may well have prevented the cascaded effect that these "simple" failures produced, resulting thereby in system-wide breakdowns leading ultimately to these major accidents that we have come to know and fear. Safety and reliability are thus two prime factors motivating the study of this problem, especially, in the case of high risk potential systems as the ones mentioned above; system availability is an additional motivating factor when we consider systems such as power plants and petrochemical industries where a shutdown due to failure could lead to major disruption of services and have serious economic impacts. Finally, we note that effective methods of failure diagnosis can not only help avoid the undesirable effects of failures, but can also enhance the operational goals of industries. Improved quality of performance, product integrity and reliability, and reduced cost of equipment maintenance and service are some major benefits that accurate diagnosis schemes can provide, especially for service and product oriented industries such as home and building environment control, office automation, automobile manufacturing, and semiconductor manufacturing. Thus, we see that accurate and timely methods of failure diagnosis can enhance the safety, reliability, availability, quality, and economy of industrial processes.

The complex nature of industrial systems not only increases the potential for failures, but also, makes the diagnosis problem more difficult and challenging. Most systems are equipped with indicators such as alarms, warning lights, and so on, to indicate the status of the system to operators monitoring its behavior, and aid them in their diagnostic reasoning. However, cost and technological feasibility limit the number of sensors and hence the number of system variables that can be directly monitored. Thus information about the state of the system that operators (or technicians) have are indirect and they have to *infer* the diagnosis from mental models of the process. Given the complex and often non-apparent interactions and coupling between system components, such inferencing becomes a very challenging task, especially if decisions have to be made fast. Finally, even if one had all the necessary sensors available, the problem remains as difficult. Assimilating the data provided by a large number of, often, seemingly contradictory sensors (especially under faulty situations), and correlating this data with possible failures is by no means a trivial task. This sometimes results in operators ignoring alarms. Further, there have been in the history of industrial accidents, cases of missed/misread indications, and faulty interpretations of sensor information by operators [35].

In view of the above mentioned factors, it is not difficult to see that *automated mechanisms for the timely and accurate diagnosis of failures are very essential*. Indeed, this need is well understood and appreciated both in industry and in academia. A great deal of research effort has been and is being spent in the design and development of automated diagnostic systems, and a variety of schemes, differing both in their theoretical framework and in their design and implementation philosophy, have been proposed.

From the conceptual view-point most existing methods of failure diagnosis can be classified as (i) fault-tree based methods; (ii) quantitative, analytical model-based methods; (iii) expert systems and other knowledge-based methods; (iv) model-based reasoning methods; and (v) discrete event systems (DES) based methods. From the implementation standpoint these diagnostic systems can be classified as off-line or on-line. Off-line methods assume that the system is in a testbed and is to be tested for possible prior failures, while in on-line diagnosis, the system is assumed to be operational and the diagnostic subsystem is designed so as to continuously monitor the system behavior, identify and isolate failures. In the following section, we provide a brief overview of some of the more popular schemes for failure diagnosis. A detailed discussion of several of these methods has appeared in [37] (see especially the introduction section). Also discussed in [37] are some other approaches to failure detection based on statistical decision theory, control charts, and vibration and noise analysis. We refer the reader to [11] for a good review of the knowledge-based approaches including expert systems and model-based reasoning methods. An overview of the salient features of the aforementioned methods can also be found in [44].

1.2 Approaches to Failure Diagnosis

In a vast majority of industrial systems fault detection and isolation is based on simple limit checking of absolute values or of trends of measurable system variables with threshold logic; alarms are used to indicate deviations of the monitored variables from preset limits. Failure propagation in the system, in general, leads to a large number of alarms being set off in rapid succession. The problem then is to identify the source and location of the failure from the alarm indications. This process is normally referred to as alarm analysis or failure propagation. The most widely used scheme for alarm analysis, especially in the process control industry, is based on *fault trees* [23], [24], [54], [57], [56]. Fault trees provide a graphical representation of cause-effect relationships of faults in a system. Starting from a goal violation, or, a system failure event that is indicated by an alarm condition, a fault tree is built by reasoning backwards from the system failure to basic or *primal* failures that represent the root cause of the failure. The primary drawbacks of this approach are: (i) fault trees require a great deal of effort in their construction; and (ii) they pose difficulties in handling feedback systems [45], [30]. Though there has been a lot of research on automatic synthesis of fault trees from system models [57], [23], [36], this is not a completely resolved problem to the best of our knowledge.

A vast majority of the approaches to failure diagnosis proposed in the control systems literature (see [15] [55] [58] and references therein; see also [2] and [51]) are based on *analytical redundancy*. The analytical redundancy method can roughly be divided into two major steps: (i) generation of residuals and (ii) decision and fault isolation. The residual generation process typically involves generating residual signals by comparing predicted values of system variables (from mathematical models of the system) with the actual observed values. These signals are nominally near zero and get accentuated when failures do occur. In the decision and fault isolation stage, the residuals are examined for the likelihood of faults; Appropriate decision functions as, for example, likelihood ratio functions, formed out of the residuals are used in a fault decision logic where fault signatures obtained from a fault model are used to isolate the failure. A major advantage of this approach is the ability to detect, not only abrupt faults (or *hard* failures) but also slowly developing (or *incipient*) faults via trend analysis. The primary drawbacks of this approach are the computational expenditure for the detailed on-line modeling of the process, and more importantly, the sensitivity of the detection process with respect to modeling errors and measurement noise. The issue of robust failure detection using analytical models has been and is being investigated in detail.

Failure diagnosis by expert systems is a popular method in use (see [45] and references therein). This approach is eminently suited for systems that are difficult to model, i.e., systems involving subtle and complicated interactions (among and within components) whose outcomes are hard to predict. Traditional expert systems for diagnosis are rule-based systems in which the heuristic knowledge of experts is captured in the form of empirical associations which relate symptoms to the faults that produce them. The chief drawback of expert systems is that a considerable amount of time may elapse before enough knowledge is accumulated to develop the necessary set of heuristic rules for reliable diagnosis, coupled with the fact that this approach is very domain dependent, i.e., expert systems are not easily portable from one system to another. Further, it is difficult to validate an expert system. We note here that methods of failure diagnosis which combine the analytical model-based methods discussed above with the expert systems approach have also been proposed (see, e.g., [15], [18]). Also we note that many expert systems are based on structures and techniques relating to fault trees [45], [36].

A relatively new approach to failure diagnosis that is being investigated in the AI literature is that of model-based reasoning. The fundamental paradigm of this approach, much like the analytical redundancy methods, is that of observation and prediction [11]. These model-based methods employ a general purpose model of the *structure* and *behaviour* of the system which are constructed using standard AI technology such as predicate logic, frames, constraints, and rules. The algorithms for diagnosis are also based on standard techniques in AI, like theorem proving, heuristic search, constraint satisfaction, and qualitative simulation. In general, the model based methods deal only with models of correct behavior. There is no *a priori* specification of how components might fail, and a failure is taken to be any anomaly as compared to normal behaviour. This method of diagnosis is highly suitable for

troubleshooting analog and digital circuits. However, applicability of this method in other areas, especially for dynamic systems, is yet to be fully demonstrated. Most of the model-based reasoners that have been proposed for failure diagnosis of dynamic systems (see, e.g., [10], [13], [12], [17], [29], [30], [33], [40], [9], [4], [14] [50]) are based on the general diagnostic formalisms proposed for static systems. In some of these approaches, the system behavior is predicted using qualtitative differential equations while in others constraint equations normally used to describe static behavior (such as those based on predicate calculus) are augmented to explicitly include time. Though several approaches such as the ones mentioned above have been proposed, model-based reasoning for dynamic systems still remains an active research area.

We conclude this section with a brief summary of the approaches to failure monitoring and diagnosis that have been studied in the recent DES literature. In [25] and [26], the authors propose a state-based DES approach to failure diagnosis. The problems of off-line and on-line diagnosis are addressed separately and notions of diagnosability in both of these cases are presented. In off-line diagnosis, the system to be diagnosed is not in normal operation and can be thought of as being in a "test-bed". The diagnostic procedure involves issuing a sequence of test commands, observing the resulting outputs, and drawing inferences on the set of possible states the system could be in. Thus, the off-line diagnosis problem can be considered equivalent to the problem of "verification". In on-line diagnosis, the system is assumed to be in normal operation. The goal here, as in the case of off-line diagnosis, is to issue a sequence of commands and identify uniquely, up to a partition, the state of the system. However, unlike the case of off-line diagnosis, one now has to account for the possible occurrences of other uncontrollable events during the diagnostic process. The authors give an algorithm for computing a diagnostic control, i.e., a sequence of test commands for diagnosing system failures. This algorithm is guaranteed to converge if the system satisfies the conditions for on-line diagnosability.

Extensions of the above work can be found in [3] where the authors study testability of DES (which is equivalent to the off-line diagnosability problem studied in [25]) and present algorithms, (i) for determining the optimal set of sensors which would ensure testability of a given system and (ii) for determining, given a fixed set of sensors, the infimal partition of the state space with respect to which the system is testable.

In [19], the authors present an approach to fault monitoring in manufacturing systems characterized by occurrences of an unspecified number of concurrent event sequences. They propose the method of template monitoring which uses timing and sequencing relationships to determine if events expected to happen do occur and to detect inappropriate, or out-of-sequence, occurrences of events. A major advantage of the template monitoring scheme is cited to be the ease of its implementation on distributed control architectures.

In [56], the authors propose a Petri net based method for failure detection and diagnosis in manufacturing systems. They propose the use of Petri net models of the system for the purpose of failure detection and fault trees for failure isolation.

This concludes our discussion on approches to failure diagnosis that have appeared in the literature. Each of the above methods possesses certain advantages and disadvantages, and is best applicable under specific circumstances. Which of these approaches one selects for a given system depends not only on the characteristics of the system and the knowledge available about the system but also on the nature of the failures one wants to diagnose. Finally, we remark that several of these methods are complementary. With this background on the importance of the failure diagnosis problem and the various methodologies to solve this problem, we now pass on to the approach to diagnosis that this thesis proposes. In the sections that follow we highlight the main features and the scope of the proposed approach and also outline the main contributions of this thesis.

1.3 The Proposed Approach to Failure Diagnosis

We propose in this thesis a new approach to failure diagnosis in the framework of discrete event systems (DES). This work builds on the DES-based approach to diagnosis that was first proposed in [46]. DES are dynamic systems whose behavior is governed by the occurrence of physical events that cause abrupt changes in the state of the system. These systems are characterized by a discrete state space of logical values and event driven dynamics. (For a tutorial introduction to the study of DES we refer the reader to [6].) Most large scale dynamic systems can be viewed as DES at some level of abstraction. Such abstraction can be done for the purpose of supervisory control, or, for the purpose of failure diagnosis, as in our case. Therefore, the proposed method of fault diagnosis is applicable not only to systems that fall naturally in the class of DES (communication networks, computer systems, and manufacturing 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 is ideally suited for the diagnosis of large complex systems like heating, ventilation and air conditioning (HVAC) units, power plants, semiconductor manufacturing processes and automobile manufacturing. Further, this approach to diagnosis is appropriate for "sharp" failures that involve significant changes in the status of the system components but do not necessarily bring the system to a halt. Examples of such failures are stuck failures of valves, stalling of actuators, clogging of filters, bias failures of sensors, overflow of output buffers, loss of packets in a communication network, and so on. We note that our emphasis in this work is on on-line diagnosis of system failures.

1.3.1 Conceptual System Architecture

Figure 1.1 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

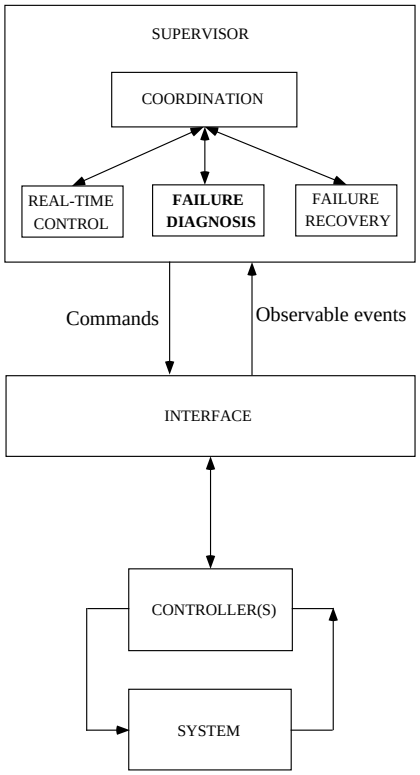


Figure 1.1: The conceptual system architecture

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

1.3.2 Overview of the Approach

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 models that we use are *logical* models (see, e.g., [6]) that describe the system behavior simply by the order in which events occur, without specific references to the times of occurrences of these events. In this regard, our approach is based on *finite state machines* (FSMs), (see, e.g., [39]) and *formal languages* (see, e.g., [20]). 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 an FSM built from the system model. This machine performs diagnosis when it observes on-line the behavior of the

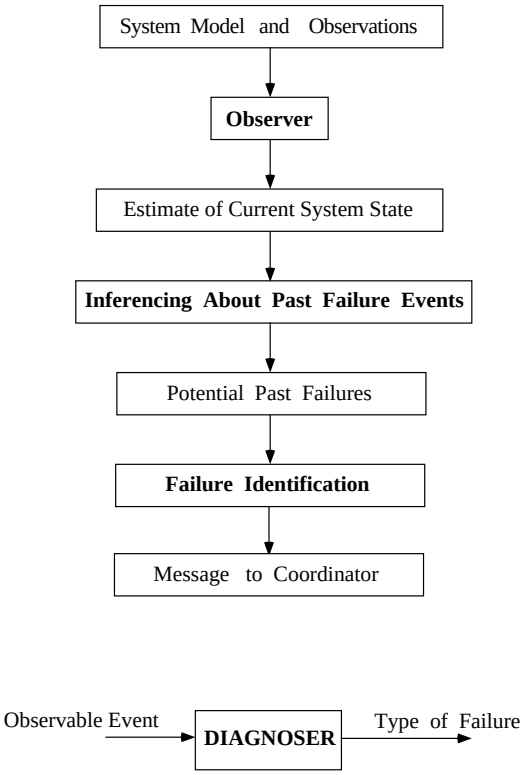


Figure 1.2: The Diagnostic Process

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. Figure 1.2 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 Figure 1.2. We note that this method does not make any assumptions on the number of failures; it is general enough to accomodate multiple system failures.

1.3.3 Scope of the Approach

One of the important factors that distinguishes our work from most prior work in this area is the following: the framework in which we formulate and study the failure diagnosis problem allows us not only to provide a methodology for detection and isolation of failures, but it also allows us to analyze the *diagnosability* properties of a system. More specifically, it allows us to answer the following question: "given a system, and given a set of diagnostic requirements, can this system meet the requirements, i.e., will it be possible to diagnose the failures we are interested in?" Most other approaches stop with providing a methodology for performing diagnostics; indeed, some of them (e.g., fault trees and expert systems) are not even set in a theoretical framework that will permit such analysis. In this thesis, we

formulate precisely the notion of diagnosability of a system; simply speaking, a system is said to be diagnosable if it is possible to detect with finite delay occurrences of failures in the system from a record of its observed behavior. We also establish necessary and sufficient conditions for a system to be diagnosable and discuss how these conditions can be checked by using the diagnoser. We show that diagnosability of a system can be determined by checking for certain special cycles in the diagnoser called *indeterminate cycles*. Note that the diagnoser thus serves two purposes: it can be used to verify the diagnosability properties of a system *and* it can be used to perform failure diagnosis.

The ability to formally analyze the diagnosability properties of a system has important design implications as explained below. In most industrial systems, design of the monitoring and diagnostic subsystem is done *after* the initial system design, and the diagnostic subsystem is added-on as a separate module to the existing system. In other words, diagnosability considerations are often not explicitly taken into account in the system design, quite probably due to the fact that there is no systematic way of judging how a specific design will affect the diagnosability properties of a system. Characterising the notion of diagnosability and establishing the conditions under which a system will meet the given diagnostic requirements, allows us to incorporate diagnostic considerations in a systematic fashion into the system design and to build *diagnosable* systems. Two possible approaches to building diagnosable systems are as follows: (i) by determining the set of sensors necessary for diagnosis and (ii) by appropriate design of the system controller. In this thesis, we study the latter approach. The need to take diagnosability considerations into account right at the design stage is more strongly brought forth in this case; as we shall see later in this thesis, diagnosability of a system often depends on the control protocol used, and decoupling the design of the controller and that of the diagnostic system can significantly affect the diagnosability properties of a system. In this thesis, we propose an integrated approach to control and diagnosis (which we refer to as the *active diagnosis* problem), and we provide a systematic procedure for designing supervisory controllers [6] that ensure that the system not only meets its desired control objectives but also is diagnosable.

To summarize, the proposed approach provides:

1. a systematic procedure to implement on-line failure diagnosis;
2. a framework to study the diagnosability properties of a system; and
3. a systematic approach to building diagnosable systems.

1.4 Organization of this Thesis

This thesis is organized as follows. In Chapter 2 we present a methodology for building discrete event models of physical systems for the purpose of failure diagnosis. In Chapter 3 we study the property of diagnosability of systems. We present two related definitions,

namely, diagnosability and I-diagnosability in the framework of formal languages. We introduce the diagnoser, present the necessary and sufficient conditions for diagnosability and I-diagnosability of systems, and show how the diagnoser can be used to test for diagnosability. In the next chapter we show how the diagnoser can be used to perform on-line failure diagnosis. In Chapter 5 we illustrate application of the theory developed in the preceding chapters to failure diagnosis in HVAC systems. In Chapter 6 we present the problem of active diagnosis. Finally, in Chapter 7 we summarize the main contributions of this thesis, compare the proposed approach to some of the other approaches to the failure diagnosis problem that we discussed earlier in this chapter, and outline directions for future research.

We note here that the main results presented in this thesis were first presented in [43], [44], and [42]. We also note that all of the computations for the examples presented in this thesis, such as generating the global system model, building the diagnoser, checking for diagnosability, and so on, were performed using routines in the UMDES-LIB software library that has been developed by the discrete event systems software group at the University of Michigan.

CHAPTER 2

Modeling for Failure Diagnosis

The justification of a mathematical model is solely and precisely that it is expected to work.

... John von Neumann

2.1 Introduction

In this chapter, we present a methodology for modeling physical systems in a DES framework. We assume that the system of interest consists of several distinct physical components and is equipped with a set of sensors. Starting from discrete event models of the individual components and from discrete-valued sensor maps we present a systematic procedure for generating a global model for the system that captures the interaction among the components and also incorporates in it the information provided by the sensors. This model accounts for both the normal and the failed behavior of the system. Failures are treated as special unobservable events in the model, and in the subsequent chapters of this thesis we will pose failure diagnosis as the problem of inferring about past occurrences of these failure events based on observed event sequences. We now present a four-step procedure for building the global system model and illustrate this procedure with examples. Finally, we conclude with some general remarks on the methodology.

2.2 Methodology for Discrete Event Model Building

Suppose that the system to be diagnosed has N individual components; typically, these components consist of equipment and controllers. The first step of our modeling methodology is to build FSM models for these components. Let

$$G_i = (X_i, \Sigma_i, \delta_i, x_{0i}) \tag{2.1}$$

refer to the FSM model of the i^{th} 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 i^{th} 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., [22]). 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,

$$\begin{aligned}
\Sigma &= \Sigma_1 \cup \Sigma_2 \\
X &= X_1 \times X_2 \\
x_0 &= (x_{01}, x_{02}) \\
\delta(\sigma, (x_1, x_2)) &= \begin{cases} (\delta_1(\sigma, x_1), \delta_2(\sigma, x_2)) & \text{if } \sigma \in e_1(x_1) \cap e_2(x_2) \\ (\delta_1(\sigma, x_1), x_2) & \text{if } \sigma \in e_1(x_1) \Leftrightarrow \Sigma_2 \\ (x_1, \delta_2(\sigma, x_2)) & \text{if } \sigma \in e_2(x_2) \Leftrightarrow \Sigma_1 \\ \text{undefined otherwise.} \end{cases}
\end{aligned}$$

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

$$\tilde{G} = (\tilde{X}, \tilde{\Sigma}, \tilde{\delta}, \tilde{x}_0) \tag{2.2}$$

denote the synchronous composition of the component models $G_i,\ i = 1, \ldots, N$. Observe that we need only consider the accessible part[2] of $\tilde{G}$. $\tilde{G}$ then models the joint operation of these components. Here,

$$\tilde{X} \subseteq \prod_i X_i \ \text{ and } \ \tilde{\Sigma} = \bigcup_i \Sigma_i. \tag{2.3}$$

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

$$Y = \prod_{j=1}^{M} Y_j \tag{2.4}$$

---

[2]The accessible part of $G = (X, \Sigma, \delta, x_0) = (X_A, \Sigma, \delta_A, x_0)$ where $X_A = \{x \in X : \delta(x_0, s) = x \text{ for some } s \in \Sigma^*\}$ and δ_A is the restriction to $X_A \times \Sigma$ of δ.

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

$$h(x) = (h_1(x), h_2(x), \ldots, h_M(x)). \tag{2.5}$$

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 $\tilde{\delta}(x, \sigma) = x'$ 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 $< \sigma, h(x') >$ and let $\delta(x, < \sigma, h(x') >) = x'$. The new event $< \sigma, h(x') >$ is observable in Σ.

2. If σ is unobservable and if $h(x) = h(x')$, then σ is left unchanged in G and $\delta(x, \sigma) = x'$. 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) $\delta(x, \sigma) = x_{new}$ and

   (b) $\delta(x_{new}, < h(x) \rightarrow h(x') >) = x'$

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

The reason for the above transformation is explained as follows. Incorporating the sensor information into the event set and renaming the events of the synchronous composition model $\tilde{G}$ allows us to have a pure event-based model in the sense that all relevant information about the system is captured in the event set as opposed to having information partly in the event set and partly as output maps defined on the states. In other words, all relevant information for the diagnosis problem is contained in the language generated by the system model G. This allows us to formulate and study the diagnosability properties of systems in the general framework of formal languages in the subsequent chapters of this thesis. The reason of the introduction of the new states x_{new} in Step 3 of the transformation process is as follows. When the occurrence of an unobservable event σ leads to a change of sensor reading from $< h(x) \rightarrow h(x') >$, incorporating the sensor change in the event set results in a composite event of the form $< \sigma, h(x) \rightarrow h(x') >$ where the first component σ is unobservable while the second component $< h(x) \rightarrow h(x') >$ is observable. This poses a difficulty in classifying the event as one of two types: unobservable or observable. Therefore, we split the event into two events, $< \sigma >$ and $< h(x) \rightarrow h(x') >$, and add a new state in the model G connecting these two events; the first event is treated as an unobservable event while the second event is classified as observable.

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

1. $< \sigma, h(x') >$: observable
2. $< \sigma >$: unobservable and
3. $< h(x) \rightarrow h(x') >$: observable

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

$$X = \tilde{X} \cup X_{new}. \tag{2.6}$$

The system to be diagnosed is now represented by the discrete event model

$$G = (X, \Sigma, \delta, x_0). \tag{2.7}$$

Note that the model G accounts for the normal and failed behavior of the system. Also note that all information relevant to the diagnosis problem, including the sensor information, is captured in the event set of this model. Typically, the observable events in the system are 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 are failure events or other events which cause changes in the system state not recorded by sensors.

To summarize, the model building procedure consists of the following steps:

Step 1: Build FSM models of the individual system components.

Step 2: Obtain the synchronous composition of these FSMs to model the joint operation of the components.

Step 3: Generate the global sensor map that lists the discrete sensor readings for each state of the synchronous composition model.

Step 4: Rename the transitions of the synchronous composition model to capture the sensor map in the event set and thereby obtain the composite system model.

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. (See Example 2 that follows.)

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 behaviour of a system over its entire range of operations including start-up and shutdown procedures. In the second example we model deviations from the steady-state of a system.

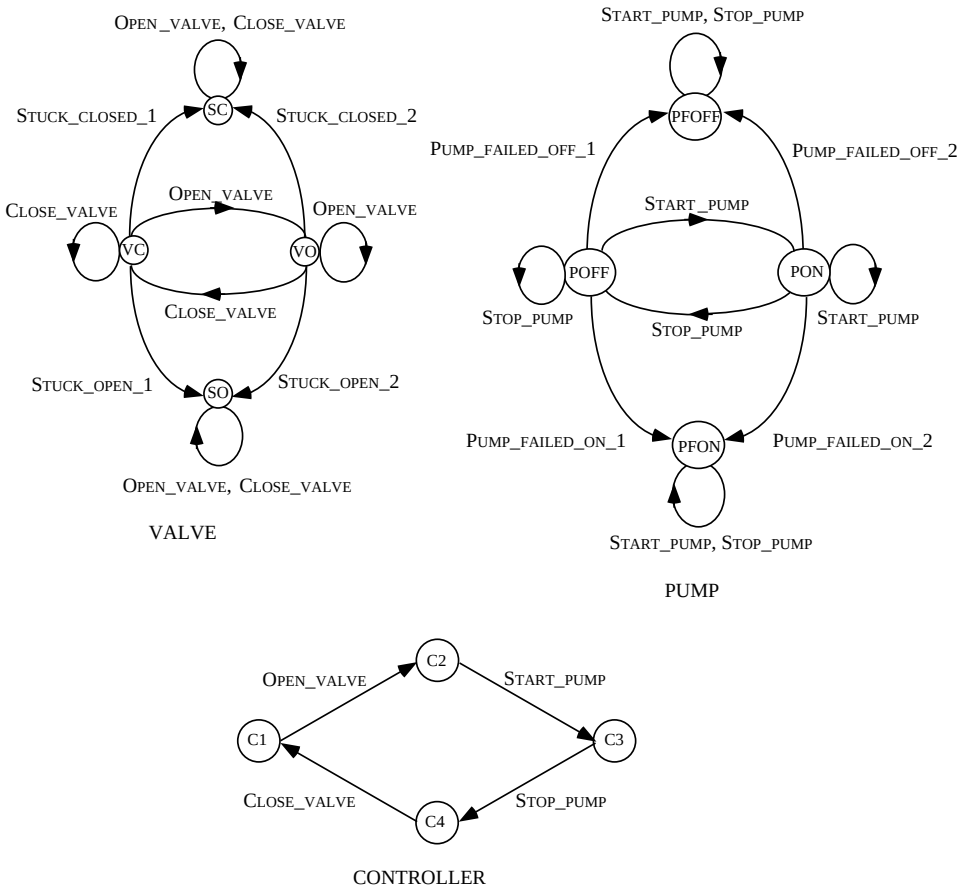


Figure 2.1: Component models for the HVAC system

2.3 Examples

Example 1 Consider an elementary HVAC system consisting of a pump, a valve, and a controller. Figure 2.1 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, 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 Figure 2.2 is obtained by the synchronous composition of the valve, pump, and controller models of Figure 2.1. 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

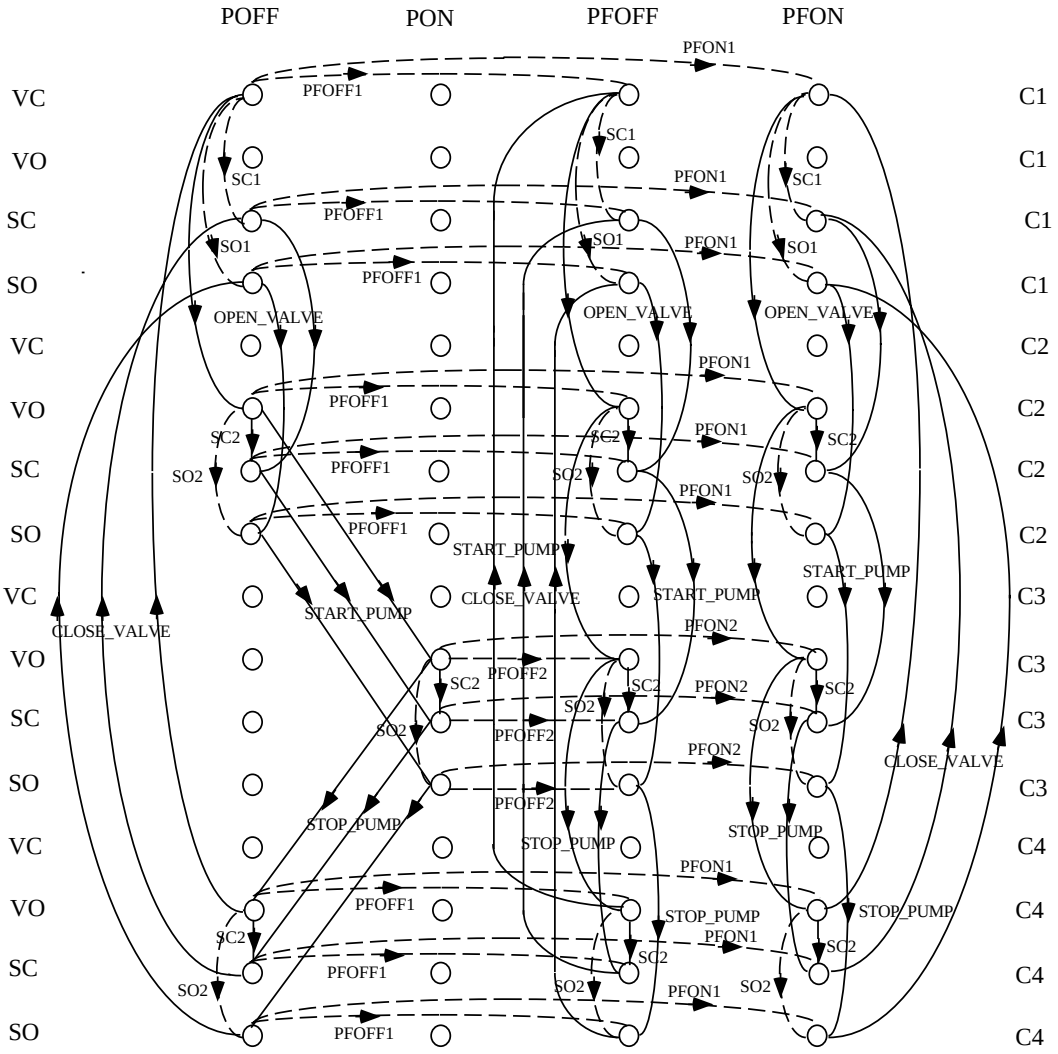


Figure 2.2: Synchronous composition of the component models for the HVAC system

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 $Y_1 = \{$ NP, PP $\}$ and $Y_2 = \{$NF, F$\}$ denote the set of outputs of the pressure sensor and flow sensor, respectively. NP and PP denote no pressure and positive pressure while NF and F denote no flow and flow, respectively. Table 2.1 lists the global sensor map h. Note that the map h is defined only for the accessible states of $\tilde{G}$ in Figure 2.2. 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 Figure 2.3. The shaded circles in Figure 2.3 denote the additional states x_{new}; as before, observable events are indicated by solid lines and unobservable events by dotted lines. The table in Figure 2.3 lists the events in this

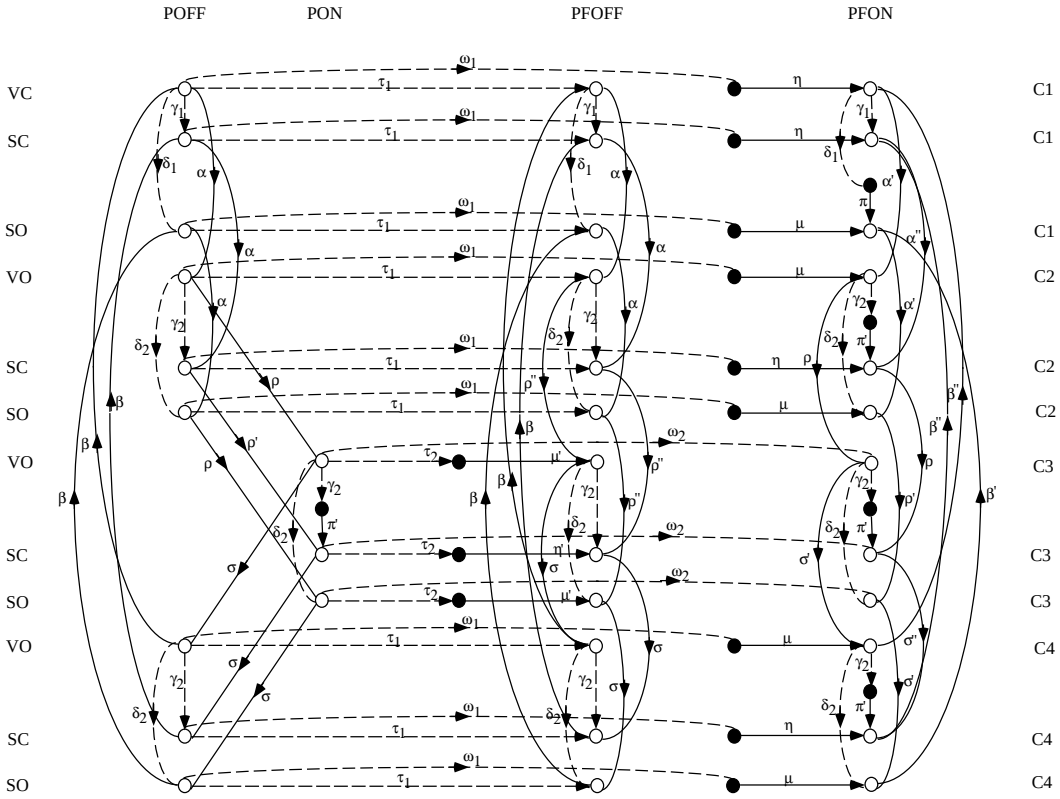
$h($ POFF, VC, $\bullet$ $)$	$=$	NP, NF	$h($ PFOFF, VC, $\bullet)$	$=$	NP, NF
$h($ POFF, VO, $\bullet$ $)$	$=$	NP, NF	$h($ PFOFF, VO, $\bullet$ $)$	$=$	NP, NF
$h($ POFF, SC, $\bullet$ $)$	$=$	NP, NF	$h($ PFOFF, SC, $\bullet$ $)$	$=$	NP, NF
$h($ POFF, SO, $\bullet$ $)$	$=$	NP, NF	$h($ PFOFF, SO, $\bullet$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

Table 2.1: The global sensor map for the HVAC system

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

Example 2 Consider the nitric acid cooling system shown in Figure 2.4 (cf. [23]). We now present discrete event models for this system that capture deviations of its behaviour 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 Figure 2.5.

Note that, as in Example 1, the component models include both normal and failure events. Also note that Figure 2.5 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 open state V1CO to the completely closed state, V1CC. However, if the event NAS_FAILURE, which corresponds to failure of the nitric acid shutdown system, occurs, 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



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 >

Figure 2.3: The composite model for the HVAC system

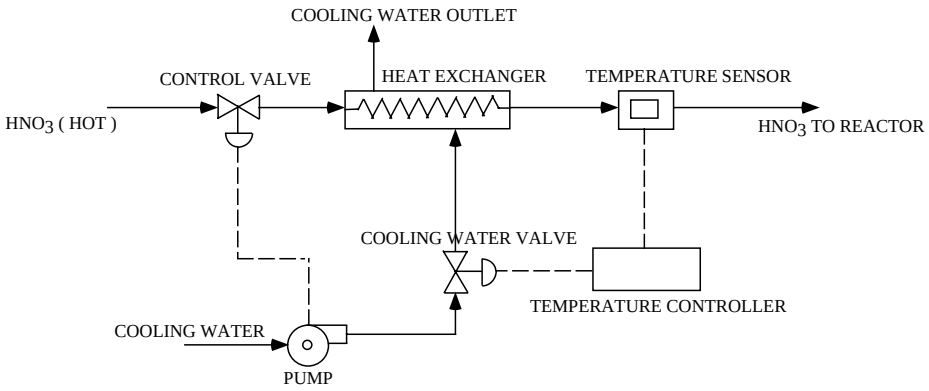


Figure 2.4: A nitric acid cooling system

major disturbance "low air pressure" at the cooling water valve, causes the cooling water valve to get into a complelety 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 Figure 2.5, 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 Figure 2.5. This FSM (not shown here due to space limitations) has 368 states and 1486 transitions. 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,

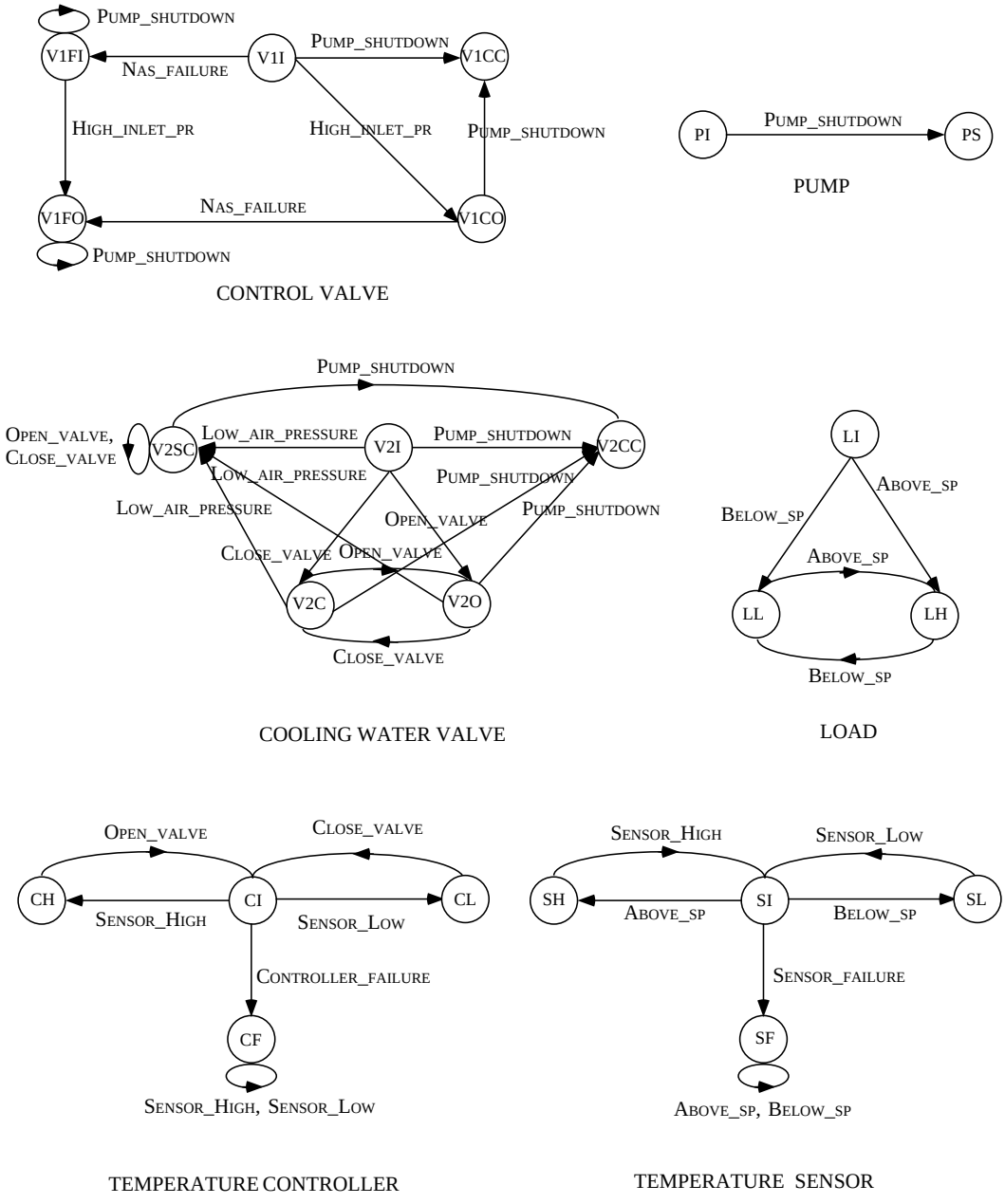


Figure 2.5: Component models for the nitric acid cooling system

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(•, V1FI/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

Table 2.2: Sensor Maps for the Nitric Acid Cooling System

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 2.2 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 Equation 2.5. The entries in Table 2.2 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.

As before, the composite system model can be obtained following the procedure of

Section 2.2. It has 646 states and 1764 transitions. Again, it is not shown here due to space considerations. ■

2.4 Conclusion

We have proposed in this chapter 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. From the point of view of the knowledge that is used in building the system model, one can say that the functional and (operational) procedural knowledge of the system are captured in the FSM models of the components, and operational status knowledge is captured in the sensor maps. Alternately speaking, the FSM models of the components provide generic component knowledge while the FSM model of the controller and the sensor maps provide the specific process related information. We note that while the approach to modeling that we discussed here has been developed for the purpose of diagnostics, its scope is not restricted to this problem; this method can be used to develop discrete event models of industrial systems for other purposes such as control as well.

Two pieces of user defined inputs are necessary for our modeling procedure, namely, the individual component models and the sensor maps. Given these two inputs, the rest of the procedure can be carried out algorithmically. Building the individual component models, and the sensor maps, 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. However, given any application domain, one can build a library of generic component models that include normal and faulty behavior. Then the only steps of the procedure that need to be addressed for a specific application are generation of the sensor map, and building the FSM model of the system controller.

Finally, note that it is not necessary for the sensor maps h_j to be single-valued functions, as defined in Section 2.2, that associate to any state of the system (synchronous composition) model one unique discrete value, as defined in Section 2.2. The sensor maps may be allowed to be multi-valued with *no change* in the modeling procedure. The only difference in the resulting composite model is that there would be two or more events of the form $<$ Command X, Sensor Reading 1 $>$, $<$ Command X, Sensor Reading 2 $>$, etc., or of the form, $<$ Sensor Reading 1 $\rightarrow$ Sensor Reading 2 $>$, $<$ Sensor Reading 1 $\rightarrow$ Sensor Reading 3 $>$ etc., leading *to* states with which multiple sensor readings are associated, and events of the form $<$ Sensor Reading 1 $\rightarrow$ Sensor Reading 2 $>$, $<$ Sensor Reading 3 $\rightarrow$ Sensor Reading 2 $>$ *from* states with which multiple sensor readings are associated. The need for such multi-valued maps arises in situations where it may not be possible to assign a unique sensor value for certain states of the system model due to coarse discretization of the system states. Consider, for example, a temperature control system that has both a cooling subsystem and a heating subsystem. Suppose that the valve controlling the flow of the cooling medium gets stuck

closed and the valve controlling the flow of the heating medium gets stuck open at the same time. The resultant system temperature may be nominal, higher than nominal, or lower than nominal depending on the relative physical positions of the two stuck valves. If we associate only one state in the discrete event model of the cooling valve to its different physically possible stuck closed positions, and likewise, only one state for all possible stuck open positions of the heating valve, then we need to allow the temperature sensor map to take all of the above three possible values, in states where the heating valve is stuck open and the cooling valve is stuck closed. The actual value realised may vary from one run of the system to another. Coarse discretization of the states of the physical system and relatively more finer discretization of the sensor values is another possible reason for requiring multi-valued sensor maps. As mentioned earlier, generalising the sensor maps to be multivalued calls for no further modification to the modeling procedure and thus the method is general enough to accomodate uncertain or incomplete sensor information.

CHAPTER 3

Diagnosability of DES

Never worry about theory as long as the machinery does what it's supposed to do.

... Science fiction writer, Robert Heinlein

3.1 Introduction

In this chapter, we study the property of *diagnosability* of systems. Recall that in Chapter 2 we presented a systematic procedure for obtaining an FSM model for any industrial system, that contains all information relevant for the diagnosis problem, including the sensor information, in its event set. This discrete event system and the language that it generates will be the starting point of our discussions here.

The organization of this chapter is as follows. In Section 3.2 we present the system model and introduce some notation that we will follow throughout the rest of the thesis. In Section 3.3 we introduce the notion of diagnosability. Roughly speaking, a system (or language) is said to be diagnosable if it is possible to detect, with finite delay, occurrences of the (unobservable) failure events from observed event sequences. We present two related definitions, namely, diagnosability and I-diagnosability, and illustrate these definitions by means of simple examples. This is followed by a comparison with related work in the DES literature, namely, other approaches to diagnosability, and the problems of observability and invertibility. In Section 3.4 we present the necessary and sufficient conditions for diagnosability and I-diagnosability of systems. These conditions are stated on the *diagnoser* (also referred to as the basic diagnoser) or variations thereof. The diagnoser is an FSM built from the FSM model of the system and can be thought of as an extended *observer* [31] for the system; in addition to estimates of the system state that an observer provides, the diagnoser provides information about past occurrences of failures in the system via labels attached to the state estimates. The diagnoser serves two important purposes: (i) off-line verification of the diagnosability properties of a system and (ii) on-line detection and isolation of failures. In this chapter we show how the diagnoser can be used to test for diagnosability of a system; we show that checking for diagnosability amounts to checking

for certain special type of cycles referred to as *indeterminate cycles* in the diagnoser or its variations. In the next chapter on on-line failure diagnosis we discuss how the diagnoser can be used to perform diagnostics when it observes on-line the behavior of the system. Finally, in Section 3.5 we illustrate the main ideas of this chapter with simple examples of physical systems.

3.2 Preliminaries and Notation

3.2.1 The System Model

The system of interest is modeled as an FSM

$$G = (X, \Sigma, \delta, x_0) \tag{3.1}$$

where X is the state space, Σ is the set of events, δ is the partial transition function, and x_0 is the initial state of the system. The model G accounts for the normal *and* failed behaviour of the system. The behaviour of the system is described by the *prefix-closed language* [39] $L(G)$ generated by G. Henceforth, we shall denote $L(G)$ by L. L is a subset of Σ^*, where Σ^* denotes the Kleene closure [20] of the set Σ and includes the empty trace denoted by ϵ.

Some of the events in Σ are observable, i.e., their occurrence 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} represents the set of unobservable events. The observable events in the system may be one of the following : commands issued by the controller, 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 that cause changes in the system state not recorded by sensors. (Recall the modeling methodology of Chapter 2.)

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 that in the traces generated by the system, only the events in Σ_o are observed. Often, we may not require that every failure event be identified uniquely; 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, or, when the control action that follows diagnosis is the same for all of these failures. In such cases, we partition the set of failure events into different failure types and require unique identification not of the failure event itself, but of the *type* of failure, when such an event occurs in the system. Let

$$\Sigma_f = \Sigma_{f1} \ \dot\cup \ldots \dot\cup \ \Sigma_{fm} \tag{3.2}$$

where $\Sigma_{fi}, \ i = \{1, \ldots, m\}$ denote disjoint sets of failure events corresponding to different failure types, and let Π_f denote this partition. Hereafter, when we write that "a failure of type F_i has occurred", we will mean that some event from the set Σ_{fi} has occurred.

We make the following assumptions on the system under investigation.

(A1) The language L generated by G is *live*. This means that there is a transition defined at each state x in X, i.e., the system cannot reach a point at which no event is possible.

(A2) There does not exist in G any cycle of unobservable events, i.e.,

$$\exists n_o \in N \text{ such that } \forall\ ust\ \in L,\ s \in \Sigma_{uo}^* \Rightarrow\ ||s|| \leq n_o$$

where $||s||$ is the length of trace s.
The liveness assumption on L is made for the sake of simplicity. This assumption can be justified for the uncontrolled system behavior (as is the case here) by appropriate modeling. In Chapter 6 of this thesis, where we deal with controlled DES, we relax this assumption and discuss diagnosability issues pertaining to non-live languages. Assumption (A2) ensures that observations occur with some regularity. Since detection of failures is based on observable transitions of the system we require that G does not generate arbitrarily long sequences of unobservable events.

3.2.2 Notation

In this thesis, we assume that all languages of interest are prefix-closed. Let $\overline{s}$ denote the prefix-closure of any trace $s \in \Sigma^*$. We denote by L/s the postlanguage of L after s, i.e.,

$$L/s = \{t \in \Sigma^* \mid st \in L\}. \tag{3.3}$$

We define the *projection* $P : \Sigma^* \rightarrow \Sigma_o^*$ in the usual manner [39]:

$$\begin{aligned} P(\epsilon) &= \epsilon \\ P(\sigma) &= \sigma \quad \text{if } \sigma \in \Sigma_o \\ P(\sigma) &= \epsilon \quad \text{if } \sigma \in \Sigma_{uo} \\ P(s\sigma) &= P(s)\ P(\sigma) \quad s \in \Sigma^*,\ \ \sigma \in \Sigma. \end{aligned} \tag{3.4}$$

Thus, P simply 'erases' the unobservable events in a trace. The inverse projection operator P_L^{-1} is defined as:

$$P_L^{-1}(y) = \{s \in L : P(s) = y\}. \tag{3.5}$$

Let s_f denote the final event of trace s. We define

$$\Psi(\Sigma_{fi}) = \{s \in L : s_f \in \Sigma_{fi}\}, \tag{3.6}$$

i.e., $\Psi(\Sigma_{fi})$ denotes the set of all traces of L that end in a failure event belonging to the class Σ_{fi}. Consider $\sigma \in \Sigma$ and $s \in \Sigma^*$. We use the notation $\sigma \in s$ to denote the fact that σ is an event in the trace s. With slight abuse of notation, we write $\Sigma_{fi} \in s$ to denote the fact that $\sigma_f \in s$ for some $\sigma_f \in \Sigma_{fi}$, or, formally, $\overline{s} \cap \Psi(\Sigma_{fi}) \neq \emptyset$.

We define

$$X_o = \{x_0\} \cup \{x \in X : x \text{ has an observable event into it}\}. \tag{3.7}$$

Let $L(G, x)$ denote the set of all traces that originate from state x of G. We define

$$L_o(G, x) = \{s \in L(G, x) : s = u\sigma, \quad u \in \Sigma_{uo}^* \quad , \sigma \in \Sigma_o\} \tag{3.8}$$

and

$$L_\sigma(G, x) = \{s \in L_o(G, x) : s_f = \sigma\}. \tag{3.9}$$

$L_o(G, x)$ denotes the set of all traces that originate from state x and end at the first observable event. $L_\sigma(G, x)$ denotes those traces in $L_o(G, x)$ that end with the particular observable event σ.

In the following sections, we will need to use a specially constructed FSM G', that generates the language $P(L)$, where

$$P(L) = \{t : t = P(s) \text{ for some } s \in L\}. \tag{3.10}$$

G' will in general be non-deterministic and it is constructed as follows:

$$G' = (X_o, \Sigma_o, \delta_{G'}, x_0) \tag{3.11}$$

where X_o, Σ_o and x_0 are as defined previously. The transition relation of G' is given by $\delta_{G'} \subseteq (X_o \times \Sigma \times X_o)$ and is defined as follows:

$$(x, \sigma, x') \in \delta_{G'} \text{ if } \delta(x, s) = x' \text{ for some } s \in L_\sigma(G, x) \tag{3.12}$$

It is straightforward to verify that $L(G') = P(L)$. Figures 3.6, 3.7 and 3.8 illustrate the construction of G' from G for three different systems.

3.3 The Notion of Diagnosability

In this section we present two definitions of diagnosability, with the first definition more stringent than the second. We shall henceforth refer to the first notion simply as diagnosability and to the second one as I-diagnosability.

3.3.1 Diagnosability

Formally, we define diagnosability as follows:

Definition 1 *A prefix-closed and live language L is said to be **diagnosable** with respect to the projection P and with respect to the partition Π_f on Σ_f if the following holds:*

$$(\forall i \in \Pi_f)\ (\exists n_i \in N\ )\ (\forall s \in \Psi(\Sigma_{fi}))\ (\forall t \in L/s) \qquad [\ ||t|| \geq n_i \Rightarrow D\ ]$$

where the diagnosability condition D is:

$$\omega \in P_L^{-1}[P(st)] \Rightarrow \Sigma_{fi} \in \omega\ .$$

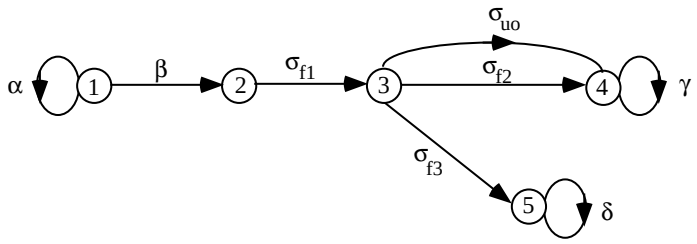


Figure 3.1: Example of a system with multiple failures

The above definition of diagnosability means the following. Let s be any trace generated by the system that ends in a failure event from the set Σ_{fi} and let t be any sufficiently long continuation of s. Condition D then requires that every trace belonging to the language that produces the same record of observable events as the trace st should contain in it a failure event from the set Σ_{fi}. This implies that along every continuation t of s 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 s. Alternately speaking, diagnosability requires that every failure event leads to observations distinct enough to enable unique identification of the failure type with a finite delay.

The case of multiple failures from the same set of the partition deserves special attention. When more than one failure of the same type, say, F_i, occurs along a trace s of L, the above definition of diagnosability does not require that each of these occurrences be detected. It suffices to be able to conclude, within finitely many events after the occurrence of the first failure, that along s, a failure from the set Σ_{fi} happened. In later sections we shall see how this feature distinguishes the case of possible multiple failures from the case of no multiple failures from any set of the partition.

Note that Definition 1 applies to any language $L \subseteq \Sigma^*$, regular [20] or not. We now illustrate by a simple example the above notion of diagnosability.

Example 3 Consider the system represented in Fig. 3.1. Here, α, β, γ and δ are observable events, σ_{uo} is an unobservable event while σ_{f1}, σ_{f2} and σ_{f3} represent failure events. Let the initial state x_0 of the system be state 1. If one chooses the partition $\Sigma_{f1} = \{\sigma_{f1}, \sigma_{f2}\}$ and $\Sigma_{f2} = \{\sigma_{f3}\}$, i.e., it is not required to distinguish between failures σ_{f1} and σ_{f2}, then the above system is diagnosable with $n_1 = 2$ and $n_2 = 1$. On the other hand, if the partition is $\Sigma_{f1} = \{\sigma_{f1}\}, \Sigma_{f2} = \{\sigma_{f2}\}$, and $\Sigma_{f3} = \{\sigma_{f3}\}$, then the system is not diagnosable since it is not possible to deduce the occurrence of failure σ_{f2}. ■

3.3.2 I-diagnosability

The preceding definition of diagnosability requires condition D to hold for all traces of L containing a failure event. We now propose a relaxed definition of diagnosability (termed I-diagnosability) that requires the diagnosability condition D to hold not for all traces containing a failure event but only for those in which the failure event is followed

by certain *indicator* observable events associated with every failure type. This modification is motivated by physical considerations such as the following: consider, for example, an HVAC system with a variable air volume (VAV) controller [1]. This system can operate in one of two possible modes: heating or cooling. In the cooling mode of operation, the zone temperature is controlled by regulating the position of the damper supplying conditioned air to the zone. In the heating mode, the damper is kept at a minimum open position and the zone temperature is controlled by regulating the reheat valve that supplies hot water to a heat exchanger which increases the temperature of the air supplied to the zone. The reheat valve is normally in a closed position during the cooling mode. Suppose now that the reheat valve gets stuck closed. As long as the controller remains in the cooling mode, it does not exercise the reheat valve. More precisely, the valve is not asked to open during this mode of operation. Under such conditions, it is obvious that one cannot diagnose the stuck-closed failure of the valve. Such a system is considered not diagnosable according to the previous definition. In the case of the modified definition, we associate as an indicator event, "open reheat valve" with the valve failure event "stuck-closed", and require the system to execute the "open reheat valve" event before deciding on its diagnosability. The system is considered diagnosable if after the execution of the corresponding indicator event it is possible to detect the valve failure, while it is termed not diagnosable if even after the indicator event is executed the valve failure remains undetectable. To summarize, I-diagnosability requires detection of failures only after the occurrence of an indicator event corresponding to the failure.

We first 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 $I_f : \Sigma_f \to 2^{\Sigma_I}$ denote the indicator map. Next we choose a partition Π_f on Σ_f such that

$$\bigcup_{i \in \Pi_f} \Sigma_{fi} = \Sigma_f$$

as before, with the additional constraint that for each $i = 1, \ldots, m$,

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

and define

$$I(\Sigma_{fi}) = I_f(\sigma_f) \ \text{ for any } \ \sigma_f \in \Sigma_{fi}. \tag{3.13}$$

We now have a set of observable indicator events $I(\Sigma_{fi})$ associated with each failure type F_i.

We now propose the following definition of I-diagnosability.

Definition 2 *A prefix-closed and live language L is said to be **I-diagnosable** with respect to the projection P, the partition Π_f on Σ_f, and the indicator map I if the following holds:*

$$(\forall i \in \Pi_f)\ (\exists n_i \in N\ )\ (\forall s \in \Psi(\Sigma_{fi})\ (\forall t_1 t_2 \in L/s : st_1 \in \Psi[I(\Sigma_{fi})]) \quad [\ ||t_2|| \geq n_i \Rightarrow D\ ]$$

where the diagnosability condition D is:

$$\omega \in P_L^{-1}[P(st_1t_2)] \Rightarrow \Sigma_{fi} \in \omega \ .$$

Note that $\Psi[I(\Sigma_{fi})]$ denotes the set of all traces of L that end in an observable event from the set $I(\Sigma_{fi})$. Therefore, in the case of the I-diagnosability, we require that occurrences of failure events of the type F_i should be detected in at most n_i transitions of the system *after* the occurrence of an indicator event from the set $I(\Sigma_{fi})$.

Example 4 Consider the system represented in Fig. 3.1. Suppose that the indicator events are chosen as follows: $I(\Sigma_{f1}) = \{\gamma\}$, $I(\Sigma_{f2}) = \{\delta\}$ and $I(\Sigma_{f3}) = \{\delta\}$. Let the desired partition be $\Sigma_{f1} = \{\sigma_{f1}\}, \Sigma_{f2} = \{\sigma_{f2}\}$ and $\Sigma_{f3} = \{\sigma_{f3}\}$. This system is I-diagnosable with $n_1 = 0$ and $n_3 = 0$. It is to be noted that though it is not possible to deduce the occurrence of failure σ_{f2}, the indicator event corresponding to σ_{f2}, i.e., δ does not follow this failure event and hence the diagnosability condition is not violated. ■

3.3.3 Comparison with Related Work

Partial observation problems in DES have been investigated by several researchers. While the problem of diagnosability itself has not been studied in detail, the related notions of observability, observability with delay, and invertibility have been the subject of several papers, among them [5], [8], [28], [31], [32], [34], [38]. Though closely related to these other problems, diagnosability is a distinctly different notion for the following reasons: partitioning of the failure events; need to identify every failure type with a finite delay; possibility of multiple failures; possible presence of unobservable events other than the failure events; no requirement of diagnosis or detection during normal system operation, and absence of "locking-on" phenomenon (explained below). In this section, we first discuss other approaches to diagnosability that have been proposed in recent DES literature [25], [3]. Afterwards, we discuss briefly the differences between diagnosability and the other notions mentioned above.

Other Approaches to Diagnosability

Feng Lin, in [25] (also see [26]), proposes a state-based approach to diagnosability. He assumes partial state information available via an output function. He addresses separately the problems of off-line and on-line diagnosis. In off-line diagnosis, the system to be diagnosed is not in normal operation and can be thought of as being in a "test-bed". The diagnostic procedure involves issuing a sequence of test commands, observing the resulting outputs, and drawing inferences on the set of possible states the system could be in. The off-line diagnosis problem can be considered equivalent to the problem of "verification". In on-line diagnosis, the system is assumed to be in normal operation. The goal of diagnostics, as before, is to issue a sequence of commands and identify uniquely, up to a partition, the

state of the system. However, unlike the case of off-line diagnosis, one now has to account for the possible occurrences of other uncontrollable events during the diagnostic process. The author gives an algorithm for computing a diagnostic control, or, a sequence of test commands for diagnosing system failures. This algorithm is guaranteed to converge if the system is indeed on-line diagnosable.

In [3], Bavishi and Chong study extensions of the above work. In particular, they consider testability of DES (which is equivalent to the off-line diagnosability problem studied in [25]) and present algorithms, (i) for determining the optimal set of sensors which would ensure testability of a given system and (ii) given a fixed set of sensors, for determining the infimal partition of the state space, with respect to which the system is testable.

Related Notions in DES

Language Observability Lin and Wonham study in [28] the supervisory control problem with partial event observations. They introduce a language based definition of observability and state conditions for the existence of a solution to the supervisory control problem in terms of observability and controllability of languages. The control problem addressed there does not require explicit determination of the occurrences of unobservable events or identification of the system state. Thus, the notion of observability introduced there is different from the problem of diagnosability.

Observability of State Machines In his paper on observability of DES [38], Ramadge explicitly addresses the problem of state identification for discrete event systems. In his framework, the system is modeled by a nondeterministic automaton with full event observability and partial state observability via an output map defined on the states (as in a Moore automaton). The problem is to reconstruct exactly the state of the system after the occurrence of every event. The motivation for the observability problem addressed there is an observer-state feedback approach to controller synthesis. The work in [38] is set in a different framework and is incomparable with the diagnosability problem studied here.

Özveren and Willsky adopt in [31] a slightly different notion of observability from that of Ramadge. They assume a partial event observation model with no direct state observations. A system is termed *observable* if, using a record of observable events, it is possible to determine the current state exactly at *intermittent* (but not necessarily fixed) points in time, separated by a bounded number of events. An *observer* is a DES which produces estimates of the state of the system after the occurrence of every observable event. In [31], the authors also address the problem of observability with delay. A system is said to be *observable with delay*, if, at intermittent points in time, it is possible to have perfect knowledge, not of the current state of the system but of the state some finite number of transitions into the past. In our framework, diagnosability is posed as an event detection problem. Viewed as a problem of state identification, diagnosability is a stronger notion

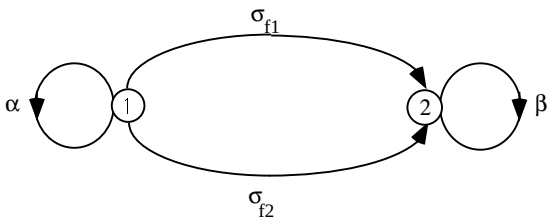


Figure 3.2: Example of a non-diagnosable system that is observable with delay

than observability with delay since the former requires that every failure state should be identifiable uniquely (up to a partition). In contrast, in [31], there is no notion of a particular state or set of states being observable. A system is observable (or observable with delay) as long as there exists at least one state which is uniquely identifiable at intermittent points in time. On the other hand, diagnosability only requires that the failure states be identifiable with finite delay; there are no similar requirements on the normal states. Thus, a system could execute arbitrarily long sequences of events, while in normal (failure-less) operation, with no single state being uniquely determinable even with delay. Further, a system could fail to be observable (with or without delay) if in the post-failure operation, there exists no state that is uniquely identifiable. However, this system could still be diagnosable, since we require unique identification not of every failure state but only of every set of the partition. Examples 5 and 6 that follow illustrate the differences between diagnosability and observability with delay.

Example 5 Figure 3.2 represents a system which is observable with delay but not diagnosable. Here α and β are observable events while σ_{f1} and σ_{f2} are unobservable failure events. This system is not diagnosable if the desired partition is $\Sigma_{f1} = \{\sigma_{f1}\}$ and $\Sigma_{f2} = \{\sigma_{f2}\}$. ■

Example 6 Figure 3.3 represents a system where the converse holds. In this figure α and β are observable while σ_{uo} is unobservable. The only failure event is σ_f. Here, a possible output sequence is β^*. When this sequence is observed, neither the current state nor the state any finite number of transitions in the past can be identified uniquely. On the other hand, it is possible to conclude the occurrence of a failure whenever the event sequence $\alpha^*\beta\beta\beta^*$ is observed. Hence, this is a diagnosable system which is not observable with delay. ■

In [5], Caines, Greiner and Wang study the state estimation problem for partially observed automata. The system is modeled as input-state-output automaton with partial state information available via an output function. A state output automaton is taken to be a special case of the above automaton where the input set is a singleton. They address the problems of initial state observability and current state observability using two different kinds of observers: classical dynamical observers and logic-based dynamical observers. The classical dynamic observer is a finite state automaton which takes for its input the observed

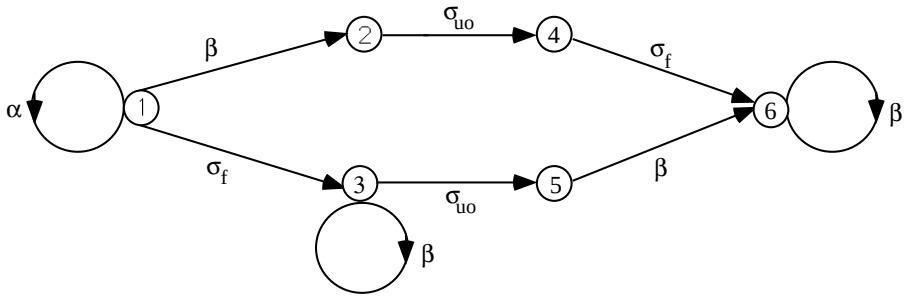


Figure 3.3: Example of a diagnosable system that is not observable with delay

system behaviour, namely, the sequence of input-output pairs, and generates a sequence of state estimates (either of the initial state or of the current state). The logic-based observer, on the other hand, is a logic-based dynamical system built in the framework of predicate calculus. This observer generates a sequence of logic propositions that describe the properties of the system. An interesting feature of these logic-based observers is their adaptability to changes in the system model. Observability as studied in [5] and observability as discussed in [38] and in [31] differ in the following important aspect. In [5], the authors assume that once the current state of the system is determined, then it is known for all future time, i.e., once the observer estimate converges to the true state of the system, it will thenceforth stay *locked on* and will always provide the correct system state as its output, for all observed input-output behavior.

Invertibility In [32], Özveren and Willsky introduce the concept of invertibility which is closely related to the problem of diagnosability. A language is said to be *invertible* if, at any time, using knowledge of the observed event sequence up to that time, we can reconstruct the full event sequence (corresponding to this observed sequence) up to a finite, bounded number of events in the past. Invertibility is a stronger notion than diagnosability. For a system to be diagnosable, we do not require reconstruction of entire event sequences; we are interested in identifying the occurence of specific failure events only. Further, when the failure events are partitioned into sets, one is interested only in identifying if one of a set of events has happened. Also, as mentioned before, in the case of multiple failures from the same set of the partition, diagnosability does not require detection of every single occurrence of these failures; it is enough to be able to conclude that a failure event from that set has occurred at least once. Hence, a system that is diagnosable could be non-invertible. Example 7 below illustrates a non-invertible system which is diagnosable.

Example 7 Figure 3.4 depicts a non-invertible system which is diagnosable. Here it is not possible to distinguish between the occurrence of traces $\sigma_{f1}\sigma_{uo1}\beta$, $\sigma_{f1}\sigma_{f2}\beta$ and $\sigma_{f2}\sigma_{uo2}\beta$. Hence the system is not invertible. However, if the required partition is $\Sigma_{f1} = \{\sigma_{f_1}, \sigma_{f_2}\}$, the system is diagnosable. ■

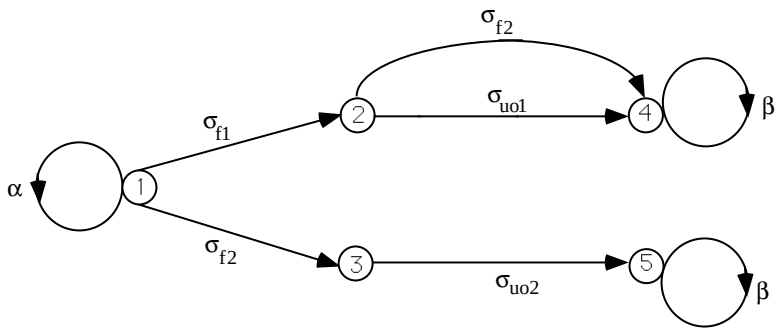


Figure 3.4: Example of a non-invertible system that is diagnosable

The problem of *eventual invertibility* of timed DES modeled by *generalized semi-Markov schemes* is addressed by Park and Chong in [34]. In this modeling framework the timed behaviour of a system is described by an automaton in conjunction with a set of event lifetimes. Partial state as well as partial event information is assumed available. In addition, all transition firing times are assumed to be observable. The problem of eventual invertibility is to determine from observations of events, states, and transition epochs, the corresponding event lifetimes up to a finite time in the past. The authors establish in [34] the equivalence between the problem of extracting event lifetimes and that of constructing the event trajectory from observations of the system behaviour.

This concludes the comparison of our notion of diagnosability with other related notions that have appeared in the literature.

3.4 Necessary and Sufficient Conditions for Diagnosability

In this section, we present necessary and sufficient conditions for a language L to be diagnosable, followed by similar conditions for L to be I-diagnosable. In the case of diagnosability, we investigate separately the case of no multiple failures of the same type and that of possible multiple failures of the same type. The former corresponds to the situation where along every trace s of L, no more than one failure from the same set of the partition can occur; the latter corresponds to the situtation where it is possible to have multiple failures from the same set of the partition occurring along any trace s. The reason for the separate investigation of these two cases will become apparent as we proceed. In each of the three cases, i.e., diagnosability of a language with non-multiple failures, diagnosability of a language with multiple failures, and I-diagnosability, the necessary and sufficient conditions are stated on the diagnoser or variations thereof. In order to test for these conditions, we use, in addition to the diagnoser, the machine G' introduced in 3.2.2. In all three cases presented below, we introduce the appropriate diagnoser, state some of its properties, and finally state and prove the conditions for diagnosability.

3.4.1 Conditions for Diagnosability-The Case of No Multiple Failures

The Diagnoser G_d

We define the set of *failure labels* $\Delta_f = \{F_1, F_2, ...F_m\}$ where $|\Pi_f| = m$ and the complete set of possible labels

$$\Delta = \{N\} \ \cup \ 2^{\{\Delta_f \cup \{A\}\}}. \tag{3.14}$$

Here N is to be interpreted as meaning "normal", A as meaning "ambiguous" (to be explained shortly), and $F_i, \ i \in \{1, \ldots, m\}$ as meaning that a failure of the type F_i has occurred. Recall from Section 3.2.2 the definition of X_o and define

$$Q_o = 2^{X_o \times \Delta}. \tag{3.15}$$

The diagnoser for G is the FSM

$$G_d = (Q_d, \Sigma_o, \delta_d, q_0) \tag{3.16}$$

where Q_d, Σ_o, δ_d and q_0 have the usual interpretation. The initial state of the diagnoser q_0 is defined to be $\{(x_0, \{N\})\}$. The transition function δ_d of the diagnoser is constructed as explained below. The state space Q_d is the resulting subset of Q_o composed of the states of the diagnoser that are reachable from q_0 under δ_d. Since the state space Q_d of the diagnoser is a subset of Q_o, a state q_d of G_d is of the form

$$q_d = \{(x_1, \ell_1), \ldots, (x_n, \ell_n)\}$$

where $x_i \in X_o$ and $\ell_i \in \Delta$, i.e., ℓ_i is of the form $\ell_i = \{N\}$, $\ell_i = \{A\}$, $\ell_i = \{F_{i_1} F_{i_2}, \ldots, F_{i_k}\}$, or $\ell_i = \{A, F_{i_1} F_{i_2}, \ldots, F_{i_k}\}$ where in the last two cases $\{i_1, i_2, \ldots, i_k\} \subseteq \{1, 2, \ldots, m\}$.

An *observer* for G (see [31]) gives estimates of the current state of the system after the occurrence of every observable event. The diagnoser G_d can be thought of as an *extended observer* where we append to every state estimate a label of the form mentioned above. The labels attached to the state estimates carry failure information and failures are diagnosed by checking these labels. We assume the system G is normal to start with, hence we define $q_0 = \{(x_o, \{N\})\}$.

Before defining the transition function δ_d of the diagnoser, we define the following three functions: the Label Propagation function LP, the Range function R, and the Label Correction function LC.

Definition 3 *The* ***Label Propagation Function*** $LP : X_o \times \Delta \times \Sigma^* \to \Delta$.
Given $x \in X_o$, $\ell \in \Delta$, and $s \in L_o(G, x)$, LP propagates the label ℓ over s, starting from x and following the dynamics of G, i.e., according to $L(G, x)$. It is defined as follows:

$$LP(x, \ell, s) = \begin{cases} \{N\} \text{ if } \ell = \{N\} \ \wedge \ \forall i \ [\Sigma_{fi} \notin s] \\ \{A\} \text{ if } \ell = \{A\} \ \wedge \ \forall i \ [\Sigma_{fi} \notin s] \\ \{F_i : F_i \in \ell \ \vee \ \Sigma_{fi} \in s\} \text{ otherwise.} \end{cases}$$

Definition 4 *The **Range Function** $R : Q_o \times \Sigma_o \rightarrow Q_o$ is defined as follows:*

$$R(q,\sigma) = \bigcup_{(x,\ell)\in q} \bigcup_{s\in L_\sigma(G,x)} \{(\delta(x,s), LP(x,\ell,s))\}.$$

Definition 5 *The **Label Correction Function** $LC : Q_o \rightarrow Q_o$ is defined as follows:*

$$\begin{aligned} LC(q) \;=\; & \{(x,\ell) \in q : x \text{ appears only once in all the pairs in } q\} \;\cup \\ & \{(x, \{A\} \cup \ell_{i1} \cap \ldots \cap \ell_{ik}) \;\; \text{whenever } \exists \text{ two or more pairs } (x,\ell_{i1}),\ldots,(x,\ell_{ik}) \\ & \text{in } q\}. \end{aligned}$$

The use of the Label Correction function LC and the label A is explained as follows. The label acquired by any state x along a trace s indicates the occurrence or otherwise of a failure when the system moves along trace s and transitions into state x. Suppose that there exist two pairs $(x,\ell),(x,\ell')$ in $R(q,\sigma)$ for some state q of the diagnoser. Then this implies that the state x could have resulted from a failure event of a particular type, say F_i, or not. Under this condition, we attach the label A to x to denote the fact that there is an ambiguity. In other words, the A label is to be interpreted as meaning "either F_i or not F_i" for $i \in \{1,\ldots,m\}$. It is to be noted here that we do not distinguish between cases "F_i or F_j" , "F_j or F_k", "N or F_i", and so on. In all of these situations, we simply use the label A.

The transition function of the diagnoser $\delta_d : Q_o \times \Sigma_o \rightarrow Q_o$ is now defined as

$$q_2 = \delta_d(q_1,\sigma) \Leftrightarrow q_2 = LC[R(q_1,\sigma)] \tag{3.17}$$

with $\sigma \in e_d(q_1)$ where

$$e_d(q_1) = \bigcup_{(x,\ell)\in q_1} \{P(s) : s \in L_o(G,x)\}. \tag{3.18}$$

In words, $e_d(q_1)$ is the active event set of G_d at q_1, i.e., the set of all possible transitions of the diagnoser at the state q_1.

To summarize, the diagnoser G_d is constructed as follows. Let the current state of the diagnoser (i.e., 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, given by $S(x,\sigma) = \{\delta(x,s\sigma) \text{ where } s \in \Sigma_{uo}^*\}$.

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 $\ell = \{N\}$ and s contains no failure events, then the label ℓ' is also $\{N\}$.

   (b) If $\ell = \{A\}$ and s contains no failure events, then the label ℓ' is also $\{A\}$.

(c) If $\ell = \{A, F_i\}$ and s contains no failure events, then the label ℓ' is $\{F_i\}$.

(d) If $\ell = \{N\}$ or $\{A\}$ and s contains failure events from Σ_{fi}, Σ_{fj}, then $\ell' = \{F_i, F_j\}$.

(e) If $\ell = \{F_i, F_j\}$ or $\{A, F_i, F_j\}$ and s contains failure events from Σ_{fk}, then $\ell' = \{F_i, F_j, F_k\}$.

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', A, F_i, F_j) all $(x', \ell'), (x', \ell'') \in q_2$ such that F_i and F_j are components of both ℓ' *and* ℓ''. That is, if the same state estimate x' appears more than once in q_2 with different labels, we associate with x' all common components of these labels, and in addition, we attach to x the *ambiguous* label A.

Note that in cases (c), (d), and (e) above, we do not propagate the A label from one state to the next. While this leads to a reduction in the state space of the diagnoser, it leads to no loss of information necessary for determining the diagnosability properties of a language or, for implementing diagnostics. The reasons for this will become evident in the subsequent sections.

It is not difficult to see from the above construction of the diagnoser that $L(G_d) = P(L)$. We now give a simple example illustrating the construction of the diagnoser.

Example 8 Figure 3.5 illustrates a system G and its diagnoser G_d. Here $\alpha, \beta, \gamma, \delta$ and σ are observable events while $\sigma_{uo}, \sigma_{f1}, \sigma_{f2}$ and $\sigma_{f2'}$ are unobservable. $\Sigma_{f1} = \{\sigma_{f1}\}$ and $\Sigma_{f2} = \{\sigma_{f2}, \sigma_{f2'}\}$. In all illustrations that follow, we represent (x, ℓ) pairs simply as $x\ell$ for clarity. Also, the initial state x_0 of G is chosen to be state 1. ■

Remark: In the above construction procedure we have assumed knowledge of the initial state of the system, since the diagnoser is assumed to run in parallel with the system from the start of operation. However, it is to be noted that the above procedure remains valid even in the case of unknown initial state.

Properties and Definitions of the Diagnoser G_d

We now state a few properties of the diagnoser that follow from its construction. These properties and the definitions that follow will be used subsequently to state and prove the conditions for diagnosability.

(P1) By construction, any $x_i \in X_o$ appears in at most one pair (x_i, ℓ_i) in any state of Q_d.

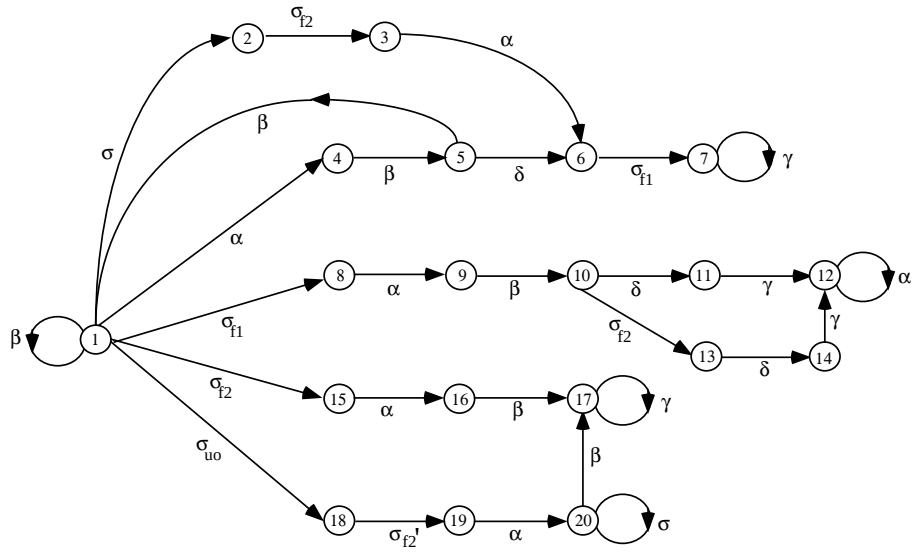
(P2) Let $q \in Q_d$. Then

$$(x_1, \ell_1),\ (x_2, \ell_2) \in q \quad \Leftrightarrow$$

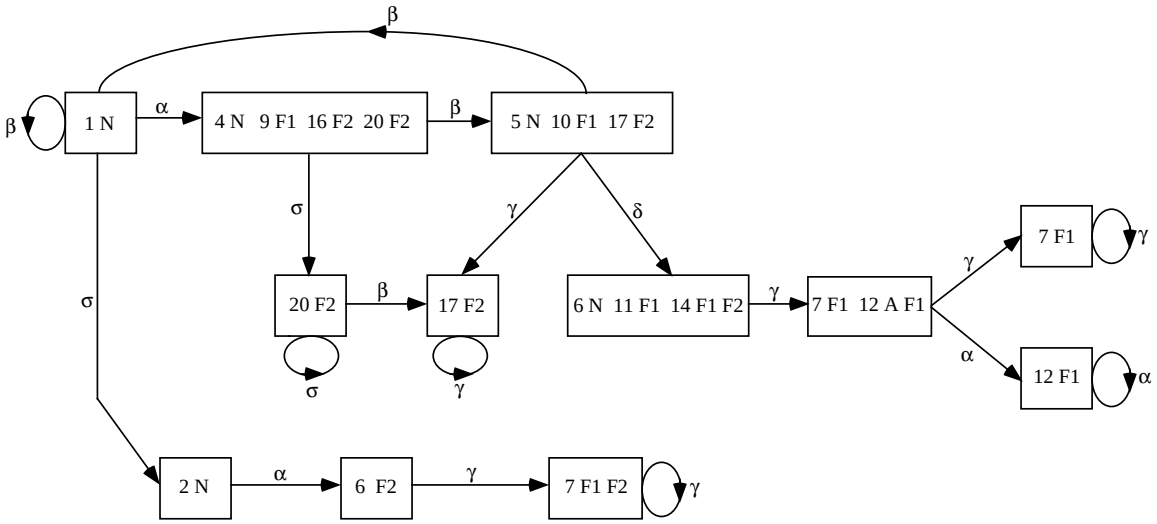
$\exists s_1, s_2 \in L$ such that $s_{1f}, s_{2f} \in \Sigma_o,\ \delta(x_0, s_1) = x_1,\ \delta(x_0, s_2) = x_2$ and $P(s_1) = P(s_2)$.

(P3) Let $q_1, q_2 \in Q_d$ and $s \in \Sigma^*$ such that $(x_1, \ell_1) \in q_1$, $(x_2, \ell_2) \in q_2$, $\delta(x_1, s) = x_2$ and $\delta_d(q_1, P(s)) = q_2$. Then

$$(F_i \notin \ell_2)\ \wedge\ (A \notin \ell_2) \Rightarrow F_i \notin \ell_1.$$



THE SYSTEM G



THE DIAGNOSER G_d

Figure 3.5: Example illustrating construction of the diagnoser G_d

Property (P3) above simply states that the failure labels F_i propagate from state to state, unless replaced by the A label as a consequence of the label correction function LC. Hence, along any trace s of L, if a state x receives an F_i label, every successor x' of x also carries the F_i label, unless ambiguity arises, in which case, x' receives the A label. Also note that if along a trace $s \in L$, a state x carries the label N, then so do all of its predecessors.

Definition 6 *1. A state $q \in Q_d$ is said to be F_i-**certain** if $\forall (x, \ell) \in q, \quad F_i \in \ell$.*

*2. A state $q \in Q_d$ is said to be F_i-**uncertain** if $\exists (x, \ell), (y, \ell') \in q$, such that $F_i \in \ell$ and $F_i \notin \ell'$.*

*3. A state $q \in Q_d$ is said to be **ambiguous** if $\exists (x, \ell) \in q$, such that $A \in \ell$.*

Note that in the above definition of an F_i-uncertain state, $x \neq y$ by Property (P1). Also note that if a state q is not F_i-uncertain, it does not necessarily imply that q is F_i-certain, since a state $q \in Q_d$ such that $\forall (x, \ell) \in q, F_i \notin \ell$ is neither F_i-certain nor F_i-uncertain. The following results are a direct consequence of the construction of the diagnoser.

Lemma 1 *(i) Let $\delta_d(q_0, u) = q$. If q is F_i-certain, then $(\forall \omega \in P_L^{-1}(u)) \quad \Sigma_{fi} \in \omega$.*

(ii) If a state $q \in Q_d$ is F_i-uncertain, then $\exists \; s_1, s_2 \in L$ such that : $\Sigma_{fi} \in s_1, \quad \Sigma_{fi} \notin s_2$, $P(s_1) = P(s_2), \quad \delta_d(q_0, P(s_1)) = q, \quad$ and $\delta(x_0, s_1) \neq \delta(x_0, s_2)$.

(iii) If a state $q \in Q_d$ is ambiguous, then $\exists s_1, s_2 \in L$ and $\exists i \in \Pi_f$ such that: $\Sigma_{fi} \in s_1$, $\Sigma_{fi} \notin s_2, \quad P(s_1) = P(s_2), \quad \delta_d(q_0, P(s_1)) = q, \quad$ and $\delta(x_0, s_1) = \delta(x_0, s_2)$.

From the definition of an F_i-certain state and the above lemma, it is obvious that if the current state of the diagnoser is F_i-certain, then we can conclude that a failure of the type F_i has occurred, regardless of what the current state of G is. This is precisely the type of diagnosis that is addressed in this paper. On the other hand, presence of an F_i-uncertain state in G_d corresponds to the situation where there are two traces s_1 and s_2 in L such that s_1 contains a failure event of type F_i while s_2 does not and in addition, the traces s_1 and s_2 produce the same record of observable events. Whenever the diagnoser hits an F_i-uncertain state, we conclude that a failure of the type F_i may have occurred but it is not possible to ascertain from the observed event sequence up to that point whether the failure has indeed occurred. Finally, presence of an ambiguous state in G_d corresponds to the situation where there are two traces s_1 and s_2 in L such that the set of all possible continuations of s_1 in L is the same as that of s_2, s_1 contains a failure event of a particular type, say F_i, while s_2 does not, and in addition the traces s_1 and s_2 produce the same record of observable events. We shall henceforth refer to such traces as *F_i-ambiguous traces*.

Definition 7 *A set of states $x_1, x_2, \ldots, x_n \in X$ is said to form a cycle in G if $\exists s \in L(G, x_1)$ such that $s = \sigma_1 \sigma_2 \ldots \sigma_n$ and $\delta(x_l, \sigma_l) = x_{(l+1) \; mod \; n}, \quad l = 1, 2, \ldots, n$.*

The following definition of an F_i-indeterminate cycle is based upon examination of cycles in G_d and G'.

Definition 8 *A set of F_i-uncertain states $q_1, q_2, \ldots, q_n \in Q_d$ is said to form an* ***F_i-indeterminate*** *cycle if*

1. *$q_1, q_2, \ldots, q_n$ form a cycle in G_d with $\delta_d(q_l, \sigma_l) = q_{l+1}$, $l = 1, \ldots, n \Leftrightarrow 1$, $\delta_d(q_n, \sigma_n) = q_1$, where $\sigma_l \in \Sigma_o$, $l = 1, \ldots, n$; and*

2. *$\exists (x_l^k, \ell_l^k), (y_l^r, \tilde{\ell}_l^r) \in q_l$, $l = 1, \ldots, n$, $k = 1, \ldots, m$, and $r = 1, \ldots, m'$ such that*

   (a) *$F_i \in \ell_l^k$, $F_i \notin \tilde{\ell}_l^r$ for all l, k, and r;*

   (b) *The sequences of states $\{x_l^k\}$, $l = 1, \ldots, n$, $k = 1, \ldots, m$ and $\{y_l^r\}$, $l = 1, \ldots, n$, $r = 1, \ldots, m'$ form cycles in G' with*

$$\begin{aligned} (x_l^k, \sigma_l, x_{(l+1)}^k) &\in \delta_{G'}, \quad l = 1, \ldots, n \Leftrightarrow 1, \quad k = 1, \ldots, m, \\ (x_n^k, \sigma_n, x_1^{k+1}) &\in \delta_{G'}, \quad k = 1, \ldots, m \Leftrightarrow 1, \text{ and} \\ (x_n^m, \sigma_n, x_1^1) &\in \delta_{G'}, \end{aligned}$$

   and

$$\begin{aligned} (y_l^r, \sigma_l, y_{(l+1)}^r) &\in \delta_{G'}, \quad l = 1, \ldots, n \Leftrightarrow 1, \quad r = 1, \ldots, m,' \\ (y_n^r, \sigma_n, y_1^{r+1}) &\in \delta_{G'}, \quad r = 1, \ldots, m' \Leftrightarrow 1, \text{ and} \\ (y_n^{m'}, \sigma_n, y_1^1) &\in \delta_{G'}. \end{aligned}$$

In other words, an F_i-indeterminate cycle in G_d is a cycle composed exclusively of F_i-uncertain states for which there exist:

1. a corresponding cycle (of observable events) in G' involving only states that carry F_i in their labels in the cycle in G_d (this is the sequence $\{x_l^k\}$) 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 (this is the sequence $\{y_l^r\}$).

Observe that in the above definition, m and m' denote the number of times the cycle $q_1, q_2, \ldots, q_n$ in G_d is completed before the cycle in G' is completed, i.e, nm and nm' are the cycle lengths in G' for $\{x_l^k\}$ and $\{y_l^r\}$ respectively.

An Fi-indeterminate cycle in G_d indicates the presence in L of two traces s_1 and s_2 of arbitrarily long length, such that they both have the same observable projection, and s_1 contains a failure event from the set Σ_{fi} while s_2 does not. Alternately speaking, an F_i-indeterminate cycle has the propery that if the diagnoser enters this cycle following the occurrence of a failure event, then it can continue to remain forever in this cycle. The notion of an F_i-indeterminate cycle is the most crucial element in the development of necessary and sufficient conditions for diagnosability. We now present examples to better illustrate this notion.

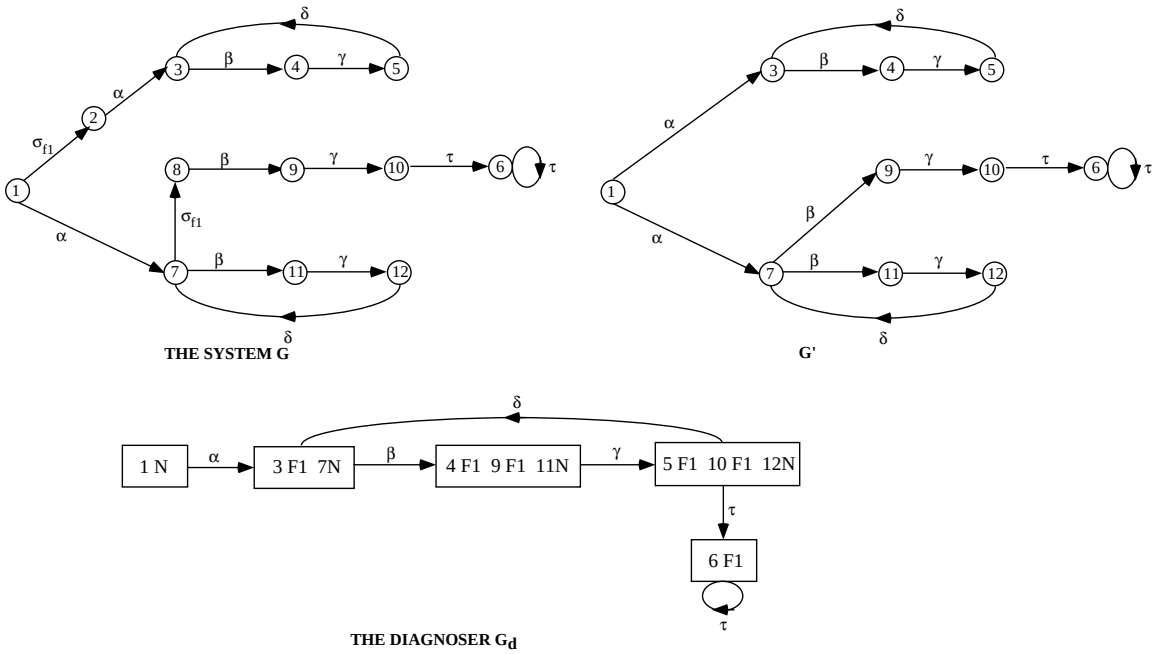


Figure 3.6: Example of a system with an F_1-indeterminate cycle in its diagnoser G_d

Example 9 Figure 3.6 depicts a system and its diagnoser G_d. Let $\Sigma_f = \Sigma_{f1} = \{\sigma_{f1}\}$. The diagnoser has a cycle of F_1-uncertain states, with the corresponding event sequence $\beta\gamma\delta$. Corresponding to this cycle in the diagnoser, there are two cycles in the state machine G': the first involves states $3, 4$ and 5 which appear with an F_1 label in the cycle in the diagnoser and the second involves states $7, 11$ and 12 which carry a N label in the cycle in the diagnoser. Thus the cycle in G_d is an F_1-indeterminate cycle with $m = m' = 1$, $x_1^1 = 3,\ x_2^1 = 4,\ x_3^1 = 5$ and $y_1^1 = 7,\ y_2^1 = 11,\ y_3^1 = 12$. ■

Example 10 Consider the system and its diagnoser G_d depicted in Figure 3.7. Let $\Sigma_f = \Sigma_{f1} = \{\sigma_{f1}\}$. This diagnoser has a cycle of F_1-uncertain states. In fact, on closer inspection, one sees that the diagnosers of the systems in Figures 3.6 and 3.7 are identical. However, this cycle is not F_1-indeterminate as there is no corresponding cycle in G' involving states that carry the F_1 label in the cycle in G_d, namely states $3, 4, 5, 9$ and 10. ■

In the above examples, the cycle in G_d corresponds directly to a cycle in G', in the sense that the loop in G' is completed with just one completion of the loop in the diagnoser G_d, i.e, $m = m' = 1$. We now give an example of a system where more than one traversal of the loop in G_d is required to complete the loop in G'.

Example 11 Consider the system and the diagnoser G_d in Figure 3.8. It is seen by inspection of the diagnoser G_d that it contains a cycle of F_1-uncertain states with corresponding event sequence $\gamma\beta$. It can also be verified by direct inspection that this cycle is F_1-indeterminate. In this case, the set $\{x_l^k\}$ in Definition 8 is $\{3, 4\}$ while the set $\{y_l^r\}$ is

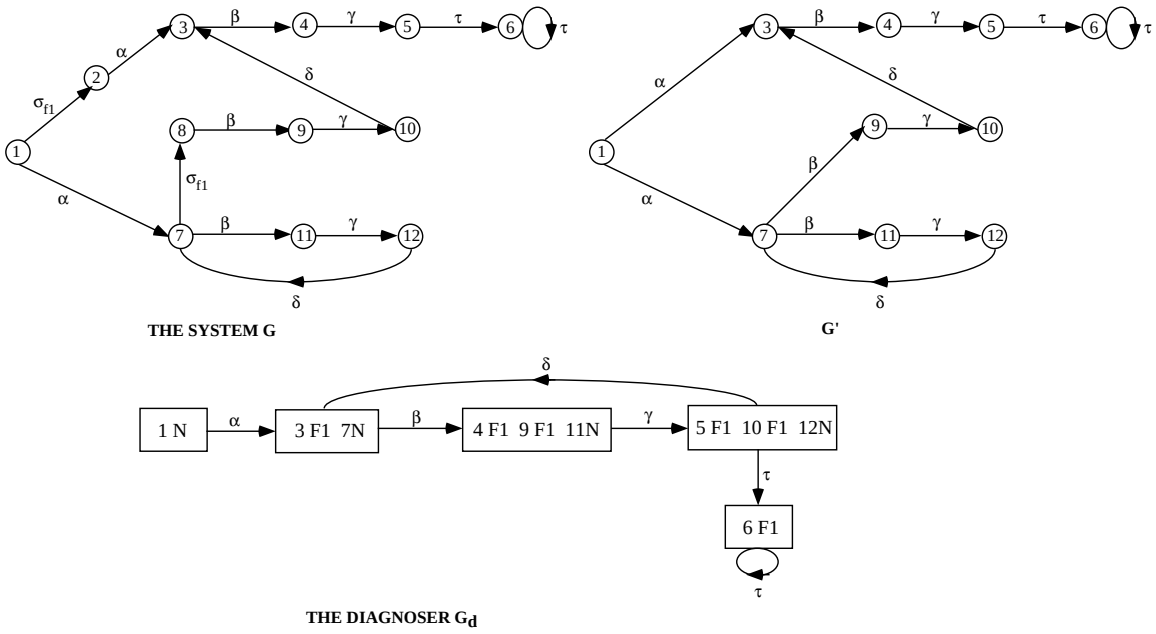


Figure 3.7: Example of a system with a cycle of F_1-uncertain states in its diagnoser G_d

$\{5,6,8,9\}$ (or, $\{8,9,5,6\}$). Note that the cycle in G_d has to be traversed twice before the cycle formed by the states $5, 6, 8,$ and 9 is completed. Thus, we have multiplicities $m = 1$ and $m' = 2$ for this indeterminate cycle. ■

We are now ready to state the necessary and sufficient conditions for diagnosability in the case of no multiple failures.

Necessary and Sufficient Conditions

Theorem 1 *A language L without multiple failures of the same type is diagnosable if and only if its diagnoser G_d satisfies the following two conditions:*

(C1) *There are no F_i-indeterminate cycles in G_d, for all failure types F_i.*

(C2) *No state $q \in Q_d$ is ambiguous.*

Proof: Necessity: We first prove that if L is diagnosable, then it satisfies Condition (C1). By contradiction, assume there exist states $q_1, q_2, \ldots, q_n \in Q_d$ such that they form an F_i-indeterminate cycle and let $\delta_d(q_i, \sigma_i) = q_{(i+1) \bmod n}$. Let $(x_l^k, \ell_l^k), (y_l^r, \tilde{\ell}_l^r) \in q_l$, $l = 1, \ldots, n$, $k = 1, \ldots, m$, and $r = 1, \ldots, m'$ form corresponding cycles in G' with $F_i \in \ell_l^k$, $F_i \notin \tilde{\ell}_l^r$. Then we have

$$\begin{aligned}
\delta(x_l^k, s_l^k \sigma_l) &= x_{(l+1)}^k, \quad l = 1, \ldots, n \Leftrightarrow 1, \quad k = 1, \ldots, m, \\
\delta(x_n^k, s_n^k \sigma_n) &= x_1^{k+1}, \quad k = 1, \ldots, m \Leftrightarrow 1, \\
\delta(x_n^m, s_n^m \sigma_n) &= x_1^1,
\end{aligned}$$

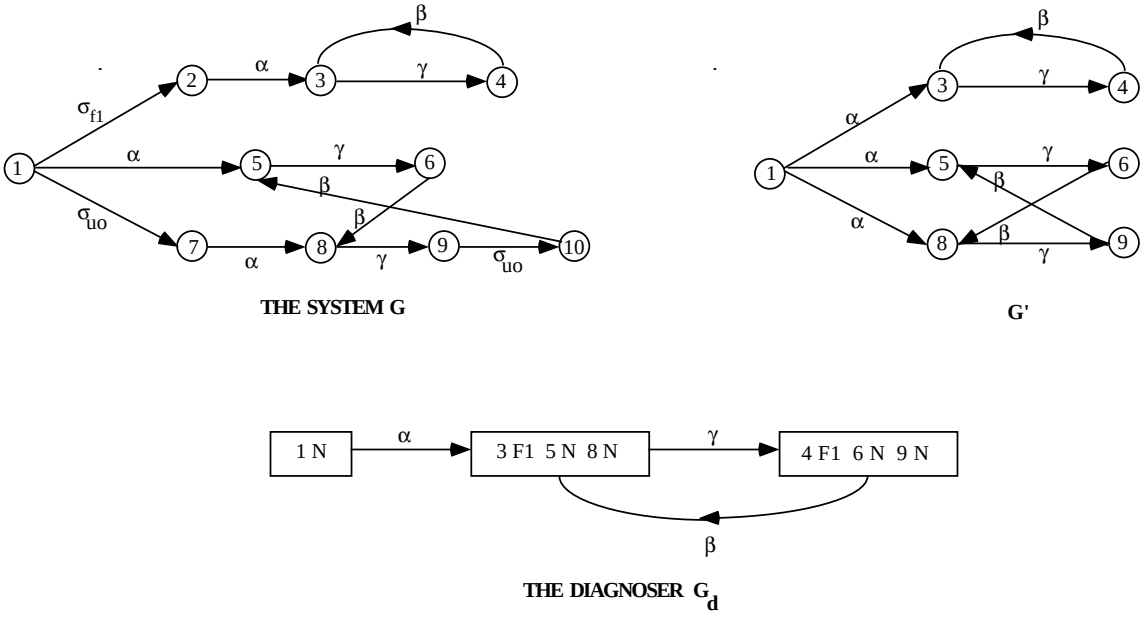


Figure 3.8: Another example of a system with an F_1-indeterminate cycle in its diagnoser

and

$$\begin{aligned}
\delta(y_l^r, \tilde{s}_l^r\sigma_l) &= y_{(l+1)}^r, \quad l = 1,\ldots,n \Leftrightarrow 1, \quad r = 1,\ldots,m,' \\
\delta(y_n^r, \tilde{s}_n^r\sigma_n) &= y_1^{r+1}, \quad r = 1,\ldots,m' \Leftrightarrow 1, \text{ and} \\
\delta(y_n^{m'}, \tilde{s}_n^{m'}\sigma_n) &= y_1^1
\end{aligned}$$

where

$$m, m' \in N\ ,\ s_l^k \in L(G, x_l^k),\ \tilde{s}_l^r \in L(G, y_l^r) \text{ and } s_l^k, \tilde{s}_l^r \in \Sigma_{uo}^*.$$

Since (x_1^1, ℓ_1^1), $(y_1^1, \tilde{\ell}_1^1) \in q_1$, $\exists s_0, \tilde{s}_0 \in L$ such that $\delta(x_0, s_0) = x_1^1$, $\delta(y_0, \tilde{s}_0) = y_1^1$ and $P(s_0) = P(\tilde{s}_0)$ from Property (P2). Further, since $F_i \in \ell_1^1$, then $\Sigma_{fi} \in s_0$ and since $F_i \notin \tilde{\ell}_l^r$, we have that $\Sigma_{fi} \notin \tilde{s}_0$ and $\Sigma_{fi} \notin \tilde{s}_l^r$ for all l, r.

Consider the two traces

$$\begin{aligned}
\omega &= s_0(s_1^1\sigma_1 s_2^1\sigma_2 \ldots s_n^1\sigma_n s_1^2\sigma_1 s_2^2\sigma_2 \ldots s_n^2\sigma_n \ldots\ldots s_1^m\sigma_1 s_2^m\sigma_2 \ldots s_n^m\sigma_n)^{km'} \\
\tilde{\omega} &= \tilde{s}_0(\tilde{s}_1^1\sigma_1 \tilde{s}_2^1\sigma_2 \ldots \tilde{s}_n^1\sigma_n \tilde{s}_1^2\sigma_1 \tilde{s}_2^2\sigma_2 \ldots \tilde{s}_n^2\sigma_n \ldots\ldots \tilde{s}_1^{m'}\sigma_1 \tilde{s}_2^{m'}\sigma_2 \ldots \tilde{s}_n^{m'}\sigma_n)^{km}
\end{aligned}$$

for arbitrarily large k. We have that $\omega,\ \tilde{\omega} \in L$, $P(\omega) = P(\tilde{\omega}) = P(s_0)(\sigma_1\sigma_2 \ldots \sigma_n)^{kmm'}$, and $\Sigma_{fi} \in \omega$ while $\Sigma_{fi} \notin \tilde{\omega}$. Let $s \in \bar{s}_0$ be such that $s \in \Psi(\Sigma_{fi})$ and let $t \in L/s$ be such that $\omega =$

st. By choosing k to be arbitrarily large, we can get $||t|| > n$ for any given $n \in N$. Thus, we have $\tilde{\omega} \in P_L^{-1}[P(st)]$ and $\Sigma_{fi} \notin \tilde{\omega}$. Therefore, the chosen s violates the definition of diagnosability for F_i. Hence L is not diagnosable.

We now prove that if L is diagnosable, then it satisfies condition (C2). By contradiction, let $q \in Q_d$ be ambiguous. Then for some $i \in \Pi_f$, $\exists$ F_i-ambiguous traces $s_1, s_2 \in L$ satisfying Lemma 1-(iii). Let $\delta(x_0, s_1) = x$. Since $\delta(x_0, s_1) = \delta(x_0, s_2)$, $t \in L/s_1$ iff $t \in L/s_2$. Since, by assumption, multiple failures from the same set of the partition do not occur, and since $\Sigma_{fi} \in s_1$, $\Sigma_{fi} \notin t \ \ \forall t \in L/s_1$. Hence, $s_2 t \in P_L^{-1}[P(s_1 t)]$ and $\Sigma_{fi} \notin s_2 t \ \forall t \in L/s_1$. Choosing $s = s_1$ and $\omega = s_2 t$, we see that Definition 1 is violated and L is not diagnosable.

Sufficiency: Assume that the diagnoser G_d for L satisfies conditions (C1) and (C2). Pick any $s \in L$ and any F_i such that $s \in \Psi(\Sigma_{fi})$ and let $\delta(x_0, s) = x$. Pick any $t_1 \in L_o(G, x)$. From Assumption (A2) of a finite bound n_o on the length any sequence of unobservable events in L, $||t_1|| \leq n_o$. Let $\delta(x_0, st_1) = x_1$ and correspondingly in G_d, let $\delta_d(q_0, P(st_1)) = q_1$. Since $\Sigma_{fi} \in st_1$, and since we assume that there are no ambiguous states in G_d, we have $(x_1, \ell_1) \in q_1$ with $F_i \in \ell_1$.

We now have two distinct cases to consider: (I) q_1 is F_i-certain and (II) q_1 is F_i-uncertain.
Case I: Suppose q_1 is F_i-certain. Then, by Lemma 1-(i),

$$(\forall \omega \in P_L^{-1}[P(st_1)]) \ \ \Sigma_{fi} \in \omega.$$

Hence L is diagnosable for F_i with $n_i = n_o$. Since this is true for any F_i, L is diagnosable.
Case II: Suppose q_1 is F_i-uncertain. Consider any $(z, \ell) \in q_1$ such that $F_i \in \ell$. We shall then refer to z as an "x-state" of q_1. Likewise, if $(z', \ell') \in q_1$ such that $F_i \notin \ell'$, then we shall denote z' as a "y-state" of q_1. We have assumed that there are no F_i-indeterminate cycles in G_d. Recalling the definition of an F_i-indeterminate cycle, this assumption means that one of the following is true. (i) There are no cycles of F_i-uncertain states in G_d. (ii) There exists one or more cycles of F_i-uncertain states $q_1, q_2, \ldots, q_n$ in G_d but corresponding to any such cycle in G_d, there do not exist two sequences $\{x_l^k\}$ and $\{y_l^r\}$, $l = 1, \ldots, n$, and $k, r \in N$ such that *both* of these form cycles in G', where the sequence $\{x_l^k\}$ is composed of "x-states" of q_l, and the sequence $\{y_l^r\}$ is composed of "y-states" of q_l, $l = 1, \ldots, n$.
Case (i) : Suppose that there are no cycles of F_i-uncertain states in G_d. Then this implies that every F_i-uncertain state should lead to an F_i-certain state in a bounded number of transitions by Condition (C2) and by Property (P3) of label propagation.
Case (ii) : Suppose that there exists a cycle of F_i-uncertain states $q_1, q_2, \ldots, q_n$ in G_d as in (ii) above. We now show that whenever a failure happens, i.e, when the true state of the system is an "x-state", it is not possible to loop for arbitrarily long in this cycle in G_d and thereby never detect the failure.

Pick any "y-state" $y_l \in q_l$ and let the corresponding label be $\tilde{\ell}_l$. Since $F_i \notin \tilde{\ell}_l$, the pair $(y_l, \tilde{\ell}_l) \in q_l$ could only have resulted from a pair $(y_{l-1}, \tilde{\ell}_{l-1}) \in q_{l-1}$ such that $F_i \notin \tilde{\ell}_{l-1}$ and not from any $(x_{l-1}, \ell_{l-1}) \in q_{l-1}$ where $F_i \in \ell_{l-1}$, because of Condition (C2) and Property (P3). That is, the "y-state" y_l cannot be a successor of any "x-state" x_{l-1} along the

corresponding trace in G'. Thus, by backward induction, we can always build a cycle of states in G' involving some or all of the "y-states" of $q_l,\ l = 1, \ldots, n$. These "y-state"s then constitute the sequence $\{y_l^r\}$. But since the cycle of F_i-uncertain states $q_1, q_2, \ldots, q_n$ is not F_i-indeterminate, there *cannot* be a corresponding cycle in G' involving the "x-states" of q_l, i.e., there cannot exist a sequence $\{x_l^k\}$ of "x-state"s that form a cycle in G'. Hence, if we pick any "x-state" x_l in any state q_l in the cycle in G_d, then a sufficiently long trace $p \in L(G, x_l)$ (guaranteed by the liveness assumption (A1)) will leave the cycle of F_i-uncertain states. Specifically, let l^x be the number of "x-states" in any $q_l,\ l = 1, \ldots, n$. Then, we can stay in the cycle formed by states $q_1, q_2, \ldots, q_n$ for as long as $\sum_{l=1}^{n} l^x$ transitions of G_d, before leaving it. Since this is true for any cycle of F_i-uncertain states in G_d, we can conclude that we will eventually hit an F_i-certain state from q_l.

Therefore, for both situations (i) and (ii), we conclude that $\forall t_2 \in L(G, x_1)$ of sufficiently long length, $\delta_d(q_1, P(t_2)) = \delta_d(q_0, P(st_1t_2)) = q_2$ is F_i-certain. Let $t = t_1t_2$. We conclude that $\exists n_i \in N$ such that $\forall t \in L/s$,

$$||t|| \geq n_i \Rightarrow (\forall \omega \in P_L^{-1}[P(st)])\ \ \Sigma_{fi} \in \omega.$$

Hence L is diagnosable. Further, we can obtain a bound on $n_i,\ \forall i \in \Pi_f$ as follows. First, recall that $||t_1|| \leq n_o$. Next, define

$$C_i = \sum_{q \in Q_d:\ q \text{ is } F_i\text{-uncertain}} \#x\text{-states in } q. \tag{3.19}$$

Finally, recall that at most n_o unobservable events can occur between any two observable events in L. Hence we have that

$$n_i \leq C_i \times n_o + n_o. \tag{3.20}$$

Q.E.D.

Example 12 Consider the systems represented in Figures 3.6 and 3.7. In both of these systems multiple failures of the same type do not occur along any trace. The diagnoser corresponding to the system in Figure 3.7 does not have any F_i-indeterminate cycle or ambiguous state and hence this system is diagnosable. The bound on the delay n_1 for this system is calculated from Equation 3.20 to be 6; inspection of the system reveals that the actual value of $n_1 = 6$. Inspection of the diagnoser shows that the detection delay for this system is also 6. (Here, $n_1 + n_o = 7$.) The system represented in Figure 3.6 is not diagnosable since the diagnoser G_d for this system contains an F_i-indeterminate cycle as explained earlier. Figure 3.5 represents another system that is not diagnosable. This again is an example of a system in which multiple failures are not possible; inspection of the diagnoser for this system reveals the presence of an ambiguous state. ■

Remark: One could interpret conditions (C1) and (C2) of Theorem 1 as generalizations to the case of diagnosability of Özveren and Willsky's conditions for invertibility stated in [32].

3.4.2 Conditions for Diagnosability-The Case of Multiple Failures

We now consider the case of possible multiple failures from the same set of the partition. First, recall that when more than one failure event of the same failure type occurs along any trace of the system, our definition of diagnosability does not require that all of these events be detected. We only require that it be possible to conclude with finite delay (after the first occurrence of a failure) that a failure event of that type happened. This is what distinguishes the case of multiple failures from the case of no multiple failures and leads to the following consequences on the diagnosability of a language.

In the case of no multiple failures discussed in the last section, we saw that a necessary condition for L to be diagnosable is that no state of Q_d is ambiguous. In other words, L should contain no two F_i-ambiguous traces $\forall i \in \Pi_f$. Such a requirement is not necessary when we allow for the possibility of multiple failures. Recall from Lemma 1-(iii) that any two F_i-ambiguous traces s_1 and s_2 produce the same record of observable events, and in addition, share the same future behaviour. Thus, no future observations can help identify which of the two traces was actually executed by the system. However, if every trace in the post-language of these ambiguous traces contains failure events of the same type that caused the ambiguity, namely, failures from the set Σ_{fi} occurring in a bounded number of transitions of the system following the first occurrence of the failure, and if it were possible to detect with finite delay the occurrence of these failures, the language L would still satisfy our condition of diagnosability. Hence, presence of two ambiguous traces does not necessarily imply that L is not diagnosable. To determine in the case of multiple failures if L is indeed diagnosable, one needs to record what failure types caused the ambiguity and test if these failure types reappear. For these reasons, the "basic" diagnoser introduced in Section 3.4.1 is not adequate for checking diagnosability of a language in which mutiple failures of the same type are possible. In this regard, we now introduce some modifications to the diagnoser G_d of Section 3.4.1 and obtain the diagnoser G_d^{mf} as follows.

The Diagnoser G_d^{mf}

We define the new set of possible labels

$$\Delta^{mf} = \{N\} \cup 2^{\Delta_f} \tag{3.21}$$

(as opposed to $\{N\} \cup 2^{\{\Delta_f \cup \{A\}\}}$ in the previous case). The modified diagnoser for G is the FSM

$$G_d^{mf} = (Q_d^{mf}, \Sigma_o, \delta_d^{mf}, q_0). \tag{3.22}$$

Here $q_0 = \{(x_0, \{N)\}\}$ as before, and The Label Propagation function LP^{mf}, the Range function R, the transition function δ_d^{mf}, and the state space Q_d^{mf} of G_d^{mf} are defined as follows.

Definition 9 *The Label Propagation function* $LP^{mf} : X_o \times \Delta^{mf} \times \Sigma^* \to \Delta^{mf}$.
Given $x \in X_o$, $\ell \in \Delta^{mf}$, *and* $s \in L_o(G, x)$, LP^{mf} *propagates the label* ℓ *over* s, *starting*

from x and following the dynamics of G, i.e., according to $L(G,x)$. It is defined by

$$LP^{mf}(x,\ell,s) = \begin{cases} \{N\} & \text{if } \ell = \{N\} \ \wedge \ \forall i \ [\Sigma_{fi} \notin s] \\ \{F_i : F_i \in \ell \ \vee \ \Sigma_{fi} \in s\} & \text{otherwise.} \end{cases}$$

The Range function $R : Q_o \times \Sigma_o \rightarrow Q_o$ is the same as before but with the new LP^{mf}.

$$R(q,\sigma) = \bigcup_{(x,\ell)\in q} \ \bigcup_{s\in L_\sigma(G,x)} \{(\delta(x,s), LP^{mf}(x,\ell,s))\}.$$

The Label Correction function LC, which assigns the A label, is now dropped.

The transition function $\delta_d^{mf} : Q_o \times \Sigma_o \rightarrow Q_o$ is now defined as

$$q_2 = \delta_d^{mf}(q_1,\sigma) \Leftrightarrow q_2 = R(q_1,\sigma) \tag{3.23}$$

with $\sigma \in e_d(q_1)$ defined as before. The state space Q_d^{mf} is the resulting subset of Q_o composed of the states of the diagnoser that are reachable from q_0 under the transition function δ_d^{mf}.

Properties and Definitions of G_d^{mf}

We now restate some of the properties of the diagnoser and the definitions of Section 3.4.1 taking into account the modifications discussed above. Note that we have now dropped the Label Correction function LC. Therefore, Property (P1) no longer holds, i.e., there may exist $q \in Q_d$ such that $(x,\ell),(x,\ell') \in q$ with $\ell \neq \ell'$. Property (P2) remains true for the case of multiple failures. Property (P3) is restated as follows.

(P3-MF) Let $q_1, q_2 \in Q_d$ and $s \in \Sigma^*$ such that $(x_1,\ell_1) \in q_1$, $(x_2,\ell_2) \in q_2$, $\delta(x_1,s) = x_2$ and $\delta_d(q_1,P(s)) = q_2$. Then

$$F_i \notin \ell_2 \Rightarrow F_i \notin \ell_1.$$

Property (P3-MF) simply states that the failure labels F_i propagate from state to state and if along any trace s of L a state x receives an F_i label, then every successor x' of x also carries the F_i label.

The definition of an F_i-certain state holds as before and the definition of an ambiguous state is now irrelevant. We add to the definitions of an F_i-uncertain state and an F_i-indeterminate cycle, respectively, the qualifiers "x not necessarily distinct from y" and "x_l^k not necessarily distinct from y_l^r". Therefore, states $q \in Q_d$ that were ambiguous in the case of no multiple failures are now F_i-uncertain states.

Lemma 1 is restated as follows:

Lemma 2 *(i) Let $\delta_d(q_0,u) = q$. If q is F_i-certain, then $(\forall \omega \in P_L^{-1}(u)) \ \ \Sigma_{fi} \in \omega$.*

(ii) If a state $q \in Q_d$ is F_i-uncertain , then this implies that $\exists\ s_1, s_2 \in L$ such that $\Sigma_{fi} \in s_1$, $\Sigma_{fi} \notin s_2$, $P(s_1) = P(s_2)$, and $\delta_d(q_0,P(s_1)) = q$.

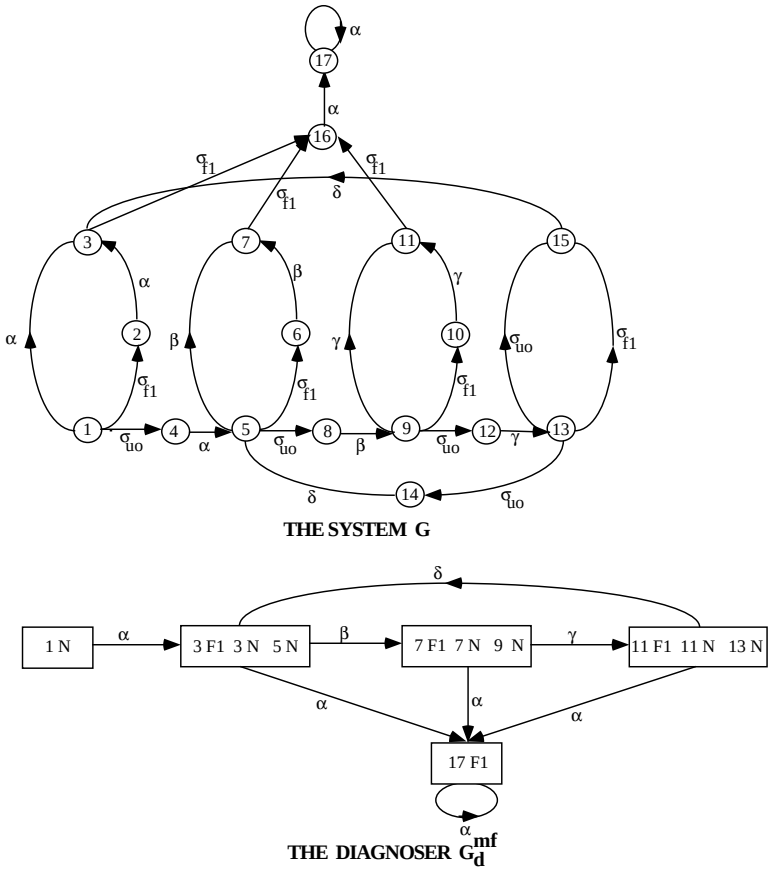


Figure 3.9: Example illustrating construction of the diagnoser G_d^{mf} for the case of multiple failures

The proof of the above lemma is obvious by the construction of the diagnoser G_d^{mf}. Note that Lemma 1-(ii) and 1-(iii) of Section 3.4.1 have been restated together as Lemma 2-(ii) since ambiguous states have now become F_i-uncertain states.

Example 13 Figure 3.9 illustrates construction of the diagnoser G_d^{mf} for the case of multiple failures. In this system α, β, γ and δ are observable events while σ_{uo} is unobservable. The only failure event in the system is σ_{f1} and hence the partition is given by $\Sigma_{f1} = \{\sigma_{f1}\}$.

Necessary and Sufficient Conditions

Theorem 2 *A language L is diagnosable if and only if its diagnoser G_d^{mf} satisfies the following condition:*

(C-MF) *There are no F_i-indeterminate cycles in G_d, for all failure types F_i.*

Proof: The proof of the necessity of the above condition is identical to the proof of the necessity of Condition (C1) of Theorem 1 since the latter proof does not require that the

x_l^ks and the y_l^rs be distinct. The proof of the sufficiency of Condition (C-MF) is essentially the same as the proof of the sufficiency of Conditions (C1) and (C2) of Theorem 1. The only difference is that the absence of ambiguous states is true by assumption in the case of Theorem 1 whereas it is true by construction in the present case. Hence, reasoning along lines identical to the proof of Theorem 1, we conclude that the condition of no F_i-indeterminate cycles in G_d^{mf}, for all failure types F_i, is necessary and sufficient for L to be diagnosable in the case of multiple failures. Further, reasoning as before, we have the following bound on the delay n_i, $\forall i \in \Pi_f$:

$$n_i \leq C_i^{mf} \times n_o + n_o \tag{3.24}$$

where

$$C_i^{mf} = \sum_{q \in Q_d^{mf}:\ q \text{ is } F_i\text{-uncertain}} \#x\text{-states in } q. \tag{3.25}$$

However, this is a very conservative bound. In Chapter 4 we shall provide a better bound on the delay n_i. **Q.E.D.**

Example 14 Figure 3.9 represents a system where multiple failures of the same type are possible. This system is diagnosable since it is easily verified that the cycle of F_1-uncertain states in the corresponding diagnoser G_d^{mf} is not F_1-indeterminate. ■

3.4.3 Conditions for I-Diagnosability

We now study necessary and sufficient conditions for a language to be I-diagnosable. Recall from Section 3.3.2 that in the case of I-diagnosability we are interested in detecting failure events only after the occurrence of the corresponding indicator events, i.e., we require the diagnosability condition D to hold only for those traces in which an indicator event follows a failure event. Based on this requirement, we introduce the following modifications to the basic diagnoser G_d and obtain the diagnoser G_d^I as follows.

The Diagnoser G_d^I

We define, as before, the set of failure labels $\Delta_f = \{F_1, F_2, ...F_m\}$ where $|\Pi_f| = m$. In addition, we now define a set of *indicator labels* $\Delta_i = \{I_1, I_2, ..., I_m\}$. We interpret $\{I_{i1}, \ldots, I_{ik}\}$ as meaning that indicator events of the type I_{i1} through I_{ik} have occurred. The complete set of possible labels is now defined as

$$\Delta^I = \{N\} \ \cup \ 2^{\Delta_f \ \cup \ \Delta_i} \tag{3.26}$$

with the restriction that

$$(\forall \ell \in \Delta^I) \ \ I_i \in \ell \Rightarrow F_i \in \ell$$

(explained in the subsequent paragraphs).

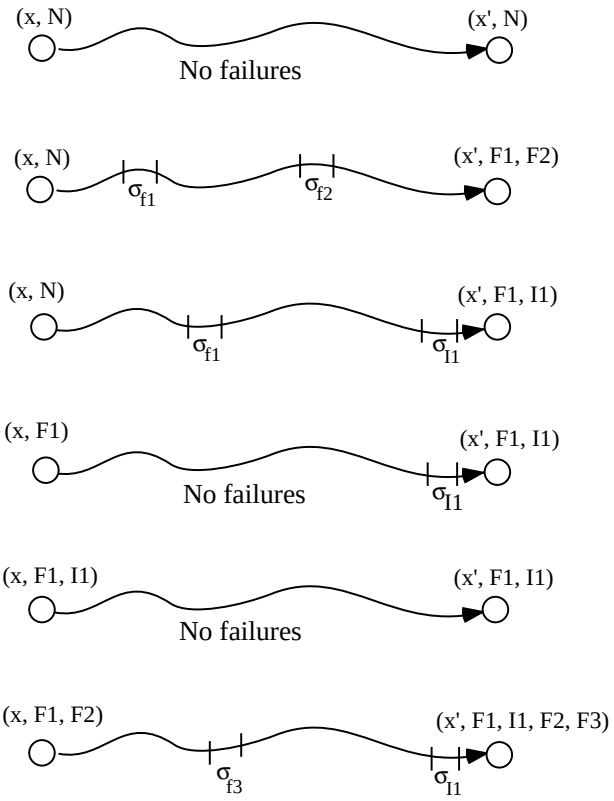


Figure 3.10: Propagation of labels along traces of L

The modified diagnoser G_d^I is the FSM

$$G_d^I = (Q_d^I, \Sigma_o, \delta_d^I, q_0) \tag{3.27}$$

with the initial state $q_0 = \{(x_0, \{N\})\}$ as in Section 3.4.1. The Label Propagation function LP^I, the Range function R, the Label Correction function LC^I, the transition function δ_d^I, and the state space Q_d^I of G_d^I are defined as follows.

Definition 10 *The Label Propagation Function* $LP^I : X_o \times \Delta^I \times \Sigma^* \to \Delta^I$.
Given $x \in X_o$, $\ell \in \Delta^I$, *and* $s \in L_o(G, x)$, LP^I *propagates the label* ℓ *over* s, *starting from* x *and following the dynamics of* G, *i.e., according to* $L(G, x)$. *It is defined by*

$$LP^I(x, \ell, s) = \begin{cases} \{N\} \text{ if } \ell = \{N\} \ \wedge \ \forall i \ [\Sigma_{fi} \notin s] \\ \{F_i : F_i \in \ell \ \vee \ \Sigma_{fi} \in s\} \ \bigcup \\ \quad \{I_i : I_i \in \ell \ \vee \ [I(\Sigma_{fi}) \in s \ \wedge \ (F_i \in \ell \ \vee \ \Sigma_{fi} \in s)]\} \text{ otherwise.} \end{cases}$$

Figure 3.10 illustrates propagation of labels according to LP^I as defined above. Here σ_{f1} refers to a failure event of type F_1 and σ_{I1} refers to an indicator event of type I_1. The Range Function $R : Q_o \times \Sigma_o \to Q_o$ is defined to be

$$R(q, \sigma) = \bigcup_{(x,\ell) \in q} \ \bigcup_{s \in L_\sigma(G,x)} \{(\delta(x, s), LP^I(x, \ell, s))\}.$$

Definition 11 *The Label Correction Function $LC^I : Q_o \rightarrow Q_o$ is defined as follows:*

$$LC^I(q) \quad = \quad q \Leftrightarrow \{(x,\ell) \in q : (x,\ell') \in q \ \wedge \ \forall i \ [F_i \in \ell \Leftrightarrow F_i \in \ell'] \ \wedge \ [\ell \subset \ell']\}$$

The use of the Label Correction function LC^I is explained as follows. Suppose there exist two pairs (x,ℓ) and (x,ℓ') as described above, in $R(q,\sigma)$ for some state q of G_d^I. This implies the presence in L of two traces s_1 and s_2 such that they have identical projections and lead to the same state x, and s_1 contains an indicator event of type I_i following a failure event of type F_i while s_2 does not. Since for I-diagnosability, we are concerned only with traces in which the indicator event follows the failure, we can drop the pair (x,ℓ) which does not contain the I_i label with no loss of generality.

The transition function $\delta_d^I : Q_o \times \Sigma_o \rightarrow Q_o$ is defined as

$$q_2 = \delta_d^I(q_1,\sigma) \Leftrightarrow q_2 = LC^I[R(q_1,\sigma)] \tag{3.28}$$

with $\sigma \in e_d(q_1)$ defined as before. The state space Q_d^I is the resulting subset of Q_o composed of the states of the diagnoser that are reachable from q_0 under the transition function δ_d^I. A state q_d of G_d^I is now of the form

$$q_d = \{(x_1,\ell_1),\ldots,(x_n,\ell_n)\}$$

where $x_i \in X_o$ and $\ell_i \in \Delta^I$, i.e., ℓ_i is of the form $\ell_i = \{N\}$ or $\ell_i = \{F_{i_1}, F_{i_2}, \ldots, F_{i_k}, I_{j_1}I_{j_2}, \ldots, I_{j_l}\}$ where $\{i_1,i_2,\ldots,i_k\} \subseteq \{1,2,\ldots,m\}$ and $\{j_1,j_2,\ldots,j_l\} \subseteq \{i_1,i_2,\ldots,i_k\}$.

We make the following observations on the modified diagnoser G_d^I.

1. In addition to failure information, the labels now carry information on occurrences of indicator events following the failure events.

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 I_i labels propagate from state to state just like the F_i labels.

4. We do not use the A label here. As mentioned earlier, we are now concerned only with traces where the failure event is followed by an appropriate indicator event. Therefore, there could be present in L two F_i-ambiguous traces for some $i \in \Pi_f$ and yet L could be diagnosable if no trace in the post-language of these traces contains an indicator event from the set $I(\Sigma_{fi})$. Hence, in order to check for I-diagnosability, we need to remember which failure types caused the ambiguity, even in the case of no multiple failures. Therefore, we do not need to distinguish between the case of possible multiple failures and the case of no multiple failures in this section.

Example 15 Figure 3.11 illustrates the construction of the diagnoser G_d^I. Here, σ_i, $i \in \{1,\ldots,4\}$ and σ_{I1} are observable events while σ_{uo} is unobservable. The indicator event correponding to the failure event σ_{f1} is $I(\sigma_{f1}) = \{\sigma_{I1}\}$ and the partition is $\Sigma_{f1} = \{\sigma_{f1}\}$. ■

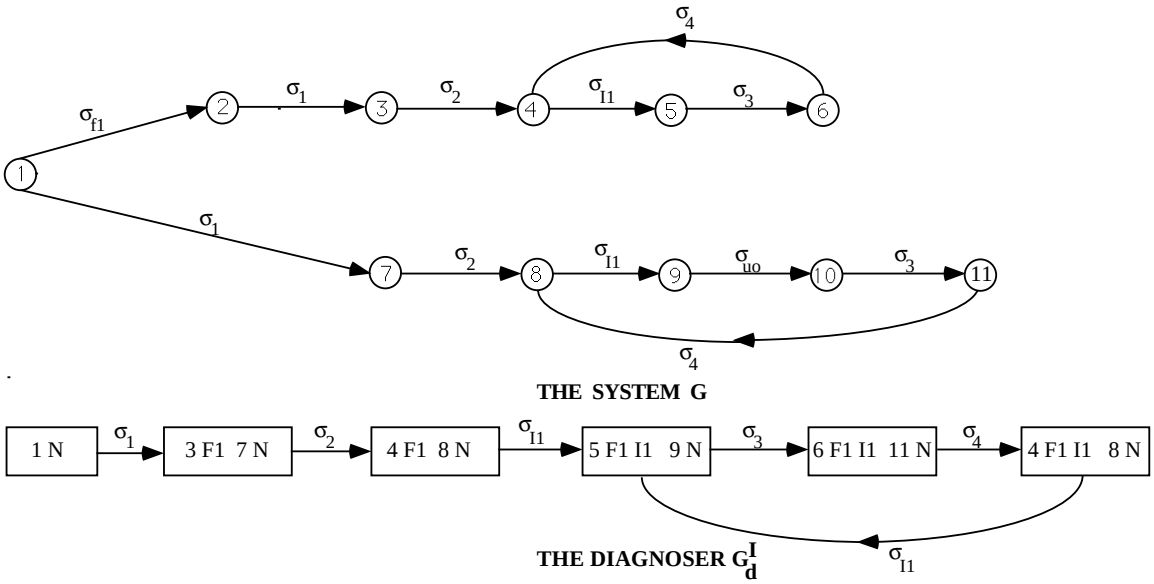


Figure 3.11: Example illustrating construction of the diagnoser G_d^I

Properties and Definitions of G_d^I

Since we do not use the A label, properties (P1) through (P3) of the diagnoser correspond to those discussed in Section 3.4.2. Likewise, the remarks on the definition of an F_i-certain state, an F_i-uncertain state, and an F_i-indeterminate cycle, and Lemma 2 cited in Section 3.4.2 remain valid. We now introduce the notions of (F_i, I_i)-uncertain states and (F_i, I_i)-indeterminate cycles.

Definition 12 *A state $q \in Q_d^I$ is said to be (F_i, I_i)-**uncertain** if $\exists (x, \ell), (y, \ell') \in q$, such that $\{F_i, I_i\} \subseteq \ell$ and $F_i \notin \ell'$.*

Lemma 3 *If a state $q \in Q_d$ is (F_i, I_i)-uncertain, then this implies that $\exists\ s_1 = p_1 t_1 \in L$ and $s_2 \in L$ such that : $p_1 \in \Psi(\Sigma_{fi})$, $I(\Sigma_{fi}) \in t_1$, $\Sigma_{fi} \notin s_2$, $P(s_1) = P(s_2)$, and $\delta_d^I(q_0, P(s_1)) = q$.*

The above lemma simply states that presence of an (F_i, I_i)-uncertain state in G_d^I corresponds to the situation where there are two traces s_1 and s_2 in L such that s_1 contains a failure event of type F_i followed by an indicator event corresponding to this failure type while s_2 does not contain a failure event of type F_i. In addition, the traces s_1 and s_2 produce the same record of observable events. Proof of this lemma follows directly from the construction of G_d^I.

Definition 13 *A set of (F_i, I_i)-uncertain states $q_1, q_2, \ldots, q_n \in Q_d^I$ is said to form an (F_i, I_i)-**indeterminate** cycle if*

1. *$q_1, q_2, \ldots, q_n$ form a cycle in G_d^I with $\delta_d^I(q_l, \sigma_l) = q_{l+1}$, $l = 1, \ldots, n \Leftrightarrow 1$ and $\delta_d^I(q_n, \sigma_n) = q_1$ where $\sigma_l \in \Sigma_o$, $l = 1, \ldots, n$; and*

2. $\exists (x_l^k, \ell_l^k), (y_l^r, \tilde{\ell}_l^r) \in q_l$, $l = 1, \ldots, n$, $k = 1, \ldots, m$, *and* $r = 1, \ldots, m'$ *(x not necessarily distinct from y), such that*

   (a) $\{F_i, I_i\} \subseteq \ell_l^k$, $F_i \notin \tilde{\ell}_l^r$ *for all* l, k, *and* r;

   (b) *The sequences of states* $\{x_l^k\}$, $l = 1, \ldots, n$, $k = 1, \ldots, m$ *and* $\{y_l^r\}$, $l = 1, \ldots, n$, $r = 1, \ldots, m'$ *form cycles in* G' *with*

$$\begin{aligned} (x_l^k, \sigma_l, x_{(l+1)}^k) &\in \delta_{G'}, \quad l = 1, \ldots, n \Leftrightarrow 1, \quad k = 1, \ldots, m, \\ (x_n^k, \sigma_n, x_1^{k+1}) &\in \delta_{G'}, \quad k = 1, \ldots, m \Leftrightarrow 1, \quad \textit{and} \\ (x_n^m, \sigma_n, x_1^1) &\in \delta_{G'}, \end{aligned}$$

   and

$$\begin{aligned} (y_l^r, \sigma_l, y_{(l+1)}^r) &\in \delta_{G'}, \quad l = 1, \ldots, n \Leftrightarrow 1, \quad r = 1, \ldots, m,' \\ (y_n^r, \sigma_n, y_1^{r+1}) &\in \delta_{G'}, \quad r = 1, \ldots, m' \Leftrightarrow 1, \quad \textit{and} \\ (y_n^{m'}, \sigma_n, y_1^1) &\in \delta_{G'}. \end{aligned}$$

An (F_i, I_i)-indeterminate cycle in G_d^I indicates the presence in L of two traces s_1 and s_2 of arbitrarily long length, such that they both have the same observable projection, and s_1 contains a failure event from the set Σ_{fi} followed by an indicator event from the set $I(\Sigma_{fi})$ while s_2 does not contain any event from the set Σ_{fi}.

Example 16 Consider the system shown in Figure 3.11. Inspection of the diagnoser G_d^I for this system reveals the presence of an (F_i, I_i)-indeterminate cycle. Here the set $\{x_l^k\}$ of Definition 13 is $\{5, 6, 4\}$ (these states carry the label $\{F_1, I_1\}$ in the diagnoser G_d^I), the set $\{y_l^r\}$ is $\{9, 11, 8\}$ (these states carry the label $\{N\}$), and $m = m' = 1$. ■

Necessary and Sufficient Conditions

Theorem 3 *A language* L *is I-diagnosable if and only if the diagnoser* G_d^I *satisfies the following condition:*

(C-I) *There are no* (F_i, I_i)*-indeterminate cycles in* G_d^I*, for all failure types* F_i*.*

Proof: This proof is very similar to the proof of Theorem 1 with the exceptions that we now consider the I labels and (F_i, I_i)-indeterminate cycles and that there are no ambiguous states. For the sake of clarity, it is presented in its entirety.

Necessity: We prove necessity by contradiction. Assume there exist states $q_1, q_2, \ldots, q_n \in Q_d^I$ such that they form an (F_i, I_i)-indeterminate cycle and let $\delta_d^I(q_i, \sigma_i) = q_{(i+1) \bmod n}$. Let $(x_l^k, \ell_l^k), (y_l^r, \tilde{\ell}_l^r) \in q_l$, $l = 1, \ldots, n$, $k = 1, \ldots, m$, and $r = 1, \ldots, m'$ form corresponding cycles in G' with $\{F_i, I_i\} \subseteq \ell_l^k$, $F_i \notin \tilde{\ell}_l^r$. Then we have

$$\begin{aligned} \delta(x_l^k, s_l^k \sigma_l) &= x_{(l+1)}^k, \quad l = 1, \ldots, n \Leftrightarrow 1, \quad k = 1, \ldots, m \\ \delta(x_n^k, s_n^k \sigma_n) &= x_1^{k+1}, \quad k = 1, \ldots, m \Leftrightarrow 1 \text{ and} \\ \delta(x_n^m, s_n^m \sigma_n) &= x_1^1 \end{aligned}$$

and

$$\begin{aligned}\delta(y_l^r, \tilde{s}_l^r\sigma_l) &= y_{(l+1)}^r, \quad l = 1,\ldots,n \Leftrightarrow 1, \quad r = 1,\ldots,m' \\ \delta(y_n^r, \tilde{s}_n^r\sigma_n) &= y_1^{r+1}, \quad r = 1,\ldots,m' \Leftrightarrow 1 \text{ and} \\ \delta(y_n^{m'}, \tilde{s}_n^{m'}\sigma_n) &= y_1^1\end{aligned}$$

where

$$m, m' \in N\ ,\ s_l^k \in L(G, x_l^k),\ \tilde{s}_l^r \in L(G, y_l^r) \text{ and } s_l^k, \tilde{s}_l^r \in \Sigma_{uo}^*.$$

Since (x_1^1, ℓ_1^1), $(y_1^1, \tilde{\ell}_1^1) \in q_1$, $\exists\, s_0, \tilde{s}_0 \in L$ such that $\delta(x_0, s_0) = x_1^1$, $\delta(y_0, \tilde{s}_0) = y_1^1$ and $P(s_0) = P(\tilde{s}_0)$ from Property (P2). Further, since $\{F_i, I_i\} \subseteq \ell_1^1$, then $\Sigma_{fi} \in s_0$, and $\exists st_1 \in \overline{s_0}$ such that $s \in \Psi(\Sigma_{fi})$ and $st_1 \in \Psi[I(\Sigma_{fi})]$, i.e., the trace st_1 contains a failure event of the type F_i and ends in an indicator event associated with the failure type F_i. Also, since $F_i \notin \tilde{\ell}_l^r$, we have that $\Sigma_{fi} \notin \tilde{s}_0$ and $\Sigma_{fi} \notin \tilde{s}_l^r$, for all l, r.

Consider the two traces

$$\omega = s_0(s_1^1\sigma_1 s_2^1\sigma_2 \ldots s_n^1\sigma_n s_1^2\sigma_1 s_2^2\sigma_2 \ldots s_n^2\sigma_n \ldots\ldots s_1^m\sigma_1 s_2^m\sigma_2 \ldots s_n^m\sigma_n)^{km'}$$

and

$$\tilde{\omega} = \tilde{s}_0(\tilde{s}_1^1\sigma_1 \tilde{s}_2^1\sigma_2 \ldots \tilde{s}_n^1\sigma_n \tilde{s}_1^2\sigma_1 \tilde{s}_2^2\sigma_2 \ldots \tilde{s}_n^2\sigma_n \ldots\ldots \tilde{s}_1^{m'}\sigma_1 \tilde{s}_2^{m'}\sigma_2 \ldots \tilde{s}_n^{m'}\sigma_n)^{km}$$

for arbitrarily large k. We have that $\omega, \tilde{\omega} \in L$, and $P(\omega) = P(\tilde{\omega}) = P(s_0)(\sigma_1\sigma_2\ldots\sigma_n)^{kmm'}$. Let $t_2 \in L/st_1$ be such that $\omega = st_1t_2$. By choosing k to be arbitrarily large, we can get $||t_2|| > n$ for any given $n \in N$. Thus, $\tilde{\omega} \in P_L^{-1}[P(st)]$ and $\Sigma_{fi} \notin \tilde{\omega}$. Therefore, the chosen s violates the definition of I-diagnosability for F_i. Hence L is not I-diagnosable.

Sufficiency: Assume that the diagnoser G_d^I for L satisfies Condition (C-I). Pick any $s \in L$ and any F_i such that $s \in \Psi(\Sigma_{fi})$ and any $t_1 \in L/s$ such that $t_{1f} \in I(\Sigma_{fi})$. Let $\delta(x_0, st_1) = x_1$ and correspondingly in G_d^I, let $\delta_d^I(q_0, P(st_1)) = q_1$. Since $\Sigma_{fi} \in st_1$ and $t_{1f} \in I(\Sigma_{fi})$, we have $(x_1, \ell_1) \in q_1$ with $\{F_i, I_i\} \subseteq \ell_1$.

We now have two distinct cases to consider: (I) q_1 is F_i-certain and (II) q_1 is (F_i, I_i)-uncertain.

Case I: Suppose q_1 is F_i-certain. Then, by Lemma 2-(i)

$$(\forall \omega \in P_L^{-1}[P(st_1)])\ \ \Sigma_{fi} \in \omega.$$

Hence L is I-diagnosable for F_i with $n_i = 0$. Since this is true for any F_i, L is I-diagnosable.

Case II: Suppose q_1 is (F_i, I_i)-uncertain. We have assumed that there are no (F_i, I_i)-indeterminate cycles in G_d^I. Recalling the definition of an (F_i, I_i)-indeterminate cycle, this assumption means that one of the following is true. (i) There are no cycles of (F_i, I_i)-uncertain states in G_d^I. (ii) There exists one or more cycles of (F_i, I_i)-uncertain states in G_d^I but corresponding to any of these cycles in G_d^I, there do not exist two sequences $\{x_l^k\}$ and $\{y_l^r\}$, $l = 1,\ldots,n$ and $k, r \in N$ such that *both* of these form cycles in G', where the sequence $\{x_l^k\}$ is composed of states that appear with an $\{F_i, I_i\}$ label in the cycle in G_d^I while the sequence $\{y_l^r\}$ is composed of states that do not appear with an F_i label.

Reasoning along lines similar to the proof of sufficiency of conditions (C1) and (C2) of Theorem 1, we conclude as before that $\forall t_2 \in L(G, x_1)$ of sufficiently long length, $\delta_d^I(q_1, P(t_2)) = \delta_d^I(q_0, P(st_1t_2)) = q_2$ is F_i-certain. Note that q_2 cannot be F_i-uncertain. This is because q_1 is (F_i, I_i)-uncertain, and no (F_i, I_i)-uncertain state of G_d^I can lead to an F_i-uncertain state since the I_i labels propagate from state to state. Hence we conclude that $\exists n_i \in N$ such that $\forall t_2 \in L/st_1$,

$$||t_2|| \geq n_i \Rightarrow (\forall \omega \in P_L^{-1}[P(st_1t_2)]) \;\; \Sigma_{fi} \in \omega.$$

Hence L is I-diagnosable. Further,

$$n_i \leq C_i^I \times n_o \tag{3.29}$$

where

$$C_i^I = \sum_{q \in Q_d^I:\ q \text{ is } (F_i, I_i)\text{-uncertain}} \#x\text{-states in } q. \tag{3.30}$$

We note here that this bound on the delay n_i is conservative; in Section 4 we provide a better bound. **Q.E.D.**

Example 17 Figure 3.11 provides an example of a system that is not I-diagnosable since the corresponding diagnoser G_d^I contains an (F_1, I_1)-indeterminate cycle. ■

This concludes the discussion on necessary and sufficient conditions for diagnosability and I-diagnosability. Note that checking for diagnosability and I-diagnosability amounts to cycle detection in the diagnosers and in G' and any of the standard cycle detection algorithms (which are of polynomial complexity) may be used.

3.5 Additional Examples

Example 18 Consider a simple system consisting of a pump, a valve, and a controller. We assume that the valve has two failure modes, a stuck-open failure mode and a stuck-closed failure mode; the stuck open failure occurs only from the open state of the valve and the stuck closed failure occurs only from its closed state. The component models for this system are shown in Figure 3.12.

Suppose that the system is equipped with just one sensor, a flow sensor that can read one of two possible values: F, indicating that there is a flow and NF, indicating that there is no flow. The sensor map for this system is listed in Table 3.1. Note that the •s in Table 3.1 stand for the state of the controller, and are used to indicate that the sensor map is independent of the controller state.

The composite system model G (obtained via the modeling procedure of Chapter 2) is depicted in Figure 3.13.

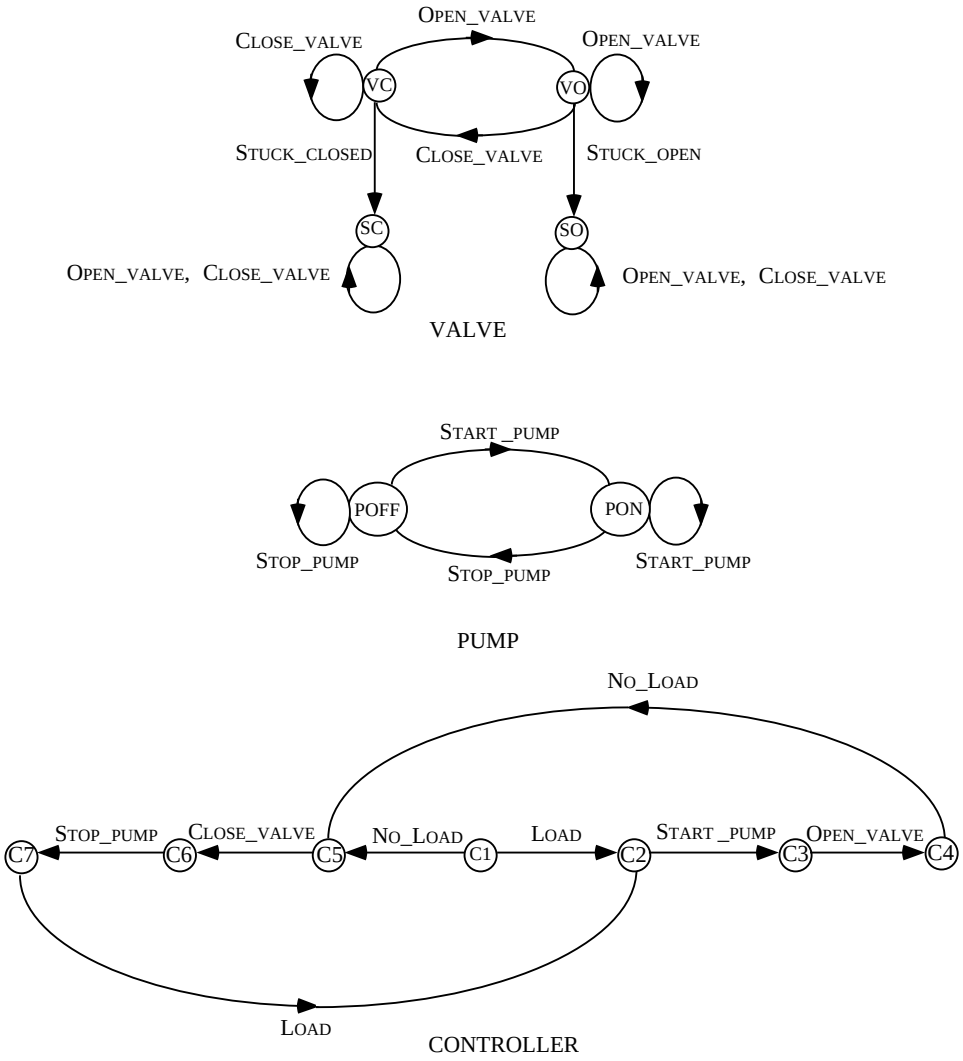


Figure 3.12: Component models for the pump-valve system of Example 18

$h($ VC, POFF, $\bullet$ $)$	=	NF
$h($ SC, POFF, $\bullet$ $)$	=	NF
$h($ SO, POFF, $\bullet$ $)$	=	NF
$h($ VC, PON, $\bullet$ $)$	=	NF
$h($ VO, PON, $\bullet$ $)$	=	F
$h($ SC, PON, $\bullet$ $)$	=	NF
$h($ SO, PON, $\bullet$ $)$	=	F

Table 3.1: The global sensor map for the pump-valve system of Example 18

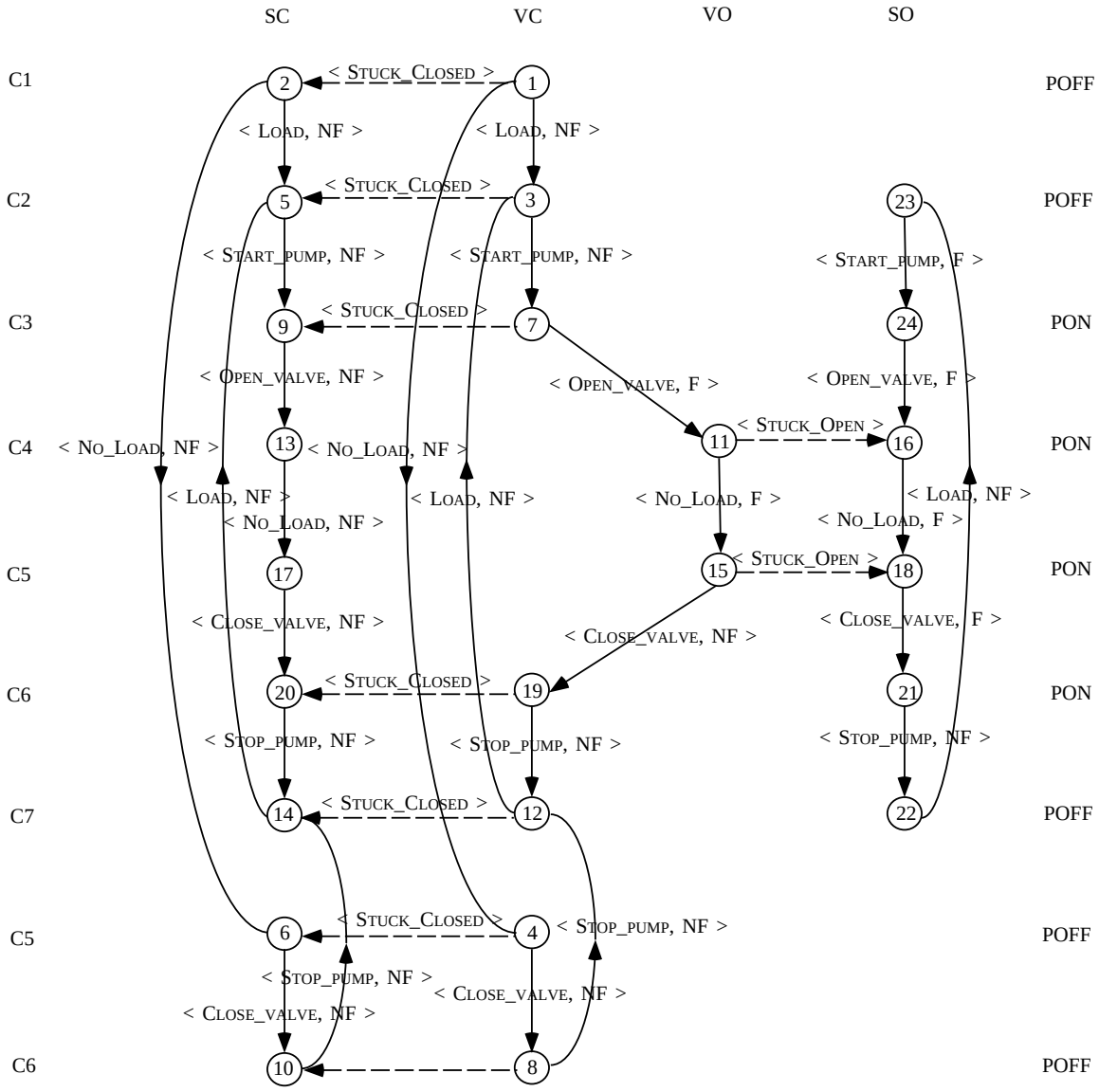


Figure 3.13: The composite system model G for the pump-valve system of Example 18

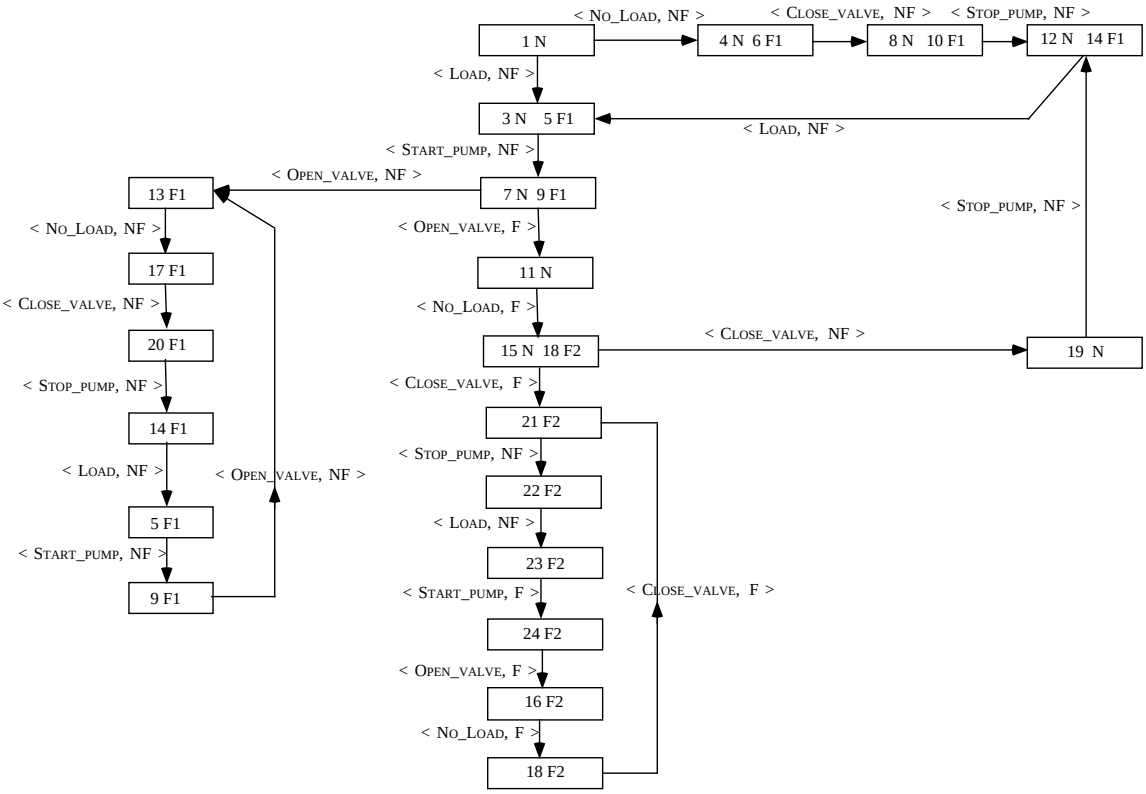


Figure 3.14: The diagnoser G_d for the pump-valve system of Example 18

Let the partition Π_f on the set of failures be chosen as follows: $\Sigma_{f1} = \{\text{STUCK\_CLOSED}\}$ and $\Sigma_{f2} = \{\text{STUCK\_OPEN}\}$. Figure 3.14 illustrates the diagnoser G_d for this system. Inspection of the diagnoser reveals that there are no cycles of F_1-uncertain states or F_2-uncertain states and hence there are no F_1 or F_2-indeterminate cycles in the diagnoser. Therefore, we conclude that the system of Figure 3.13 is diagnosable. ■

Example 19 Consider the pump-valve system presented in Example 18. Suppose that the controller of Figure 3.12 is replaced with the controller of Figure 3.15. The global sensor map and the composite system model G for this system are shown in Table 3.2 and Figure 3.16, respectively.

$h($ VC, POFF, $\bullet$ $)$	$=$	NF
$h($ VO, POFF, $\bullet$ $)$	$=$	NF
$h($ SC, POFF, $\bullet$ $)$	$=$	NF
$h($ SO, POFF, $\bullet$ $)$	$=$	NF
$h($ VO, PON, $\bullet$ $)$	$=$	F
$h($ SC, PON, $\bullet$ $)$	$=$	NF
$h($ SO, PON, $\bullet$ $)$	$=$	F

Table 3.2: The global sensor map for the pump-valve system of Example 19

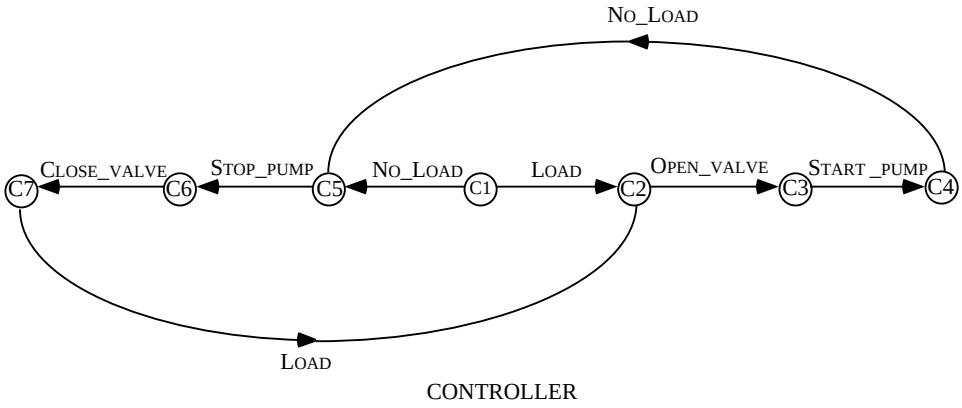


Figure 3.15: Controller model for the pump-valve system of Example 19

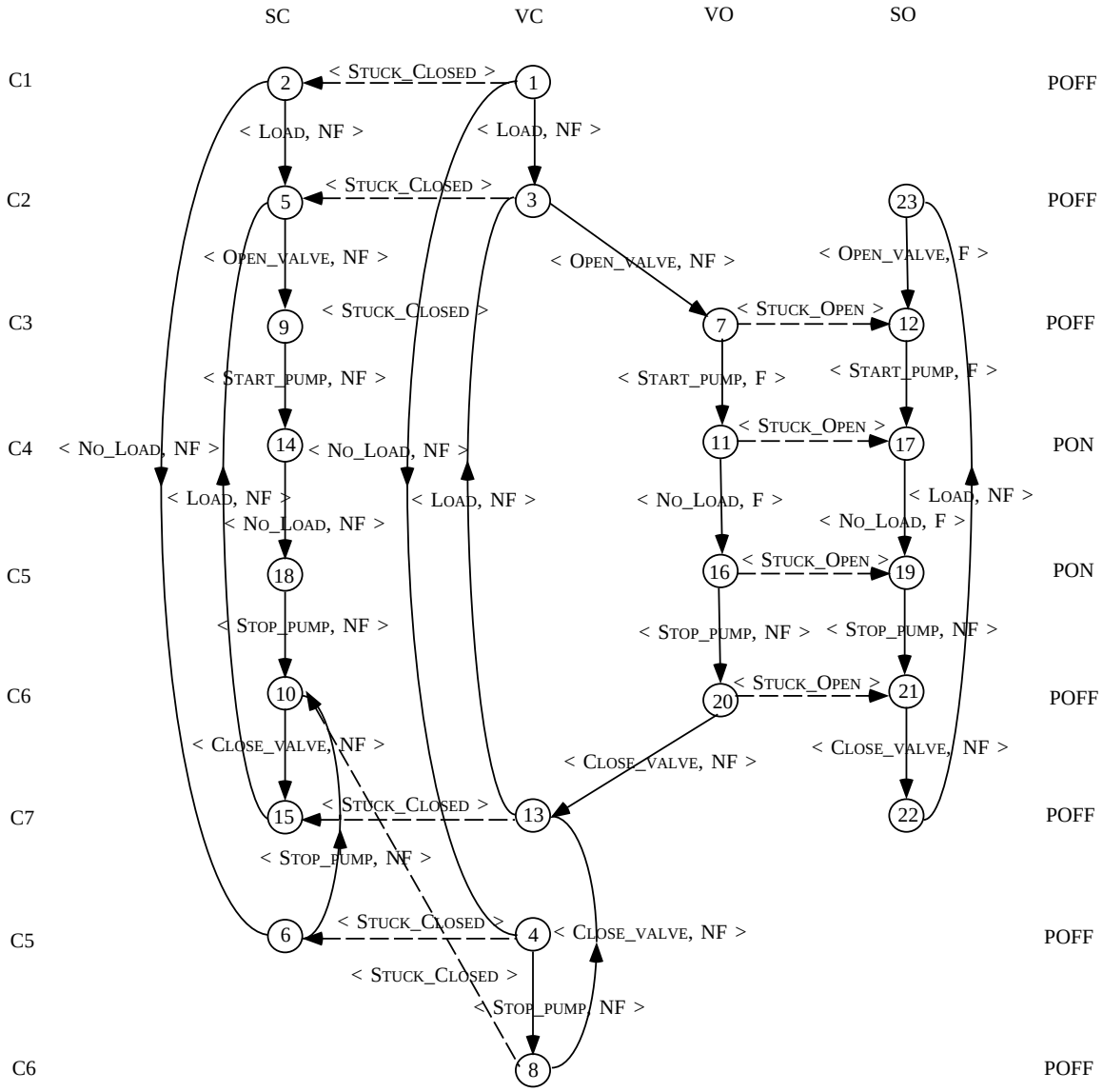


Figure 3.16: The composite system model G for the pump-valve system of Example 19

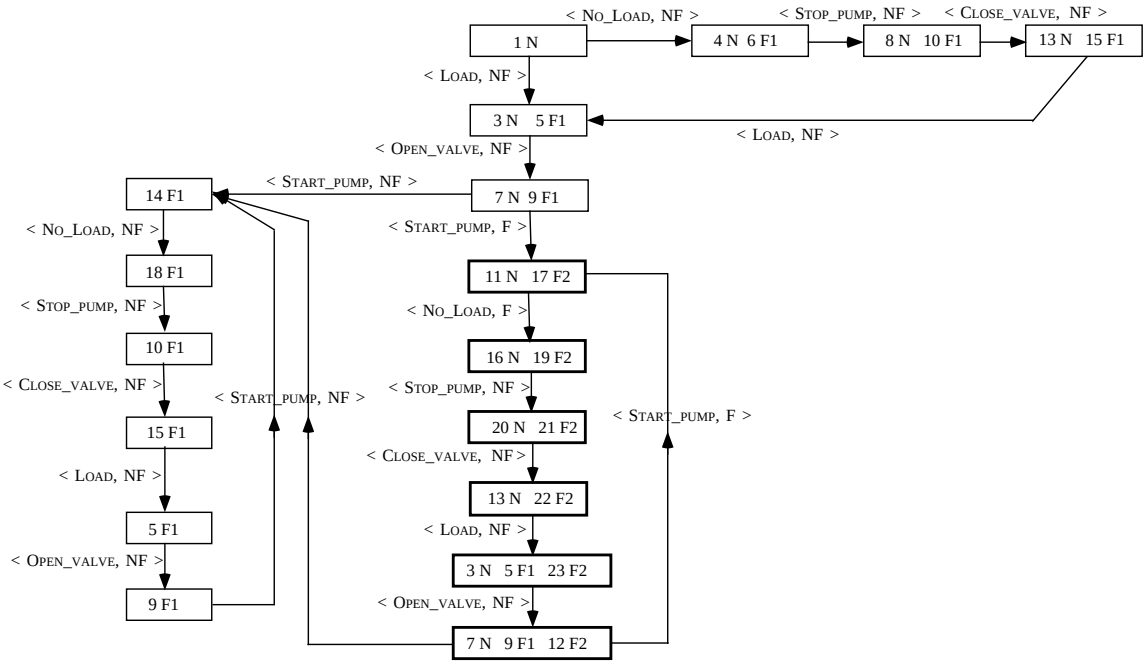


Figure 3.17: The diagnoser G_d for the pump-valve system of Example 19

As before, let $\Sigma_{f1} = \{\text{Stuck\_Closed}\}$ and $\Sigma_{f2} = \{\text{Stuck\_Open}\}$. Figure 3.17 depicts the diagnoser G_d for this system. Inspection of the diagnoser G_d and the system model G reveals the presence of an F_2-indeterminate cycle in the diagnoser (formed by the highlighted states in Figure 3.17). We conclude, therefore, that this system is not diagnosable. More precisely, it is not possible to diagnose occurrences of the valve stuck-open failure in this system.

Suppose next that we choose the indicator events corresponding to the valve failures as follows:

$I(\Sigma_{f1}) = \{ < \text{Open\_valve}, \bullet > \}$,

$I(\Sigma_{f2}) = \{ < \text{Close\_valve}, \bullet > \}$.

(Here $< \text{Open\_valve}, \bullet >$ stands for both the events $< \text{Open\_valve}, \text{F}>$, and $< \text{Open\_valve}, \text{NF}>$.)

The I-diagnoser G_d^I for this system is shown in Figure 3.18. Again, inspection of G_d^I and the system model G shows that there exists an (F_2, I_2)-indeterminate cycle in G_d^I. Note that this cycle does indeed include the indicator event corresponding to the valve stuck-open failure, namely $< \text{Close\_valve}, \bullet >$. This means that even after the valve is issued a close command, it is not possible to diagnose whether the valve is stuck open or not. Therefore, this system is not I-diagnosable. The reason for this is intuitively obvious, if we examine the controller of Figure 3.15. Note that this controller asks for the pump to be stopped even before the valve is closed. The pump being the prime mover, once it is shut off, the flow sensor would always read a no-flow value regardless of the state of the valve. Therefore, with the controller of 3.15, it is not possible to determine whether the valve is normal or

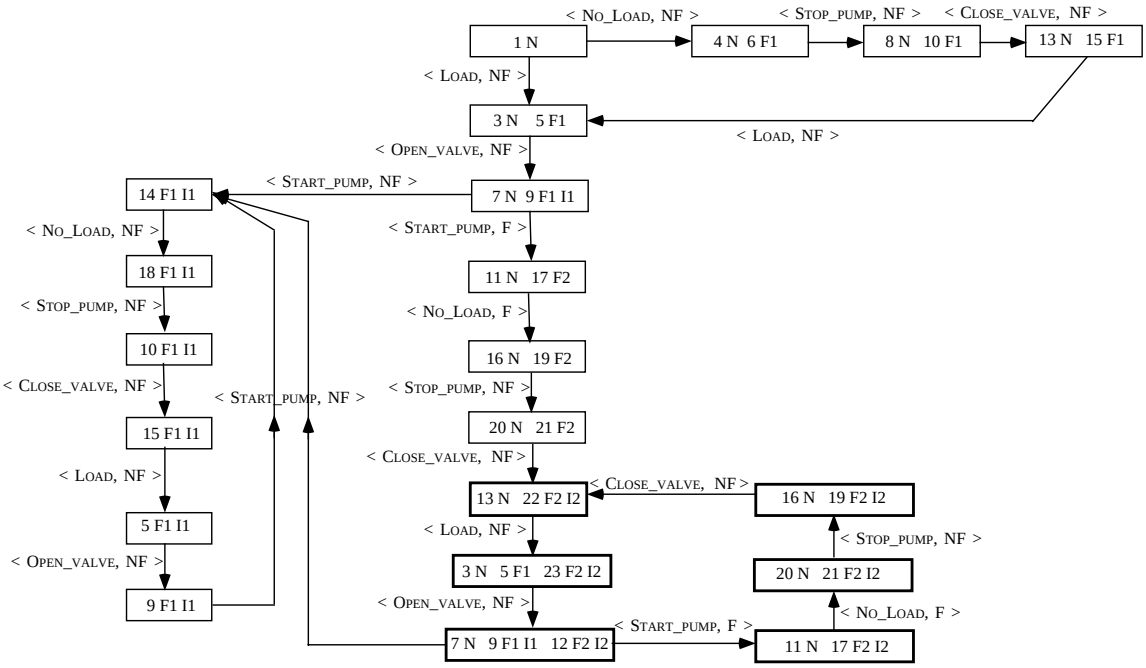
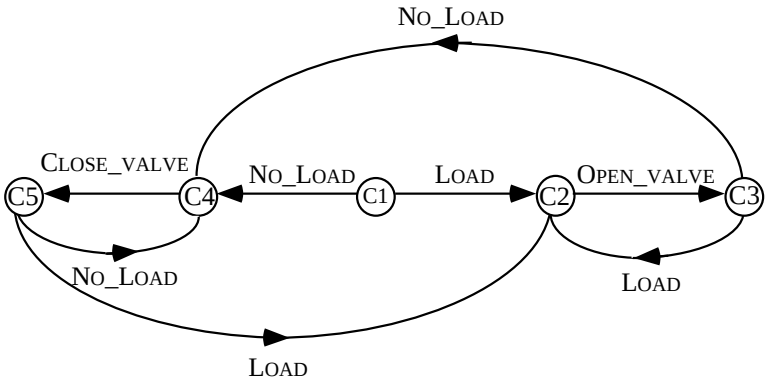


Figure 3.18: The diagnoser G_d^I for the pump-valve system of Example 19

stuck-open, and the system of Figure 3.16 is neither diagnosable nor I-diagnosable. ■

Example 20 As a final example, consider the system of Figure 3.12 but now assume that the pump is always in its on state and further assume that the controller is replaced by the one in Figure 3.19.

The global sensor map and the composite system model G for this system are shown in Table 3.3 and Figure 3.20, respectively. For the sake of brevity, we have omitted the state of the pump in Table 3.3 and in Figure 3.20 since we assume that the pump is always on.



CONTROLLER

Figure 3.19: Controller model for the pump-valve system of Example 20

$h($ VC, $\bullet$ $)$	$=$	NF
$h($ VO, $\bullet$ $)$	$=$	F
$h($ SC, $\bullet$ $)$	$=$	NF
$h($ SO, $\bullet$ $)$	$=$	F

Table 3.3: The global sensor map for the pump-valve system of Example 20

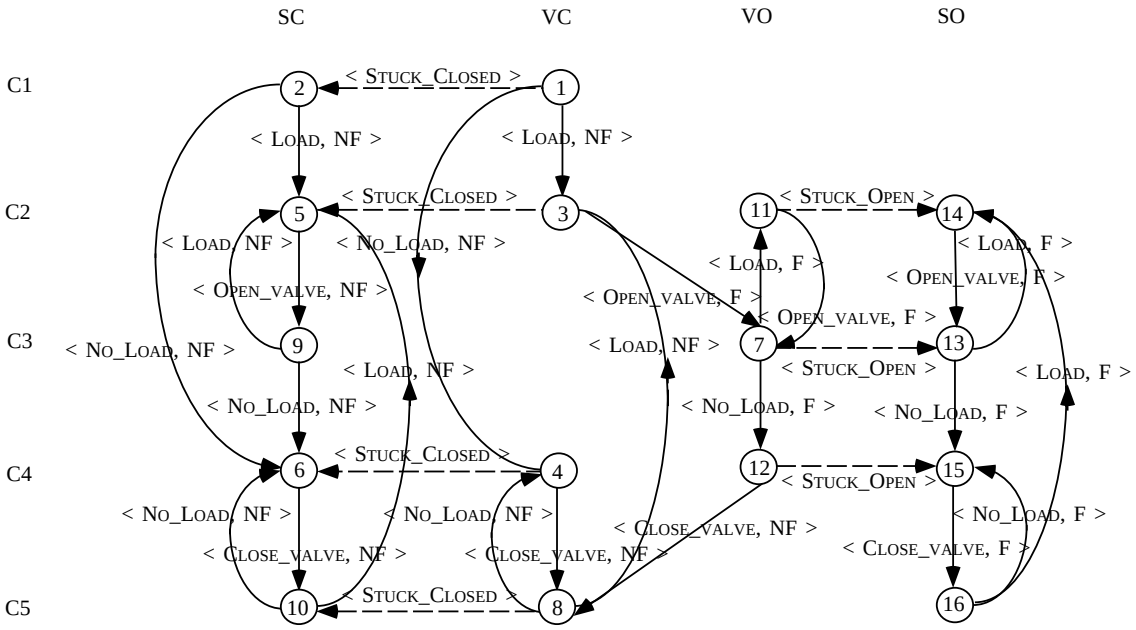


Figure 3.20: The composite system model G for Example 20

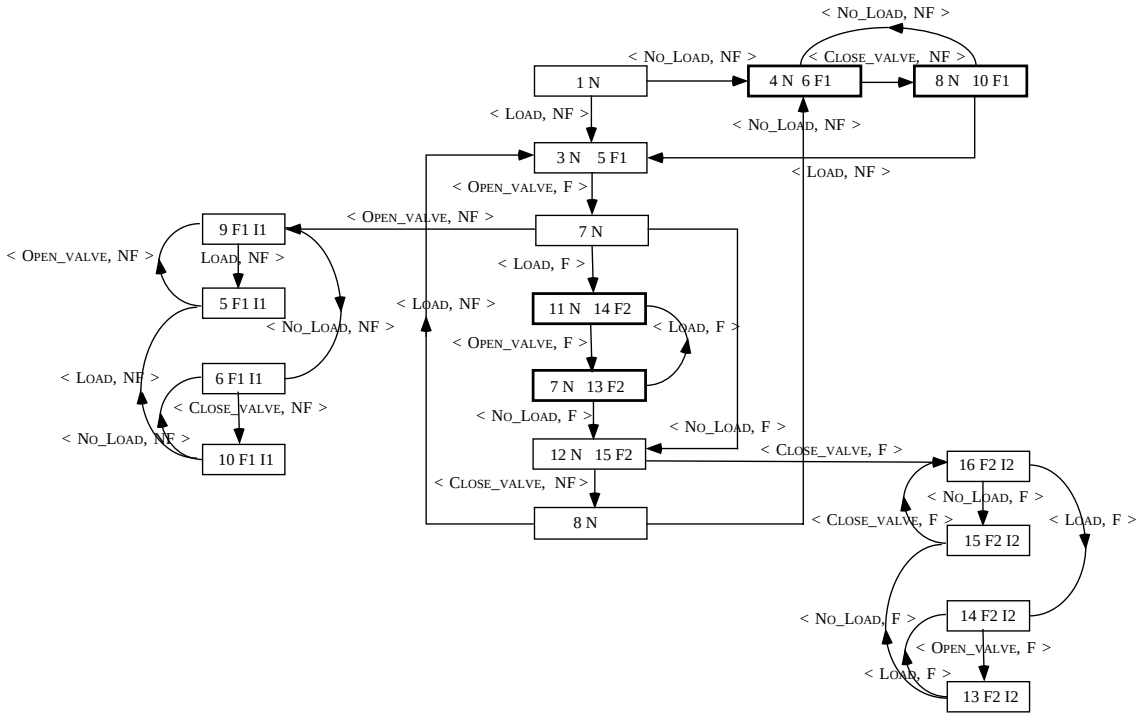


Figure 3.21: The diagnoser G_d^I for Example 20

As before, let $\Sigma_{f1} = \{$Stuck_Closed$\}$ and $\Sigma_{f2} = \{$Stuck_Open$\}$, and let $I(\Sigma_{f1})$ = $\{ <$ Open_valve, $\bullet >\}$ and $I(\Sigma_{f2}) = \{<$ Close_valve, $\bullet > \}$. The I-diagnoser G_d^I for this system is depicted in Figure 3.21. Examination of the diagnoser and the system model shows that there is both an F_1-indeterminate cycle and an F_2-indeterminate cycle in the diagnoser (shown by the highlighted states in Figure 3.21). Note, however, that these cycles are not (F_1, I_1)-indeterminate and (F_2, I_2)-indeterminate, respectively. Thus, the system of Figure 3.20 is not diagnosable but it is I-diagnosable. Intuitively, this means the following. As long as there is no load on the system the controller does not issue an open-valve command and hence it is not possible to determine if the valve is stuck-closed or not. However, since we worry about the valve stuck-closed failure only after the open-valve command is issued (recall that $<$ Open_valve, $\bullet >$ is the indicator event corresponding to this failure), this situation does not make the system non-I-diagnosable. Likewise, the controller does not issue a close-valve command as long as there is a load on the system. This means that it is not possible to diagnose the stuck-open failure as long as the load remains. Again, this does not violate I-diagnosability since we worry about the stuck-open failure only after a close-valve command is issued. ■

3.6 Conclusion

In this chapter we have introduced the notions of diagnosability and I-diagnosability of systems in the framework of formal languages. We have compared this notion with the

problems of observability, observability with delay, and invertibility, all of which fall in the general class of partial observation problems, and we have illustrated by means of examples that diagnosability is a distinctly different notion. We have provided construction procedures for diagnosers and presented necessary and sufficient conditions for diagnosability and I-diagnosability. These conditions can be verified using standard cycle detection algorithms on the diagnosers and the machine G'. Finally, we have presented some simple examples of physical systems and illustrated the main concepts discussed in this chapter.

The theory presented in this chapter is based on two assumptions on the system model. As mentioned earlier, the first assumption, on the liveness of the system, can be relaxed and the definition of diagnosability can be extended to include terminating traces as well; the necessary and sufficient conditions for diagnosability can also be modified appropriately. These results are presented in Chapter 6. The second assumption, on the absence of arbitrarily long traces of unobservable events in L, can also be relaxed if we require that the failures be detected within a bounded number of occurrences of *observable* events following the failure. Appropriate modifications of the theory presented here are straightforward.

CHAPTER 4

On-line Diagnosis of Diagnosable Systems

He has two of the three qualities necessary for the ideal detective - the powers of observation and deduction. He is only wanting in knowledge and that will come in time.

... Sherlock Holmes

4.1 Introduction

In the preceding chapter, we have introduced the notion of dignosability of systems and discussed how the diagnoser can be used to determine if a given system is diagnosable. As was mentioned there, the diagnoser can be used not only to test for the diagnosability of a system but can also be used to perform on-line diagnostics; failures in the system can be diagnosed simply by checking the labels associated with the state estimates that the diagnoser provides. In this chapter we first show how the three diagnosers introduced in Chapter 3 can be used to diagnose failures in diagnosable systems. This result follows as a direct consequence of the necessary and sufficient conditions for diagnosability, and the liveness assumption that we made on the language generated by the system. Next, we prove a more important result which is as follows. *The basic diagnoser G_d is adequate for diagnosing failures in diagnosable and I-diagnosable systems, with or without multiple failures.* In other words, once it is established that the system is diagnosable or I-diagnosable, we can restrict attention to G_d (as opposed to the diagnosers G_d^{mf} and G_d^I) for performing diagnostics; occurrences of failures in the system can be detected with a finite delay by inspecting the states of this diagnoser. This result is important from an implementation viewpoint, as G_d will in general have far fewer states than its counterparts G_d^{mf} and G_d^I.

We now present three lemmas corresponding to the three cases of diagnosability with non-multiple failures, diagnosability with multiple failures, and I-diagnosability, and show how the appropriate diagnosers can be used to perform on-line failure diagnosis. This is followed by a theorem which states the main result of this chapter, namely, the adequacy of the diagnoser G_d for diagnosing failures in diagnosable systems.

4.2 On-line Failure Diagnosis

Lemma 4 *Consider a prefix-closed and live language L. Let $\Sigma_{fi}, i = 1, 2, \ldots, m$ denote disjoint sets of failure events in Σ. Assume that multiple failures of the same type do not occur in the traces in L. If L is diagnosable with delay n_i corresponding to failure type F_i, then the diagnoser G_d detects occurrences of failure events of the type F_i in at most $n_i + n_o$ events of L following the occurrence of the failure events.*

Proof: Let L be diagnosable with delay n_i corresponding to failure type F_i. Recall conditions (C1) (no indeterminate cycles in the diagnoser G_d) and (C2) (no ambiguous states in the diagnoser G_d) of Theorem 1 (presented in Chapter 3) for a language to be diagnosable. From the proof of sufficiency of Conditions (C1) and (C2), and from the liveness assumption on L, it follows that every trace of L containing a failure event of type F_i leads to an F_i-certain state of the diagnoser in a bounded number of transitions. We now show that this happens in at most $n_i + n_o$ transitions of the system following the failure event. Consider any $s \in \Psi(\Sigma_{fi})$ and consider any $t \in L/s$ such that $||t|| \geq n_i$. Since L is diagnosable with delay n_i, we have that $(\forall \omega \in P_L^{-1}[P(st)]),\ \Sigma_{fi} \in \omega$. First suppose that $t_f \in \Sigma_o$. It follows then from the construction of the diagnoser G_d that $\delta_d(q_0, P(st))$ is F_i-certain. Next suppose that $t_f \notin \Sigma_o$. Since the state of the diagnoser G_d corresponding to the trace st is defined only after the occurrence of the first observable event following st, and since the length of any sequence of unobservable events in L is bounded by n_o, we have that $\forall v \in L : \ (v = stu\sigma_o)\ (u \in \Sigma_{uo}^*)\ (\sigma_o \in \Sigma_o)$, $||u\sigma_o|| \leq n_o$ and $\delta_d(q_0, P(v))$ is F_i-certain. Recall that an F_i-certain state of the diagnoser is one where every state estimate contains the label F_i and hence whenever the diagnoser hits an F_i-certain state we can conclude for sure that a failure of type F_i has occurred, regardless of what the current state of the system is. It then follows that the diagnoser G_d detects occurrences of failures of type F_i with a delay of at most $n_i + n_0$ events. **Q.E.D.**

Lemma 5 *Consider a prefix-closed and live language L. Let $\Sigma_{fi}, i = 1, 2, \ldots, m$ denote disjoint sets of failure events in Σ. If L is diagnosable with delay n_i corresponding to failure type F_i, then the diagnoser G_d^{mf} detects occurrences of failure events of the type F_i in at most $n_i + n_o$ events of L following the occurrence of the failure events.*

Proof: Straightforward by recalling condition (C-MF) (no indeterminate cycles in the diagnoser G_d^{mf}) of Theorem 2 from Chapter 3, and by reasoning along lines similar to the proof of Lemma 4. **Q.E.D.**

Lemma 6 *Consider a prefix-closed and live language L. Let $\Sigma_{fi}, i = 1, 2, \ldots, m$ denote disjoint sets of failure events in Σ and let $I(\Sigma_{fi})$ denote the corresponding sets of indicator events. If L is I-diagnosable with delay n_i corresponding to failure type F_i, then the diagnoser G_d^I detects occurrences of failure events of the type F_i in at most $n_i + n_o$ events of L after the occurrence of indicator events of type I_i following the failure events.*

Proof: Let L be I-diagnosable with delay n_i corresponding to failure type F_i. Proof of the sufficiency of Condition (C-I) of Theorem 3 (in Chapter 3) reveals that if L is I-diagnosable, then every trace of L containing a failure event of type F_i, followed by an indicator event of type I_i, leads to an F_i-certain state of G_d^I in a bounded number of transitions. We now show that this happens in at most $n_i + n_o$ transitions of the system following the indicator event. Consider any $s \in \Psi(\Sigma_{fi})$ and consider any $t_1 t_2 \in L/s$ such that $st_1 \in I[\Psi(\Sigma_{f1})]$ and $||t_2|| \geq n_i$. Since L is diagnosable with delay n_i, we have that $(\forall \omega \in P_L^{-1}[P(st_1t_2)])\ \Sigma_{fi} \in \omega$. First suppose that $t_{2_f} \in \Sigma_o$. It follows then from the construction of the diagnoser G_d^I that $\delta_d^I(q_0, P(st_1t_2))$ is F_i-certain. Next suppose that $t_{2_f} \notin \Sigma_o$. Since the state of the diagnoser G_d^I corresponding to the trace st_1t_2 is defined only after the occurrence of the first observable event following st_1t_2, and since the length of any sequence of unobservable events in L is bounded by n_o, we have that $\forall v \in L : (v = st_1t_2u\sigma_o)\ (u \in \Sigma_{uo}^*)\ (\sigma_o \in \Sigma_o)$, $||u\sigma_o|| \leq n_o$ and $\delta_d^I(q_0, P(v))$ is F_i-certain. As before, recalling the definition of an F_i-certain state, the result follows directly. **Q.E.D.**

Theorem 4 *Consider a prefix-closed and live language L. Let $\Sigma_{fi}, i = 1, 2, \ldots, m$ denote disjoint sets of failure events in Σ. If L is diagnosable (respectively, I-diagnosable) with delay n_i corresponding to failure type F_i, then the diagnoser G_d detects occurrences of failure events of the type F_i in at most $n_i + n_o$ events of L after the occurrence of the failure events (respectively, after the occurrence of indicator events of type I_i following the failure events).*

Proof: Case I: Diagnosability: We first consider the case where L is such that multiple failures of the same type do not occur along any trace. It then follows directly from Lemma 4 that the diagnoser G_d detects occurrences of failures of type F_i with a delay of at most $n_i + n_o$ events.

Consider next the case where L is such that multiple failures of the same type are possible. Suppose that we again construct the diagnoser G_d. First, note that the only difference between G_d^{mf} (discussed in Section 3.4.2) and G_d is in the treatment of ambiguous states. We have that every F_i-uncertain state of G_d^{mf} corresponds uniquely either to an ambiguous state or to an F_i-uncertain state of G_d, and, every F_i-certain state of G_d^{mf} corresponds to a unique F_i-certain state of G_d. To be more specific, an F_i-uncertain state $q \in Q_d^{mf}$ such that $q = \{(x, \ell), (y, \ell')\}$ with $x \neq y$ will also be a state of G_d; an F_i-uncertain state $q \in Q_d^{mf}$ such that $q = \{(x, \ell), (x, \ell')\}$ will correspond to the state $q' = \{(x, \{A\} \cup \ell \cap \ell')\}$ of G_d; finally, two states $q_1, q_2 \in Q_d^{mf}$ such that $q_1 = \{(x, \ell), (x, \ell'), (y_1, \ell_1), \ldots, (y_k, \ell_k)\}$ and $q_2 = \{(x, \tilde{\ell}), (x, \tilde{\ell}'), (y_1, \ell_1), \ldots, (y_k, \ell_k)\}$ where q_1 is F_i-uncertain (due to ℓ and ℓ') , q_2 is F_j-uncertain (due to $\tilde{\ell}$ and $\tilde{\ell}'$), and $\ell \cap \ell' = \tilde{\ell} \cap \tilde{\ell}'$ will both correspond to the same ambiguous state $q_3 = \{(x, \{A\} \cup \ell \cap \ell'), (y_1, \ell_1), \ldots, (y_k, \ell_k)\} \in Q_d$. Note here that $L(G_d^{mf}, q) = L(G_d, q')$ and $L(G_d^{mf}, q_1) = L(G_d^{mf}, q_2) = L(G_d, q_3)$. Hence, if one considers a mapping of the states of G_d^{mf} onto the states of G_d, this map preserves the transition structure of G_d^{mf} in the sense of (i) preserving the language generated by G_d^{mf} and (ii) preserving the essential information

for implementing diagnostics because whenever G_d^{mf} hits an F_i-certain state, so would G_d. It follows from Lemma 5 and the above reasoning that if L is diagnosable with delay n_i corresponding to failure type F_i, then the diagnoser G_d^{mf}, and consequently, the diagnoser G_d hits an F_i-certain state in at most $n_i + n_o$ events following the failure event. Thus, G_d detects occurrences of failures of the type F_i with a delay of at most $n_i + n_o$ events.

Case II: I-diagnosability Let L be I-diagnosable with delay n_i corresponding to failure type F_i. Suppose that we again construct the basic diagnoser G_d for L. As in the case of G_d^{mf}, every F_i-uncertain state of G_d^I correponds uniquely to either an ambiguous state or an F_i-uncertain state of G_d, and every F_i-certain state of G_d^I corresponds to a unique F_i-certain state of G_d. Every (F_i, I_i)-uncertain state of G_d^I correponds uniquely either to an ambiguous state or to an F_i-uncertain state of G_d. For example, any two states $q_1, q_2 \in Q_d^I$ of the form $q_1 = \{(x, \{F_i, I_i\}), (y_1, \ell_1), \ldots, (y_k, \ell_k)\}$ and $q_2 = \{(x, \{F_i\}), (y_1, \ell_1), \ldots, (y_k, \ell_k)\}$ correspond to the same state $q_2 = \{(x, \{F_i\}), (y_1, \ell_1), \ldots, (y_k, \ell_k)\}$ in G_d. Note again that $L(G_d^I, q_1) = L(G_d^I, q_2) = L(G_d, q_2)$. Hence, as before, if one considers a mapping of the states of G_d^I onto the states of G_d, this map preserves the transition structure of G_d^I in the sense of (i) preserving the language generated by G_d^I and (ii) preserving the essential information for implementing diagnostics because whenever G_d^I hits an F_i-certain state, so would G_d. From Lemma 6, we conclude that every trace of L containing a failure event of type F_i, followed by an indicator event of type I_i, leads to an F_i-certain state of G_d^I and consequently, to an F_i-certain state of G_d in at most $n_i + n_o$ events after the occurrence of the indicator event of the corresponding type. **Q.E.D.**

Based on the above theorem, we now improve upon the bounds on the delay n_i provided in Sections 3.4.2 and 3.4.3 (cf. proofs of Theorems 2 and 3) for diagnosability in the case of mutiple failures, and for I-diagnosability, respectively. Recall from the proof of sufficiency of condition (C-MF) of Theorem 2 that a bound on the delay n_i is given by $C_i^{mf} \times n_o + n_o$ where $C_i^{mf} = \sum_{q \in Q_d^{mf}:\ q \text{ is } F_i\text{-uncertain}} \#x\text{-states in } q$. We now provide a better bound on n_i which is given by $n_i \leq C_i \times n_o + n_o$ as in the case of no multiple failures. Note that this bound depends only on the states of the basic diagnoser G_d and not on the states of G_d^{mf}. The improved bound can be obtained as follows. First, recall that in order to obtain a bound on n_i for the case of multiple failures, we count the number of F_i-uncertain states in G_d^{mf} (that it is possible to visit before hitting an F_i-certain state). Next, recall from the proof of Theorem 4 that there exist in G_d^{mf} states of the form q_1 and q_2 as described there which have the property that $L(G_d^{mf}, q_1) = L(G_d^{mf}, q_2)$. It is obvious then that is not necessary to count more than once "duplicate" states like q_1 and q_2 because any trace passing through q_1 cannot pass through q_2, and vice-versa. Further, note that since both q_1 and q_2 correspond to the same state q_3 in G_d, these duplicate states get accounted for only once when we compute the bound in G_d.

Likewise, in the case of I-diagnosability, we can obtain a bound on the delay n_i that is better than the one presented in Section 3.4.3, namely, $n_i \leq C_i^I \times n_o$ where $C_i^I = \sum_{q \in Q_d^I:\ q \text{ is } F_i\text{-uncertain}} \#x\text{-states in } q$. The new bound depends only on the states of the

diagnoser G_d and is given by $n_i \leq C_i \times n_o$. Note, as in the case of multiple failures discussed above, that "duplicated" states in G_d^I, of the form q_1 and q_2 described in the proof of Theorem 4 get accounted for twice when one counts the number of (F_i, I_i)-uncertain states that might be traversed before hitting an F_i-certain state in G_d^I, whereas these get accounted for (and correctly) only once in G_d. Hence we have the improved bound stated above.

We conclude, therefore, that in all cases, the bound on the detection delay n_i can be given as follows:

$$n_i \leq C_i \times n_o + n_o \tag{4.1}$$

where $C_i = \sum_{q \in Q_d:\ q \text{ is } F_i\text{-uncertain}} \#x\text{-states in } q$.

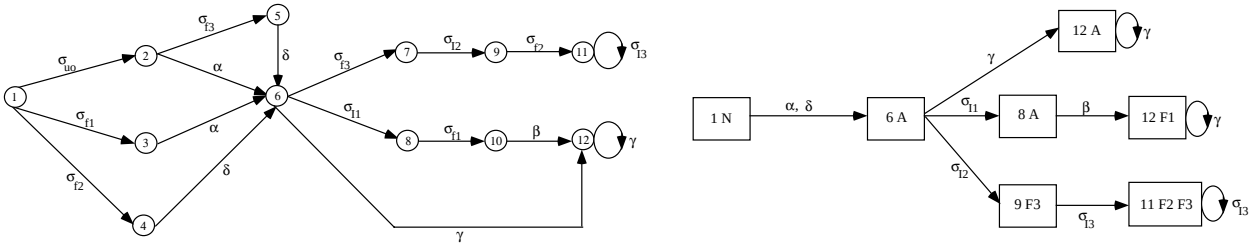
We now provide examples that illustrate implementation of diagnostics for diagnosable and I-diagnosable systems using the diagnoser G_d.

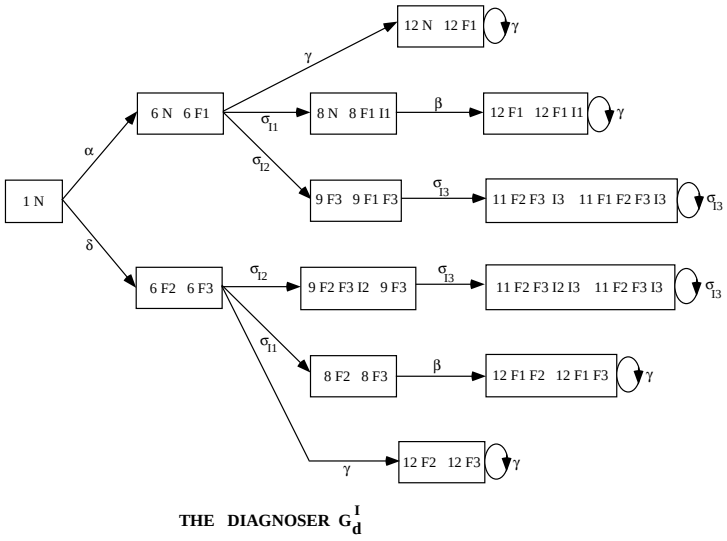
Example 21 Figure 4.1 depicts a system G, its diagnoser G_d^I and the corresponding diagnoser G_d that is implemented. Here, the events $\alpha, \beta, \gamma, \delta, \sigma_{I1}, \sigma_{I2}$, and σ_{I3} are observable while σ_{uo} and the failure events $\sigma_{f1}, \sigma_{f2}, \sigma_{f3}$ are unobservable. The indicator events are chosen to be $I(\sigma_{f1}) = \{\sigma_{I1}\}$, $I(\sigma_{f2}) = \{\sigma_{I2}\}$, and $I(\sigma_{f3}) = \{\sigma_{I3}\}$; the partition is chosen to be $\Sigma_{f1} = \{\sigma_{f1}\}$, $\Sigma_{f2} = \{\sigma_{f2}\}$, and $\Sigma_{f3} = \{\sigma_{f3}\}$. Inspection of G_d^I clearly shows that L is I-diagnosable. Knowing this fact, one is able to conclude that when G_d enters (and stays in) the state $\{(12, \{A\})\}$, no failures violating I-diagnosability have happened. Next, it is clear by inspecting the system G and the diagnoser G_d^I, that when the trace $\alpha\ \sigma_{I2}\ \sigma_{I3}\ \sigma_{I3}^*$ is observed, the diagnoser enters into an F_1-indeterminate cycle and hence it is not possible to conclude whether a failure of type F_1 has happened or not. This, however, is not an (F_1, I_1)-indeterminate cycle since the corresponding trace in G, which contains the failure event σ_{f1}, does not contain the indicator event σ_{I1}. It is interesting to note that the corresponding state in the diagnoser G_d, $\{(11, \{F2, F3\})\}$, reveals nothing about the failure σ_{f1}.

Also shown in Figure 4.1 is the diagnoser G_d^{mf}. Note that Condition (C-MF) is violated in G_d^{mf} and hence L is not diagnosable. Each of the states $\{(12, \{N\}), (12, \{F1\})\}$, $\{(12, \{F2\}), (12, \{F3\})\}$, $\{(12, \{F1, F2\}), (12, \{F1, F3\})\}$, and $\{(11, \{F2, F3\}), (11, \{F1, F2, F3\})\}$ forms an F_i-indeterminate cycle.

Finally, we make the observation that given an I-diagnosable language L, it is possible to have traces in L that satisfy the diagnosability condition D, but in which an indicator event of the appropriate type does not follow the failure event. Consider, for example, the trace $\sigma_{uo}\ \sigma_{f3}\ \delta\sigma_{f3}\ \sigma_{I2}\ \underline{\sigma_{f2}}\ \sigma_{I3}$ in Figure 4.1 and note that the corresponding state of G_d^I is F_2-certain. ■

Example 22 Consider the system of Example 18 in Chapter 3. We saw that this system is diagnosable. We now illustrate how the diagnoser G_d (depicted in Figure 3.14 and presented here again in Figure 4.2 for convenience) can be used to perform on-line diagnosis for this system.





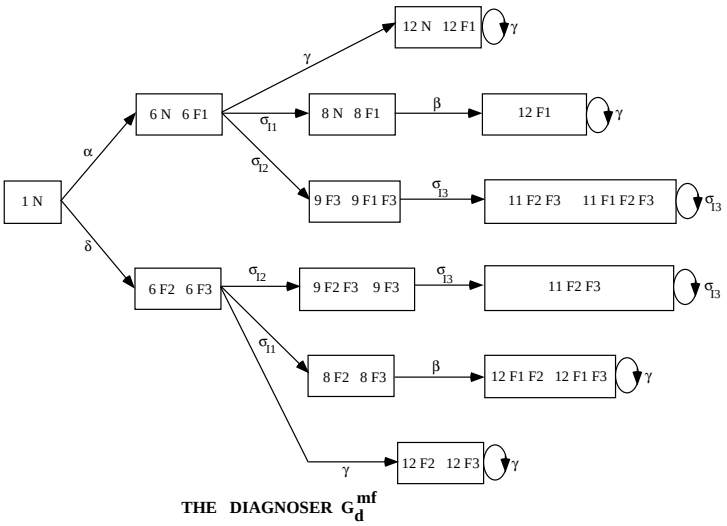


Figure 4.1: Example illustrating implementation using the diagnoser G_d

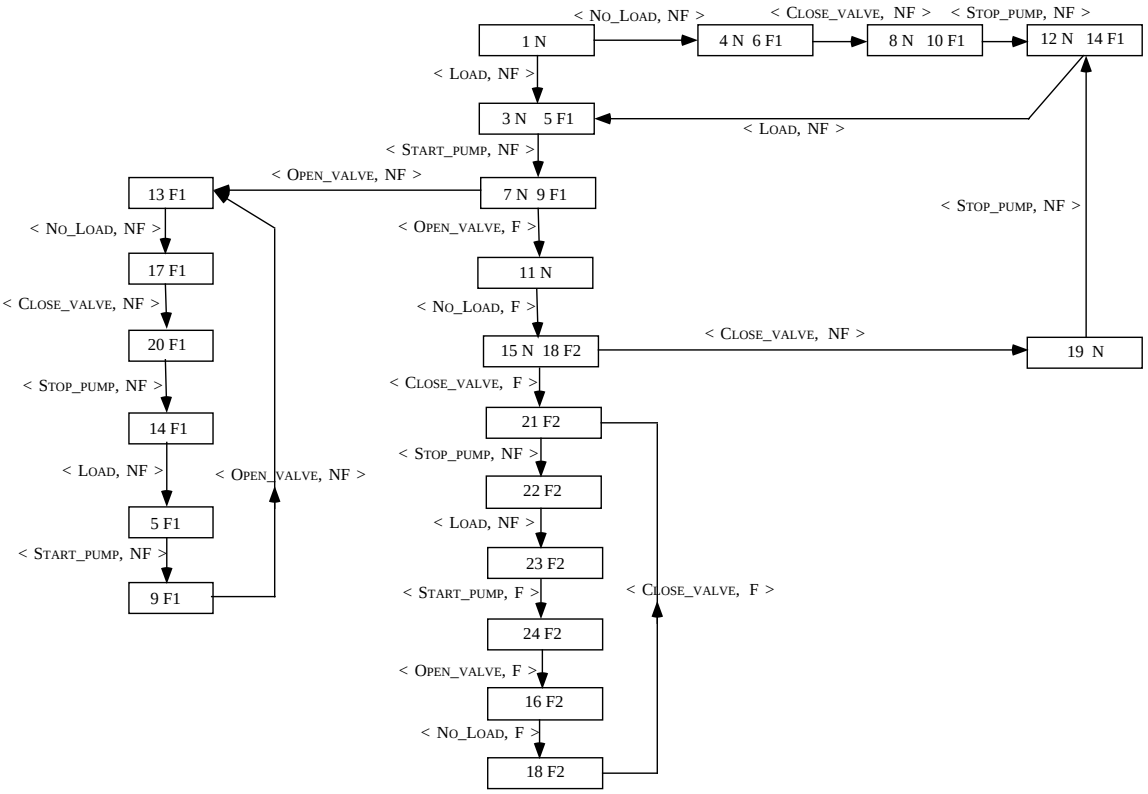


Figure 4.2: The diagnoser G_d for the pump-valve system of Example 18

Let us follow a few sample traces in the diagnoser of Figure 4.2. Suppose that the system generates the sequence of observations, <Load, NF>, <Start_Pump, NF>, <Open_Valve, NF>. Since the flow sensor reads a "no flow" value following the open valve command, we can immediately conclude that the valve is stuck closed. This information can be obtained from the diagnoser G_d by noting that the state of the diagnoser following the above event sequence, namely, the state $\{(13, \{F_1\})\}$ is F_1-certain and that F_1 corresponds to the valve stuck closed failure. Suppose, on the other hand, that the observed event sequence is <Load, NF>, <Start_Pump, NF>, <Open_Valve, F>. If we observe a flow following the open valve command, then we can conclude that the valve is normally open. The diagnoser, in this case, hits the state $\{(11, \{N\})\}$. Next, suppose that we observe the event <No_Load, F> after the event <Open_Valve, F> . At this point, the valve could be in one of two possible states, the normally open state or the stuck open state, since either of these two states could give rise to a "flow" value for the flow sensor. This information is provided by the diagnoser by the fact that it transitions into the F_2-uncertain state $\{(15, \{N\}), (18, \{F2\})\}$ following the event <No_Load, F>. Finally, suppose that the diagnoser next sees the event <Close_Valve, F>. It concludes immediately that the valve is stuck open since it reaches the F_2-certain state $\{(21, \{F2\})\}$ following this event. ■

4.3 Conclusion

In this chapter we have shown how the basic diagnoser G_d can be used to perform on-line failure diagnosis of diagnosable systems. Coupled with the modeling methodology of Chapter 2 this gives us a systematic and algorithmic procedure for building a diagnostic module for any industrial system. One crucial issue regarding the applicability of our theory to real systems is the problem of computational complexity. Since we are dealing with a partially observed system, it is clear that 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, our experience so far tends to indicate that the system often has enough structure so that the worst case computational bounds may be rarely attained. In fact for most "real" systems that we have studied so far, the diagnoser was seen to have a comparable number of states (often, even fewer states!) than the system model. Further, 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 G_d could be built on-line from the current state of G_d and the relevant part of G, with polynomial complexity at each stage. Finally, we note that regardless of whether a system is diagnosable or not, one can use the diagnoser for on-line failure diagnosis. If the system is indeed diagnosable, the diagnoser will detect and isolate all failures with a finite delay. If the system is not diagnosable, then the diagnoser could still be used to obtain the best diagnostic information that the system can provide. In other words, even in those cases where the diagnoser cannot detect the exact failure event, it would still be able to list the set of failures that could have *possibly* happened in the system, given the observed behavior, and thereby aid the operators or technicians in their diagnostic task. Further, the information provided by the diagnoser about the non-diagnosable failures of a system can guide the designer who is interested in altering the diagnosability properties of the system, i.e., in building a system that is diagnosable. In Chapter 6 of this thesis we will discuss in detail the issue of designing diagnosable systems.

CHAPTER 5

Application to HVAC Systems

Theory attracts practice as the magnet attracts iron.

... Karl Friedrich Gauss

5.1 Introduction

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) (see e.g., [1], [48]). 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: (i) 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. (ii) 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. (iii) 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.

Figure 5.1 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. In this chapter 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. The DES models of these systems, the corresponding diagnosers, and their analysis are presented below. We note that these models are based on the discrete event models for HVAC systems presented in [47].

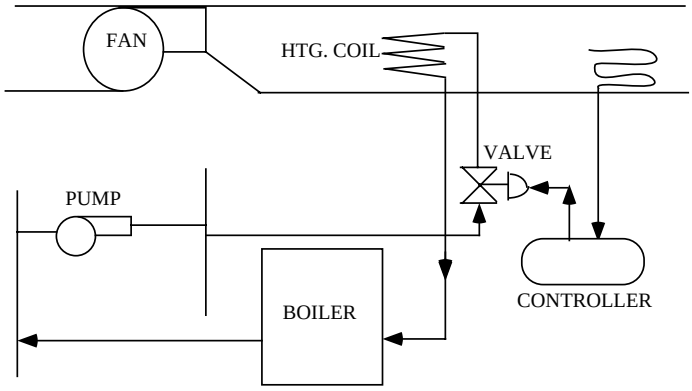


Figure 5.1: (Part of) A HVAC system

5.2 System I

Consider the HVAC system of Figure 5.1 with the component models as depicted in Figure 5.2. Here V1 and V2 represent the normally closed and open 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

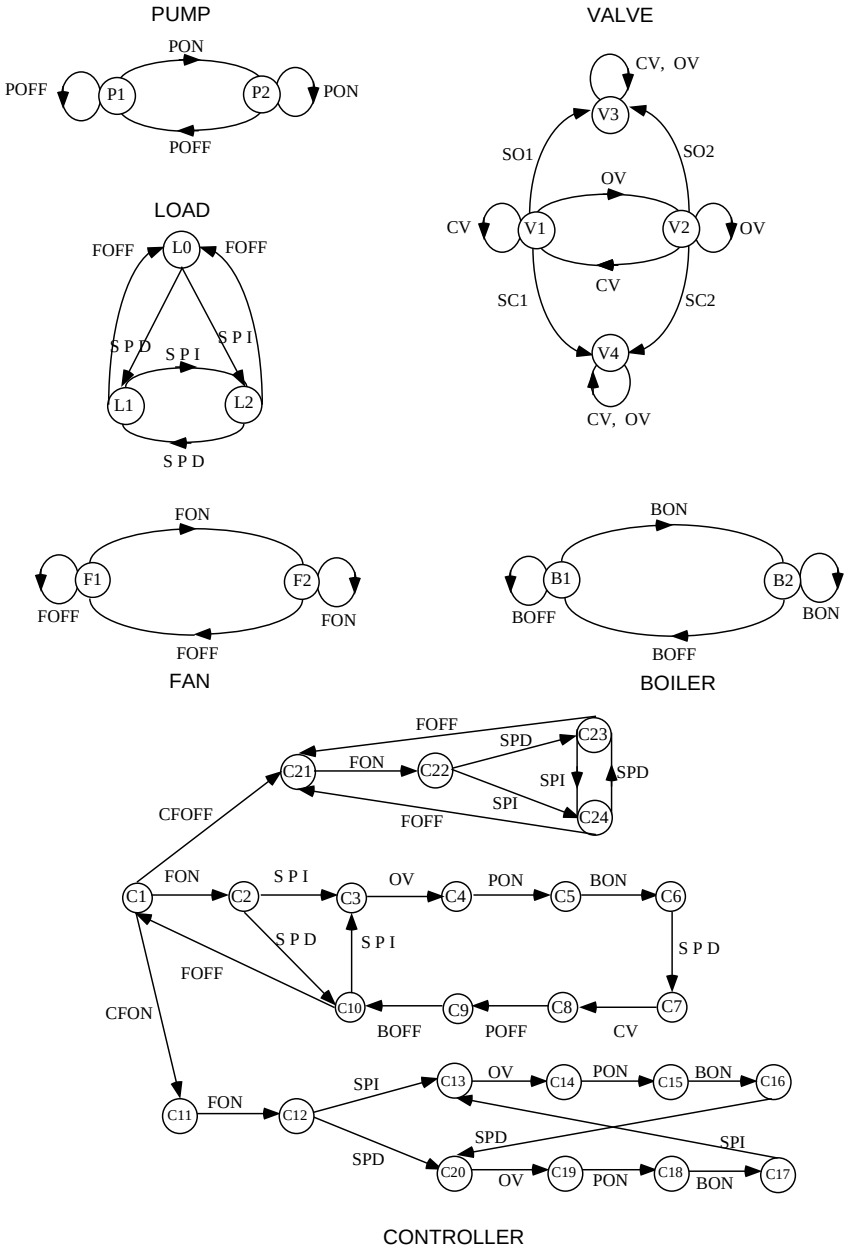


Figure 5.2: Component models: HVAC System I

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. For a complete listing of the transition table see Appendix A.

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:

$\Sigma_{f1} = \{ < \text{SO1} >, < \text{SO2} > \},$

$\Sigma_{f2} = \{ < \text{SC1} >, < \text{SC2} > \},$

$\Sigma_{f3} = \{ < \text{CFON} > \},$

$\Sigma_{f4} = \{ < \text{CFOFF} > \}.$

The indicator events associated with these failure types are:

$I(\Sigma_{f1}) = \{ < \text{CV}, \bullet > \},$

$I(\Sigma_{f2}) = \{ < \text{OV}, \bullet > \},$

$I(\Sigma_{f3}) = \{ < \text{CV}, \bullet > \},$

$I(\Sigma_{f4}) = \{ < \text{OV}, \bullet > \}.$

The diagnoser G_d^I of this system has 158 states. Figure 5.3 illustrates part of G_d^I. Inspection of Figure 5.3 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, $\bullet$ $>$ denotes that there is no load on the system. However, the controller issues the open valve command. 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, (listed in Appendix A) reveals that loop C is F_1-indeterminate. However, it is obvious by inspection 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

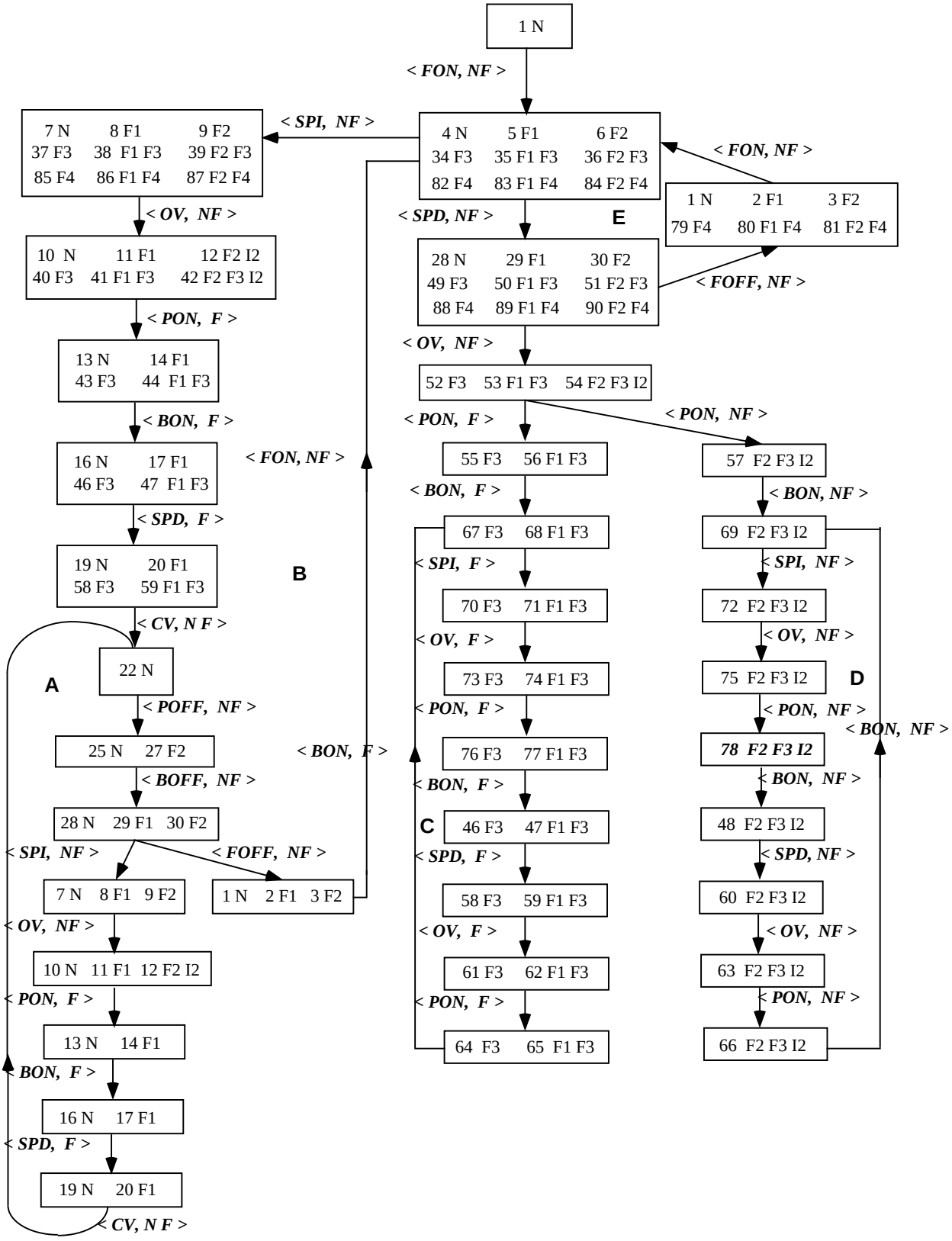


Figure 5.3: Part of the diagnoser G_d^I for HVAC System I

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 normally open or stuck open and it is not possible to distinguish between these two states from observed event sequences. However, since the close valve command is never issued by the controller when it fails on, 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 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 > < FOFF, NF >)*. The correponding 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 of the above mentioned failures. However, it is also clear that in this case, 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 $i \in \{1, \ldots, 4\}$.

Examination of all other such cycles in the diagnoser (not shown in Figure 5.3) reveals that there are no (F_i, I_i)-indeterminate cycles. We conclude therefore, by the results of Section 3.4, 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.

5.3 System II

Consider again the HVAC system of Figure 5.1. We now assume that the valve and the pump can fail while the other components including the controller have no failure events. Figure 5.4 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(positive pressure) and PNP(negative pressure) and a valve flow sensor, whose outputs are F (flow) and NF (no flow).

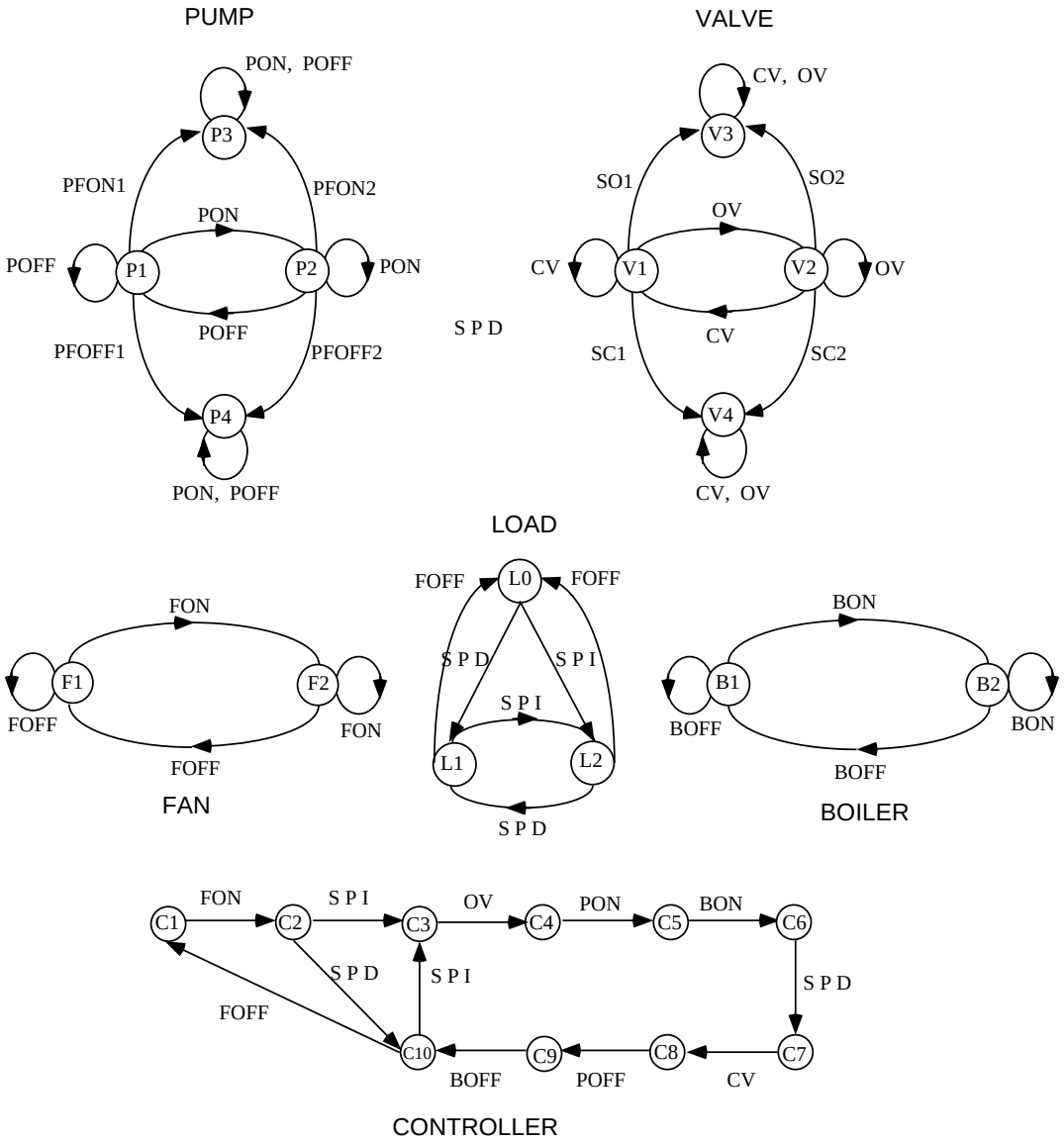


Figure 5.4: Component models: HVAC System II

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 Figure 5.4 and the remaining 40 are the new states introduced during the transition renaming process. A complete listing of the transition table can be found in Appendix B.

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

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

and

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

The diagnoser G_d^I for this system has 245 states. Figure 5.5 illustrates a part of G_d^I. Loops A and B in Figure 5.5 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. However, we note that all uncertainities 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 $\rightarrow$ NF$>$ event is observed is F_2-certain. However, at this point, we do not know if the pump is normal or failed-on. If now we observe the event $<$ PPP $\rightarrow$ 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 negative 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)-

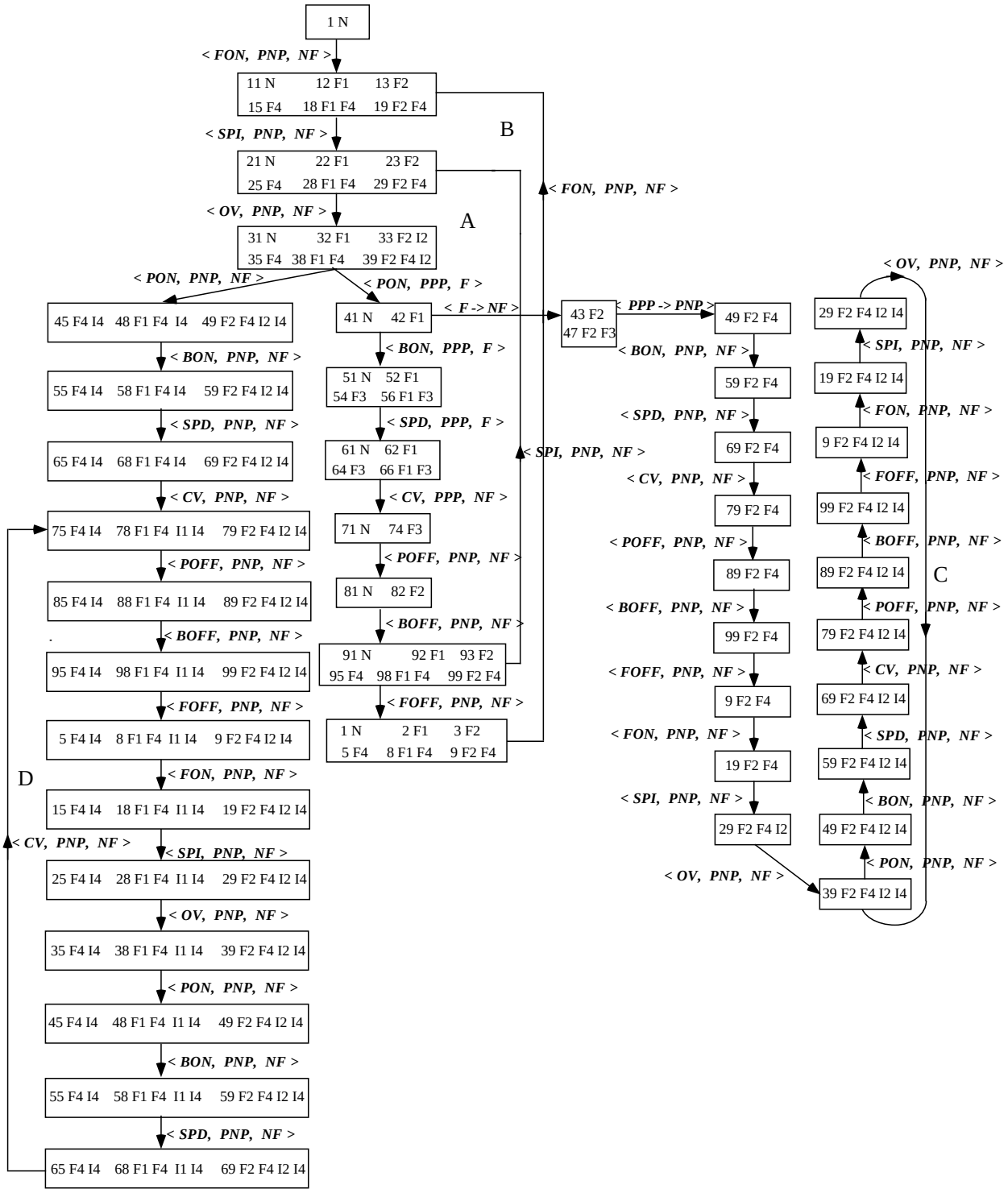


Figure 5.5: Part of the diagnoser G_d^I for HVAC System II

indeterminate, as revealed by a careful examination of the diagnoser and the system model (listed in Appendix B). This implies that the system of Figure 5.4 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 Figure 5.4 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.

5.4 Conclusion

In this chapter we illustrated our approach to failure diagnosis using realistic models of two different HVAC systems. We note that the component models of these 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.

The proposed approach to failure diagnosis has also been applied to an HVAC system with a variable air volume (VAV) terminal box controller (see, e.g., [1]). This effort was undertaken as a study project at Johnson Controls, Inc. [41], and a diagnostic module, based on our approach, was designed to handle component failures of the VAV terminal box. Implementation of this diagnostic module is currently under investigation.

CHAPTER 6

Active Diagnosis: An Integrated Approach to Control and Diagnosis

Order is heav'n's first law.

... Alexander Pope

6.1 Introduction

In the preceding chapters of this thesis we have focussed primarily on answering the question, "given a system with several possible failure modes, how do we detect and diagnose these failures?", and we have proposed a methodology for on-line failure diagnosis of industrial systems using the diagnoser. In this chapter, we extend the scope of the failure diagnosis problem one step further, and attempt to answer the question, "given a system with multiple failure modes, and given a set of diagnostic requirements, how do we ensure that the system satisfies these requirements?". In other words, we investigate the problem of *building* systems that are *diagnosable*. Recalling our discussions on the property of diagnosability in Chapter 3 we see that diagnosability of a system depends both on the projection P that generates output event sequences and the partition Π_f on the failure events. The partition, as was mentioned earlier, is based on diagnostic requirements or specifications for a system, and is often a fixed entity. Given a fixed partition, it is then possible to alter the diagnosability properties of a system by changing the observable event set Σ_o which is equivalent to altering the set of sensors on the system. Thus, one way to ensure diagnosability is to equip the system with the appropriate set of sensors; the challenge then is to determine the optimal, feasible set of sensors that will meet the requirements.

An alternate approach to the design of diagnosable system is by appropriate design of the system controller. In other words, the system controller is to be designed in such a way that it not only satisfies other specified control objectives but it also results in a diagnosable system. In most industrial systems today, design of the on-line monitoring and diagnostic subsystem is done *after* the initial system design, and the diagnostic subsytem is added-on as a separate module to the existing system. Diagnosability considerations are often not explicitly taken into account in the system design and in particular, design of the controller

and that of the diagnostic system are decoupled. Depending on the nature of the controller and the system, this decoupling can significantly affect the diagnosability properties of the system. Recall the examples in Section 3.5 of Chapter 3. We presented there a simple pump-valve system with two different controllers that responded to the presence and absence of a load on the system by taking the same control action (differing however in the order in which the system components were activated) and showed that one controller resulted in a diagnosable system while the other resulted in a non-diagnosable system. Thus, when diagnosability is not a design specification, it is possible to design two different control protocols for a given system such that the closed-loop system in both cases satisfies all the desired design objectives and yet one of these systems may be diagnosable while the other is non-diagnosable.

In this chapter we present an integrated approach to control and diagnosis, which we refer to as the *active diagnosis* problem. The term active is used to distinguish this method from passive diagnosis that was studied in the preceding chapters of this thesis, wherein the role of the diagnostic module was simply to observe the system behavior and draw inferences about potential failures. The active diagnosis problem, on the other hand, is one of combined observation and control. Here we are interested in using control actions to alter the diagnosability properties of a given system, i.e., in restricting the behavior of a non-diagnosable system, by appropriate control, to obtain a diagnosable system. The control issues are posed and addressed in the framework of *supervisory control theory*. Supervisory control theory deals with the design of controllers for a given DES that ensure that the controlled system meets certain qualitative specifications. These specifications define the *legal language* for the system. A *supervisor* for a DES is then an external agent or controller that, based on its partial view of the system, dynamically enables or disables the controllable events of the system in order to ensure that the resulting closed loop language lies within the legal language. For an introduction to the basic ideas of supervisory control theory we refer the reader to [39], [53], and [6] and the references therein.

Proceeding along the lines of the standard supervisory control problem, we adopt the following procedure to solve the active diagnosis problem. Given the non-diagnosable language generated by the system of interest, we first select an "appropriate" sublanguage of this language as the legal language. Choice of the legal language is a design issue and will typically depend on considerations such as acceptable system behavior (which ensures that we do not restrict the system behavior more than necessary in order to eventually make it diagnosable) and detection delay for the failures. As we shall see later in this chapter, the diagnoser can provide guidelines on the choice of this legal language. Once the appropriate legal language is chosen, we then design a controller, which we refer to as a *diagnostic controller*, that achieves a closed-loop language that is within the legal language and is diagnosable. This controller can be designed based on the formal framework and the synthesis techniques that supervisory control theory provides, with the additional constraint of diagnosability. Recalling that the standard problem of supervisory control under partial

observations can be stated as the problem of determining the supremal controllable and observable sublanguage[3] of the legal language, the active diagnosis problem can also be stated as determining the supremal controllable, observable, and diagnosable sublanguage[3] of the legal language and the supervisor that synthesises this language.

Finally, we note that the starting point of the active diagnosis problem can either be an uncontrolled physical system or process, or it can be a controlled system itself. An example of the latter case is where the system has to satisfy prespecified legal language constraints, in addition to being diagnosable. In this case, a controller that ensures legality can first be designed following the usual supervisor design procedures and the closed-loop language generated by the physical system with the above supervisory controller can serve as the starting point for the active diagnosis problem.

This chapter is organized as follows. In Section 6.2 we discuss some enhancements of the basic system model introduced in 3.2.1 and introduce some notation that we shall use in this chapter. In Section 6.3 we discuss diagnosability of non-live languages. Section 6.4 presents the main results of this chapter: the formulation of the active diagnosis problem, a solution procedure along with details of its implementation, and an illustrative example. We also introduce in Section 6.4 a special class of sublanguages of the non-diagnosable language generated by the system that can be used as initial conditions for the active diagnosis problem, in the case where the diagnoser does not contain any nested or interleaved indeterminate cycles (as defined in that section). In Section 6.5 we demonstrate how the theory developed in this chapter can be used to design a diagnostic controller for a simple pump-valve system. Finally, we give some concluding remarks in Section 6.6.

6.2 Preliminaries and Notation

6.2.1 The System Model

As before, the system of interest is modeled as an FSM

$$G = (X, \Sigma, \delta, x_0) \tag{6.1}$$

where X, Σ, δ, and x_0 have the usual interpretation. In addition to partitioning the event set Σ into observable and unobservable events, we also partition it into controllable and uncontrollable events. The controllable events are those that are controllable by an external agent, i.e., those events that can be enabled (permitted to occur) or disabled, while the uncontrollable events are those that cannot be prevented from occurring. Let $\Sigma = \Sigma_c \ \dot{\cup} \ \Sigma_{uc}$ where Σ_c represents the set of controllable events and Σ_{uc} represents the set of uncontrollable events. The controllable events of the system are typically command events issued by the controller while the uncontrollable events are the failure events and the changes of sensor readings.

[3] whenever such a supremal element exists

In addition to the assumptions **(A1)** and **(A2)** on the liveness and the absence of cycles of unobservable events that we made on the system previously, we now make the following additional assumption.

(A3) $\Sigma_c \subseteq \Sigma_o$, i.e., no unobservable event can be prevented from occurring by control.

This assumption is essential for the existence of a solution to the active diagnosis problem as posed here.

6.2.2 Notation

Let $C(G_d)$ denote the set of all indeterminate cycles in the diagnoser G_d corresponding to G. Each $C^i \in C(G_d)$ is given by $C^i = \{(q_1^i, q_2^i, \ldots, q_{k_i}^i), (\sigma_1^i, \sigma_2^i, \ldots, \sigma_{k_i}^i)\}$ where the set of states $\{q_j^i\}_{j=1}^{k_i}$ forms an indeterminate cycle with corresponding event sequence $\{\sigma_j^i\}_{j=1}^{k_i}$. Suppose that the set of states $S = \{q_i\}_{i=1}^n$ forms an indeterminate cycle in G_d with corresponding event sequence $\{\sigma_i\}_{i=1}^n$. Suppose further that $\exists z_1, z_2 \in Q_d \Leftrightarrow S$ and $\alpha_1, \alpha_2 \in \Sigma_o$ such that $\delta_d(z_1, \alpha_1) = q_i$ and $\delta_d(z_2, \alpha_2) = q_j$, $1 \leq i, j \leq n$, $i \neq j$. We refer to such states z_1 and z_2 as *entry states* to the cycle. Then we will consider as two distinct cycles, $C^1 = \{(q_i, q_{i+1}, \ldots, q_n, q_1, \ldots, q_{i-1})\ (\sigma_i, \sigma_{i+1}, \ldots, \sigma_n, \sigma_1, \ldots, \sigma_{i-1})\}$, and $C^2 = \{(q_j, q_{j+1}, \ldots, q_n, q_1, \ldots, q_{j-1})\ (\sigma_j, \sigma_{j+1}, \ldots, \sigma_n, \sigma_1, \ldots, \sigma_{j-1})\}$. In other words any set S of n elements that constitutes an indeterminate cycle could give rise to up to n distinct cycles $C^1, C^2, \ldots, C^n$ depending on the number of distinct entry states into the cycle, from states outside of the cycle (see Example 26 in Section 6.4.5).

Given a system G, a partial observation supervisor S_P for G is a map

$$S_P : P[L(G)] \rightarrow 2^{\Sigma_c} \cup \Sigma_{uc}.$$

The resulting closed loop system is denoted by S_P/G. A realization of the supervisor S_P for G which achieves $L(S_P/G) = K$ is given by the pair (R, φ) where $R = (X_R, \Sigma_o, \delta_R, x_{R,0})$ is a recognizer for $P(K)$ and $\varphi : X_R \rightarrow 2^{\Sigma_c} \cup \Sigma_{uc}$. Under the assumption that $\Sigma_c \subseteq \Sigma_o$, the map $\varphi(x)$ can simply be given by the active event set of R at state x. In this case the supervisor S_P may simply be realized by the FSM R, the feedback map φ being implicit in the transition structure of R.

Given a language M over the alphabet Σ and a language $K \subseteq M$ we denote by $K^{\uparrow C}$ the supremal controllable sublanguage of K with respect to M and $\Sigma_{uc} \subseteq \Sigma$ (where the appropriate M and Σ_{uc} are understood by context). Likewise we denote by $K^{\uparrow CO}$ the supremal controllable and observable sublanguage of K with respect to M, P and Σ_{uc}. Also if H is the FSM that generates K we denote by $H^{\uparrow C}$ the FSM that generates $K^{\uparrow C}$.

We use the notation $G_1 \sqsubseteq G_2$ to denote that the FSM G_1 is a submachine (i.e., subgraph) of G_2 (cf. [21]). Finally, given two FSMs G_1 and G_2, we take $G_1 = G_2$ to mean that G_1 is identical to G_2 up to a renaming of the states.

Finally, we note that in the discussions that follow we do not distinguish between the case where L is such that multiple failures of the same type are possible along its traces

and the case where L is such that multiple failures are not possible, and hence, we will use the diagnoser G_d^{mf} for both of the above two cases. However, for ease of notation, we will refer to the diagnoser G_d^{mf} simply as G_d in what follows. We also note that if we are only concerned with I-diagnosability of the system, then we can use the diagnoser G_d^I in place of the diagnoser G_d^{mf}. All of the results presented here hold true in this case as well with the only exception that all references to F_i-indeterminate cycles that appear in the following sections have to be replaced with references to (F_i, I_i)-indeterminate cycles.

6.3 On Diagnosability of Non-live Languages

The theory of diagnosability presented in Chapter 3 was based on the assumption that the language L is live. Even when the system of interest G is such that it satisfies the liveness assumption, it is possible that when the behavior of the system is restricted under control, as in the case of the active diagnosis problem that is discussed in this chapter, the language generated by the controlled system may not be live, since the use of control could result in blocking or deadlock. Hence, we now generalize the results of Chapter 3 to account for non-live languages as well. In this section, we present the generalized definition of diagnosability, a simple mechanism to *extend* any non-live language to a live language, and hence a procedure to check for diagnosability of non-live languages using the results developed in Chapter 3 for live languages. In addition to allowing us to deal with non-live languages arising as a result of control, the results of this section also allow us to deal with systems that are not live to begin with, i.e., systems that do not satisfy assumption **(A1)** as we shall see from what follows.

Definition 14 *A prefix-closed language L is said to be **diagnosable** with respect to the projection P and with respect to the partition Π_f on Σ_f if the following holds:*

$$(\forall i \in \Pi_f)\ (\exists n_i \in N\ )\ (\forall s \in \Psi(\Sigma_{fi}))\ (\forall t \in L/s)$$

$$[\ (||t|| < n_i)(L/st = \emptyset) \Rightarrow D_1\ ]\ \wedge\ [\ ||t|| \geq n_i \Rightarrow D_2\ ]$$

where the diagnosability conditions D_1 and D_2 are

$$D_1 : \forall \omega \in P_L^{-1}[P(st)] : L/\omega = \emptyset\ (\Sigma_{fi} \in \omega)$$

and

$$D_2 : \forall \omega \in P_L^{-1}[P(st)]\ (\Sigma_{fi} \in \omega)\ .$$

The above definition of diagnosability is the same as that for live languages presented in Chapter 3 except that we now require the diagnosability condition to hold for terminating traces as well. Note that this definition applies to any language $L \subseteq \Sigma^*$, regular or not, as does the definition of diagnosability for live languages presented in Chapter 3.

Stated in words, Definition 14 means the following. Let s be any trace generated by the system that ends in a failure from the set Σ_{fi} and let t be any continuation of s. Suppose first that t is of sufficiently long length. Condition D_2 then requires that every trace ω that belongs to the language L and produces the same record of observable events as the trace st should contain a failure event from the set Σ_{fi}. Suppose next that t is not of sufficiently long length and that st is a terminating trace in L. Condition D_1 then requires that every trace ω that belongs to the language L, produces the same record of observable events as the trace st, *and* is also a terminating trace in L, should contain a failure event from the set Σ_{fi}. Conditions D_1 and D_2 together imply that along every continuation t of s one can detect the occurrence of a failure of the type F_i with a finite delay. Note that even if the language L has a terminating trace s that ends in a failure event of type F_i, L may still be diagnosable as long as there does not exist in L a trace s' such that s' is also terminating and generates the same projection as the trace s but does not contain a failure event of type F_i.

In order to diagnose failures in a non-live system and to check for diagnosability of the language it generates we use the results developed in Chapter 3 for live languages as follows. Given a non-live language L we extend it to a live language L^{live} by adding a new event "*Stop*" to Σ where $Stop \in \Sigma_o \cap \Sigma_{uc}$ and by defining

$$L^{live} = L \ \cup_{i \geq 0} \ \{sStop^i : L/s = \emptyset\}. \tag{6.2}$$

It is obvious from the above definition that L^{live} is live. By comparing Definition 14 above with Definition 1 in Chapter 3, it can be seen that

$$L \text{ diagnosable } \Leftrightarrow L^{live} \text{ diagnosable.}$$

In other words, checking for the diagnosability of L is equivalent to checking for the diagnosability of L^{live}.

For the remainder of this section on diagnosability of non-live languages, we assume that L is regular. Let $G^{live} = (X, \Sigma \cup \{Stop\}, \delta^{live}, x_0)$ be the FSM generating L^{live}. The transition function δ^{live} of G^{live} is defined as follows. First, let

$$X^{dead} = \{x \in X : \delta(x, \sigma) \text{ is undefined } \forall \sigma \in \Sigma\}. \tag{6.3}$$

Then define

$$\begin{aligned} \delta^{live}(x, \sigma) &= \delta(x, \sigma) \ \ \forall x \in X \Leftrightarrow X^{dead}, \ \forall \sigma \in \Sigma \\ \delta^{live}(x, Stop) &= \begin{cases} x \text{ if } x \in X^{dead} \\ \text{undefined otherwise.} \end{cases} \end{aligned}$$

Thus, G^{live} is obtained from G by adding at every dead state of G, a self-loop due to the event *Stop*. It is straightforward to see that $L(G^{live}) = L^{live}$. We then construct from G^{live} the diagnoser $G_d^{live} = (Q_d^{live}, \Sigma_o \cup Stop, \delta_d^{live}, q_0)$ corresponding to L^{live}. This diagnoser is

used to perform on-line failure diagnosis of the system G; also the diagnosability of L can be tested by checking for indeterminate cycles in G_d^{live}.

While the diagnoser G_d^{live} may be built from G^{live} following the usual construction procedure of the diagnoser, it may also be obtained as a simple extension of $G_d = (Q_d, \Sigma_o, \delta_d, q_0)$, the diagnoser corresponding to G, if G_d is already available. First, note that the only difference between G_d^{live} and G_d is due to the *Stop* events defined at the dead states X^{dead} of G. Next, for $q \in Q_d$ define

$$D(q) = \{(x, \ell) \in q : (\forall u \in L(G, x))[u \in \Sigma_{uo}^* \ \wedge \ \delta(x, u) \in X^{dead}]\}. \tag{6.4}$$

Let $Q^{dead}(G_d)$ denote the set of all states q in G_d such that $D(q)$ is non-empty. Note that $q \in Q^{dead}(G_d)$ does not necessarily imply that q is a dead state of G_d; it simply means that there exists an (x, ℓ) pair in q where x is such that no further observable event is possible in G once G gets into state x. Consider any $q \in Q^{dead}(G_d)$ and any $(x, \ell) \in D(q)$. Let $\delta(x, u) = y$; since $y \in X^{dead}$, then, in G^{live}, the corresponding live machine, we have $\delta^{live}(x, uStop) = y$ and $\delta^{live}(y, Stop) = y$. This implies that in the diagnoser G_d^{live} corresponding to G^{live}

$$\delta_d^{live}(q, Stop) = L(q) \tag{6.5}$$

and

$$\delta_d^{live}(L(q), Stop) = L(q) \tag{6.6}$$

where

$$L(q) = \{(y, \ell') : (\delta^{live}(x, uStop) = y)(\ell' = \tilde{LP}(\ell, u)) \text{ for some } (x, \ell) \in D(q), u \in \Sigma_{uo}^*\}. \tag{6.7}$$

and[4]

$$\tilde{LP}(\ell, u) \ = \ \begin{cases} \ell \text{ if } \forall i(\Sigma_{fi} \notin u) \\ \ell \cup \{F_i : \Sigma_{fi} \in u\} \text{ otherwise.} \end{cases}$$

Finally note that if q is such that $\forall (x, \ell) \in q, \ \ x \in X^{dead}$, then the state $L(q)$ is the same as q, i.e., $\delta_d^{live}(q, Stop) = q$. Thus we see that G_d^{live} is identical to G_d except that at each state $q \in Q^{dead}(G_d)$ as above, we either have a self-loop due to the *Stop* event, or, we have a transition defined due to the *Stop* event to a state $L(q)$ as above and a self-loop at $L(q)$ due to the *Stop* event. Therefore, once the set $Q^{dead}(G_d)$ is identified, G_d^{live} can be built simply as an extension of G_d.

Example 23 Consider the non-live system G of Figure 6.1 with $\Sigma_f = \{\sigma_{f1}\}$ and $\Sigma_{uo} = \Sigma_f \cup \{\sigma_{uo}\}$. The live extension G_d^{live} of G, the diagnoser G_d corresponding to G, and the diagnoser G_d^{live} corresponding to G^{live} are depicted in Figure 6.1. ■

[4]Note that $\tilde{LP}$ is equivalent to the Label Propagation Function LP^{mf} of the diagnoser defined in Chapter 3; it propagates the failure information labels associated with a state to its successors.

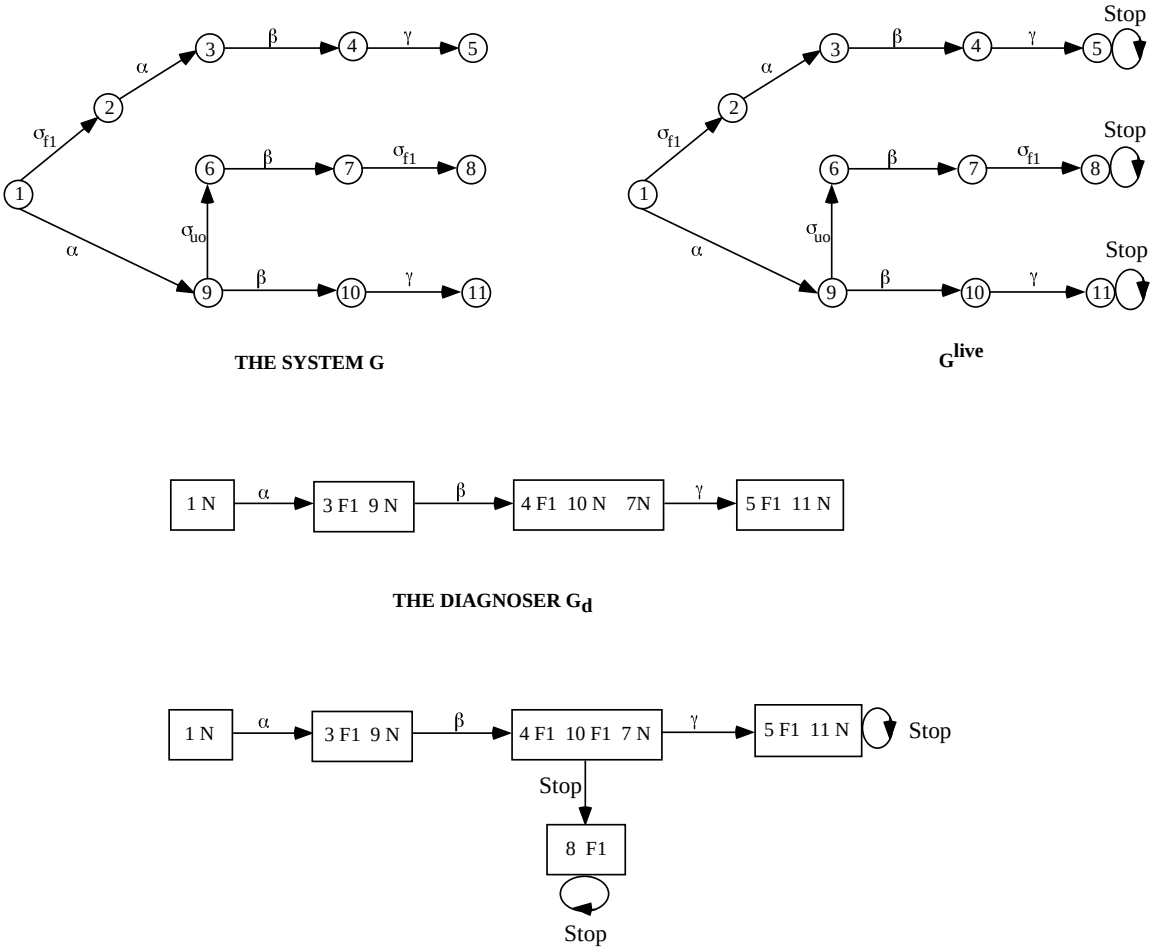


Figure 6.1: The diagnoser G_d^{live} for a non-live system G

We make the following observations on the diagnoser G_d^{live}.

1. Any terminating trace t of L leads to a state $q \in Q_d^{live}$ such that $\delta_d^{live}(q_0, P(tStop)) = q$ and $\delta_d^{live}(q, Stop) = q$.

2. Suppose that t is a terminating trace in L. Further, suppose that $\Sigma_{fi} \in t$ and t violates the diagnosability condition D_1 of Definition 14. This implies that $\exists\ \omega \in P_L^{-1}[P(t)]$ such that $L/\omega = \emptyset$ and $\Sigma_{fi} \notin \omega$. Let $\delta(x_o, t) = x$, $\delta(x_o, \omega) = y$, and $\delta_d^{live}(q_0, P(tStop)) = q$. Then we have that $(x, \ell) \in q$, $(y, \ell') \in q$, $F_i \in \ell$, $F_i \notin \ell'$, and $\delta_d^{live}(q, Stop) = q$. In other words, every terminating trace in L that violates the definition of diagnosability leads to an F_i-uncertain state in G_d^{live} which forms an F_i-indeterminate cycle due to a single event, namely, *Stop*.

3. Let t be a terminating trace of L such that $\Sigma_{fi} \in t$ and such that t does not violate the definition of diagnosability. This implies that $\forall \omega \in P_L^{-1}(P(t))$ such that $\Sigma_{fi} \notin \omega$, $L/\omega \neq \emptyset$. In other words, if $\delta_d(q_0, P(t)) = q$ and if $\exists\ \omega \in P_L^{-1}(P(t))$ such that $\Sigma_{fi} \notin \omega$, then there exists $\sigma \in \Sigma_o$ such that $\delta_d(q, \sigma)$ is defined. Thus, if no observable event occurs in the system after the diagnoser enters state q, then we can conclude that a failure of type F_i has occurred. This inferencing is captured in the live diagnoser G_d^{live} by the fact that $\delta_d^{live}(q_0, P(tStop)) = \delta_d^{live}(q, Stop)$ is an F_i-certain state. Thus, occurrence of the "artificial" *Stop* event at state q of the diagnoser G_d^{live}, which is the same as the occurrence of *no event* at the state of the "real" diagnoser G_d, leads us to diagnose the occurrence of a failure of type F_i.

Example 24 Consider the system G and the diagnoser G_d^{live} of Example 23 (Figure 6.1). The trace $t_1 = \sigma_{f1}\alpha\beta\gamma$ violates condition D_1 of diagnosability since there exists $\omega \in L$ such that $\omega = \alpha\beta\gamma$, $P(t_1) = P(\omega)$, ω is a terminating trace and $\Sigma_{fi} \notin \omega$. Note that the trace $P(t_1)$ leads to the F_1-uncertain state $\{(5, \{F1\}), (11, \{N\})\}$ of G_d^{live} with a self loop due to the *Stop* event. It is easy to verify by inspection of G^{live} that this state forms an F_1-indeterminate cycle.

Consider next the trace $t_2 = \alpha\sigma_{uo}\beta\sigma_{f1}$ that is a terminating trace of L. It is easy to see by inspection of G that this trace t_2 does not violate condition D_1 of diagnosability. Further, we see that if the event sequence $\alpha\beta$ is observed, with no further event thereafter, then we can conclude for sure that the system executed the trace t_2 and hence we can conclude that the failure σ_{f1} occurred. This diagnostic information can be obtained from the diagnoser G_d^{live} by noting that the trace t_2 followed by the *Stop* event leads to the F_1-certain state $\{(8, \{F1\})\}$. ■

This concludes our discussion on the diagnosability of non-live languages. Based on the above discussion, we see that even when the language L generated by the system G is not live, it can be extended to a live language in a straightforward manner. Therefore, we note that assumption **(A1)** that we have made about the system G leads to no loss of generality.

6.4 The Active Diagnosis Problem

In this section, we formulate the active diagnosis problem, provide a solution procedure, discuss the implementation of the procedure, demonstrate its correctness, and finally illustrate this procedure with an example.

6.4.1 Problem Formulation

We formulate the Active Diagnosis Problem (ADP) as follows:

Active Diagnosis Problem: Given the regular, live language L generated by the system G, and given a regular language $K \subseteq L$ such that every live sublanguage of K is diagnosable, find a (partial observation) supervisor S_P for G such that $L(S_P/G) = L^{act}$ where

[C1] $L^{act} \subseteq K$;

[C2] L^{act} is diagnosable; and

[C3] L^{act} is as large as possible.

From standard results on supervisory control under partial observations [28], we know that a supervisor S_P for G, such that $L(S_P/G) = L^{act}$, exists if and only if L^{act} is controllable with respect to L and Σ_{uc}, and observable with respect to L and P. Therefore, stated alternately, the ADP is to find the supremal controllable, observable, and diagnosable sublanguage of (the legal language) K. Note however, that diagnosabiity is not a property that is preserved under union of languages[5], and therefore the supremal diagnosable sublanguage of a given non-diagnosable language need not always exist. However, we will show later, *by a constructive proof*, that such a supremal element does indeed exist under the assumptions made in this chapter, i.e, the ADP is well formulated. We shall refer to the supervisor S_P that solves the above ADP as a *diagnostic controller* for G.

In Section 6.4.5 we will present a class of languages $K_l \subseteq L$, $l \geq 0$ that satisfy the above-mentioned property of K, i.e., every live sublanguage of K_l, for any given l, is diagnosable. This class of languages is defined for the case where the diagnoser G_d corresponding to G does not contain any interleaved or nested indeterminate cycles (as will be explained later), and can be obtained using the diagnoser.

6.4.2 Solution Procedure

Initialization:

[5]Consider $L_1 = \sigma_f \alpha^*$ and $L_2 = \alpha^*$ where $\Sigma_f = \{\sigma_f\}$ and $\alpha \in \Sigma_o$. Then L_1 and L_2 are trivially diagnosable but $L_1 \cup L_2 = \sigma_f \alpha^* + \alpha^*$ is not diagnosable as can be seen by choosing $s = \sigma_f$, $t = \alpha^k$ and $\omega = \alpha^k$, for an arbitrarily large k, in Definition 14.

Step 0-1 Obtain a FSM generator of K, henceforth referred to as G^{legal}.

Step 0-2 Build the diagnoser G_d^{legal} corresponding to G^{legal}.

Step 0-3 Let $i = 0; H_d(0) = G_d^{legal}$; and $M_d(0) = L(H_d(0))$.

Iteration:

Step 1 Compute the supremal controllable sublanguage $M_d^{\uparrow C}(i)$ of $M_d(i)$ with respect to $P(L)(= L(G_d))$ and $\Sigma_{uc} \cap \Sigma_o$.

Let $H_d^{\uparrow C}(i)$ denote the FSM generating $M_d^{\uparrow C}(i)$.

Step 2 Compute $M(i) = P_L^{-1}[M_d^{\uparrow C}(i)]$. Let the FSM $H(i)$ be such that $L(H(i)) = M(i)$.

Step 3 Extend $M(i)$ to the live language $M^{live}(i)$ following the procedure described in Section 6.3. Let $H^{live}(i)$ denote the FSM that generates $M^{live}(i)$.

Step 4 Build the diagnoser $H_d^{live}(i)$ corresponding to $H^{live}(i)$.

Step 5 If $M^{live}(i)$ is diagnosable, stop. The solution to the ADP is $L^{act} = M(i)$ and the corresponding supervisor S_P is realised by the FSM $H_d^{\uparrow C}(i)$.

Step 6 If $M^{live}(i)$ is not diagnosable, then

(i) Obtain $\tilde{H}_d$ by eliminating from $H_d^{\uparrow C}(i)$ all states q such that (a) there is a transition defined at q in $H_d^{live}(i)$ due to the *Stop* event and (b) this transition is part of an indeterminate cycle in $H_d^{live}(i)$ or it leads to a state that is part of an indeterminate cycle in $H_d^{live}(i)$.

(ii) Let $H_d(i+1) = Acc(\tilde{H}_d)$ where $Acc(G)$ denotes the accessible part of G and let $M_d(i+1) = L(H_d(i+1))$.

(iv) Increment i to $i+1$. Go to Step 1.

The role of Step 6-(i) in solving the ADP will become clearer as we proceed, in particular, from Lemma 7 and the proof of Theorem 5.

6.4.3 Implementation of the Solution Procedure

Step 0-1 Given the regular language K, a FSM G^{legal} that generates this language can always be obtained (see, e.g., [20]). We also build a *refined* system model G^{ref} with the following properties:

- $L(G^{ref}) = L$;
- $G^{legal} \sqsubseteq G^{ref}$.

Given G^{legal}, the refined system model G^{ref} can be obtained from the system model G following the refinement procedure in [21].

Step 0-2 G_d^{ref} and G_d^{legal}, the diagnosers corresponding to G^{ref} and G^{legal}, respectively, can be built from G^{ref} and G^{legal}, respectively, following the usual construction procedure of the diagnoser. From Step 0-1 above, it follows that $G_d^{legal} \sqsubseteq G_d^{ref}$.

Step 1 Given two FSMs G_1 and G_2 such that $G_1 \sqsubseteq G_2$, several finite step procedures to obtain $G_1^{\uparrow C}$ (where $G_1^{\uparrow C}$ is the generator of $L(G_1)^{\uparrow C}$ with respect to $L(G_2)$ and the set of uncontrollable events in G_2) exist. We follow the procedure in [21]. This procedure can be used to compute $H_d^{\uparrow C}(i)$ from G_d^{ref} and $H_d(i)$, if it can be established that $H_d(i) \sqsubseteq G_d^{ref}$ for all $i \geq 0$. The above submachine requirement is satisfied for $H_d(0)(= G_d^{legal})$ as per the comments in Step 0-2 above; hence we can follow the procedure of [21] to obtain $H_d^{\uparrow C}(0)$. Next, as can be seen from [21] $H_d^{\uparrow C}(0)$ resulting from the procedure is a submachine of $H_d(0)$. From Step 6 of the solution procedure that computes $H_d(1)$ we see that $H_d(1) \sqsubseteq H_d(0)$ which then implies that $H_d(1) \sqsubseteq G_d^{ref}$. Following the same arguments as above we see that $H_d(i) \sqsubseteq G_d^{ref}$, for all $i \geq 0$ and hence the procedure of [21] can be used to compute $H_d^{\uparrow C}(i)$, for all $i \geq 0$.

Remark: Note also from the above discussion that $H_d(i+1) \sqsubseteq H_d(i)$, for all $i \geq 0$. Since $H_d(0)$ is an FSM, then the above fact implies that the solution procedure of Section 6.4.2 is guaranteed to converge in a finite number of iterations.

Step 2 $H(i)$ can be obtained from $H_d^{\uparrow C}(i)$ by (i) adding self-loops at every state q of $H_d^{\uparrow C}(i)$ due to all $\sigma \in \Sigma_{uo}$ and (ii) performing the product of the resulting machine with G^{ref}. With slight abuse of notation we refer to the first operation as $P^{-1}(H_d^{\uparrow C}(i))$. Then $H(i) = P^{-1}(H_d^{\uparrow C}(i)) \times G^{ref}$.

Step 3 $H^{live}(i)$ can be obtained from $H(i)$ following the procedure described in Section 6.3.

Step 4 This step requires building the diagnoser $H_d^{live}(i)$ corresponding to $H^{live}(i)$. While $H_d^{live}(i)$ can be constructed from $H^{live}(i)$ at each iteration, following the usual construction procedure of the diagnoser, $H_d^{live}(i)$ can be obtained as a simple extension of $H_d^{\uparrow C}(i)$ as explained below. First we present the following lemma.

Lemma 7 *For each iteration $i \geq 0$ of the solution procedure, $H_d^{\uparrow C}(i)$ is the diagnoser corresponding to $H(i)$.*

Proof: From Step 1 above and from Step 1 of the solution procedure we note that

$$H_d^{\uparrow C}(i) \sqsubseteq G_d^{ref} \text{ and } L(H_d^{\uparrow C}(i)) = M_d^{\uparrow C}(i). \tag{6.8}$$

Let $D(M(i))$ denote the diagnoser corresponding to the language $M(i)$ represented by the FSM $H(i)$. Then

$$L(D(M(i)) = P(M(i)) = P(P_L^{-1}[M_d^{\uparrow C}(i)]) = M_d^{\uparrow C}(i) \tag{6.9}$$

where the second equality follows from Step 2 of the solution procedure. Since $G^{ref} = P^{-1}(G_d^{ref}) \times G^{ref}$, $H(i) = P^{-1}(H_d^{\uparrow C}(i)) \times G^{ref}$, and since $H_d^{\uparrow C}(i)$ is a submachine of G_d^{ref}, then $H(i)$, the generator of $M(i)$, is a submachine of G^{ref}. This implies that the diagnoser corresponding to $H(i)$ is a submachine of the diagnoser corresponding to G^{ref}, i.e.,

$$D(M(i)) \sqsubseteq G_d^{ref}. \tag{6.10}$$

From Equations (6.8), (6.9), and (6.10) we see that both FSMs $D(M(i))$ and $H_d^{\uparrow C}(i)$ are submachines of the same machine G_d^{ref}, and they generate the same language $M_d^{\uparrow C}(i)$. Since all of the above FSMs are deterministic [20], it follows that $D(M(i)) = H_d^{\uparrow C}(i)$. **Q.E.D.**

Since $H_d^{\uparrow C}(i)$ is the diagnoser corresponding to $H(i)$ and $H_d^{live}(i)$ is the diagnoser corresponding to $H^{live}(i)$, $H_d^{live}(i)$ can be built by extending $H_d^{\uparrow C}(i)$ as described in Section 6.3 given the "dead" states $Q^{dead}(H_d^{\uparrow C}(i))$ where $Q^{dead}(.)$ is as defined in Section 6.3. Recalling that G_d^{ref} is the diagnoser of the live language L, $Q^{dead}(H_d^{\uparrow C}(i))$ can be identified by (i) comparing the transitions defined at each state of $H_d^{\uparrow C}(i)$ with those defined in G_d^{ref} and (ii) for those states $q \in Q^{dead}(H_d^{\uparrow C}(i))$ such that that there exists a transition out of q in G_d^{ref} which is not present in $H_d^{\uparrow C}(i)$, checking if $D(q)$ is non-empty. If $D(q)$ is non-empty, then the state q qualifies as an element of $Q^{dead}(H_d^{\uparrow C}(i))$.

Step 5 Checking if $M^{live}(i)$ is diagnosable involves checking for indeterminate cycles in $H_d^{live}(i)$. As we shall show in Section 6.4.4 every indeterminate cycle in $H_d^{live}(i)$, if any, is caused by a self-loop at some state of $H_d^{live}(i)$ due to the *Stop* event; hence checking for indeterminate cycles in $H_d^{live}(i)$ amounts to identifying F_i-uncertain states with a self-loop due to *Stop*, for any failure type F_i.

Step 6 Construction of $\tilde{H}_d$, and hence $H_d(i+1)$, from $H_d^{\uparrow C}(i)$ is straightforward once the indeterminate cycles in $H_d^{live}(i)$ are obtained as described in Step 5.

6.4.4 Correctness of the Solution Procedure

We now prove that the iterative procedure presented in Section 6.4.2 converges in a finite number of steps to the solution of the ADP.

Theorem 5 *The iterative solution procedure of Section 6.4.2 for solving the ADP converges in a finite number of iterations. $M(i)$ at convergence, is the supremal controllable, observable, and diagnosable sublanguage of K, and is a regular language. The supervisor S_P that achieves the closed-loop language $M(i)$ can be realized by $H_d^{\uparrow C}(i)$, the diagnoser corresponding to the generator $H(i)$ of $M(i)$.*

Remark: $H_d^{live}(i)$, the diagnoser corresponding to $H^{live}(i)$ at convergence, can be used to perform on-line failure diagnosis of the closed loop system S_P/G (in the manner described

in Chapter 4). Thus the solution procedure of Section 6.4.2 provides both a controller that ensures diagnosability of the closed-loop system and a diagnoser for on-line failure diagnosis.

Before proving the above theorem, we present the following technical results that we shall use in the proof.

Lemma 8 *Consider $L \subseteq \Sigma^*$ and $M \subseteq \Sigma_o^*$. Then $K = P^{-1}(M) \cap L$ is observable with respect to L and P.*

Proof: Consider $s, s' \in K$ such that $P(s) = P(s')$. Let $s\sigma \in K$ and $s'\sigma \in L$. Then we need to prove that $s'\sigma \in K$. Now,

$$\begin{aligned} s\sigma \in K &\Rightarrow P(s\sigma) \in P(K) \\ &\Rightarrow P(s'\sigma) \in P(K) \\ &\Rightarrow P(s'\sigma) \in M \cap P(L) \quad (\text{since} P[P^{-1}(M)] = M) \\ &\Rightarrow s'\sigma \in P^{-1}(M) \\ &\Rightarrow s'\sigma \in K \quad (\text{since } s'\sigma \in L \text{ and } K = P^{-1}(M) \cap L). \end{aligned}$$

Q.E.D.

Lemma 9 *Consider $L \subseteq \Sigma^*$ and $M \subseteq \Sigma_o^*$ and define $K = P^{-1}(M) \cap L$. Then K is controllable with respect to L and Σ_{uc} whenever M is controllable with respect to $P(L)$ and $\Sigma_{uc} \cap \Sigma_o$.*

Proof: Consider $s \in K$, $\sigma \in \Sigma_{uc}$ and $s\sigma \in L$. Then we need to prove that $s\sigma \in K$. Now, $s \in K \Rightarrow P(s) \in P(K) \quad \Rightarrow \quad P(s) \in M$ and $s\sigma \in L \Rightarrow P(s\sigma) \in P(L)$. Since M is controllable with respect to $P(L)$ and $\Sigma_{uc} \cap \Sigma_o$,

$$\begin{aligned} P(s) \in M \ \wedge \ P(s\sigma) \in P(L) &\Rightarrow P(s\sigma) \in M \\ &\quad \text{(by controllability of } M \text{ if } \sigma \in \Sigma_o\text{; trivially true otherwise)} \\ &\Rightarrow s\sigma \in P^{-1}(M) \\ &\Rightarrow s\sigma \in K \quad (\text{since } s\sigma \in L \text{ and } K = P^{-1}(M) \cap L). \end{aligned}$$

Q.E.D.

Lemma 10 *Consider $L \subseteq \Sigma^*$ and $K \subseteq L$. Let $\Sigma_c \subseteq \Sigma_o$. Then $P_L^{-1}[P(K)^{\uparrow C}] = K^{\uparrow CO}$ where "$\uparrow C$" is with respect to $P(L), \Sigma_{uc} \cap \Sigma_o$ and "$\uparrow CO$" is with respect to L, Σ_{uc} for "C", and L, P for "O".*

Proof: First, note that since $\Sigma_c \subseteq \Sigma_o$, then from results in supervisory control theory [27], we know that the supremal controllable and observable sublanguage $K_l^{\uparrow CO}$ of K_l exists[6].

[6]Under the assumption that $\Sigma_c \subseteq \Sigma_o$, normality and controllability together imply observability, and hence the supremal controllable, observable sublanguage is equal to the supremal controllable, normal sublanguage.

Next, we prove that $P_L^{-1}[P(K)^{\uparrow C}] = K^{\uparrow CO}$ as follows. From Lemmas 8 and 9 we have that $P_L^{-1}[P(K)^{\uparrow C}]$ is controllable and observable. Hence, by the definition of $K^{\uparrow CO}$, we have that $P_L^{-1}[P(K)^{\uparrow C}] \subseteq K^{\uparrow CO}$. Next, consider $s \in K^{\uparrow CO}$ (which implies that $P(s) \in P(K^{\uparrow CO}) \subseteq P(K)$). Then we have that

$$\begin{aligned}
\overline{s}\,\Sigma_{uc} \cap L \subseteq K^{\uparrow CO} &\Rightarrow \overline{s}\,\Sigma_{uc}^* \cap L \subseteq K^{\uparrow CO} \\
&\Rightarrow P(\overline{s})P(\Sigma_{uc}^*) \cap P(L) \subseteq P(K^{\uparrow CO}) \subseteq P(K) \\
&\Rightarrow P(s) \in P(K)^{\uparrow C} \\
&\quad \text{(follows from Lemma C.1[7] of [7] since } P(K) \text{ is closed)} \\
&\Rightarrow P_L^{-1}[P(s)] \subseteq P_L^{-1}[P(K)^{\uparrow C}] \\
&\Rightarrow s \in P_L^{-1}[P(K)^{\uparrow C}].
\end{aligned}$$

Therefore, $K^{\uparrow CO} \subseteq P_L^{-1}[P(K)^{\uparrow C}]$, which completes the proof. **Q.E.D.**

We are now ready to demonstrate the correctness of the solution procedure.

Proof of Theorem 5: Given a regular language $K \subseteq L$ such that every live sublanguage of K is diagnosable, the ADP as stated in Section 6.4.1, is to determine the supremal controllable, observable, and diagnosable sublanguage of K and the supervisor that synthesizes this language. Recalling our assumption that $\Sigma_c \subseteq \Sigma_o$, we know that the supremal controllable and observable sublanguage $K^{\uparrow CO}$ of K exists. Hence we first attempt to compute $K^{\uparrow CO}$. From Lemma 10, we know that $P_L^{-1}[P(K)^{\uparrow C}] = K^{\uparrow CO}$ and hence $K^{\uparrow CO}$ can be computed by first computing $P(K)^{\uparrow C}$. From Steps 0-3 through Step 2 of the solution procedure we see that $M_d(0) = L(H_d(0)) = L(G_d^{legal}) = P(K)$ and hence $M_d(0)^{\uparrow C} = P(K)^{\uparrow C}$. This implies that $M(0) = P_L^{-1}(M_d(0)^{\uparrow C}) = P_L^{-1}[P(K^{\uparrow C})] = K^{\uparrow CO}$. Therefore $M(0)$ is the supremal controllable and observable sublanguage of K, and it satisfies two of the three requirements of the solution to the ADP, namely controllability and observability.

Next we need to check if $M(0)$ satisfies the third requirement of diagnosability. In order to do this we extend $M(0)$ to the live language $M^{live}(0)$, build the diagnoser $H_d^{live}(0)$ corresponding to $M^{live}(0)$ and check if $M^{live}(0)$, and consequently $M(0)$, are diagnosable. (Steps 3, 4 & 5).

- If $M(0)$ is diagnosable, then we are done. $M(0)$ is the supremal controllable, observable, and diagnosable sublanguage of K and the supervisor S_P that results in the closed-loop language $M(0)$ is realised by $H_d^{\uparrow C}(0)$, the diagnoser corresponding to the generator $H(0)$ of $M(0)$. Furthermore, on-line failure diagnosis of the closed loop system S_P/G can be performed using the FSM $H_d^{live}(0)$ which is the diagnoser corresponding to $H^{live}(0)$. Finally, note that if $M(0)$ is live, then it immediately follows from the assumption on the language K (that every live sublanguage of K

[7]Lemma C.1 of [7]: Let $K = \overline{K} \in L(G)$ and $s \in K$. Then $s \notin K^{\uparrow C} \Leftrightarrow (\exists \text{ prefix } l \text{ of } s)(\exists t \in \Sigma_{uc}^*)(lt \in L(G) \Leftrightarrow K)$.

is diagnosable) that $M(0)$ is diagnosable and in this case the diagnoser $H_d^{live}(0)$ is $H_d^{\uparrow C}(0)$.

- If $M(0)$ is not diagnosable, then we proceed as follows. Since $M(0) \subseteq K$, then we know from the assumption on K that $M(0)$ is not live; therefore, in order to obtain a diagnosable sublanguage of $M(0)$, we must eliminate all "illegal" terminating traces in $M(0)$ that violate the definition of diagnosability. We now take into account the following two considerations. First, in order to ensure observability of the language resulting after eliminating the above traces, all traces in $M(0)$ that have the same observable projection as these "illegal" terminating traces also need to be eliminated. Next, recall that no unobservable event in the system can be disabled by control. Therefore, it follows that all prefixes of the illegal traces, up to and including the prefix that ends with the last observable event of each trace, *have* to be eliminated in order to satisfy the three requirements of the ADP[8].

  Recall from the discussion on non-live languages in Section 6.3 that every terminating trace s in $M(0)$ (and traces that share the same observable projection) lead to a state q in the diagnoser $H_d^{\uparrow C}(0)$ such that there is a transition out of q due to the *Stop* event in the corresponding "live" diagnoser $H_d^{live}(0)$; further, if s violates diagnosability, then this *Stop* event out of the state q leads to an indeterminate cycle in $H_d^{live}(0)$ which consists of a single F_i-uncertain state with a self-loop due to the *Stop* event, for some failure type F_i. Thus eliminating the illegal traces of $M(0)$ that violate diagnosability (and certain of their prefixes as above) is equivalent to

  1. removing from $H_d^{\uparrow C}(0)$ all states q as above (Step 6-(i)),
  2. obtaining the accessible part of the resulting machine, denoted by $H_d(1)$ in the procedure (Step 6-(ii)), and
  3. obtaining $P_L^{-1}(L(H_d(1)))$.

  Now $P_L^{-1}(L(H_d(1)))$, while diagnosable and observable, may not be achievable by control since the last observable event of the illegal traces, that needs to eliminated as per the above discussion, may not be controllable. Hence we proceed with the iteration with this language in the place of K_l, or equivalently, with $H_d(1)$ in the place of $H_d(0) = G_d^{legal}$. As discussed in Section 6.4.3, the solution procedure is guaranteed to converge in a finite number of iterations.

Finally, note that at each step of the above (finitely-convergent) procedure we remove from K *only* those traces that violate one or other of the required conditions, i.e., controllability, observability, or diagnosability. In Steps 2 and 3 we obtain the *supremal* controllable

[8] Let $s = p\sigma_o\sigma_{uo}^1 \ldots \sigma_{uo}^n$ be an illegal terminating trace of $M(0)$; let σ_o be an observable event and let σ_{uo}^i, $i = 1, \ldots, n$ be unobservable. Then all of the traces $s, p\sigma_o\sigma_{uo}^1 \ldots \sigma_{uo}^{n-1}$, $p\sigma_o\sigma_{uo}^1 \ldots \sigma_{uo}^{n-2}$, ..., $p\sigma_o\sigma_{uo}^1$, and $p\sigma_o$ have to be eliminated from $M(0)$.

and observable sublanguage, and in Step 6 elimination of the illegal traces is done in a *minimally restictive* manner, i.e., only those traces that have to be eliminated in order to satisfy the requirements of the ADP are removed from K. Consequently, at convergence, any language $K' : M(i) \subset K' \subseteq K$ contains at least one trace that violates one or more of the above conditions. Hence we have that $M(i)$ at convergence is the supremal diagnosable, controllable, and observable sublanguage of K. Further, since $M(i)$ is realized by an FSM, it is a regular language. **Q.E.D.**

6.4.5 On the Choice of K and the Class of Languages K_l

In this section, we introduce a class of languages $K_l \subseteq L$, $l \geq 0$ which can be used as initial conditions (i.e., as the language K) for the active diagnosis problem. This class of languages is defined for the case where the diagnoser G_d is such that no two indeterminate cycles in G_d are *interleaved* or *nested*, i.e., no two indeterminate cycles share a common state in a manner such that it is possible for the diagnoser to keep alternating between these two cycles. Examples of such interleaved cycles include pairs of cycles of the form $C^1 = \{(1,2),(\sigma_1,\sigma_2)\}$, $C^2 = \{(1,3),(\sigma_3,\sigma_4)\}$, or, $C^1 = \{(1,2),(\sigma_1,\sigma_2)\}$, $C^2 = \{(1,2,3),(\sigma_1,\sigma_3,\sigma_4)\}$, or, $C^1 = \{(1,2,3,4),(\sigma_1,\sigma_2,\sigma_3,\sigma_4)\}$, $C^2 = \{(2,3),(\sigma_2,\sigma_5)\}$, and so on. (The definition of the languages K_l in the case where G_d has interleaved cycles is more involved and is not discussed here.) In this section, we define the languages K_l, discuss some of their properties, and present a procedure to obtain FSM generators of these languages.

Definition 15 *1. A trace $\omega \in L(G_d)$ is said to go through an indeterminate cycle in G_d k times if $\omega = s(\sigma_1\sigma_2\ldots\sigma_n)^k t$ where: $s,t \in \Sigma_o^*$; $\delta_d(q_0,s) = q_1$; $\delta_d(q_i,\sigma_i) = q_{(i+1) \bmod n}, i = 1,\ldots,n$; and $\{q_i\}_{i=1}^n$ forms an indeterminate cycle in G_d.*

2. A trace $\omega \in L$ is said to go through an indeterminate cycle in its diagnoser G_d k times if $P(\omega) \in L(G_d)$ goes through an indeterminate cycle in G_d k times.

Definition 16 *Given a non-diagnosable and live language L, define*

$$K_l = L \Leftrightarrow \{st \in L : s \text{ goes through an indeterminate cycle in } G_d \ l+1 \text{ times }\}.$$

Therefore, K_0 consists of all traces in L except those that complete an indeterminate cycle in G_d; K_1 consists of those traces in L that complete an indeterminate cycle in G_d at most once, and K_l consists of traces in L that complete an indeterminate cycle in G_d at most l times. Note that K_l is defined with respect to the diagnoser G_d. Therefore, it depends on the system G and not only on the language L generated by G. In other words, given a language L and two different FSMs G and G' such that they both generate the language L, any K_l obtained from G may be different than that obtained from G'. However, it is not difficult to see that these two languages will differ only in the number of times the cycles corresponding to an indeterminate cycle in the diagnoser are included.

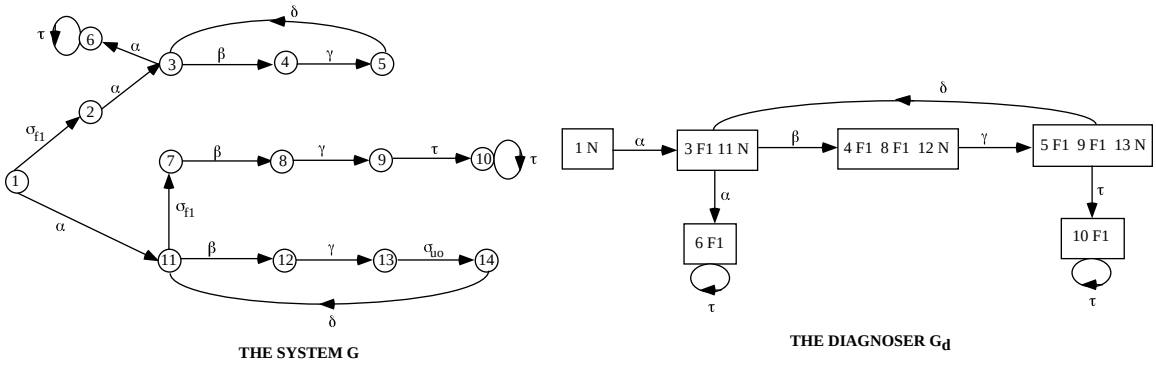


Figure 6.2: Example illustrating the definition of K_l

Example 25 Consider the system G and the corresponding diagnoser G_d represented in Figure 6.2. Let $\Sigma_f = \Sigma_{f1} = \{\sigma_{f1}\}$ and let $\Sigma_{uo} = \Sigma_f \cup \{\sigma_{uo}\}$. It is straightforward to see that the diagnoser in Figure 6.2 has an cycle F_1-indeterminate cycle with the corresponding event sequence $\beta\gamma\delta$. We have for this system (with a slight abuse of notation in the usage of *)

$$K_0 = Pr(\{\alpha\beta\gamma\sigma_{uo}, \alpha\sigma_{f1}\beta\gamma\tau\tau^*, \sigma_{f1}\alpha\beta\gamma, \sigma_{f1}\alpha\alpha\tau^*\})$$

and

$$K_i = Pr(\{\alpha(\beta\gamma\sigma_{uo}\delta)^i\beta\gamma\sigma_{uo}, \alpha(\beta\gamma\sigma_{uo}\delta)^i\sigma_{f1}\beta\gamma\tau\tau^*, \sigma_{f1}\alpha(\beta\gamma\delta)^i\beta\gamma, \sigma_{f1}\alpha(\beta\gamma\delta)^i\alpha\tau^*\})$$

where $Pr(L) = \overline{L}$. ■

Properties of the languages K_l

We now present some properties of the languages K_l defined above.

Property 1 *The languages K_l, $l \geq 0$, are regular.*

Proof: The languages K_l can be realized by FSMs. (Later in this section we present a procedure to obtain a finite state generator of any $K_l, l \geq 0$, given the system G and the set of indeterminate cycles in the diagnoser G_d.) Hence it follows that K_l, $l \geq 0$, are regular. **Q.E.D.**

Property 2 *Given a non-diagnosable and live language L, and K_l, $l \geq 0$*

$$(M \subseteq K_l) \ \wedge \ (M \ live\ ) \Rightarrow M \quad diagnosable.$$

Proof: Since K_l is obtained by removing from L all traces that go through an indeterminate cycle in G_d more than l times, it is obvious that K_l and hence $M \subseteq K_l$ do not contain any non-terminating trace of L that violate the definition of diagnosability. (Recall Theorem 1 on the necessary and sufficient conditions for diagnosability presented in Chapter 3.) This then implies that if M is live, then it is diagnosable. **Q.E.D.**

Alternately speaking, Property 2 states that if $M \subseteq K_l$ is non-diagnosable, then M cannot be live and furthermore any trace in M that violates the definition of diagnosability has to be a terminating trace. Note that M does not have to be regular.

Property 3 *Given a non-diagnosable language L and K_l, $l \geq 0$,*

$$K_0, K_1, \ldots K_l, \ldots \text{ diagnosable } \not\Rightarrow \bigcup_{l=1}^{\infty} K_l \text{ diagnosable.}$$

Proof: Obvious by noting that $\bigcup_{l=1}^{\infty} K_l = L$. **Q.E.D.**

This result simply states that even if every sublanguage of L obtained by including only those traces in L that visit the states of an indeterminate cycle in the diagnoser G_d a finite and bounded number of times (if at all) is diagnosable, L itself is non-diagnosable since it contains traces that can visit an indeterminate cycle in G_d an arbitrarily large number of times.

For the remainder of this section we assume that the multiplicities of the cycles in the system model G corresponding to an indeterminate cycle in the diagnoser G_d (cf. Example 11 in Chapter 3) are equal to one. This assumption is not restrictive because it can be shown that Properties 4 and 6 that follow hold true for the general case of multiplicity greater than one as well. However, proofs of these properties for the general case are not presented here since these proofs are quite involved and since these properties are discussed primarily to motivate the choice of a particular K_l as initial condition for the active diagnosis problem.

Property 4 *Given a non-diagnosable language L and K_l, $l \geq 0$*

$$K_0 \text{ diagnosable } \Leftrightarrow K_l \text{ diagnosable.}$$

Proof: ($\Leftarrow$) Suppose K_0 is not diagnosable. Then we have from Property 2 that K_0 is not live and that there is a terminating trace in K_0 that violates Definition 14. Let s_0 be this trace; then $s_0 \in K_0$ is such that $K_0/s_0 = \emptyset$, $\Sigma_{fi} \in s_0$ for some failure type F_i, and $\exists \tilde{s_0} \in K_0$ such that $K_0/\tilde{s_0} = \emptyset$, $P(s_0) = P(\tilde{s_0})$, and $\Sigma_{fi} \notin \tilde{s_0}$. Now since K_0 is obtained from L (which is live) by removing traces that go through an F_i-indeterminate cycle in the diagnoser G_d of L, $s_0, \tilde{s_0}$ are of the form $s_0 = su_1\sigma_1 \ldots u_{n-1}\sigma_{n-1}u_n$, and $\tilde{s_0} = \tilde{s}v_1\sigma_1 \ldots v_{n-1}\sigma_{n-1}v_n$ where $\Sigma_{fi} \in s$, $s_f \in \Sigma_o$, $P(s) = P(\tilde{s})$, $u_j, v_j \in \Sigma_{uo}^*$, $\sigma_j \in \Sigma_o$, $\Sigma_{fi} \notin \tilde{s_0}$, $\delta_d(q_0, P(s)) = q_1$, $\delta_d(q_j, \sigma_j) = q_{(j+1) \bmod n}$ and $\{q_j\}_{j=1}^n$ form an F_i-indeterminate cycle in G_d. Consider next the traces $s_l, \tilde{s_l} \in K_l$, corresponding to the traces $s_0, \tilde{s_0} \in K_0$ such that $s_l = s(u_1\sigma_1 \ldots u_n\sigma_n)^l u_1\sigma_1 \ldots u_{n-1}\sigma_{n-1}u_n$ and $\tilde{s_l} = \tilde{s}(v_1\sigma_1 \ldots v_n\sigma_n)^l v_1\sigma_1 \ldots v_{n-1}\sigma_{n-1}v_n$. Since $\delta(x_0, s_0) = \delta(x_0, s_l)$ by the definition of an indeterminate cycle, then $K_l/s_l = K_0/s_0 = \emptyset$ and $K_l/\tilde{s_l} = K_0/\tilde{s_0} = \emptyset$. Therefore, s_l violates Definition 14 for K_l and hence K_l is not diagnosable.

($\Rightarrow$) Interchange K_l and K_0 in the above proof and follow the same arguments. **Q.E.D.**

This property means that if the language obtained by cutting an indeterminate cycle in G_d before it gets completed once is not diagnosable, then so is the language obtained by

including the loop l times and then cutting it before it gets completed for the $l+1^{th}$ time, and vice versa. Therefore, as far as diagnosability is concerned, there is nothing to be gained by simply extending a non-diagnosable language by extra traversals of an indeterminate cycle.

Definition 17 *Given a diagnosable language L and $s \in L$, define*

$$delay\,(s,L) \;=\; \begin{cases} Max\,\{\,||t|| : (t \in L/s) \wedge (st \text{ satisfies } D_1 \wedge D_2) \wedge \\ \qquad (\forall u \in \overline{st}, u \text{ does not satisfy } D_1 \vee D_2)\} \quad \text{if } s_f \in \Sigma_{fi} \text{ for some } i \in \Pi_f \\ \text{undefined otherwise.} \end{cases}$$

In other words, delay(s, L) denotes the maximum number of event occurrences possible in L after the trace s before which the occurrence of the failure event s_f cannot be diagnosed.

Property 5 *Given a non-diagnosable language L, and K_l, $l \geq 0$, diagnosable, then $\forall p \in K_l$, delay $(p, K_l) \leq$ delay (p, K_{l+1}).*

Proof: Straightforward from Definition 17 of delay and from the fact that $K_l \subseteq K_{l+1}$. Note that the above inequality becomes an equality if the maximum delay in detecting a failure event in K_{l+1} occurs along a trace s such that s is also contained in K_l. **Q.E.D.**

Property 6 *Given a non-diagnosable language L, and K_l, $l \geq 0$, diagnosable, there exist $m > 0$ and $p \in K_0$ such that, for all $n \geq 1$,*

$$delay\,(p, K_0) < \quad delay\,(p, K_m) < \quad delay\,(p, K_{m+n}).$$

Proof: Consider $p \in K_0$ with $p_f \in \Sigma_{fi}$ for some failure type F_i, such that there exists $s_0 \in K_0$ where $s_0 = p\omega u_1\sigma_1 \ldots u_{n-1}\sigma_{n-1}u_n, \omega_f \in \Sigma_o, u_j \in \Sigma_{uo}^*, \sigma_j \in \Sigma_o, \delta_d(q_0, P(p\omega)) = q_1$, $\delta_d(q_j, \sigma_j) = q_{(j+1) \bmod n}$ and $\{q_j\}_{j=1}^n$ form an F_i-indeterminate cycle in G_d. We know that such a p and s_0 exist by the definition of K_0. Let $t \in K_0/p$ be such that the maximum delay in detecting the failure p_f occurs along the trace t in K_0 and let delay$(p, K_0) = M$. Pick any t' in K_0/s_0 such that detection of the failure p_f along the trace s_0 occurs after the system executes the trace s_0t' but not before, i.e., $\forall \omega_1 \in P_{K_0}^{-1}[P(s_1t')]\,(\Sigma_{fi} \in \omega_1)$ and $(\forall v \in \overline{t'})\ \exists \omega_2 \in P_{K_0}^{-1}[P(sv)] : \Sigma_{fi} \notin \omega_2$. Note that t' may be the empty trace in the case where $K_0/s_0 = \emptyset$; in this case the failure is diagnosed right after s_0 by noting that no further event occurs in the system (recall the results of Section 6.3 on diagnosing failures in a non-live language). Consider next the trace $s_m \in K_m$ where $s_m = p\omega(u_1\sigma_1 \ldots u_n\sigma_n)^m u_1\sigma_1 \ldots u_{n-1}\sigma_{n-1}u_n$. Since $\delta(x_0, s_0) = \delta(x_0, s_m)$, then the delay in detecting the failure p_f along the trace s_m in K_m is given by $|\omega| + (|u_1\sigma_1 \ldots u_n\sigma_n u_n| \times m) + |u_1\sigma_1 \ldots u_{n-1}\sigma_{n-1}u_n| + |t'|$. It is not difficult to see that by choosing a large enough m we can get $|\omega| + (|u_1\sigma_1 \ldots u_n\sigma_n u_n| \times m) + |u_1\sigma_1 \ldots u_{n-1}\sigma_{n-1}u_n| + |t'| > M$, i.e., the maximum delay in detecting the failure event p_f occurs along the trace $\omega(u_1\sigma_1 \ldots u_n\sigma_n)^m u_1\sigma_1 \ldots u_{n-1}\sigma_{n-1}u_n t'$. It then follows that delay$(p, K_m) <$ delay(p, K_{m+n}) for all $n \geq 1$ since delay$(p, K_{m+n}) = |\omega| + (|u_1\sigma_1 \ldots u_n\sigma_n u_n| \times (m+n) + |u_1\sigma_1 \ldots u_{n-1}\sigma_{n-1}u_n| + |t'|$. **Q.E.D.**

Property 6 says that when the languages $K_l, l \geq 0$ are diagnosable, then for large l, the maximum delays in detecting failures occurs along traces that go through indeterminate cycles in the diagnoser; further, each additional traversal of the cycle results in additional delay in detecting the failure.

We now show how the languages K_l form good initial conditions to the ADP posed in Section 6.4.1. Recall that given a non-diagnosable system, the goal of the ADP is to restrict the language generated by the system to a diagnosable sublanguage. Ideally, we would like this language to be as large as possible since we do not want to restrict the behavior of the system more than necessary in order to make it diagnosable. Further, we know from the discussions in Chapter 3 that in order to obtain a diagnosable sublanguage of any given non-diagnosable language, we need to eliminate the F_i-indeterminate cycles in the corresponding diagnoser, for all failure types F_i. The languages K_l, by definition, are obtained by eliminating the traces that go through indeterminate cycles in the diagnoser, and further these languages differ from L *only* in those traces associated with indeterminate cycles. Therefore they are good candidates for the legal language K if it can be shown that they satisfy the assumption made on K in the formulation of the ADP, namely, that every live sublanguage of K is diagnosable. While the languages K_l may themselves not be diagnosable, (since eliminating from L those traces that go through indeterminate cycles in the diagnoser may result in non-live languages, i.e., K_l may contain terminating traces that violate diagnosability) we have from Property 2 that any live sublanguge of K_l is diagnosable. Therefore, the languages K_l satisfy the required assumption.

Thus, we see that each of the languages K_l of Definition 16 is a candidate for the initial condition K. The question then is: which of these K_l does one choose? The following factors motivate the choice of K_l for a particular problem.

- From Property 4, we have that if K_i is diagnosable, then K_j is diagnosable, and vice-versa for all i, j. Hence diagnosability considerations do not have to be taken into account in the choice of K_l.

- Consider the two languages K_m and K_{m+1} and let L_m (L_{m+1}) denote the closed-loop behavior resulting when K_m (K_{m+1}) is chosen as the desired K_l in the ADP. Then it is obvious that $L_m \subset L_{m+1}$ since $K_m \subset K_{m+1}$, i.e., we get a larger closed-loop behavior if we choose K_{m+1}.

- From Properties 5 and 6, the detection delay of all failures in K_{m+1} is greater than or equal to the corresponding delay in K_m, and for a large enough m, the detection delay in K_{m+1} is strictly greater than that in K_m.

Thus, the choice of K_l is dictated primarily by two considerations: closed-loop behavior and failure detection delay. This choice reflects the designer's trade-off between minimal detection delay and maximal closed-loop behavior. Once the appropriate K_l is chosen based on the above design considerations, we look for the largest sublanguage L^{act} of K_l that is

diagnosable and that can be achieved by control. The solution to the ADP is then given by that supervisor S_P that synthesizes this closed-loop language L^{act}.

Procedures to Obtain a Generator G^{legal} of K_l and the Refined Machine G^{ref}

We conclude this section by presenting (i) a procedure to obtain a FSM that generates K_l (henceforth referred to as G^{legal}) from the system model G, and from the diagnoser G_d; and (ii) a procedure to obtain the refined system model G^{ref} such that $L(G^{ref}) = L(G)$ and such that G^{ref} includes as a submachine the generator G^{legal} of K_l. Recall from Section 6.4.3 that the FSM G^{ref} is necessary to implement the solution procedure for the ADP presented in Section 6.4.2.

Procedure to obtain the generator G^{legal} of K_l Given the language L generated by the system G, and given K_l, $l \geq 0$, the generator G^{legal} of K_l can be obtained from the system model G, and from the diagnoser G_d corresponding to L, by the following two-step procedure.

Step 1 Refine $G_d = (Q_d, \Sigma_o, \delta_d, q_0)$ such that every indeterminate cycle in G_d is "expanded out" l times but left "open" (not completed) after the $l + 1^{th}$ copy. This step is explained in detail below (Steps 1-a to 1-e).

Let $G'_d = (Q'_d, \Sigma_o, \delta'_d, q_0)$ denote the refined machine. Then $L(G'_d) = P(K_l)$.

Step 2 Define $G^{legal} = P^{-1}(G'_d) \times G$. In other words, G^{legal} is obtained from G'_d by (i) adding self-loops at every state q of G'_d due to all $\sigma \in \Sigma_{uo}$ and (ii) performing the product of the resulting machine with G.

It is straightforward to implement Step 2 above. We now focus on Step 1 of the procedure.

Procedure to obtain the refined FSM G'_d from the FSM G_d.

The inputs to this procedure are

1. The FSM $G_d = (Q_d, \Sigma_o, \delta_d, q_0)$ and
2. The set $C(G_d)$ of F_i-indeterminate cycles in G_d, for all failure types F_i.

For each indeterminate cycle $C^i \in C(G_d)$, (where $C(G_d)$ denotes the set of all indeterminate cycles in G_d) we build an FSM H^i such that H^i refines the cycle C^i. The refined machine G'_d is then given by

$$G'_d = H^1 \times H^2 \ldots H^M$$

where M refers to the total number of indeterminate cycles in G_d.

We now present a five-step procedure to build H^i given G_d and given an indeterminate cycle

$$C^i = \{(q_1, q_2, \ldots, q_n), (\sigma_1, \sigma_2, \ldots, \sigma_n)\}.$$

Step 1-a Define $G_1 = (Q_d, \Sigma_o, \delta_1, q_0)$ where

$$\delta_1(q,\sigma) \;=\; \begin{cases} \delta_d(q,\sigma) \;\text{ if } q \neq q_1 \;\vee\; \sigma \neq \sigma_1 \\ \text{undefined otherwise.} \end{cases}$$

In other words, G_1 is the same as G_d, except that at state q_1, the transition due to σ_1 is deleted in G_1. Note that not all states of G_1 may be accessible.

Step 1-b Define $G_2 = (Q_2, \Sigma_o, \delta_2, p_1^1)$ where

$$\begin{aligned} Q_2 &= \{p_j^k : j \in \{1,n\}, k \in \{1, l+1\}\} \\ \delta_2(p_j^k,\sigma_j) &= \begin{cases} p_{j+1}^k \;\text{ if } j = 1,\ldots,n \Leftrightarrow 1, k = 1,\ldots,l+1 \\ p_1^{k+1} \;\text{ if } j = n, k = 1,\ldots,l \\ \text{undefined otherwise.} \end{cases} \end{aligned}$$

Note that n refers to the number of states in the cycle C^i and l refers to the number of copies of these states in the refined machine. Also note that the transition due to the event σ_n is not defined at the final state p_n^{l+1}. Thus G_2 has the cycle C^i expanded out l times but left "open" (not completed) after the $l+1^{th}$ copy.

Step 1-c Merge G_1 and G_2 with the state q_1 of G_1 set equal to the state p_1^1 of G_2. Let $G_{12} = (Q_{12}, \Sigma_o, \delta_{12}, q_0)$ denote the merged machine where

$$\begin{aligned} Q_{12} &= Q_1 \cup Q_2 \;(\text{ with } q_1 = p_1^1) \\ \delta_{12}(q,\sigma) &= \begin{cases} \delta_1(q,\sigma) \;\text{ if } \delta_1(q,\sigma) \;\text{ is defined} \\ \delta_2(q,\sigma) \;\text{ if } \delta_2(q,\sigma) \;\text{ is defined} \\ \text{undefined otherwise.} \end{cases} \end{aligned}$$

Note that the states of Q_1 are distinct from those of Q_2 except for the state q_1 which is set equal to p_1^1. At the state q_1 the transition due to σ_1 is not defined in G_1 while at the state p_1^1 of G_2 the only transition defined is due to σ_1. Thus we see that δ_{12} is a well-defined function.

Step 1-c simply "pastes" together the machines obtained in Steps 1-a and 1-b.

Step 1-d Complete the transition function δ_{12} of G_{12} at the states $\{p_j^k\}, k = 1,\ldots,n, j = 1,\ldots,l+1$ as follows:

$$\delta_{12}(p_j^k,\sigma) \;=\; \delta_d(q_j,\sigma)\;\forall\;\sigma \in \Sigma_o \Leftrightarrow \{\sigma_j\}, j = 1,\ldots,n, k = 1,\ldots,l+1.$$

Recall that Q_d is the state space of the diagnoser G_d, δ_d is the transition function of G_d, and $C^i = \{(q_1, q_2, \ldots, q_n), (\sigma_1, \sigma_2, \ldots, \sigma_n)\}$.

Step 1-d completes the transition function of G_{12} so as to ensure that the language generated by the complete machine is equal to the language generated by G_d excluding those traces that go through the indeterminate cycle C^i more than l times.

Step 1-e Let $H^i = Acc(G_{12})$ where $Acc(G)$ denotes the accessible part of G.

Step 1-e simply removes from G_{12} all states that are not reachable from the initial state q_0, and their corresponding transitions. ■

Procedure to obtain the refined system model G^{ref} Given the language L generated by the system G, and given K_l, $l \geq 0$, the procedure to obtain a refined system model G^{ref} that includes as a submachine the generator G^{legal} of K_l, is identical to the above procedure for generating G^{legal} except for Step 1-b which is modified as follows:

Step 1-b Define $G_2 = (Q_2, \Sigma_o, \delta_2, p_1^1)$ where

$$Q_2 = \{p_j^k : j \in \{1, n\}, k \in \{1, l+1\}\}$$

$$\delta_2(p_j^k, \sigma_j) = \begin{cases} p_{j+1}^k & \text{if } j = 1, \ldots, n \Leftrightarrow 1, k = 1, \ldots, l+1 \\ p_1^{k+1} & \text{if } j = n, k = 1, \ldots, l \\ p_1^1 & \text{if } j = n, k = l+1 \\ \text{undefined otherwise.} \end{cases}$$

Note that the transition due to the event σ_n is defined at the final state p_n^{l+1}. Thus G_2 has the cycle C^i expanded out l times and completed after the $l+1^{th}$ time since $\delta_2(p_n^{l+1}, \sigma_n)$ is defined to be p_1^1, the initial state of G_2. This is unlike in the previous procedure for obtaining the generator of K_l where the cycle is left open after the $l+1^{th}$ copy.

In this case we have $L(G_d') = L(G_d)$. It is then straightforward to see that this procedure results in a refined system model G^{ref} such that $L(G) = L(G^{ref})$ and further that $G^{legal} \sqsubseteq G^{ref}$. ■

We now illustrate the above two procedures. Since Step 2 of the procedures is straightforward, in the example that follows, we only illustrate the various operations involved in Step 1, i.e, in generating the refined model G_d' from G_d, given any K_l.

Example 26 Consider Figure 6.3. Let the FSM at the top of this figure represent G_d and let the cycle formed by the states $\{2, 3, 4, 5\}$ denote an indeterminate cycle. Then we have $C^1 = \{(2, 3, 4, 5), (a, b, c, d)\}$ and $C^2 = \{(5, 2, 3, 4), (d, a, b, c)\}$. The FSM representation of C^1 and C^2 are also shown in Figure 6.3. Suppose that we wish to build the FSM generator G^{legal} of K_1. Figures 6.4 and 6.5 illustrate the various steps (Steps 1-a to 1-e in Procedure I above) involved in obtaining the FSM H^1 corresponding to the cycle C^1. The FSM on the bottom of Figure 6.5 represents H^1. Figure 6.8 illustrates H^2 corresponding to the cycle C^2 and Figure 6.7 depicts G_d' which is equal to the product of H^1 and H^2.

Suppose next that we wish to build the refined system model G^{ref} corresponding to K_1. Figures 6.8 and 6.9 represent the FSMs H^1 and H^2 corresponding to the cycles C^1 and C^2 and Figure 6.10 denotes the corresponding G_d'.

Note that the G_d' of Figure 6.7 is submachine of the G_d' of Figure 6.10. It then follows from Step 2 of the above procedures that G^{legal} is a submachine of G^{ref}. ■

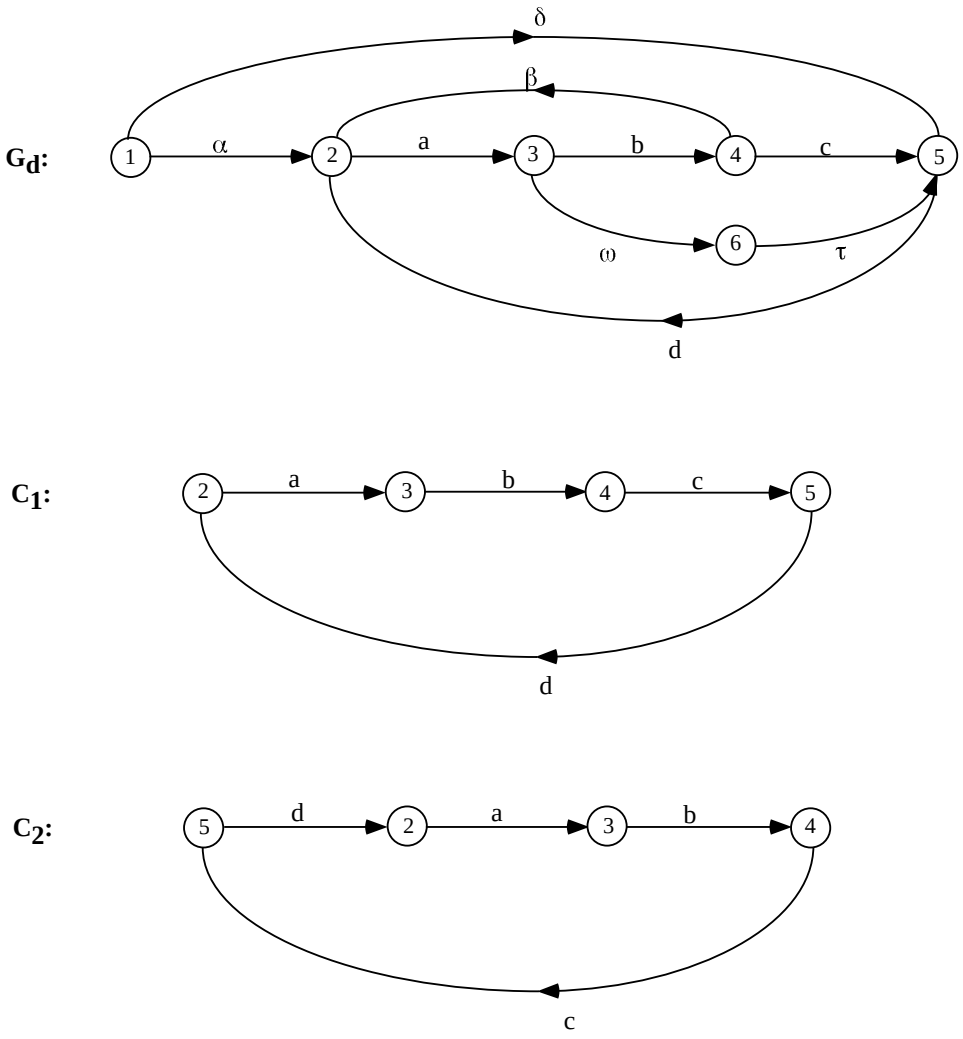


Figure 6.3: G_d and its indeterminate cycles C^1, C^2

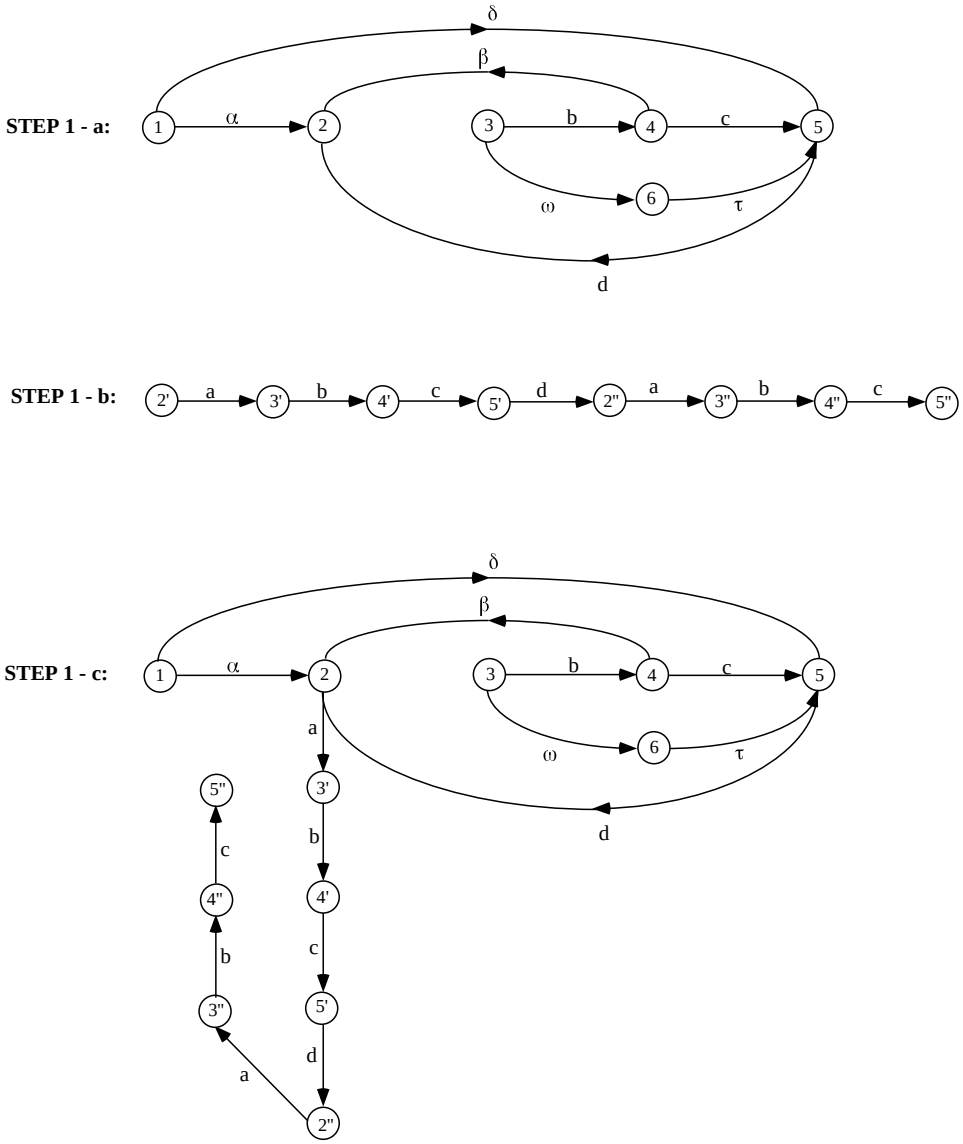


Figure 6.4: Steps in building H^1 for obtaining the FSM generator G^{legal} of K_1

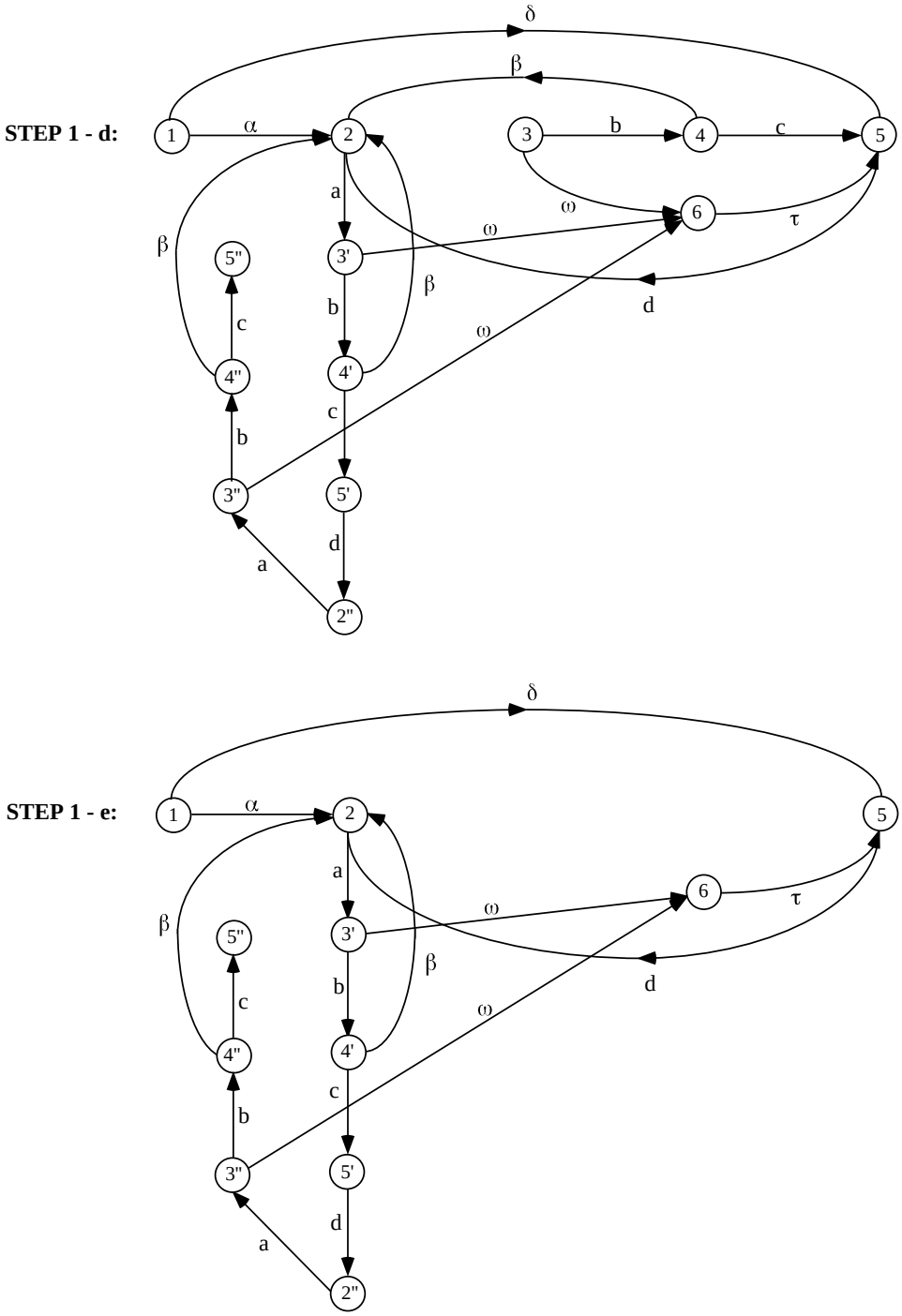


Figure 6.5: Steps in the building H^1 for obtaining the FSM generator G^{legal} of K_1

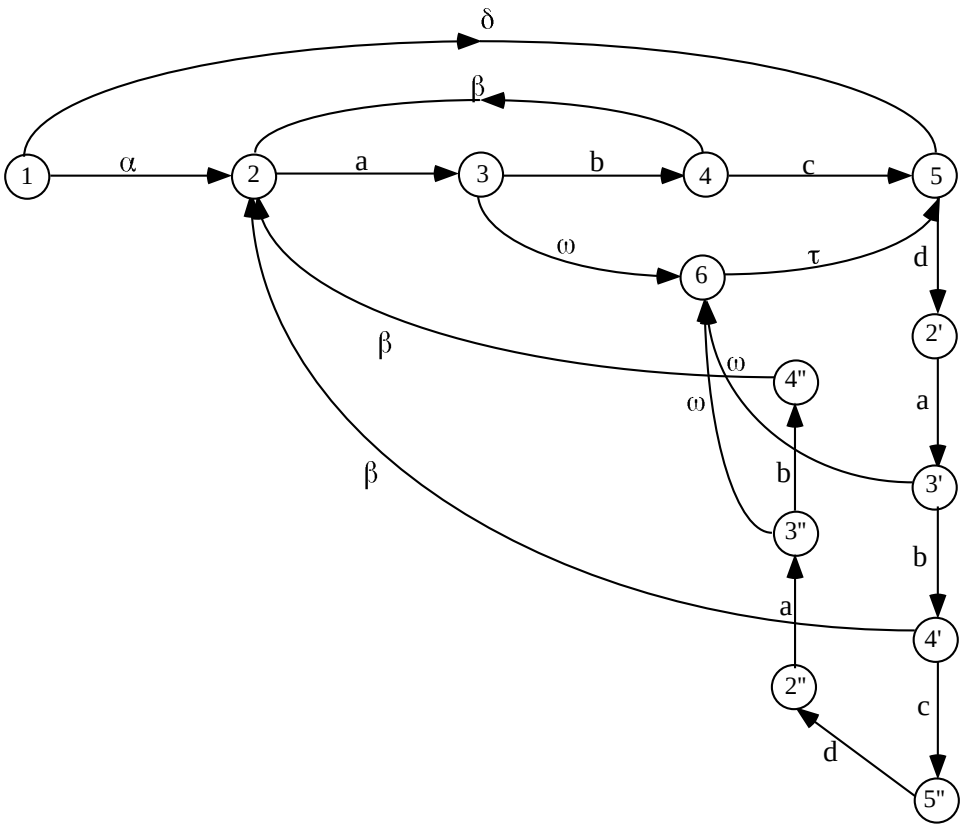


Figure 6.6: H^2 corresponding to cycle C^2 for obtaining the FSM generator G^{legal} of K_1

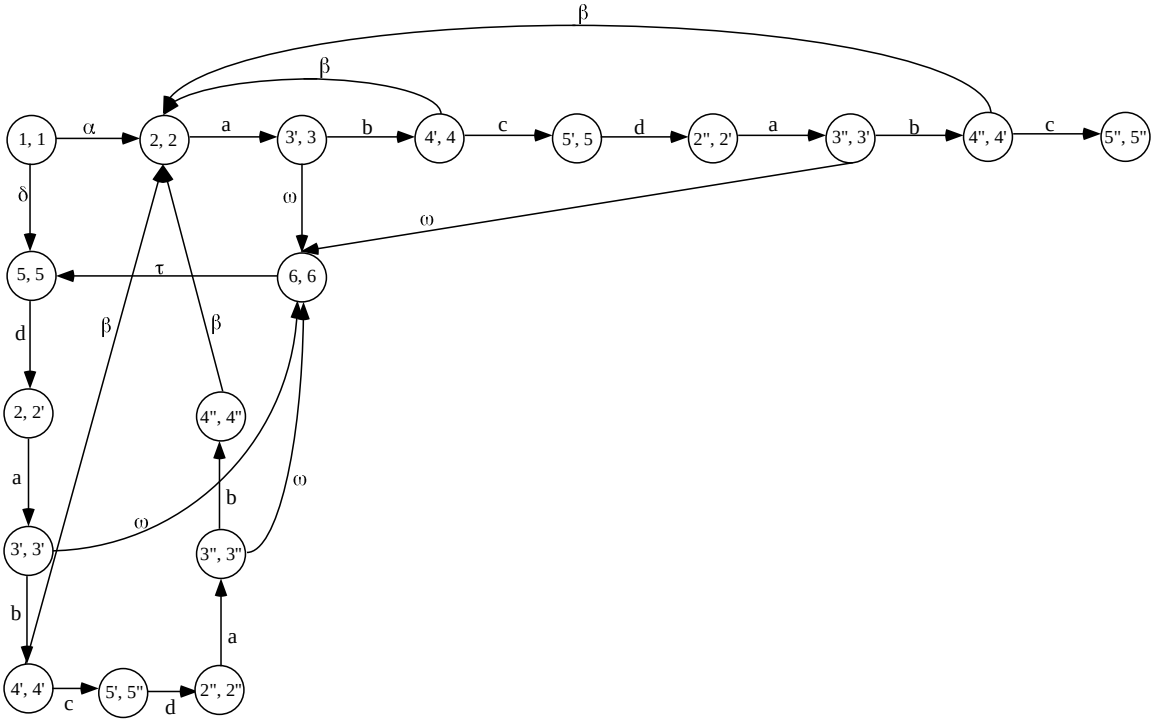


Figure 6.7: G'_d for obtaining the FSM generator G^{legal} of K_1

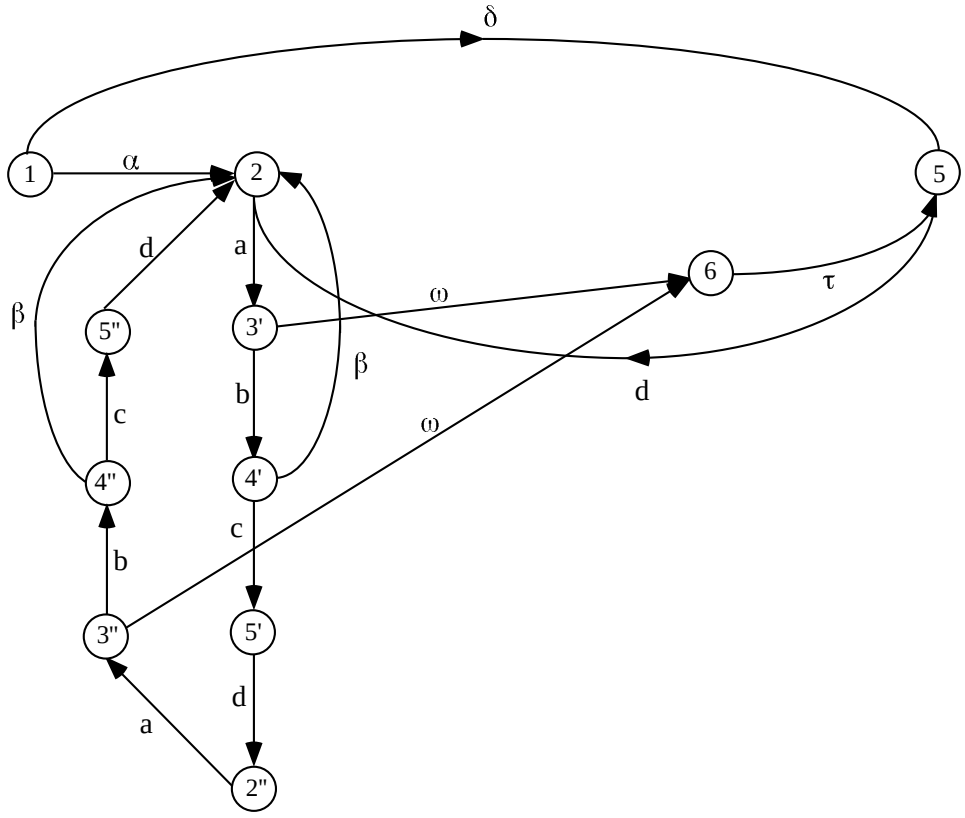


Figure 6.8: H^1 corresponding to cycle C^1 for obtaining the FSM generator G^{legal} of K_1

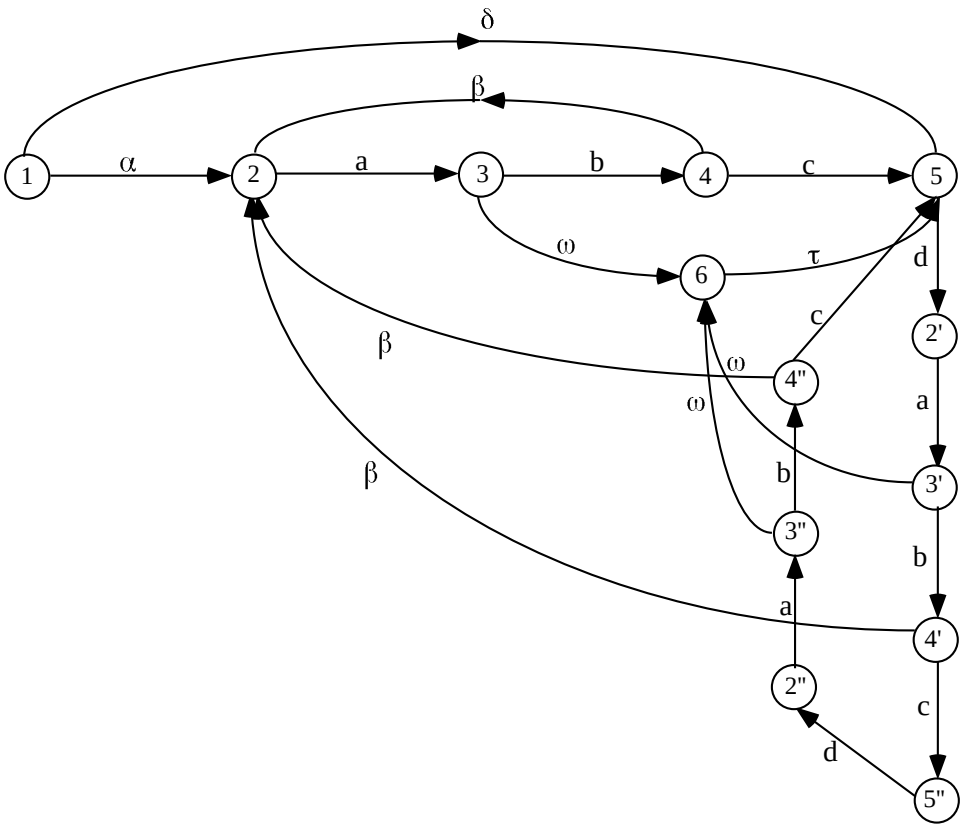


Figure 6.9: H^2 corresponding to cycle C^2 for obtaining the refined system model G^{ref}

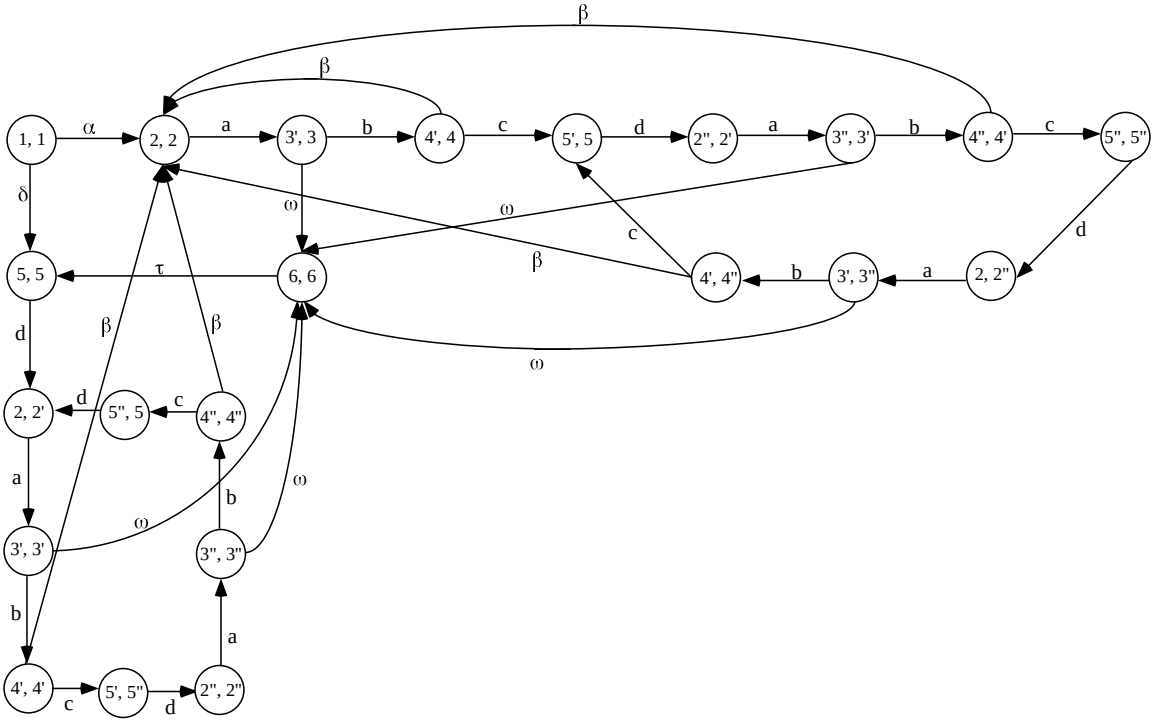


Figure 6.10: G'_d for obtaining the refined system model G^{ref}

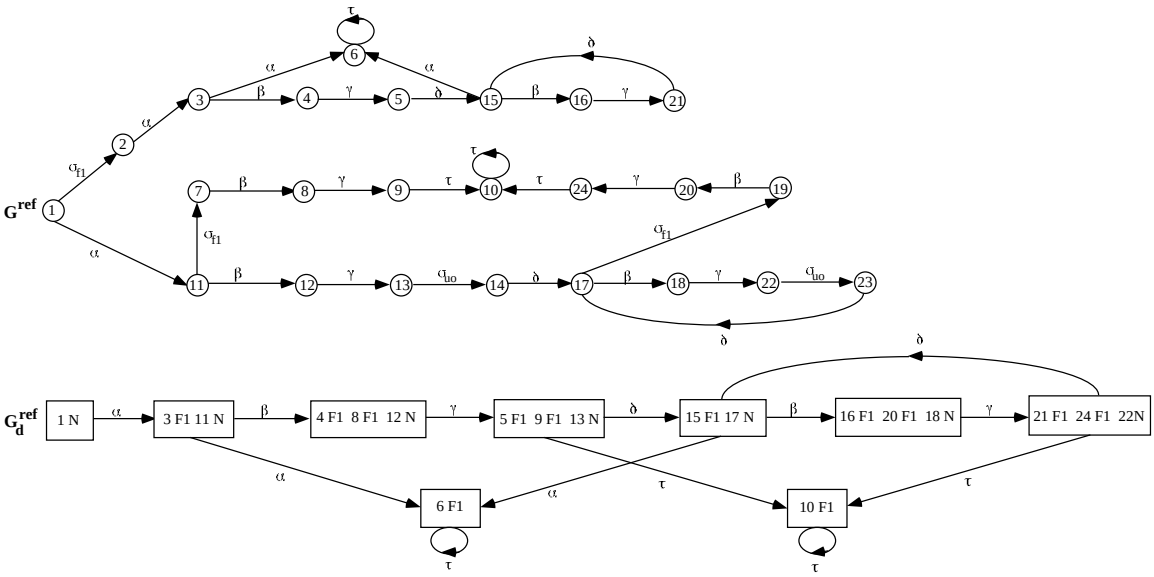


Figure 6.11: The refined system model G^{ref} and the diagnoser G_d^{ref}

6.4.6 Illustrative Example

We now present a simple example to illustrate the steps of the solution procedure for the ADP. In Section 6.5 we present a physical system and illustrate the application of the theory developed in this chapter to the design of a diagnostic controller for this system.

Example 27 Consider the system represented in Figure 6.2. As before let $\Sigma_f = \Sigma_{f1} = \{\sigma_{f1}\}$ and let $\Sigma_{uo} = \Sigma_f \cup \{\sigma_{uo}\}$. Also let $\Sigma_{uc} = \Sigma_f \cup \{\delta\}$. As seen in Example 2.1 this system has one indeterminate cycle formed by the three states $\{(3,\{F1\}),(11,\{N\})\}$, $\{(4,\{F1\}),(8,\{F1\}),(12,\{N\})\}$, and $\{(5,\{F1\}),(9,\{F1\}),(13,\{N\})\}$ with the corresponding event sequence $\beta\gamma\delta$, and it is not diagnosable.

Note that the diagnoser G_d does not contain any interleaved indeterminate cycles. We solve the ADP for this system with $K = K_1$. Figures 6.11, 6.12, and 6.13 illustrate the various steps of the solution procedure for this problem. Figure 6.11 depicts the refined system G^{ref} and the corresponding diagnoser G_d^{ref}. Note that the indeterminate cycle has been expanded out once in G_d^{ref}. G^{legal} is the same as G^{ref} in Figure 6.11 except that the event δ, which is the final event that completes the indeterminate cycle, is not defined at states 21 and 23 of G^{ref}. Likewise, G_d^{legal} is the same as G_d^{ref} except that δ is not defined at the state $\{(21,\{F1\}),(24,\{F1\}),(22,\{N\})\}$ of G_d^{ref}. Figure 6.12 corresponds to the first iteration of the solution procedure. Recalling that δ is uncontrollable we see that $H_d^{\uparrow C}(0)$ excludes from $H_d(0)(= G_d^{legal})$ the state $\{(21,\{F1\}),(24,\{F1\}),(22,\{N\})\}$ and the transitions associated with this state. It is obvious by inspection of $H_d^{live}(0)$ that $M(0) = L(H(0))$ is not diagnosable; notice the self-loop due to the *Stop* event at the F_1-uncertain state $\{(16,\{F1\}),(20,\{F1\}),(18,\{N\})\}$ in $H_d^{live}(0)$. Hence we continue to iterate.

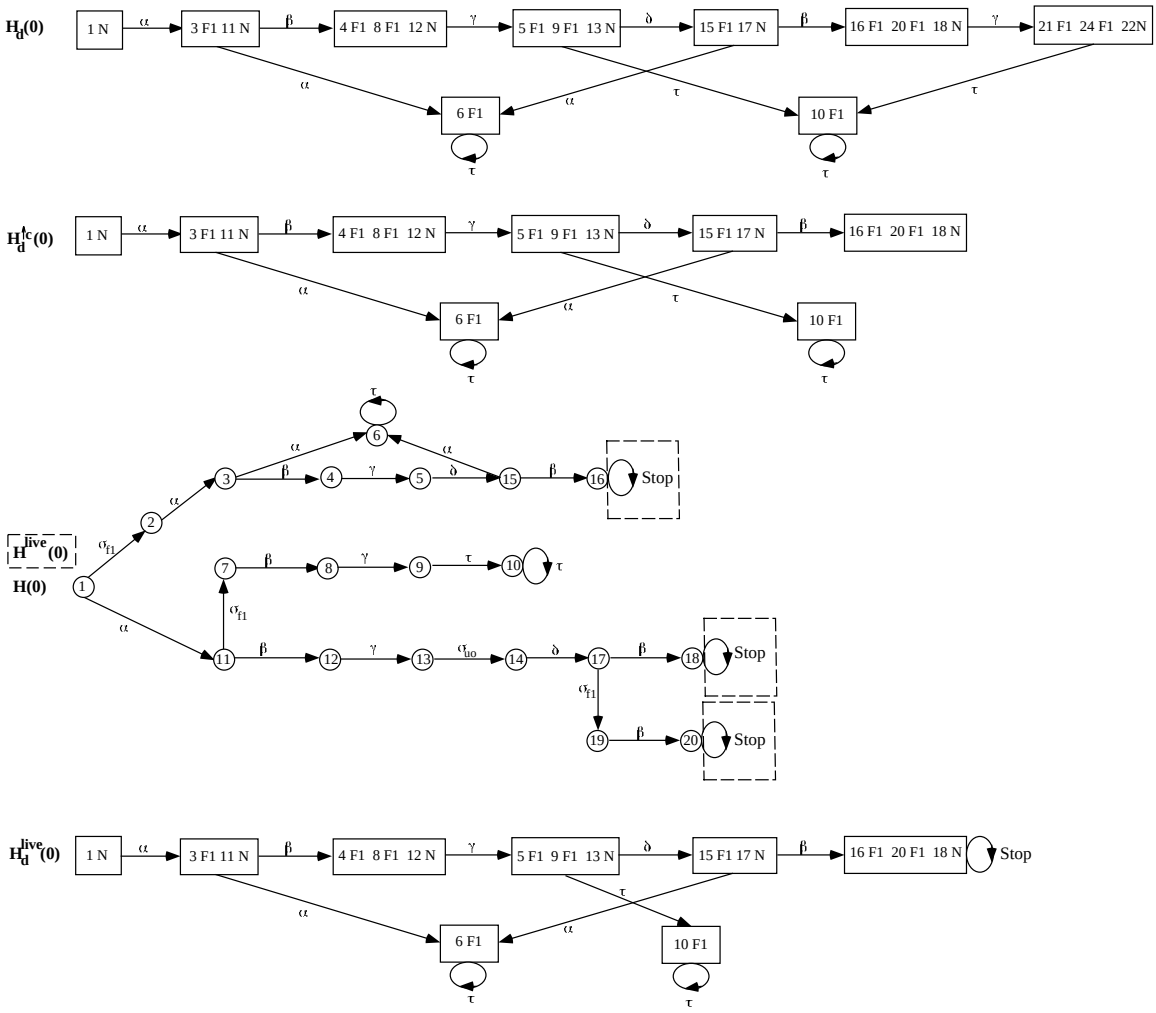


Figure 6.12: Iteration 1

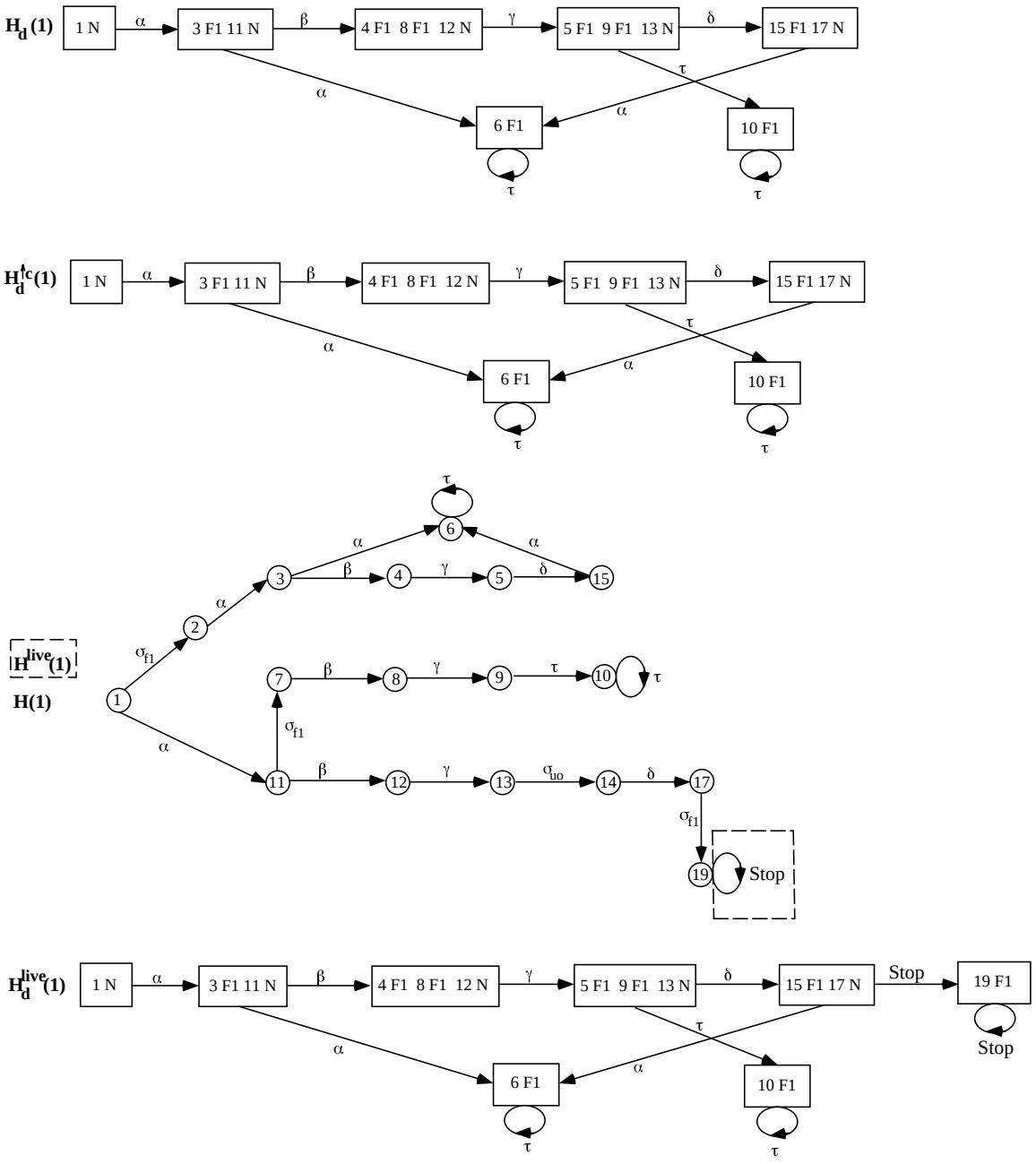


Figure 6.13: Iteration 2

Figure 6.13 depicts the various steps in the second iteration. To obtain $H_d(1)$ we remove from $H_d^{\uparrow C}(0)$ the state $\{(16,\{F1\}),(20,\{F1\}),(18,\{N\})\}$ and the associated transitions. Notice that $H_d(1)$ is a submachine of $H_d(0)$. Since the event β leading into the "illegal" state $\{(16,\{F1\}),(20,\{F1\}),(18,\{N\})\}$ is controllable, $H_d^{\uparrow C}(1)$ is the same as $H_d(1)$. Inspection of $H_d^{live}(1)$ reveals that $M(1) = L(H(1))$ is diagnosable. Thus the procedure terminates in two iterations; the solution to the ADP is $L^{act} = M(1)$ and the supervisor S_P that synthesizes $M(1)$ is realised by $H_d^{\uparrow C}(1)$, the diagnoser of $H(1)$. Further, $H_d^{live}(1)$ can be used to perform on-line diagnosis of the closed-loop formed by the system G and the controller $H_d^{\uparrow C}(1)$. ■

6.5 Application: Design of a Diagnostic Controller for a Pump-Valve System

In this section we demonstrate the application of the theory developed in this chapter to the design of a diagnostic controller for a pump-valve system.

Recall the system consisting of a pump and a valve that was studied in Examples 18 and 19 in Chapter 3. Again, we assume that the valve has two failure modes, a stuck-open failure mode and a stuck-closed failure mode and that the system is equipped with just one sensor, a flow sensor that can read one of two possible values: F, indicating that there is a flow and NF, indicating that there is no flow. Suppose that we need to design a controller for this system that achieves the following objectives:

1. When there is a load on the system, the controller must respond by starting the pump and opening the valve.

2. When there is no load on the system, the controller must respond by stopping the pump and closing the valve.

3. The sequence of start-pump, stop-pump, close-valve, open-valve commands has to be chosen in a manner such that the resulting system is diagnosable.

Figure 6.14 depicts the FSM models for the pump and the valve (repeated here from Figure 3.12 for convenience). Also shown in Figure 6.14 is a controller model. This controller captures requirements 1 and 2 above and is straighforward to build. The problem now is to design a second controller that ensures diagnosability of the closed-loop system while still meeting objectives 1 and 2. We now demonstrate how this problem can be solved in the framework of the ADP.

The first step is to obtain the system model G. In this example G is chosen to be the closed-loop system formed by the pump, the valve, and the controller of Figure 6.14. The sensor map for this system is listed in Table 6.1[9]. The modeling formalism presented

[9]As before, the •s in Table 6.1 stand for the state of the controller, and are used to indicate that the

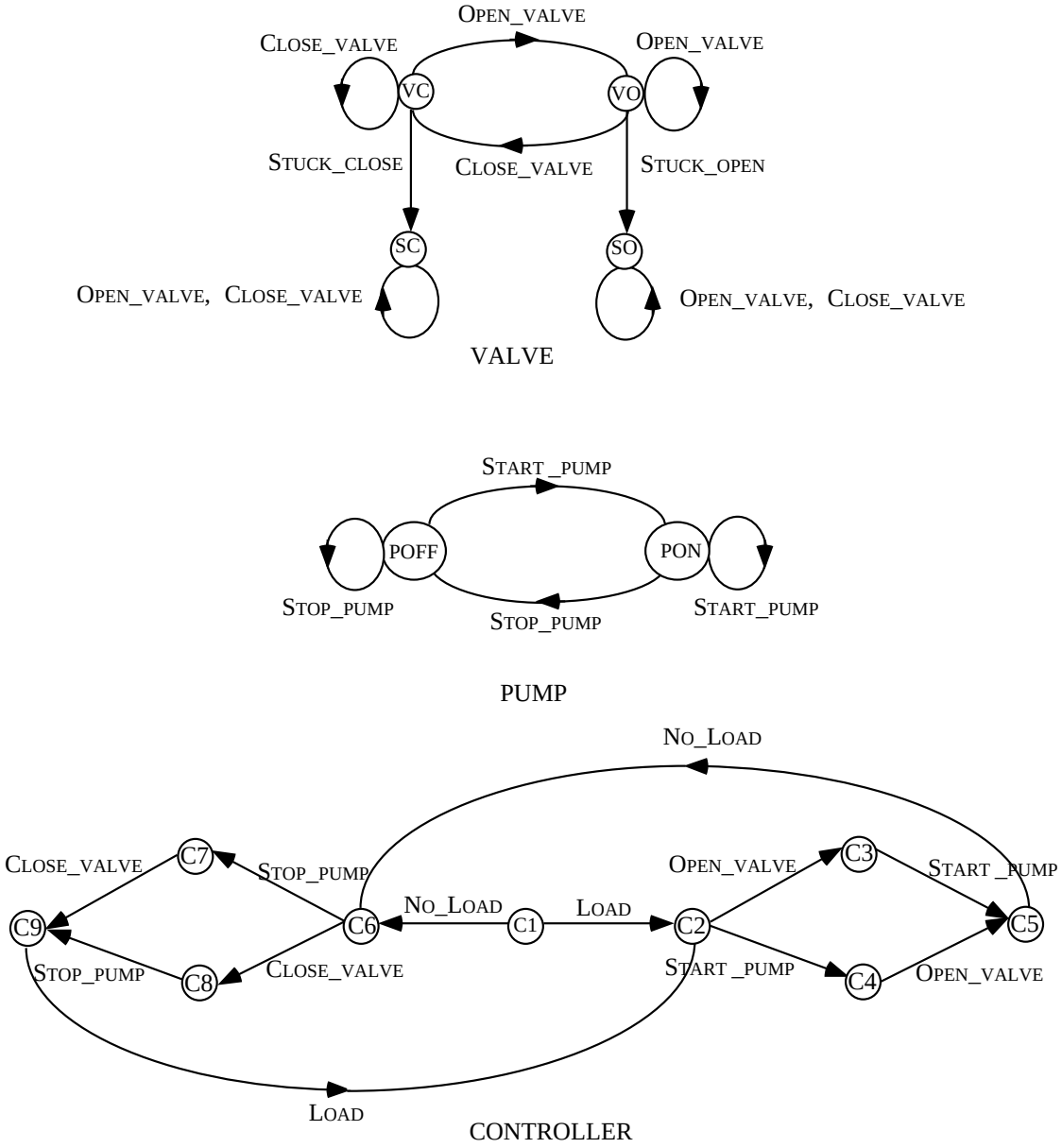


Figure 6.14: Component models for the pump-valve system

$h($ VC, POFF, $\bullet$ $)$	=	NF
$h($ VO, POFF, $\bullet$ $)$	=	NF
$h($ SC, POFF, $\bullet$ $)$	=	NF
$h($ SO, POFF, $\bullet$ $)$	=	NF
$h($ VC, PON, $\bullet$ $)$	=	NF
$h($ VO, PON, $\bullet$ $)$	=	F
$h($ SC, PON, $\bullet$ $)$	=	NF
$h($ SO, PON, $\bullet$ $)$	=	F

Table 6.1: The global sensor map for the pump-valve system

in Chapter 2 translates all sensor information into the event set. As a result, the global system model consists of "composite" events of the form <COMMAND, RESULTANT SENSOR READINGS>, as for example, the event <START_PUMP, F>. Note that while the command event is a controllable event, the event corresponding to the resultant sensor readings is uncontrollable. Hence, where it is necessary to partition the event set of the system into controllable and uncontrollable events, as in the active diagnosis problem studied here, use of the composite events poses a difficulty. Two approaches could be followed to overcome this difficulty. The first approach is to break every composite event in the global system model into two events: (i) the command and (ii) the resultant sensor readings; the command event is then treated as a controllable event while the event corresponding to the sensor readings is considered to be uncontrollable. Note that this leads to the introduction of new states in the system model. The second approach is based on the notion of control patterns [16], [52]. Simply speaking, the use of control patterns implies that the set of controllable events that are to be enabled or disabled at any point of time cannot be arbitrarily chosen but are constrained to be within prespecified subsets, i.e., certain events may only be enabled or disabled together as a group. In our modeling framework, this amounts to classifying the composite events as controllable events and requiring that all events of the form <COMMAND A, Y1>, <COMMAND A, Y2>, ..., <COMMAND A, YN> should be enabled/disabled as a group.

While the procedure of [21] for computing the supremal controllable sublanguage of a given language does not handle control patterns, it can be modified to do so in a straightforward manner. Thus either of the two approaches above can be used. However, for ease of representation, the event labels in the figures and tables that follow are left as composite events. A complete listing of the transition table for the pump-valve system is included in Appendix C. If the first approach of breaking the composite events is adopted, then the set of controllable events in this system are the command events, OPEN_VALVE, CLOSE_VALVE, STOP_PUMP, and START_PUMP, while the uncontrollable events are the failure events, STUCK_CLOSED and STUCK_OPEN, and the events corresponding to the presence and the absence of a load on the system, namely, LOAD, and NO_LOAD. On the other hand, if we use control patterns, then the controllable subsets involve events of the form <OPEN_VALVE, X>, <CLOSE_VALVE, X>, <STOP_PUMP, X>, and <START_PUMP, X> where X can take either of the two values, F, or NF. Finally, the set of failure events is partitioned into two sets $\Sigma_{f1} = \{$STUCK_CLOSED$\}$ and $\Sigma_{f2} = \{$STUCK_OPEN$\}$.

The diagnoser G_d for this system (with the composite events) consists of 32 states and is shown in Figure 6.15. (It is not difficult to see that the diagnoser for this system with the composite events broken up into two events will consist of 64 states.) A careful examination of the diagnoser G_d and the system G reveals that the cycle formed by the highlighted states in Figure 6.15 is an F_2-indeterminate cycle; as long as the diagnoser remains in this cycle,

sensor map is independent of the controller state.

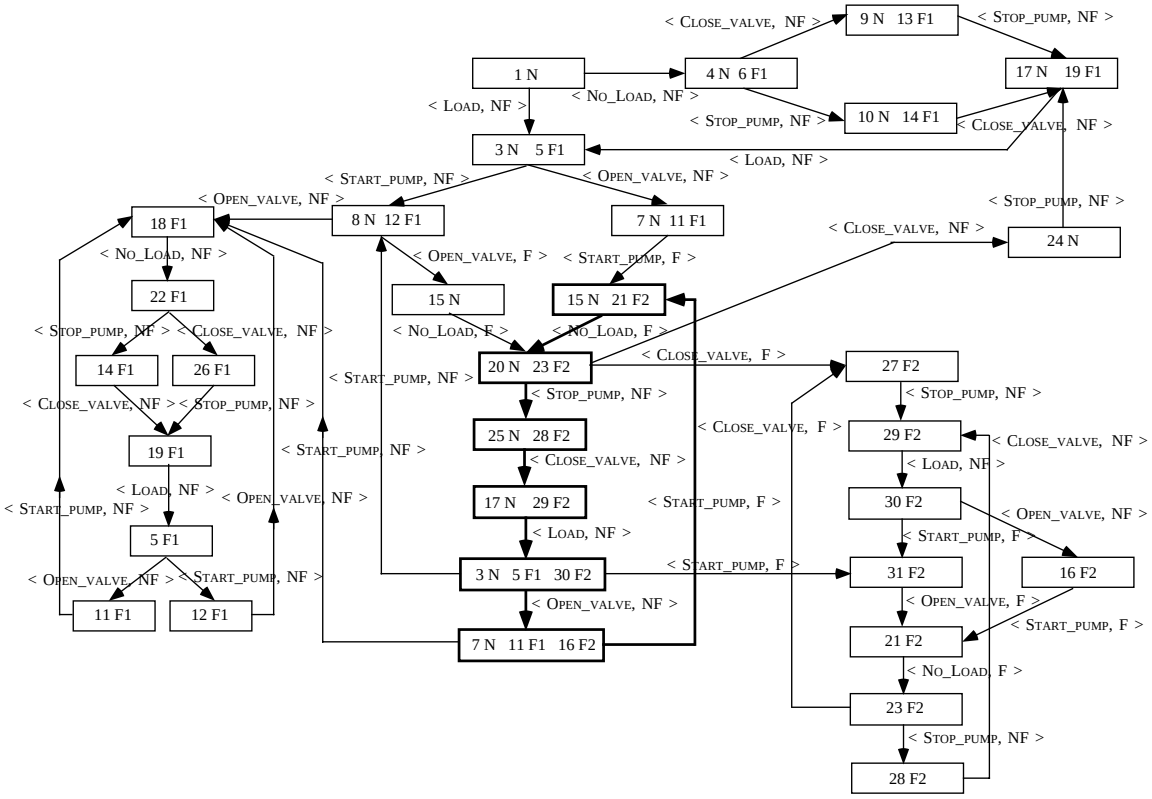


Figure 6.15: The diagnoser for the pump-valve system

one cannot detect stuck-open failures of the valve.

Since the diagnoser G_d contains only one indeterminate cycle, it follows that there are no interleaved indeterminate cycles and hence we can formulate the ADP in terms of the languages K_l. Suppose that we wish to solve the ADP for this system with $l = 0$. It is then straightforward to see that G^{ref} is the same as G, and that G_d^{ref} is the same as G_d. G_d^{legal}, the diagnoser corresponding to K_0, and hence $H_d(0)$, are obtained by disabling the event START_PUMP at the state $\{(7,\{N\}),(11,\{F1\}),(16,\{F2\})\}$ of the diagnoser (which is equivalent to disabling the events <START_PUMP, F> and the event <START_PUMP, NF> in Figure 6.15). Note that START_PUMP is a controllable event and hence $H_d^{\uparrow C}(0)$ is identical to $H_d(0)$. Now, disabling the event START_PUMP results in a deadlock both in the diagnoser $H_d^{\uparrow C}(0)$ and in the corresponding system $H(0)$ (as can be verified by inspection of the transition structure of G). In other words we get $M(0) = L(H(0))$ to be non-live. Since the state $\{(7,\{N\}),(11,\{F1\}),(16,\{F2\})\}$ is both F_1-uncertain and F_2-uncertain and since all three states, 7, 11, and 16 are dead states of the corresponding system $H(0)$ it can be seen that the diagnoser $H_d^{live}(0)$ (not shown here) has an indeterminate cycle consisting of the state $\{(7,\{N\}),(11,\{F1\}),(16,\{F2\})\}$ with a self-loop due to the *Stop* event. Thus $M(0)$ is not diagnosable and we proceed with the iteration.

It is easy to see that in the next iteration we have $H_d(1)$ to be the submachine of $H_d(0)$ obtained by eliminating the state $\{(7,\{N\}),(11,\{F1\}),(16,\{F2\})\}$ from $H_d(0)$. Since the

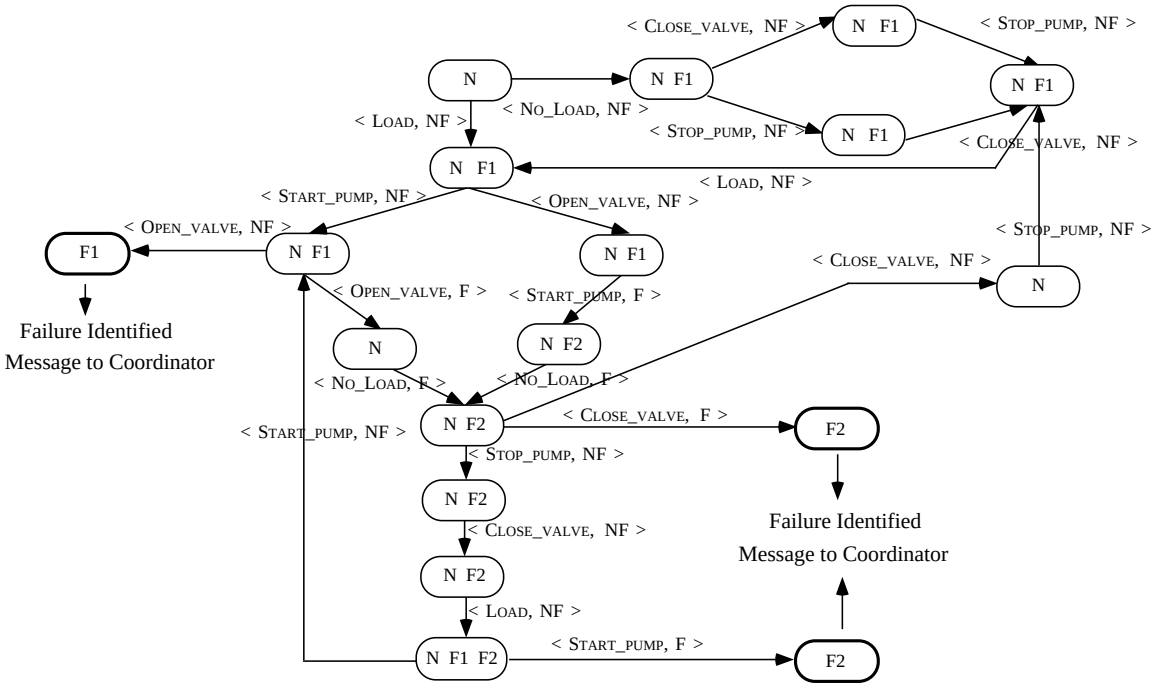


Figure 6.16: Part of the diagnostic controller for the pump-valve system

event <Open_Valve, X> is controllable, then $H_d^{\uparrow C}(1)$ is the same as $H_d(1)$. Further the resulting system $H(1)$ can be verified to have no dead states, i.e., $M(1)$ is live. It follows from Property 2 that $M(1)$ is diagnosable and hence is the solution to the ADP. The fact that $M(1)$ is diagnosable can also be ascertained by inspection of the corresponding diagnoser $H_d^{\uparrow C}(1)$. Thus the diagnostic controller for the pump-valve system of Figure 6.14 is realized by $H_d^{\uparrow C}(1)$. A partial realization (upto the point where failures are diagnosed) of the controller is depicted in Figure 6.16. Note that this controller satisfies all three design objectives: it guarantees that the system responds appropriately to the presence or absence of a load *and* it results in a diagnosable system.

Finally, it is interesting to note that this design approach results in a much less restrictive system behavior than what could be obtained by a fixed control protocol which also achieves the same design objectives. Consider for instance the control sequence, <Start_Pump>, <Open_Valve>, <Close_Valve>, <Stop_Pump>. This control sequence satisfies both design objectives. This was the control sequence in Example 18. We saw ealier that this controller results in a diagnosable system. However, it is evident that the controller of Figure 6.16 obtained as a solution of the ADP achieves a much larger closed-loop behavior. Consider next the control sequence <Open_Valve>, <Start_Pump>, <Stop_Pump>, <Close_Valve>. While this protocol also achieves the desired control objectives, it does not result in a diagnosable system, as we saw in Example 19. However, the diagnostic controller for this system allows the above control sequence to occur provided that this sequence is followed by the sequence <Start_Pump>, <Open_Valve> which results in a diagnosable system. Thus, starting from a controller that allows maximum flexibility of

the closed-loop system while achieving the desired control objectives, and then restricting its behavior to yield a diagnosable system, results in a much larger closed-loop behavior than that obtainable by a fixed control protocol.

6.6 Conclusion

In this chapter we have investigated the problem of *integrated control and diagnosis*, and presented an iterative procedure for the design of diagnostic controllers for DES. This procedure can be implemented using finite state machines, and is guaranteed to converge in a finite number of iterations. Further, the solution procedure provides both a controller that ensures diagnosability of the closed-loop system and a diagnoser for on-line failure diagnosis.

Almost universally, "control " in the context of failure diagnosis has referred to the notion of "testing" or "probing", and is more of an off-line diagnosis problem (see, e.g., [25], [3], [11], [49]); potential prior failures of the system are diagnosed by issuing test commands and observing the system responses. The challenge in these problems is the design of the "probe sequence" or the "test vector". The active diagnosis problem studied in this chapter differs significantly from prior work in this area in its basic philosophy; the emphasis here is on the design of diagnosable systems by appropriate design of the system controller. However, the solution methodology that we presented here for the design of diagnostic controllers can also be used, with some modifications, to solve the problem of test/probe vector determination for off-line diagnosis. Loosely speaking, the set of all probe sequences that achieve diagnosis of system failures may be determined, in the framework of the active diagnosis problem, by starting from a completely flexible system that allows for all feasible probe sequences and eliminating those that do not achieve diagnosis with a finite delay. This amounts to eliminating those traces that lead to indeterminate cycles in the corresponding diagnoser. Finally, one may further restrict the set of feasible test vectors by taking into consideration factors such as detection delay, cost of probing, etc.

CHAPTER 7

Conclusion

Editor: I like your book except for the ending.
Author: What's wrong with the ending?
Editor: It should be closer to the beginning.

The Creative Whack Pack, Roger Von Oeck

In this final chapter, we first summarize the main contributions of this thesis. Next, we compare our work with some of the other approaches to failure diagnosis that we discussed in Chapter 1. Finally, we outline some directions for future research.

7.1 Contributions of this Thesis

In this thesis, we have attempted to lay the foundations for a theory of diagnosis in the framework of DES. The main contributions of the thesis are:

- a systematic methodology for building discrete event models of physical systems for failure diagnosis;
- a methodology to implement on-line failure diagnosis in industrial systems;
- a formal framework to analyze and study the diagnosability properties of systems; and
- a systematic procedure to design diagnostic controllers that result in diagnosable systems.

7.2 Comparison with Other Approaches to Failure Diagnosis

We now provide a brief comparison of our approach to failure diagnosis with some of the other approaches discussed earlier, namely, analytical redundancy methods, fault trees, and model-based reasoning. Our comparison will be based on each of the main contributions of this thesis mentioned above.

On Model Building

- The analytical redundancy methods make use of detailed differential equation models of the system. A major advantage of our approach is that it does not require such in-depth modeling. Indeed, for sharp failures as the ones we are interested in diagnosing, a higher level abstraction such as discrete event models appears quite sufficient.
- 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 attempts to separate local and global behavior. Local behavior is captured in a completely modular manner by the individual component models (component interactions show up as common events of the interacting components). The global behavior is refelected in the sensor maps. Starting from FSM models of the individual components and from a listing of the sensor maps we build the complete model in a systematic manner. The FSM models are composed automatically by means of the synchronous composition; likewise, the composite system model is obtained by a systematic transition-renaming procedure from the synchronous composition model and from the sensor maps. In our opinion, the variable-based digraph models cannot be manipulated as easily and often lead to difficulties in going from local to global behaviour (e.g., the problems of "negative feedback loops" [23], [30] and "apparent non-local causality" [30]).
- 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 FSMs [33], [13], qualitative differential equations [12], [29], constraint equations [17], signed-digraphs [30], and so forth. However, in the work on model-based reasoning that we have seen, there is no systematic method to generate the complete system model as we propose. Further, the model based methods, in general, deal only with models of correct, or, normal behavior; often, there are no explicit fault models.

On the Diagnostic Process

- The 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.

- 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.
- Multiple system failures, or goal violations, do not seem to be straightforward to handle in the case of fault trees. Our approach handles single as well as multiple failures in the same framework.
- 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.
- Most of the approaches in model-based reasoning that have been proposed so far for dynamic systems are based on diagnostic schemes developed for static systems; the system behavior is predicted by simulation (the most popular method for this appears to be the use of qualitative differential equations) and the diagnostic reasoning makes use of techniques such as constraint satisfaction (see, e.g., [12], [4], [50]). Often, the initial failure hypothesis is obtained from decision trees or fault trees. Thus the overall diagnostic scheme differs considerably from the approach that we have taken.

On Diagnosability Properties of Systems

- As mentioned previously, most of the other approaches, including the model-based diagnostic approaches, do not attempt to explicitly characterize and study the diagnosability properties of systems. In [25] and in [3] the authors study the notions of off-line and on-line diagnosability and the notion of testability, respectively. However, these notions are introduced in the context of "testing" and not in the context of passive diagnosis as in our case.

On Diagnostic Controllers

- As mentioned in Chapter 6 control in the context of failure diagnosis has referred, almost universally, to the notion of testing; the active diagnosis problem studied in this chapter, however, emphasizes the the design of diagnosable systems by appropriate design of the system controller. Such an integrated approach to control and diagnosis, to the best of our knowledge, has not been attempted before.

We note at this point that the proposed method is complementary, if not an alternative, to the other approaches. Most large scale systems involve failures of different nature and characteristics; further the quantity and the quality of knowledge available may vary between different subsystems and processes that constitute the system. A diagnostic module for these systems is most likely to involve a judicious integration of different sub-modules based on different techniques of failure diagnosis, and designed for diagnosing different classes of failures. As we have seen our approach is eminently suited for the class of sharp failures, or failures that cause a distinct change in the system state. Further, we note that several of the other approaches that we mentioned earlier such as the analytical model-based methods and the statistical decision theory-based methods involve a decision logic phase for failure isolation (see, e.g., [37]); it appears that our diagnostic procedure could provide this decision logic. For instance, the residual signals that are used for failure detection by the analytical redundancy methods, could be used as "virtual sensors", in addition to physical sensors in the system, as inputs to our diagnostic process to detect and isolate failures.

7.3 Directions for Future Work

Based on our work so far, we have identified several directions for future work which are outlined below.

- As we mentioned earlier in this thesis, one approach to designing diagnosable systems is by equipping the system with the appropriate set of sensors. The theory that we have developed so far can help in answering the following question: "given a set of sensors, is the system diagnosable?" This theory, in conjunction with the software tools that we have developed, have helped us assess the usefulness of specific sensors for realistic systems [41]. What remains to be developed is a systematic way of determining the set of sensors that need to be installed in the system in order to achieve diagnosability, taking into consideration additional factors such as the cost of sensors, technological feasibility, and so on.

- Diagnosability of a system depends not only on the observable event projection (which leads to the abovementioned sensor problem) but it also depends on the partition defined on the set of failure events. Though this partition is often a fixed, given entity based on the diagnostic requirements for the system, in situations where the partition is not specified a priori, a systematic procedure to determine the finest partition of failure events with respect to which the system is diagnosable can prove valuable to the designer. Determining this finest partition (when it exists), or, determining the set of partitions which satisfies the property that for each partition in this set there exists no other finer partition with respect to which the system is diagnosable (when the finest partition does not exist) are interesting problems for future investigation.

- The thesis restricts attention to untimed logical discrete event models. The use of timing information can further enhance the capabilities of the diagnostic process, especially for systems involving sequential execution of batch processes, like automated manufacturing systems, in which events have to occur within prespecified time limits. Another motivation to study timed models is when one needs to characterize the detection delay in terms of time elapsed before detection as opposed to the number of events that can occur in the system before failures are detected.

- The diagnosis provided by the diagnoser depends on two factors: (i) the system model from which the diagnoser is synthesized, and (ii) the observation sequences that the diagnoser "sees". Two factors that influenze the accuracy of the diagnosis are then, sensitivity to unmodeled dynamics, and resilience of the diagnostic process to observation errors. There can be several sources of observation errors. For instance, the diagnoser may miss a transition, it may mistake one transition for another, or it may assume there was a transition when there was none. This leads us to the problem of *resilient* diagnosers. In [31] the authors study the problem of resilient observers. Along similar lines, a resilient diagnoser may be defined to be one that is capable of recovering from a finite burst of errors and correcting a wrong diagnosis within a finite delay. Unmodelled dynamics are likely to occur in situations where there is insufficient knowledge of the system, and in particular, when one cannot determine a priori the set of all possible failures that can occur in the system. At this point, we conjecture that unmodelled events will lead to inconsistencies in the observations seen by the diagnoser, i.e., the diagnoser may see a transition that it does not expect to see at its current state. Further studies are necessary in order to determine the nature of the inconsistencies that might arise out of such modeling errors. In practice, this problem may require an iterative model revision-diagnoser synthesis-testing process.

- Computational complexity is a major factor that can affect the applicability of our approach to large systems. Two possible ways to beat computational problems are by the use of modular diagosis and hierarchical diagnosis. These ideas have been applied to other areas of DES research such as supervisory control (see [53] and references therein). Hierarchical methods of diagnosis have also been studied by several researchers in model-based reasoning (see [11] and references therein).

We conclude this chapter with the following remarks. The application area that we have primarily focussed on in this thesis is that of HVAC systems. However, our approach is by no means restricted to these systems. The generic nature of the models that we have presented, and the generality of our modeling methodology indicate that this approach to failure diagnosis could potentially be applied to a large class of systems, including semiconductor manufacturing systems, automobiles, locomotive engines, and so on. Good modeling techniques coupled with "engineering" approaches to tackling computational complexity can

enhance the applicability of this method to handle "real" systems. To conclude, we believe that this thesis has not only laid a strong foundation for a DES-based theory of diagnosis but has also provided a viable methodology for failure diagnosis in industrial systems.

APPENDICES

APPENDIX A

Transition Table for HVAC System I of Chapter 5

	x	σ	$\delta(x,\sigma)$
1	(C1.V1.F1.P1.B1.L0)	< CFOFF >	(C21.V1.F1.P1.B1.L0)
		< CFON >	(C11.V1.F1.P1.B1.L0)
		< SC1 >	(C1.V4.F1.P1.B1.L0)
		< SO1 >	(C1.V3.F1.P1.B1.L0)
		< FON, NF >	(C2.V1.F2.P1.B1.L0)
31	(C11.V1.F1.P1.B1.L0)	< SC1 >	(C11.V4.F1.P1.B1.L0)
		< SO1 >	(C11.V3.F1.P1.B1.L0)
		< FON, NF >	(C12.V1.F2.P1.B1.L0)
79	(C21.V1.F1.P1.B1.L0)	< SC1 >	(C21.V4.F1.P1.B1.L0)
		< SO1 >	(C21.V3.F1.P1.B1.L0)
		< FON, NF >	(C22.V1.F2.P1.B1.L0)
2	(C1.V3.F1.P1.B1.L0)	< CFOFF >	(C21.V3.F1.P1.B1.L0)
		< CFON >	(C11.V3.F1.P1.B1.L0)
		< FON, NF >	(C2.V3.F2.P1.B1.L0)
32	(C11.V3.F1.P1.B1.L0)	< FON, NF >	(C12.V3.F2.P1.B1.L0)
80	(C21.V3.F1.P1.B1.L0)	< FON, NF >	(C22.V3.F2.P1.B1.L0)
3	(C1.V4.F1.P1.B1.L0)	< CFOFF >	(C21.V4.F1.P1.B1.L0)
		< CFON >	(C11.V4.F1.P1.B1.L0)
		< FON, NF >	(C2.V4.F2.P1.B1.L0)
33	(C11.V4.F1.P1.B1.L0)	< FON, NF >	(C12.V4.F2.P1.B1.L0)
81	(C21.V4.F1.P1.B1.L0)	< FON, NF >	(C22.V4.F2.P1.B1.L0)
34	(C12.V1.F2.P1.B1.L0)	< SC1 >	(C12.V4.F2.P1.B1.L0)
		< SO1 >	(C12.V3.F2.P1.B1.L0)
		< SPD, NF >	(C20.V1.F2.P1.B1.L1)
		< SPI, NF >	(C13.V1.F2.P1.B1.L2)
4	(C2.V1.F2.P1.B1.L0)	< SC1 >	(C2.V4.F2.P1.B1.L0)
		< SO1 >	(C2.V3.F2.P1.B1.L0)
		< SPD, NF >	(C10.V1.F2.P1.B1.L1)
		< SPI, NF >	(C3.V1.F2.P1.B1.L2)

	x	σ	$\delta(x,\sigma)$
82	(C22.V1.F2.P1.B1.L0)	< SC1 >	(C22.V4.F2.P1.B1.L0)
		< SO1 >	(C22.V3.F2.P1.B1.L0)
		< SPD, NF >	(C23.V1.F2.P1.B1.L1)
		< SPI, NF >	(C24.V1.F2.P1.B1.L2)
35	(C12.V3.F2.P1.B1.L0)	< SPD, NF >	(C20.V3.F2.P1.B1.L1)
		< SPI, NF >	(C13.V3.F2.P1.B1.L2)
5	(C2.V3.F2.P1.B1.L0)	< SPD, NF >	(C10.V3.F2.P1.B1.L1)
		< SPI, NF >	(C3.V3.F2.P1.B1.L2)
83	(C22.V3.F2.P1.B1.L0)	< SPD, NF >	(C23.V3.F2.P1.B1.L1)
		< SPI,NF >	(C24.V3.F2.P1.B1.L2)
36	(C12.V4.F2.P1.B1.L0)	< SPD, NF >	(C20.V4.F2.P1.B1.L1)
		< SPI, NF >	(C13.V4.F2.P1.B1.L2)
6	(C2.V4.F2.P1.B1.L0)	< SPD, NF >	(C10.V4.F2.P1.B1.L1)
		< SPI, NF >	(C3.V4.F2.P1.B1.L2)
84	(C22.V4.F2.P1.B1.L0)	< SPD, NF >	(C23.V4.F2.P1.B1.L1)
		< SPI, NF >	(C24.V4.F2.P1.B1.L2)
28	(C10.V1.F2.P1.B1.L1)	< SC1 >	(C10.V4.F2.P1.B1.L1)
		< SO1 >	(C10.V3.F2.P1.B1.L1)
		< FOFF, NF >	(C1.V1.F1.P1.B1.L0)
		< SPI, NF >	(C3.V1.F2.P1.B1.L2)
49	(C20.V1.F2.P1.B1.L1)	< SC1 >	(C20.V4.F2.P1.B1.L1)
		< SO1 >	(C20.V3.F2.P1.B1.L1)
		< OV, NF >	(C19.V2.F2.P1.B1.L1)
88	(C23.V1.F2.P1.B1.L1)	< SC1 >	(C23.V4.F2.P1.B1.L1)
		< SO1 >	(C23.V3.F2.P1.B1.L1)
		< FOFF, NF >	(C21.V1.F1.P1.B1.L0)
		< SPI, NF >	(C24.V1.F2.P1.B1.L2)
52	(C19.V2.F2.P1.B1.L1)	< SC2 >	(C19.V4.F2.P1.B1.L1)
		< SO2 >	(C19.V3.F2.P1.B1.L1)
		< PON, F >	(C18.V2.F2.P2.B1.L1)
29	(C10.V3.F2.P1.B1.L1)	< FOFF, NF >	(C1.V3.F1.P1.B1.L0)
		< SPI, NF >	(C3.V3.F2.P1.B1.L2)
53	(C19.V3.F2.P1.B1.L1)	< PON, F >	(C18.V3.F2.P2.B1.L1)
50	(C20.V3.F2.P1.B1.L1)	< OV, NF >	(C19.V3.F2.P1.B1.L1)
89	(C23.V3.F2.P1.B1.L1)	< FOFF, NF >	(C21.V3.F1.P1.B1.L0)
		< SPI, NF >	(C24.V3.F2.P1.B1.L2)
30	(C10.V4.F2.P1.B1.L1)	< FOFF, NF >	(C1.V4.F1.P1.B1.L0)
		< SPI, NF >	(C3.V4.F2.P1.B1.L2)
54	(C19.V4.F2.P1.B1.L1)	< PON, NF >	(C18.V4.F2.P2.B1.L1)

	x	σ	$\delta(x,\sigma)$
51	(C20.V4.F2.P1.B1.L1)	< OV, NF >	(C19.V4.F2.P1.B1.L1)
90	(C23.V4.F2.P1.B1.L1)	< FOFF, NF >	(C21.V4.F1.P1.B1.L0)
		< SPI, NF >	(C24.V4.F2.P1.B1.L2)
55	(C18.V2.F2.P2.B1.L1)	< BON, F >	(C17.V2.F2.P2.B2.L1)
		< SC2 >	(C18.V4.F2.P2.B1.L1)'
		< SO2 >	(C18.V3.F2.P2.B1.L1)
56	(C18.V3.F2.P2.B1.L1)	< BON, F >	(C17.V3.F2.P2.B2.L1)
57	(C18.V4.F2.P2.B1.L1)	< BON, NF >	(C17.V4.F2.P2.B2.L1)
25	(C9.V1.F2.P1.B2.L1)	< BOFF, NF >	(C10.V1.F2.P1.B1.L1)
		< SC1 >	(C9.V4.F2.P1.B2.L1)
		< SO1 >	(C9.V3.F2.P1.B2.L1)
26	(C9.V3.F2.P1.B2.L1)	< BOFF, NF >	(C10.V3.F2.P1.B1.L1)
27	(C9.V4.F2.P1.B2.L1)	< BOFF, NF >	(C10.V4.F2.P1.B1.L1)
22	(C8.V1.F2.P2.B2.L1)	< SC1 >	(C8.V4.F2.P2.B2.L1)
		< SO1 >	(C8.V3.F2.P2.B2.L1)'
		< POFF, NF >	(C9.V1.F2.P1.B2.L1)
67	(C17.V2.F2.P2.B2.L1)	< SC2 >	(C17.V4.F2.P2.B2.L1)'
		< SO2 >	(C17.V3.F2.P2.B2.L1)
		< SPI, F >	(C13.V2.F2.P2.B2.L2)
64	(C18.V2.F2.P2.B2.L1)	< BON, F >	(C17.V2.F2.P2.B2.L1)
		< SC2 >	(C18.V4.F2.P2.B2.L1)'
		< SO2 >	(C18.V3.F2.P2.B2.L1)
61	(C19.V2.F2.P2.B2.L1)	< SC2 >	(C19.V4.F2.P2.B2.L1)'
		< SO2 >	(C19.V3.F2.P2.B2.L1)
		< PON, F >	(C18.V2.F2.P2.B2.L1)
58	(C20.V2.F2.P2.B2.L1)	< SC2 >	(C20.V4.F2.P2.B2.L1)'
		< SO2 >	(C20.V3.F2.P2.B2.L1)
		< OV, F >	(C19.V2.F2.P2.B2.L1)
19	(C7.V2.F2.P2.B2.L1)	< CV, NF >	(C8.V1.F2.P2.B2.L1)
		< SC2 >	(C7.V4.F2.P2.B2.L1)'
		< SO2 >	(C7.V3.F2.P2.B2.L1)
68	(C17.V3.F2.P2.B2.L1)	< SPI, F >	(C13.V3.F2.P2.B2.L2)
65	(C18.V3.F2.P2.B2.L1)	< BON, F >	(C17.V3.F2.P2.B2.L1)
62	(C19.V3.F2.P2.B2.L1)	< PON, F >	(C18.V3.F2.P2.B2.L1)
59	(C20.V3.F2.P2.B2.L1)	< OV, F >	(C19.V3.F2.P2.B2.L1)
20	(C7.V3.F2.P2.B2.L1)	< CV, F >	(C8.V3.F2.P2.B2.L1)
23	(C8.V3.F2.P2.B2.L1)	< POFF, NF >	(C9.V3.F2.P1.B2.L1)
69	(C17.V4.F2.P2.B2.L1)	< SPI, NF >	(C13.V4.F2.P2.B2.L2)
66	(C18.V4.F2.P2.B2.L1)	< BON, NF >	(C17.V4.F2.P2.B2.L1)

	x	σ	$\delta(x,\sigma)$
63	(C19.V4.F2.P2.B2.L1)	< PON, NF >	(C18.V4.F2.P2.B2.L1)
60	(C20.V4.F2.P2.B2.L1)	< OV, NF >	(C19.V4.F2.P2.B2.L1)
21	(C7.V4.F2.P2.B2.L1)	< CV, NF >	(C8.V4.F2.P2.B2.L1)
24	(C8.V4.F2.P2.B2.L1)	< POFF, NF >	(C9.V4.F2.P1.B2.L1)
37	(C13.V1.F2.P1.B1.L2)	< SC1 >	(C13.V4.F2.P1.B1.L2)
		< SO1 >	(C13.V3.F2.P1.B1.L2)
		< OV, NF >	(C14.V2.F2.P1.B1.L2)
85	(C24.V1.F2.P1.B1.L2)	< SC1 >	(C24.V4.F2.P1.B1.L2)
		< SO1 >	(C24.V3.F2.P1.B1.L2)
		< FOFF, NF >	(C21.V1.F1.P1.B1.L0)
		< SPD, NF >	(C23.V1.F2.P1.B1.L1)
7	(C3.V1.F2.P1.B1.L2)	< SC1 >	(C3.V4.F2.P1.B1.L2)
		< SO1 >	(C3.V3.F2.P1.B1.L2)
		< OV, NF >	(C4.V2.F2.P1.B1.L2)
40	(C14.V2.F2.P1.B1.L2)	< SC2 >	(C14.V4.F2.P1.B1.L2)
		< SO2 >	(C14.V3.F2.P1.B1.L2)
		< PON, F >	(C15.V2.F2.P2.B1.L2)
10	(C4.V2.F2.P1.B1.L2)	< SC2 >	(C4.V4.F2.P1.B1.L2)
		< SO2 >	(C4.V3.F2.P1.B1.L2)
		< PON, F >	(C5.V2.F2.P2.B1.L2)
38	(C13.V3.F2.P1.B1.L2)	< OV, NF >	(C14.V3.F2.P1.B1.L2)
41	(C14.V3.F2.P1.B1.L2)	< PON, F >	(C15.V3.F2.P2.B1.L2)
86	(C24.V3.F2.P1.B1.L2)	< FOFF, NF >	(C21.V3.F1.P1.B1.L0)
		< SPD, NF >	(C23.V3.F2.P1.B1.L1)
8	(C3.V3.F2.P1.B1.L2)	< OV, NF >	(C4.V3.F2.P1.B1.L2)
11	(C4.V3.F2.P1.B1.L2)	< PON, F >	(C5.V3.F2.P2.B1.L2)
39	(C13.V4.F2.P1.B1.L2)	< OV, NF >	(C14.V4.F2.P1.B1.L2)
42	(C14.V4.F2.P1.B1.L2)	< PON, NF >	(C15.V4.F2.P2.B1.L2)
87	(C24.V4.F2.P1.B1.L2)	< FOFF, NF >	(C21.V4.F1.P1.B1.L0)
		< SPD, NF >	(C23.V4.F2.P1.B1.L1)
9	(C3.V4.F2.P1.B1.L2)	< OV, NF >	(C4.V4.F2.P1.B1.L2)
12	(C4.V4.F2.P1.B1.L2)	< PON, NF >	(C5.V4.F2.P2.B1.L2)
43	(C15.V2.F2.P2.B1.L2)	< BON, F >	(C16.V2.F2.P2.B2.L2)
		< SC2 >	(C15.V4.F2.P2.B1.L2)'
		< SO2 >	(C15.V3.F2.P2.B1.L2)
13	(C5.V2.F2.P2.B1.L2)	< BON, F >	(C6.V2.F2.P2.B2.L2)
		< SC2 >	(C5.V4.F2.P2.B1.L2)'
		< SO2 >	(C5.V3.F2.P2.B1.L2)
44	(C15.V3.F2.P2.B1.L2)	< BON, F >	(C16.V3.F2.P2.B2.L2)
14	(C5.V3.F2.P2.B1.L2)	< BON, F >	(C6.V3.F2.P2.B2.L2)
45	(C15.V4.F2.P2.B1.L2)	< BON, NF >	(C16.V4.F2.P2.B2.L2)
15	(C5.V4.F2.P2.B1.L2)	< BON, NF >	(C6.V4.F2.P2.B2.L2)

	x	σ	$\delta(x,\sigma)$
70	(C13.V2.F2.P2.B2.L2)	< SC2 >	(C13.V4.F2.P2.B2.L2)'
		< SO2 >	(C13.V3.F2.P2.B2.L2)
		< OV, F >	(C14.V2.F2.P2.B2.L2)
73	(C14.V2.F2.P2.B2.L2)	< SC2 >	(C14.V4.F2.P2.B2.L2)'
		< SO2 >	(C14.V3.F2.P2.B2.L2)
		< PON, F >	(C15.V2.F2.P2.B2.L2)
76	(C15.V2.F2.P2.B2.L2)	< BON, F >	(C16.V2.F2.P2.B2.L2)
		< SC2 >	(C15.V4.F2.P2.B2.L2)'
		< SO2 >	(C15.V3.F2.P2.B2.L2)
46	(C16.V2.F2.P2.B2.L2)	< SC2 >	(C16.V4.F2.P2.B2.L2)'
		< SO2 >	(C16.V3.F2.P2.B2.L2)
		< SPD, F >	(C20.V2.F2.P2.B2.L1)
16	(C6.V2.F2.P2.B2.L2)	< SC2 >	(C6.V4.F2.P2.B2.L2)'
		< SO2 >	(C6.V3.F2.P2.B2.L2)
		< SPD, F >	(C7.V2.F2.P2.B2.L1)
71	(C13.V3.F2.P2.B2.L2)	< OV, F >	(C14.V3.F2.P2.B2.L2)
74	(C14.V3.F2.P2.B2.L2)	< PON, F >	(C15.V3.F2.P2.B2.L2)
77	(C15.V3.F2.P2.B2.L2)	< BON, F >	(C16.V3.F2.P2.B2.L2)
47	(C16.V3.F2.P2.B2.L2)	< SPD, F >	(C20.V3.F2.P2.B2.L1)
17	(C6.V3.F2.P2.B2.L2)	< SPD, F >	(C7.V3.F2.P2.B2.L1)
72	(C13.V4.F2.P2.B2.L2)	< OV, NF >	(C14.V4.F2.P2.B2.L2)
75	(C14.V4.F2.P2.B2.L2)	< PON, NF >	(C15.V4.F2.P2.B2.L2)
78	(C15.V4.F2.P2.B2.L2)	< BON, NF >	(C16.V4.F2.P2.B2.L2)
48	(C16.V4.F2.P2.B2.L2)	< SPD, NF >	(C20.V4.F2.P2.B2.L1)
18	(C6.V4.F2.P2.B2.L2)	< SPD, NF >	(C7.V4.F2.P2.B2.L1)
91	(C5.V4.F2.P2.B1.L2)'	< F -> NF >	(C5.V4.F2.P2.B1.L2)
92	(C6.V4.F2.P2.B2.L2)'	< F -> NF >	(C6.V4.F2.P2.B2.L2)
93	(C7.V4.F2.P2.B2.L1)'	< F -> NF >	(C7.V4.F2.P2.B2.L1)
94	(C8.V3.F2.P2.B2.L1)'	< NF -> F >	(C8.V3.F2.P2.B2.L1)
95	(C18.V4.F2.P2.B1.L1)'	< F -> NF >	(C18.V4.F2.P2.B1.L1)
96	(C15.V4.F2.P2.B1.L2)'	< F -> NF >	(C15.V4.F2.P2.B1.L2)
97	(C16.V4.F2.P2.B2.L2)'	< F -> NF >	(C16.V4.F2.P2.B2.L2)
98	(C20.V4.F2.P2.B2.L1)'	< F -> NF >	(C20.V4.F2.P2.B2.L1)
99	(C19.V4.F2.P2.B2.L1)'	< F -> NF >	(C19.V4.F2.P2.B2.L1)
100	(C18.V4.F2.P2.B2.L1)'	< F -> NF >	(C18.V4.F2.P2.B2.L1)
101	(C17.V4.F2.P2.B2.L1)'	< F -> NF >	(C17.V4.F2.P2.B2.L1)
102	(C13.V4.F2.P2.B2.L2)'	< F -> NF >	(C13.V4.F2.P2.B2.L2)
103	(C14.V4.F2.P2.B2.L2)'	< F -> NF >	(C14.V4.F2.P2.B2.L2)
104	(C15.V4.F2.P2.B2.L2)'	< F -> NF >	(C15.V4.F2.P2.B2.L2)

APPENDIX B

Transition Table for HVAC System II of Chapter 5

	x	σ	$\delta(x,\sigma)$
1	(C1.V1.F1.P1.B1.L0)	< PFOFF1 >	(C1.V1.F1.P4.B1.L0)
		< PFON1 >	(C1.V1.F1.P3.B1.L0)
		< SC1 >	(C1.V4.F1.P1.B1.L0)
		< SO1 >	(C1.V3.F1.P1.B1.L0)
		< FON, PNP, NF >	(C2.V1.F2.P1.B1.L0)
2	(C1.V3.F1.P1.B1.L0)	< PFOFF1 >	(C1.V3.F1.P4.B1.L0)
		< PFON1 >	(C1.V3.F1.P3.B1.L0)
		< FON, PNP, NF >	(C2.V3.F2.P1.B1.L0)
3	(C1.V4.F1.P1.B1.L0)	< PFOFF1 >	(C1.V4.F1.P4.B1.L0)
		< PFON1 >	(C1.V4.F1.P3.B1.L0)
		< FON, PNP, NF >	(C2.V4.F2.P1.B1.L0)
91	(C10.V1.F2.P1.B1.L1)	< PFOFF1 >	(C10.V1.F2.P4.B1.L1)
		< PFON1 >	(C10.V1.F2.P3.B1.L1)'
		< SC1 >	(C10.V4.F2.P1.B1.L1)
		< SO1 >	(C10.V3.F2.P1.B1.L1)
		< FOFF, PNP, NF >	(C1.V1.F1.P1.B1.L0)
		< SPI, PNP, NF >	(C3.V1.F2.P1.B1.L2)
11	(C2.V1.F2.P1.B1.L0)	< PFOFF1 >	(C2.V1.F2.P4.B1.L0)
		< PFON1 >	(C2.V1.F2.P3.B1.L0)'
		< SC1 >	(C2.V4.F2.P1.B1.L0)
		< SO1 >	(C2.V3.F2.P1.B1.L0)
		< SPD, PNP, NF >	(C10.V1.F2.P1.B1.L1)
		< SPI, PNP, NF >	(C3.V1.F2.P1.B1.L2)
92	(C10.V3.F2.P1.B1.L1)	< PFOFF1 >	(C10.V3.F2.P4.B1.L1)
		< PFON1 >	(C10.V3.F2.P3.B1.L1)'
		< FOFF, PNP, NF >	(C1.V3.F1.P1.B1.L0)
		< SPI, PNP, NF >	(C3.V3.F2.P1.B1.L2)
12	(C2.V3.F2.P1.B1.L0)	< PFOFF1 >	(C2.V3.F2.P4.B1.L0)
		< PFON1 >	(C2.V3.F2.P3.B1.L0)'
		< SPD, PNP, NF >	(C10.V3.F2.P1.B1.L1)
		< SPI, PNP, NF >	(C3.V3.F2.P1.B1.L2)

	x	σ	$\delta(x,\sigma)$
93	(C10.V4.F2.P1.B1.L1)	< PFOFF1 >	(C10.V4.F2.P4.B1.L1)
		< PFON1 >	(C10.V4.F2.P3.B1.L1)'
		< FOFF, PNP, NF >	(C1.V4.F1.P1.B1.L0)
		< SPI, PNP, NF >	(C3.V4.F2.P1.B1.L2)
13	(C2.V4.F2.P1.B1.L0)	< PFOFF1 >	(C2.V4.F2.P4.B1.L0)
		< PFON1 >	(C2.V4.F2.P3.B1.L0)'
		< SPD, PNP, NF >	(C10.V4.F2.P1.B1.L1)
		< SPI, PNP, NF >	(C3.V4.F2.P1.B1.L2)
4	(C1.V1.F1.P3.B1.L0)	< SC1 >	(C1.V4.F1.P3.B1.L0)
		< SO1 >	(C1.V3.F1.P3.B1.L0)
		< FON, PPP, NF >	(C2.V1.F2.P3.B1.L0)
6	(C1.V3.F1.P3.B1.L0)	< FON, PPP, F >	(C2.V3.F2.P3.B1.L0)
7	(C1.V4.F1.P3.B1.L0)	< FON, PPP, NF >	(C2.V4.F2.P3.B1.L0)
94	(C10.V1.F2.P3.B1.L1)	< SC1 >	(C10.V4.F2.P3.B1.L1)
		< SO1 >	(C10.V3.F2.P3.B1.L1)'
		< FOFF, PNP, NF >	(C1.V1.F1.P3.B1.L0)
		< SPI, PPP, NF >	(C3.V1.F2.P3.B1.L2)
14	(C2.V1.F2.P3.B1.L0)	< SC1 >	(C2.V4.F2.P3.B1.L0)
		< SO1 >	(C2.V3.F2.P3.B1.L0)'
		< SPD, PPP, NF >	(C10.V1.F2.P3.B1.L1)
		< SPI, PPP, NF >	(C3.V1.F2.P3.B1.L2)
96	(C10.V3.F2.P3.B1.L1)	< FOFF, PNP, NF >	(C1.V3.F1.P3.B1.L0)
		< SPI, PPP, NF >	(C3.V3.F2.P3.B1.L2)
16	(C2.V3.F2.P3.B1.L0)	< SPD, PPP, F >	(C10.V3.F2.P3.B1.L1)
		< SPI, PPP, F >	(C3.V3.F2.P3.B1.L2)
97	(C10.V4.F2.P3.B1.L1)	< FOFF, PNP, NF >	(C1.V4.F1.P3.B1.L0)
		< SPI, PPP, NF >	(C3.V4.F2.P3.B1.L2)
17	(C2.V4.F2.P3.B1.L0)	< SPD, PPP, NF >	(C10.V4.F2.P3.B1.L1)
		< SPI, PPP, NF >	(C3.V4.F2.P3.B1.L2)
5	(C1.V1.F1.P4.B1.L0)	< SC1 >	(C1.V4.F1.P4.B1.L0)
		< SO1 >	(C1.V3.F1.P4.B1.L0)
		< FON, PNP, NF >	(C2.V1.F2.P4.B1.L0)
8	(C1.V3.F1.P4.B1.L0)	< FON, PNP, NF >	(C2.V3.F2.P4.B1.LO)
9	(C1.V4.F1.P4.B1.L0)	< FON, PNP, NF >	(C2.V4.F2.P4.B1.L0)
95	(C10.V1.F2.P4.B1.L1)	< SC1 >	(C10.V4.F2.P4.B1.L1)
		< SO1 >	(C10.V3.F2.P4.B1.L1)
		< FOFF, PNP, NF >	(C1.V1.F1.P4.B1.L0)
		< SPI, PNP, NF >	(C3.V1.F2.P4.B1.L2)
15	(C2.V1.F2.P4.B1.L0)	< SC1 >	(C2.V4.F2.P4.B1.L0)
		< SO1 >	(C2.V3.F2.P4.B1.L0)
		< SPD, PNP, NF >	(C10.V1.F2.P4.B1.L1)
		< SPI, PNP, NF >	(C3.V1.F2.P4.B1.L2)

	x	σ	$\delta(x,\sigma)$
98	(C10.V3.F2.P4.B1.L1)	< FOFF, PNP, NF >	(C1.V3.F1.P4.B1.L0)
		< SPI, PNP, NF >	(C3.V3.F2.P4.B1.L2)
18	(C2.V3.F2.P4.B1.L0)	< SPD, PNP, NF >	(C10.V3.F2.P4.B1.L1)
		< SPI, PNP, NF >	(C3.V3.F2.P4.B1.L2)
99	(C10.V4.F2.P4.B1.L1)	< FOFF, PNP, NF >	(C1.V4.F1.P4.B1.L0)
		< SPI, PNP, NF >	(C3.V4.F2.P4.B1.L2)
19	(C2.V4.F2.P4.B1.L0)	< SPD, PNP, NF >	(C10.V4.F2.P4.B1.L1)
		< SPI, PNP, NF >	(C3.V4.F2.P4.B1.L2)
81	(C9.V1.F2.P1.B2.L1)	< BOFF, PNP, NF >	(C10.V1.F2.P1.B1.L1)
		< PFOFF1 >	(C9.V1.F2.P4.B2.L1)
		< PFON1 >	(C9.V1.F2.P3.B2.L1)'
		< SC1 >	(C9.V4.F2.P1.B2.L1)
		< SO1 >	(C9.V3.F2.P1.B2.L1)
82	(C9.V3.F2.P1.B2.L1)	< BOFF, PNP, NF >	(C10.V3.F2.P1.B1.L1)
		< PFOFF1 >	(C9.V3.F2.P4.B2.L1)
		< PFON1 >	(C9.V3.F2.P3.B2.L1)'
83	(C9.V4.F2.P1.B2.L1)	< BOFF, PNP, NF >	(C10.V4.F2.P1.B1.L1)
		< PFOFF1 >	(C9.V4.F2.P4.B2.L1)
		< PFON1 >	(C9.V4.F2.P3.B2.L1)'
71	(C8.V1.F2.P2.B2.L1)	< PFOFF2 >	(C8.V1.F2.P4.B2.L1)'
		< PFON2 >	(C8.V1.F2.P3.B2.L1)
		< SC1 >	(C8.V4.F2.P2.B2.L1)
		< SO1 >	(C8.V3.F2.P2.B2.L1)'
		< POFF, PNP, NF >	(C9.V1.F2.P1.B2.L1)
61	(C7.V2.F2.P2.B2.L1)	< CV, PPP, NF >	(C8.V1.F2.P2.B2.L1)
		< PFOFF2 >	(C7.V2.F2.P4.B2.L1)'
		< PFON2 >	(C7.V2.F2.P3.B2.L1)
		< SC2 >	(C7.V4.F2.P2.B2.L1)'
		< SO2 >	(C7.V3.F2.P2.B2.L1)
62	(C7.V3.F2.P2.B2.L1)	< CV, PPP, F >	(C8.V3.F2.P2.B2.L1)
		< PFOFF2 >	(C7.V3.F2.P4.B2.L1)'
		< PFON2 >	(C7.V3.F2.P3.B2.L1)
72	(C8.V3.F2.P2.B2.L1)	< PFOFF2 >	(C8.V3.F2.P4.B2.L1)'
		< PFON2 >	(C8.V3.F2.P3.B2.L1)
		< POFF, PNP, NF >	(C9.V3.F2.P1.B2.L1)
63	(C7.V4.F2.P2.B2.L1)	< CV, PPP, NF >	(C8.V4.F2.P2.B2.L1)
		< PFOFF2 >	(C7.V4.F2.P4.B2.L1)'
		< PFON2 >	(C7.V4.F2.P3.B2.L1)
73	(C8.V4.F2.P2.B2.L1)	< PFOFF2 >	(C8.V4.F2.P4.B2.L1)'
		< PFON2 >	(C8.V4.F2.P3.B2.L1)
		< POFF, PNP, NF >	(C9.V4.F2.P1.B2.L1)

	x	σ	$\delta(x,\sigma)$
74	(C8.V1.F2.P3.B2.L1)	< SC1 >	(C8.V4.F2.P3.B2.L1)
		< SO1 >	(C8.V3.F2.P3.B2.L1)'
		< POFF, PPP, NF >	(C9.V1.F2.P3.B2.L1)
84	(C9.V1.F2.P3.B2.L1)	< BOFF, PPP, NF >	(C10.V1.F2.P3.B1.L1)
		< SC1 >	(C9.V4.F2.P3.B2.L1)
		< SO1 >	(C9.V3.F2.P3.B2.L1)'
64	(C7.V2.F2.P3.B2.L1)	< CV, PPP, NF >	(C8.V1.F2.P3.B2.L1)
		< SC2 >	(C7.V4.F2.P3.B2.L1)'
		< SO2 >	(C7.V3.F2.P3.B2.L1)
66	(C7.V3.F2.P3.B2.L1)	< CV, PPP, F >	(C8.V3.F2.P3.B2.L1)
76	(C8.V3.F2.P3.B2.L1)	< POFF, PPP, F >	(C9.V3.F2.P3.B2.L1)
86	(C9.V3.F2.P3.B2.L1)	< BOFF, PPP, F >	(C10.V3.F2.P3.B1.L1)
67	(C7.V4.F2.P3.B2.L1)	< CV, PPP, NF >	(C8.V4.F2.P3.B2.L1)
77	(C8.V4.F2.P3.B2.L1)	< POFF, PPP, NF >	(C9.V4.F2.P3.B2.L1)
87	(C9.V4.F2.P3.B2.L1)	< BOFF, PPP, NF >	(C10.V4.F2.P3.B1.L1)
75	(C8.V1.F2.P4.B2.L1)	< SC1 >	(C8.V4.F2.P4.B2.L1)
		< SO1 >	(C8.V3.F2.P4.B2.L1)
		< POFF, PNP, NF >	(C9.V1.F2.P4.B2.L1)
85	(C9.V1.F2.P4.B2.L1)	< BOFF, PNP, NF >	(C10.V1.F2.P4.B1.L1)
		< SC1 >	(C9.V4.F2.P4.B2.L1)
		< SO1 >	(C9.V3.F2.P4.B2.L1)
65	(C7.V2.F2.P4.B2.L1)	< CV, PNP, NF >	(C8.V1.F2.P4.B2.L1)
		< SC2 >	(C7.V4.F2.P4.B2.L1)
		< SO2 >	(C7.V3.F2.P4.B2.L1)
68	(C7.V3.F2.P4.B2.L1)	< CV, PNP, NF >	(C8.V3.F2.P4.B2.L1)
78	(C8.V3.F2.P4.B2.L1)	< POFF, PNP, NF >	(C9.V3.F2.P4.B2.L1)
88	(C9.V3.F2.P4.B2.L1)	< BOFF, PNP, NF >	(C10.V3.F2.P4.B1.L1)
69	(C7.V4.F2.P4.B2.L1)	< CV, PNP, NF >	(C8.V4.F2.P4.B2.L1)
79	(C8.V4.F2.P4.B2.L1)	< POFF, PNP, NF >	(C9.V4.F2.P4.B2.L1)
89	(C9.V4.F2.P4.B2.L1)	< BOFF, PNP, NF >	(C10.V4.F2.P4.B1.L1)
21	(C3.V1.F2.P1.B1.L2)	< PFOFF1 >	(C3.V1.F2.P4.B1.L2)
		< PFON1 >	(C3.V1.F2.P3.B1.L2)'
		< SC1 >	(C3.V4.F2.P1.B1.L2)
		< SO1 >	(C3.V3.F2.P1.B1.L2)
		< OV, PNP, NF >	(C4.V2.F2.P1.B1.L2)
31	(C4.V2.F2.P1.B1.L2)	< PFOFF1 >	(C4.V2.F2.P4.B1.L2)
		< PFON1 >	(C4.V2.F2.P3.B1.L2)'
		< SC2 >	(C4.V4.F2.P1.B1.L2)
		< SO2 >	(C4.V3.F2.P1.B1.L2)
		< PON, PPP, F >	(C5.V2.F2.P2.B1.L2)
22	(C3.V3.F2.P1.B1.L2)	< PFOFF1 >	(C3.V3.F2.P4.B1.L2)
		< PFON1 >	(C3.V3.F2.P3.B1.L2)'
		< OV, PNP, NF >	(C4.V3.F2.P1.B1.L2)

	x	σ	$\delta(x,\sigma)$
32	(C4.V3.F2.P1.B1.L2)	< PFOFF1 >	(C4.V3.F2.P4.B1.L2)
		< PFON1 >	(C4.V3.F2.P3.B1.L2)'
		< PON, PPP, F >	(C5.V3.F2.P2.B1.L2)
23	(C3.V4.F2.P1.B1.L2)	< PFOFF1 >	(C3.V4.F2.P4.B1.L2)
		< PFON1 >	(C3.V4.F2.P3.B1.L2)'
		< OV, PNP, NF >	(C4.V4.F2.P1.B1.L2)
33	(C4.V4.F2.P1.B1.L2)	< PFOFF1 >	(C4.V4.F2.P4.B1.L2)
		< PFON1 >	(C4.V4.F2.P3.B1.L2)'
		< PON, PPP, NF >	(C5.V4.F2.P2.B1.L2)
41	(C5.V2.F2.P2.B1.L2)	< BON, PPP, F >	(C6.V2.F2.P2.B2.L2)
		< PFOFF2 >	(C5.V2.F2.P4.B1.L2)'
		< PFON2 >	(C5.V2.F2.P3.B1.L2)
		< SC2 >	(C5.V4.F2.P2.B1.L2)'
		< SO2 >	(C5.V3.F2.P2.B1.L2)
42	(C5.V3.F2.P2.B1.L2)	< BON, PPP, F >	(C6.V3.F2.P2.B2.L2)
		< PFOFF2 >	(C5.V3.F2.P4.B1.L2)'
		< PFON2 >	(C5.V3.F2.P3.B1.L2)
43	(C5.V4.F2.P2.B1.L2)	< BON, PPP, NF >	(C6.V4.F2.P2.B2.L2)
		< PFOFF2 >	(C5.V4.F2.P4.B1.L2)'
		< PFON2 >	(C5.V4.F2.P3.B1.L2)
24	(C3.V1.F2.P3.B1.L2)	< SC1 >	(C3.V4.F2.P3.B1.L2)
		< SO1 >	(C3.V3.F2.P3.B1.L2)'
		< OV, PPP, F >	(C4.V2.F2.P3.B1.L2)
34	(C4.V2.F2.P3.B1.L2)	< SC2 >	(C4.V4.F2.P3.B1.L2)'
		< SO2 >	(C4.V3.F2.P3.B1.L2)
		< PON, PPP, F >	(C5.V2.F2.P3.B1.L2)
44	(C5.V2.F2.P3.B1.L2)	< BON, PPP, F >	(C6.V2.F2.P3.B2.L2)
		< SC2 >	(C5.V4.F2.P3.B1.L2)'
		< SO2 >	(C5.V3.F2.P3.B1.L2)
26	(C3.V3.F2.P3.B1.L2)	< OV, PPP, F >	(C4.V3.F2.P3.B1.L2)
36	(C4.V3.F2.P3.B1.L2)	< PON, PPP, F >	(C5.V3.F2.P3.B1.L2)
46	(C5.V3.F2.P3.B1.L2)	< BON, PPP, F >	(C6.V3.F2.P3.B2.L2)
27	(C3.V4.F2.P3.B1.L2)	< OV, PPP, NF >	(C4.V4.F2.P3.B1.L2)
37	(C4.V4.F2.P3.B1.L2)	< PON, PPP, NF >	(C5.V4.F2.P3.B1.L2)
47	(C5.V4.F2.P3.B1.L2)	< BON, PPP, NF >	(C6.V4.F2.P3.B2.L2)
25	(C3.V1.F2.P4.B1.L2)	< SC1 >	(C3.V4.F2.P4.B1.L2)
		< SO1 >	(C3.V3.F2.P4.B1.L2)
		< OV, PNP, NF >	(C4.V2.F2.P4.B1.L2)

	x	σ	$\delta(x,\sigma)$
35	(C4.V2.F2.P4.B1.L2)	< SC2 >	(C4.V4.F2.P4.B1.L2)
		< SO2 >	(C4.V3.F2.P4.B1.L2)
		< PON, PNP, NF >	(C5.V2.F2.P4.B1.L2)
45	(C5.V2.F2.P4.B1.L2)	< BON, PNP, NF >	(C6.V2.F2.P4.B2.L2)
		< SC2 >	(C5.V4.F2.P4.B1.L2)
		< SO2 >	(C5.V3.F2.P4.B1.L2)
28	(C3.V3.F2.P4.B1.L2)	< OV, PNP, NF >	(C4.V3.F2.P4.B1.L2)
38	(C4.V3.F2.P4.B1.L2)	< PON, PNP, NF >	(C5.V3.F2.P4.B1.L2)
48	(C5.V3.F2.P4.B1.L2)	< BON, PNP, NF >	(C6.V3.F2.P4.B2.L2)
29	(C3.V4.F2.P4.B1.L2)	< OV, PNP, NF >	(C4.V4.F2.P4.B1.L2)
39	(C4.V4.F2.P4.B1.L2)	< PON, PNP, NF >	(C5.V4.F2.P4.B1.L2)
49	(C5.V4.F2.P4.B1.L2)	< BON, PNP, NF >	(C6.V4.F2.P4.B2.L2)
51	(C6.V2.F2.P2.B2.L2)	< PFOFF2 >	(C6.V2.F2.P4.B2.L2)'
		< PFON2 >	(C6.V2.F2.P3.B2.L2)
		< SC2 >	(C6.V4.F2.P2.B2.L2)'
		< SO2 >	(C6.V3.F2.P2.B2.L2)
		< SPD, PPP, F >	(C7.V2.F2.P2.B2.L1)
52	(C6.V3.F2.P2.B2.L2)	< PFOFF2 >	(C6.V3.F2.P4.B2.L2)'
		< PFON2 >	(C6.V3.F2.P3.B2.L2)
		< SPD, PPP, F >	(C7.V3.F2.P2.B2.L1)
53	(C6.V4.F2.P2.B2.L2)	< PFOFF2 >	(C6.V4.F2.P4.B2.L2)'
		< PFON2 >	(C6.V4.F2.P3.B2.L2)
		< SPD, PPP, NF >	(C7.V4.F2.P2.B2.L1)
54	(C6.V2.F2.P3.B2.L2)	< SC2 >	(C6.V4.F2.P3.B2.L2)'
		< SO2 >	(C6.V3.F2.P3.B2.L2)
		< SPD, PPP, F >	(C7.V2.F2.P3.B2.L1)
56	(C6.V3.F2.P3.B2.L2)	< SPD, PPP, F >	(C7.V3.F2.P3.B2.L1)
57	(C6.V4.F2.P3.B2.L2)	< SPD, PPP, NF >	(C7.V4.F2.P3.B2.L1)
55	(C6.V2.F2.P4.B2.L2)	< SC2 >	(C6.V4.F2.P4.B2.L2)
		< SO2 >	(C6.V3.F2.P4.B2.L2)
		< SPD, PNP, NF >	(C7.V2.F2.P4.B2.L1)
58	(C6.V3.F2.P4.B2.L2)	< SPD, PNP, NF >	(C7.V3.F2.P4.B2.L1)
59	(C6.V4.F2.P4.B2.L2)	< SPD, PNP, NF >	(C7.V4.F2.P4.B2.L1)
191	(C10.V1.F2.P3.B1.L1)'	< PNP -> PPP >	(C10.V1.F2.P3.B1.L1)
111	(C2.V1.F2.P3.B1.L0)'	< PNP -> PPP >	(C2.V1.F2.P3.B1.L0)
192	(C10.V3.F2.P3.B1.L1)'	< PNP -> PPP >	(C10.V3.F2.P3.B1.L1)
112	(C2.V3.F2.P3.B1.L0)'	< PNP -> PPP >	(C2.V3.F2.P3.B1.L0)
193	(C10.V4.F2.P3.B1.L1)'	< PNP -> PPP >	(C10.V4.F2.P3.B1.L1)
113	(C2.V4.F2.P3.B1.L0)'	< PNP -> PPP >	(C2.V4.F2.P3.B1.L0)

	x	σ	$\delta(x,\sigma)$
194	(C10.V3.F2.P3.B1.L1)'	< NF -> F >	(C10.V3.F2.P3.B1.L1)
114	(C2.V3.F2.P3.B1.L0)'	< NF -> F >	(C2.V3.F2.P3.B1.L0)
181	(C9.V1.F2.P3.B2.L1)'	< PNP -> PPP >	(C9.V1.F2.P3.B2.L1)
182	(C9.V3.F2.P3.B2.L1)'	< PNP -> PPP >	(C9.V3.F2.P3.B2.L1)
183	(C9.V4.F2.P3.B2.L1)'	< PNP -> PPP >	(C9.V4.F2.P3.B2.L1)
171	(C8.V1.F2.P4.B2.L1)'	< PPP -> PNP >	(C8.V1.F2.P4.B2.L1)
271	(C8.V3.F2.P2.B2.L1)'	< NF -> F >	(C8.V3.F2.P2.B2.L1)
161	(C7.V2.F2.P4.B2.L1)'	< PPP -> PNP >	(C7.V2.F2.P4.B2.L1)
261	(C7.V4.F2.P2.B2.L1)'	< F -> NF >	(C7.V4.F2.P2.B2.L1)
162	(C7.V3.F2.P4.B2.L1)'	< PPP -> PNP >	(C7.V3.F2.P4.B2.L1)
172	(C8.V3.F2.P4.B2.L1)'	< PPP -> PNP >	(C8.V3.F2.P4.B2.L1)
163	(C7.V4.F2.P4.B2.L1)'	< PPP -> PNP >	(C7.V4.F2.P4.B2.L1)
173	(C8.V4.F2.P4.B2.L1)'	< PPP -> PNP >	(C8.V4.F2.P4.B2.L1)
174	(C8.V3.F2.P3.B2.L1)'	< NF -> F >	(C8.V3.F2.P3.B2.L1)
184	(C9.V3.F2.P3.B2.L1)'	< NF -> F >	(C9.V3.F2.P3.B2.L1)
164	(C7.V4.F2.P3.B2.L1)'	< F -> NF >	(C7.V4.F2.P3.B2.L1)
121	(C3.V1.F2.P3.B1.L2)'	< PNP -> PPP >	(C3.V1.F2.P3.B1.L2)
131	(C4.V2.F2.P3.B1.L2)'	< PNP -> PPP >	(C4.V2.F2.P3.B1.L2)
122	(C3.V3.F2.P3.B1.L2)'	< PNP -> PPP >	(C3.V3.F2.P3.B1.L2)
132	(C4.V3.F2.P3.B1.L2)'	< PNP -> PPP >	(C4.V3.F2.P3.B1.L2)
123	(C3.V4.F2.P3.B1.L2)'	< PNP -> PPP >	(C3.V4.F2.P3.B1.L2)
133	(C4.V4.F2.P3.B1.L2)'	< PNP -> PPP >	(C4.V4.F2.P3.B1.L2)
141	(C5.V2.F2.P4.B1.L2)'	< PPP -> PNP >	(C5.V2.F2.P4.B1.L2)
241	(C5.V4.F2.P2.B1.L2)'	< F -> NF >	(C5.V4.F2.P2.B1.L2)
142	(C5.V3.F2.P4.B1.L2)'	< PNP -> PPP >	(C5.V3.F2.P4.B1.L2)
143	(C5.V4.F2.P4.B1.L2)'	< PNP -> PPP >	(C5.V4.F2.P4.B1.L2)
124	(C3.V3.F2.P3.B1.L2)'	< NF -> F >	(C3.V3.F2.P3.B1.L2)
134	(C4.V4.F2.P3.B1.L2)'	< F -> NF >	(C4.V4.F2.P3.B1.L2)
144	(C5.V4.F2.P3.B1.L2)'	< F -> NF >	(C5.V4.F2.P3.B1.L2)
151	(C6.V2.F2.P4.B2.L2)'	< PPP -> PNP >	(C6.V2.F2.P4.B2.L2)
251	(C6.V4.F2.P2.B2.L2)'	< F -> NF >	(C6.V4.F2.P2.B2.L2)
152	(C6.V3.F2.P4.B2.L2)'	< PPP -> PNP >	(C6.V3.F2.P4.B2.L2)
153	(C6.V4.F2.P4.B2.L2)'	< PPP -> PNP >	(C6.V4.F2.P4.B2.L2)
154	(C6.V4.F2.P3.B2.L2)'	< F -> NF >	(C6.V4.F2.P3.B2.L2)

APPENDIX C

Transition Table for the Pump-Valve System of Chapter 6

	x	σ	$\delta(x,\sigma)$
1	VC,POFF,C1	< Stuck_Closed >	SC,POFF,C1
		< Load, NF >	VC,POFF,C2
		< No_Load, NF >	VC,POFF,C6
2	SC,POFF,C1	< Load, NF >	SC,POFF, C2
		< No_Load, NF >	SC,POFF,C6
3	VC,POFF,C2	< Stuck_Closed >	SC,POFF,C2
		< Open_Valve, NF >	VO,POFF,C3
		< Start_Pump, NF >	VC,PON,C4
4	VC,POFF,C6	< Stuck_Closed >	SC,POFF,C6
		< Close_Valve, NF >	VC,POFF,C7
		< Stop_Pump, NF >	VC,POFF,C8
5	SC,POFF,C2	< Open_Valve, NF >	SC,POFF,C3
		< Start_Pump, NF >	SC,PON,C4
6	SC,POFF,C6	< Close_Valve, NF >	SC,POFF,C7
		< Stop_Pump, NF >	SC,POFF,C8
7	VO,POFF,C3	< Start_Pump, F >	VO,PON,C5
		< Stuck_Open >	SO,POFF,C3
8	VC,PON,C4	< Stuck_Closed >	SC,PON,C4
		< Open_Valve, F >	VO,PON,C5
9	VC,POFF,C7	< Stuck_Closed >	SC,POFF,C7
		< Stop_Pump, NF >	VC,POFF,C9
10	VC,POFF,C8	< Stuck_Closed >	SC,POFF,C8
		< Close_Valve, NF >	VC,POFF,C9
11	SC,POFF,C3	< Start_Pump, NF >	SC,PON,C5
12	SC,PON,C4	< Open_Valve, NF >	SC,PON,C5
13	SC,POFF,C7	< Stop_Pump, NF >	SC,POFF,C9
14	SC,POFF,C8	< Close_Valve, NF >	SC,POFF,C9
15	VO,PON,C5	< No_Load, F >	VO,PON,C6
		< Stuck_Open >	SO,PON,C5

	x	σ	$\delta(x,\sigma)$
16	SO,POFF,C3	< Start_Pump, F >	SO,PON,C5
17	VC,POFF,C9	< Stuck_Closed >	SC,POFF,C9
		< Load, NF >	VC,POFF,C2
18	SC,PON,C5	< No_Load, NF >	SC,PON,C6
19	SC,POFF,C9	< Load, NF >	SC,POFF,C2
20	VO,PON,C6	< Close_Valve, NF >	VC,PON,C7
		< Stop_Pump, NF >	VO,POFF,C8
		< Stuck_Open >	SO,PON,C6
21	SO,PON,C5	< No_Load, F >	SO,PON,C6
22	SC,PON,C6	< Close_Valve, NF >	SC,PON,C7
		< Stop_Pump, NF >	SC,POFF,C8
23	SO,PON,C6	< Close_Valve, F >	SO,PON,C7
		< Stop_Pump, NF >	SO,POFF,C8
24	VC,PON,C7	< Stuck_Closed >	SC,PON,C7
		< Stop_Pump, NF >	VC,POFF,C9
25	VO,POFF,C8	< Close_Valve, NF >	VC,POFF,C9
		< Stuck_Open >	SO,POFF,C8
26	SC,PON,C7	< Stop_Pump, NF >	SC,POFF,C9
27	SO,PON,C7	< Stop_Pump, NF >	SO,POFF,C9
28	SO,POFF,C8	< Close_Valve, NF >	SO,POFF,C9
29	SO,POFF,C9	< Load, NF >	SO,POFF,C2
30	SO,POFF,C2	< Open_Valve, NF >	SO,POFF,C3
		< Start_Pump, F >	SO,PON,C4
31	SO,PON,C4	< Open_Valve, F >	SO,PON,C5

BIBLIOGRAPHY

[1] *The 1992 ASHRAE Handbook-Heating, Ventilating, and Air Conditioning Systems and Equipment*, American Society of Heating, Refrigerating, and AirConditioning Engineers, inch-pound edition, 1992.

[2] M. Basseville and I. Nikiforov, *Detection of Abrupt Changes - Theory and Application*, Prentice Hall, Englewood Cliffs, NJ, 1993.

[3] S. Bavishi and E. Chong, "Automated fault diagnosis using a discrete event systems framework," in *Proc. 9th IEEE International Symposium on Intelligent Control*, pp. 213–218, 1994.

[4] K. Bousson, L. Zimmer, and L. Travé-Massuyès, "Causal model-based diagnosis of dynamic systems," in *Working papers, DX-94, Fifth International Workshop on Principles of Diagnosis*, pp. 34–41, New Paltz, New York, October 1994.

[5] P. Caines, R. Greiner, and S. Wang, "Classical and logic based dynamic observers for finite automata," *IMA J. Math. Control Inform.*, vol. 8, pp. 45–80, 1991.

[6] C. Cassandras, S. Lafortune, and G. Olsder, "Introduction to the modelling, control and optimization of discrete event systems," in *Trends in Control. A European Perspective*, A. Isidori, editor, pp. 217–291, Springer–Verlag, September 1995.

[7] S. L. Chung, S. Lafortune, and F. Lin, "Addendum to "Limited lookahead policies in supervisory control of discrete event systems": Proofs of technical results," Technical Report CGR-92-6, College of Engineering Control Group Reports, University of Michigan, April 1992.

[8] R. Cieslak, C. Desclaux, A. Fawaz, and P. Varaiya, "Supervisory control of discrete-event processes with partial observations," *IEEE Trans. Automatic Control*, vol. 33, no. 3, pp. 249–260, March 1988.

[9] M. Cordier and S. Thiébaux, "Event-based diagnosis for evolutive systems," in *Working papers, DX-94, Fifth International Workshop on Principles of Diagnosis*, pp. 64–69, New Paltz, New York, October 1994.

[10] 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, editors, pp. 229–234, Morgan Kaufmann, 1992.

[11] R. Davis and W. Hamscher, "Model based reasoning: Troubleshooting," in *Readings in Model Based Diagnosis*, W. Hamscher, L. Console, and J. Kleer, editors, pp. 3–24, Morgan Kaufmann, 1992.

[12] D.Dvorak and B. Kuipers, "Model based monitoring of dynamic systems," in *Readings in Model Based Diagnosis*, W. Hamscher, L. Console, and J. Kleer, editors, pp. 249–254, Morgan Kaufmann, 1992.

[13] D. Decoste, "Dynamic across-time measurement interpretation," in *Readings in Model Based Diagnosis*, W. H. L. Console and J. Kleer, editors, pp. 255–261, Morgan Kaufmann, 1992.

[14] O. Dressler, "Model-based diagnosis on board: Magellan-MT inside," in *Working papers, DX-94, Fifth International Workshop on Principles of Diagnosis*, pp. 87–92, New Paltz, New York, October 1994.

[15] 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.

[16] C. Golaszewski and P. Ramadge, "Control of discrete event processes with forced events," in *Proc. 26th IEEE Conf. on Decision and Control*, pp. 247–251, Los Angeles, December 1987.

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

[18] D. Handelman and R. Stengel, "Combining expert systems and analytical redundancy concepts for fault tolerant flight control," *Journal of Guidance*, vol. 12, no. 1, pp. 39–45, 1989.

[19] L. Holloway and S. Chand, "Time templates for discrete event fault monitoring in manufacturing systems," in *Proc. 1994 American Control Conference*, pp. 701–706, 1994.

[20] J. E. Hopcroft and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, Reading, MA, 1979.

[21] S. Lafortune and E. Chen, "The infimal closed controllable superlanguage and its application in supervisory control," *IEEE Trans. Automatic Control*, vol. 35, no. 4, pp. 398–405, April 1990.

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

[23] S. Lapp and G. Powers, "Computer aided synthesis of fault trees," *IEEE Trans. Reliability Engineering*, vol. 26, no. 1, pp. 2–13, April 1977.

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

[25] F. Lin, "Diagnosability of discrete event systems and its applications," *Discrete Event Dynamic Systems: Theory and Applications*, vol. 4, no. 2, pp. 197–212, May 1994.

[26] F. Lin, J. Markee, and B. Rado, "Design and test of mixed signal circuits: a discrete event approach," in *Proc. 32th IEEE Conf. on Decision and Control*, pp. 246–251, December 1993.

[27] F. Lin and H. Mortazavian, "A normality theorem for decentralized control of discrete event systems," *IEEE Trans. Automatic Control*, vol. 39, no. 5, pp. 1089–1093, May 1994.

[28] F. Lin and W. M. Wonham, "On observability of discrete-event systems," *Information Sciences*, vol. 44, pp. 173–198, 1988.

[29] 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, editors, pp. 242–248, Morgan Kaufmann, 1992.

[30] O. O. Oyeleye, F. Finch, and M. Kramer, "Qualitative modeling and fault diagnosis of dynamic processes by MIDAS," Presented at the *AIChE Spring Annual Meeting*, Houston, Texas, 1989.

[31] C. M. Özveren and A. S. Willsky, "Observability of discrete event dynamic systems," *IEEE Trans. Automatic Control*, vol. 35, no. 7, pp. 797–806, July 1990.

[32] C. M. Özveren and A. S. Willsky, "Invertibility of discrete event dynamic systems," *Math. Control Signals Systems*, vol. 5, pp. 365–390, 1992.

[33] J. Y. Pan, "Qualitative reasoning with deep-level mechanism models for diagnoses of mechanism failures," in *Proc. IEEE Conf. on AI Applications*, pp. 295–301, Denver, Colarado, 1984.

[34] Y. Park and E. Chong, "On the eventual invertibility of discrete event systems and its applications," in *Proc. 32th IEEE Conf. on Decision and Control*, pp. 680–685, December 1993.

[35] C. Perrow, *Normal Accidents:Living with High Risk Technologies*, Basic Books, Inc., New York, 84.

[36] A. Poucet, S. Contini, K. E. Petersen, and N. K. Vestergaard, "An expert system approach to systems safety and reliability analysis," in *Fault Detection and Reliability: Knowledge Based & Other Approaches*, M. G. Singh, K. S. Hindi, G. Schmidt, and S. Tzafestas, editors, Pergamon Press, 1987.

[37] A. D. Pouliezos and G. S. Stavrakakis, *Real time fault monitoring of industrial processes*, Kluwer Academic Publishers, 1994.

[38] P. J. Ramadge, "Observability of discrete event systems," in *Proc. 25th IEEE Conf. on Decision and Control*, pp. 1108–1112, Athens, Greece, December 1986.

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

[40] S. Rich and V. Venkatasubramanian, "Model-based reasoning in diagnostic expert systems for chemical process plants," *Computers and Chemical Engineering*, vol. 11, no. 2, pp. 111–122, 1987.

[41] M. Sampath, "Discrete event systems based diagnostics for a variable air volume terminal box application," Project report, Advanced Development Team, Johnson Controls, Inc., September 1995.

[42] M. Sampath, S. Lafortune, and D. Teneketzis, "Active diagnosis of discrete event systems," Technical Report CGR-95-3, College of Engineering Control Group Reports, University of Michigan, 1995.

[43] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis, "Diagnosability of discrete event systems," Technical Report CGR-94-2, College of Engineering Control Group Reports, University of Michigan, March 1994. In *IEEE Transactions on Automatic Control*, vol. 40, no. 9, pp. 1555-1575, September 1995.

[44] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis, "Failure diagnosis using discrete event models," Technical Report CGR-94-3, College of Engineering Control Group Reports, University of Michigan, May 1994. To appear in *IEEE Transactions on Control Systems Technology* in March 96.

[45] W. T. Scherer and C. C. White, "A survey of expert systems for equipment maintenance and diagnostics," in *Fault Detection and Reliability: Knowledge Based & Other Approaches*, M. G. Singh, K. S. Hindi, G. Schmidt, and S. Tzafestas, editors, Pergamon Press, 1987.

[46] K. Sinnamohideen, "Discrete event based diagnostic supervisory control system," Presented at the *AIChE Annual Meeting*, Los Angeles, CA, November 1991.

[47] K. Sinnamohideen, "Application of discrete event systems in the diagnosis of HVAC systems," Internal Report, Johnson Controls, Inc., 1992.

[48] W. F. Stoecker, editor, *Design of Thermal Systems*, McGraw-Hill, New York, third edition, 1989.

[49] P. Struss, "Testing for discrimination of diagnosis," in *Working papers, DX-94, Fifth International Workshop on Principles of Diagnosis*, pp. 312–320, New Paltz, New York, October 1994.

[50] S. Subramanian and R. J. Mooney, "Multiple-fault diagnosis using general qualitative models with fault modes," in *Working papers, DX-94, Fifth International Workshop on Principles of Diagnosis*, pp. 321–325, New Paltz, New York, October 1994.

[51] L. Telksnys, editor, *Detection of Changes in Random Processes*, Optimization Software Inc. Publications Division, New York, 1986.

[52] J. G. Thistle, "Control of infinite behaviour of discrete- event systems," Systems Control Group Report 9012, University of Toronto, January 1991. PhD Thesis.

[53] J. G. Thistle, "Logical aspects of control of discrete-event systems: A survey of tools and techniques," in *Lecture Notes in Control and Information Sciences. 11th Int'l Conf. on Analysis and Optimization of Systems, "Discrete Event Systems", École des Mines de Paris, France, June 15-17, 1994.*, G. Cohen and J.-P. Quadrat, editors, volume 199, pp. 3–15. Springer–Verlag, 1994.

[54] N. Ulerich and G. Powers, "On-line hazard aversion and fault diagnosis in chemical processes: The digraph + fault-tree method," *IEEE Trans. Reliability Engineering*, vol. 37, no. 2, pp. 171–177, June 1988.

[55] N. Viswanadham, "Control systems : Reliability," in *Systems and Control Encyclopedia: Theory, Technology, Applications*, M. G. Singh, editor, Pergamon Press, 1987.

[56] N. Viswanadham and T. L. Johnson, "Fault detection and diagnosis of automated manufacturing systems," in *Proc. 27th IEEE Conf. on Decision and Control*, pp. 2301–2306, Austin, Texas, 1988.

[57] R. D. Vries, "An automated methodology for generating a fault tree," *IEEE Trans. Reliability Engineering*, 1990.

[58] A. S. Willsky, "A survey of design methods for failure detection in dynamic systems," *Automatica*, vol. 12, pp. 601–611, 1976.