Fault Diagnosis in Hierarchical Discrete-Event Systems

Abdolrasul Mohammadi Idghamishi

A Thesis
in
the Department
of
Electrical and Computer Engineering

Presented in Partial Fulfilment of the Requirements
for the Degree of Master of Applied Science at
Concordia University
Montreal, Quebec, Canada

January 2004

Your file Votre référence
ISBN: 0-612-91087-3
Our file Notre référence
ISBN: 0-612-91087-3

ABSTRACT

Fault Diagnosis in Hierarchical Discrete-Event Systems

Abdolrasul Mohammadi Idghamishi

A framework for on-line passive fault diagnosis in hierarchical discrete-event systems is proposed. In this approach, the system model is broken to simpler substructures called D-holons. A state based diagnoser is constructed for each D-holon. Fault diagnosis is accomplished using the state estimates provided by the D-holon diagnosers. The diagnosers may communicate among each other in order to update their state estimates. At any given time, only a subset of the diagnosers are active, and as a result, instead of the entire model of the system, only the models of the D-holons associated with the active diagnosers are used. Therefore, only part of the system model needs to be stored in computer Random Access Memory (RAM). This reduces RAM requirements and thus, could be useful in complex multi-phase systems.

The concept of D-holon provides a suitable tool to study failure diagnosability in cases where components are active in certain phases of operation and inactive in other phases. This resulted in the introduction of the concept of phase-diagnosability. A set of necessary and sufficient conditions for phase-diagnosability is obtained.

Furthermore, in order to reduce the computational complexity of the diagnosis process, a set of sufficient conditions is provided under which the diagnosis process

becomes semi-modular. It is shown that the computational complexity of constructing (time) and storing (space) the transition systems required for diagnosis in the proposed semi-modular approach is polynomial in the number of system components, whereas in the original monolithic approach the computational complexity is exponential.

ACKNOWLEDGEMENT

I am deeply indebted to my supervisor, Dr. Shahin Hashtrudi Zad, for his constant support and guidance throughout the work with this thesis. Without his help, this work would not be possible.

I would like to thank the members of my thesis committee, Dr. Peyman Gohari, Dr. Amir Aghdam and Dr. Rama Bhat , for their very helpful comments and suggestions.

Many thanks go to my friends in the Control and Robotic Group, in particular, Mohammad Moosaei and Javad Mohammadpur, for useful discussions.

Lastly, I would like to thank my family, for all of their never-ending support they have provided throughout my life.

I dedicate this thesis to my adorable wife, Tanaz, for all her love, support, patience, and encouragement.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Fault detection and isolation systems are very important in maintaining the performance and enhancing the reliability of complex and sophisticated systems. Particularly, in situations where human attendance for system modification and maintenance is difficult or impossible, fault diagnosis becomes vital. Using systematic methods for designing diagnosis systems not only increases the accuracy and reliability of diagnosis but also reduces the future costs of system revisions and maintenance. In addition, human errors are less likely in systematic diagnosis code generation than in manual code generation. Fault diagnosis systems play a very important role in aerospace, manufacturing and process industries. As a result, a large body of work has been done on fault diagnosis (see, e.g., [HCK92], [Ise97] and [Lev95]).

1.1 Background

In this thesis, we say a **failure (fault)** [1] has occurred if the system behaviour deviates from its normal operation for a bounded or unbounded period of time under certain operating conditions [Ise97]. Stuck-closed and stuck-open of valves are examples of failures. A failure can be **permanent** or **non-permanent**. After the occurrence of a permanent failure, the system stays in the faulty mode permanently. On the other hand, after the occurrence of a non-permanent failure, the system may recover to the normal condition. A broken valve can be an example of a permanent failure and a loose connection in an electric circuit may be the source of a non-permanent failure.

In this thesis, we study fault diagnosis in systems that, for diagnostic purposes, can be modelled as hierarchical discrete event systems (HDES). We assume that the system under supervision is operational and the fault detection system only uses the observable events generated in the system and no test inputs are used for diagnosing failures. Thus, we only concentrate on **on-line passive** diagnosis.

In the following, we review some of the available techniques for fault diagnosis.

Hardware redundancy is one of the most commonly-used methods for fault diagnosis and tolerance. In this approach, multiple sensors are used for measuring each plant variable. Then, a voter compares their outputs and determines the final value. If one of the sensors fails, the failure can be detected by comparing its value with other sensor values. This approach is also employed in diagnosing software code errors in the form of N-version programming. In N-version programming, multiple codes are provided for a critical part of the system. Usually, these codes are written in different programming

[1] In this thesis, "failure" and "fault" mean the same and are used interchangeably.

languages by different programmers to avoid language, compiler and human related errors. Although, these techniques are simple and fairly reliable, they impose an overhead on the system, resulting in increase of implementation cost. Moreover, they are only suitable for detecting sensor failures and programming errors. In addition, they are not suitable for detecting common-cause failures.

Expert systems are also used for diagnosing failures. Expert systems are designed based on the experience and knowledge of experts (stored as a set of rules) and use an inference engine to diagnose failures. These systems are advantageous in cases that obtaining a model for the plant is difficult. However, gathering the required expertise and information for building an expert system can be a hard and time consuming task. In addition, it may not be possible to evaluate the completeness of the expert data base.

Hardware and software redundancy and expert systems are examples of qualitative techniques which perform diagnosis without utilizing a model of the plant and are therefore, known as **model-free** methods. In addition to model-free methods, several model-based techniques for fault diagnosis have been proposed in the literature. In a model-based approach, the observed behaviour of the plant is compared against the plant model, and the condition of the system (normal/faulty) is inferred from this comparison. In the following, we discuss some of the model-based diagnosis techniques that use discrete event models.

In [Lin94], F. Lin proposes a **state-based approach** for diagnosis failures in DES. In state-based approaches, it is assumed that the state set of the system can be partitioned according to the condition (failure status) of the system. The goal of the diagnosis process is to determine the current state of the system (or at least the block of the normal/faults

partition the current state belongs to) using the available observations (sensor measurements) and then to determine the current condition of the system. [Lin94] addresses the problems of off-line and on-line active diagnosis. An algorithm is presented for computing a sequence of test commands for diagnosing system failures. If the algorithm converges the system will be on-line diagnosable.

In [SSL95] and [SSL96], M. Sampath *et al.* present a systematic approach for passive on-line fault diagnosis in finite state automata. In passive diagnosis, the diagnoser does not generate any test inputs and relies on observations only. In [SSL95], an extended observer for the system is used (**diagnoser**) to perform diagnosis. The issue of **diagnosability** is also addressed. The approach in [SSL95] is **event-based.** In an event-based method, inference is made about the occurrence of failure events based on the observed events. It is assumed that a failure is the result of an (unobservable) failure event. In [CP97], this framework has been extended to utilize information about the timing of events. In [SLT98], active diagnosis of DES has been studied in the framework of [SSL95].

In [LY96], algorithms for testing of finite state machines are reviewed. Although testing algorithms are related to the problem of fault diagnosis, the framework used in [LY96] is different: the finite state machines are usually assumed to be deterministic with a fixed condition (failure status); also it is assumed that transitions can always be observed even if they do not result in a change in output. These assumptions often do not hold in fault diagnosis of control systems.

In [HKW03], S. Hashtrudi Zad *et al.* study fault diagnosis in DES using a state-based approach. They propose a passive on-line method for diagnosing failures in discrete event

systems and construct a fault detection system (diagnoser). In this framework, an output (sensor's signal) is associated with each state of the system. Estimates for the current state (or possible states) of the system is made based on the generated output sequence. The issue of diagnosability is also studied. In this framework, the condition of the system does not have to be known at the time that the diagnoser is started. Assuming a failure is diagnosable if it occurs before the diagnoser initialization, the proposed diagnoser can eventually detect and isolate the failure. A model reduction method has also been introduced in [HKW03] to reduce the number of states of the diagnoser and the computational complexity of diagnoser design. This framework has been extended to incorporate timing information in order to improve the accuracy of diagnosis [HKW99]. Moreover, fault diagnosis in hybrid systems is addressed in [HKW00].

In most of the approaches proposed for fault diagnosis in DES, the structure of the system is assumed to be flat ([HKW03], [HKW99], [Lin94], [SSL95] and [SSL96]). In these techniques, no particular structure is assumed for the system and the entire model is used at all times. This usually results in complex solutions for the diagnosis problem. The design of these diagnosis systems (or the transition system required for diagnosis) is computationally complex and the resulting solutions (usually in the form of transition systems) are very large and require large amounts of computer memory for on-line implementation.

One way of reducing complexity is to take advantage of the system structure. Many complex systems possess a hierarchical structure. Hierarchy not only can be seen in physical systems but also can be observed in many aspects of our life. For example, in social systems, a human is a unit that belongs to a family. Family is a subset of a

community and the community itself is a part of a country. In many biological and physical systems, we can see a hierarchy too. In this thesis, we study fault diagnosis in an enhanced class of DES called hierarchical DES to overcome the aforementioned shortcoming.

In [Har87], D. Harel introduces a visual tool for modeling complex discrete-event systems called **statecharts**. Statecharts are useful for representing complicated systems. They extend the conventional state-transition diagrams of DES by adding many useful features such as hierarchy, concurrency, and communication.

The hierarchical model that we use in this thesis for describing the system is rooted in statecharts. Since statecharts have more features than we need, we use a simplified version of statecharts called the **Hierarchical Finite State Machine (HFSM)**. HFSMs extend finite state machines (FSM) only by adding hierarchy and concurrency features. HFSMs have been used in [BH93] and [Wan95] for the supervisory control problem.

In this thesis, we propose a state-based approach for fault diagnosis in HFSMs. We use the hierarchical structure to split the system into smaller subsystems called **D-holons**. At any point in time, only the model of the D-holons needed in diagnosis are used and thus, loaded into computer memory. Our approach reduces the memory requirements for computer implementation of the diagnosis system. Moreover, because of the hierarchical structure used in design, the diagnosis system is more organized and understandable.

Modular diagnosis in DES has also been the subject of research. In [DM02], R. Debouk *et al.* propose a method in which an individual diagnoser is designed for each component of the modular system assuming the components do not have common failures. These diagnosers work concurrently when the system is executing its events.

Once a diagnoser detects a failure, a coordinator in the system becomes notified and consequently, the failure is diagnosed. They also obtain a sufficient condition under which the diagnosability of each component failure can be tested using the component model only. No assumption is made on the interaction among the components. As a result, the state estimate provided by the modular diagnosis becomes conservative (larger than the estimate provided by the monolithic diagnoser). Therefore, a diagnosable failure may become undiagnosable using their modular diagnoser. Moreover, in cases where a failure is diagnosable using the modular diagnosis, the modular diagnoser may take longer to detect and isolate the failure.

In this thesis, we propose a semi-modular approach for cases in which the system components interact, but the interactions (shared events) are observable. The advantage of our approach is that at each time a state estimate identical to the one provided by the monolithic approach can be constructed for the (entire) system. Moreover, our semi-modular approach can be easily used in our hierarchical framework.

In the following section, we discuss the outline of the thesis and review the thesis contributions in more detail.

1.2 Thesis Outline

In chapter 2, we review the background material for the framework developed in this thesis. First, we briefly review discrete-event models and then, we discuss statecharts and hierarchical finite state machines. HFSM is a powerful tool for equipping discrete event models with a top-down hierarchy. Chapter 3 presents a state-based fault diagnosis approach adapted from [HKW03]. In this thesis, we extend this approach to HFSMs. In

chapter 4, first, we will introduce the concept of D-holon. Intuitively, a D-holon is a mathematical tool for describing a phase of operation in a system. D-holons are similar to some extent in structure to holons in [Wan95]. We have, however, made some modifications to make them more suitable for diagnostic purposes. We introduce a diagnosis method which uses the hierarchical structure of the system. Here, instead of constructing a single, large diagnoser for the entire system, we design smaller diagnosers, one for each D-holon, and refer to them as **D-holon diagnosers**. A D-holon diagnoser is an observer that tracks possible system transitions in the D-holon. The diagnosers may communicate among each other in order to update their state estimates. At any given time, only a subset of the diagnosers are active and thus, instead of the entire model of the system, only the models of the D-holons associated with the active diagnosers are used. As a result, only part of the system model needs to be stored in computer Random Access Memory (RAM). This reduces RAM requirements and thus, could be useful in complex multi-phase systems. We will show that diagnosers can become decoupled if certain specific conditions on the structure of the system are satisfied. In systems with decoupled D-holon diagnosers, diagnosers can be modeled as FSMs.

The concept of D-holon provides a suitable tool to study failure diagnosability in cases where components are active in certain phases of operation and inactive in other phases. This resulted in the introduction of the concept of phase-diagnosability. A set of sufficient conditions for local-diagnosability is obtained.

In chapter 5, we propose a semi-modular approach in which the process of updating state estimates in a diagnoser can be accomplished using the model of individual components. Instead of the combined model for the system, we show that this can be

done if the interactions among components are observable. We call this method semi-modular because we do not design separate diagnosers for each module even though, the update process is based on individual component models. We also show how the semi-modular diagnosis approach can be used in the construction of D-holon diagnosers for fault diagnosis in hierarchical systems.

The main advantages of using a semi modular approach are the reduction in the complexity of diagnoser design and reduction in the size of resulting transition systems required for diagnosis. The issue of computational complexity is also investigated in chapter 5. We show that the computational complexity of constructing and storing of the transition systems required for diagnosis in the proposed semi-modular hierarchical approach is polynomial in the number of system components, whereas in the original monolithic approach, the computational complexity is exponential in the number of system components.

As an illustrative example, we apply our framework to an ozone generation plant. We show that the plant can be modeled as a HFSM, and then design a modular hierarchical diagnosis system for the plant.

Chapter 2

Background Overview

2.1 Discrete Event Systems (DES)

A "Discrete Event System" is a system with discrete state space and is asynchronously event driven. Automata theory provided one of the most comprehensive set of mathematical tools for studying DES. Many of the other models such as Petri nets for describing DES are rooted in automaton theory. In automaton models, system evolution is represented by transitions from one state to another state. The reader is referred to [RW82], [RW89] and [Won93] for details. In the following, we review some basic definitions and operations in automata theory.

2.1.1 Languages and Generators

An **alphabet** Σ is a finite set of symbols. Symbols correspond to events in DES models. A symbol sequence over Σ has the form $\sigma_1\sigma_2 \ldots \sigma_n$ for $n \geq 1$, where $\sigma_i \in \Sigma$ with $1 \leq i \leq n$. Σ^+ denotes the set of all possible finite symbol sequences over Σ. $\Sigma^* := \{\varepsilon\} \cup \Sigma^+$ represents the set of all strings or words over Σ. Here, ε denotes the empty sequence (sequence with no symbols).

Definition 2.1: A **language** over alphabet Σ is any subset of Σ^*. □

The empty language is shown by ϕ.

A generator is a simple model for representing a DES.

Definition 2.2: A (deterministic) **generator** G is a 4-tuple:

$$G = (X, \Sigma, \delta, x_0)$$

in which, X is the non-empty state set. $x_0 \in X$ is the initial state. $\delta : X \times \Sigma \to X$ is the partial transition function. □

Note that, in general, at each state $x \in X$, the transition function, δ, may be defined only for a subset of the elements of Σ.

Suppose $s \in \Sigma^*$ is a sequence of events. The transition function δ can be extended to sequences of events:

$$\delta : X \times \Sigma^* \to X$$

$$\delta(x, \varepsilon) = x$$

$$\delta(x, s\sigma) = \delta(\delta(x, s), \sigma) \text{ if } \delta(x, s) \text{ and } \delta(\delta(x, s), \sigma) \text{ are define.}$$

for all $s \in \Sigma^*$, $x \in X$ and $\sigma \in \Sigma$.

A state $x \in X$ is reachable in G if there exists $s \in \Sigma^*$ such that $x = \delta(x_0, s)$. The reachable **sub-generator** of a generator G, $Rch(G)$, is a generator $Rch(G) = G_{rch} = (X_{rch}, \Sigma, \delta_{rch}, x_0)$, where X_{rch} denotes the set of all reachable states that are reachable from x_0 and δ_{rch}, the transition function, is the restriction of δ to $X_{rch} \times \Sigma$.

Generator G defined above is said to be **deterministic** because for any $x \in X$ and $\sigma \in \Sigma$, if transition σ is defined at x, then the target state $\delta(x,\sigma)$ is uniquely defined. On the other hand, G is called **nondeterministic** if it is not deterministic. In a nondeterministic generator, the co domain of δ is the power set of X; in other words $\delta : X \times \Sigma \to 2^X$.

We use the notation $\delta(x,s)!$ to indicate that $\delta(x,s)$ is defined.

Definition 2.3: A generator $G = (X, \Sigma, \delta, x_0)$ is called an **automaton** if the transition function is a total function, i.e., $\delta(x,\sigma)!$ for any $x \in X$ and $\sigma \in \Sigma$. □

In this thesis, we refer to discrete event systems which are modeled with automata (or generators) as **flat DES,** emphasizing the fact that the system model does not have a hierarchical structure.

2.1.2 Operations on Generators

In this section, we explain some important operations on generators.

Let $G_1 = (X_1, \Sigma_1, \delta_1, x_{0,1})$ and $G_2 = (X_2, \Sigma_2, \delta_2, x_{0,2})$ be two generators. The **synchronous product** of G_1 and G_2, ***sync***(G_1, G_2), is a generator in which the shared events of two generators are synchronized. Specifically,

$$sync(G_1, G_2) = (X, \Sigma, \delta, x_0)$$

where

$$X = X_1 \times X_2$$

$$\Sigma = \Sigma_1 \cup \Sigma_2$$

$$x_0 = (x_{0,1}, x_{0,2})$$

$$\delta = X \times \Sigma \to X$$

with

$$\delta((x_1, x_2), \sigma) = \begin{cases} (\delta_1(x_1, \sigma), \delta_2(x_2, \sigma)) & \text{if } \sigma \in \Sigma_1 \cap \Sigma_2 \text{ and } \delta_1(x_1, \sigma)! \text{ and } \delta_2(x_2, \sigma)! \\ \text{not defined} & \text{if } \sigma \in \Sigma_1 \cap \Sigma_2 \text{ and} (\text{not } \delta_1(x_1, \sigma)! \text{ or not } \delta_2(x_2, \sigma)!) \\ (\delta_1(x_1, \sigma), x_2) & \text{if } \sigma \in \Sigma_1 - \Sigma_2 \text{ and } \delta_1(x_1, \sigma)! \\ (x_1, \delta_2(x_2, \sigma)) & \text{if } \sigma \in \Sigma_2 - \Sigma_1 \text{ and } \delta_2(x_2, \sigma)! \\ \text{not defined} & \text{if } \sigma \in \Sigma_1 - \Sigma_2 \text{ and not } \delta_1(x_1, \sigma)! \\ \text{not defined} & \text{if } \sigma \in \Sigma_2 - \Sigma_1 \text{ and not } \delta_2(x_2, \sigma)! \end{cases}$$

Note that usually the result of synchronous product is assumed to be a reachable generator. However, we do not have this assumption in our work.

The synchronous product may be used to model the joint operation of two generators.

When the alphabets Σ_1 and Σ_2 are disjoint (i.e., $\Sigma_1 \cap \Sigma_2 = \phi$), we refer to the synchronous product as the **shuffle product**.

Another operation on two generators is ***meet***. Consider $G_1 = (X_1, \Sigma_1, \delta_1, x_{0,1})$ and $G_2 = (X_2, \Sigma_2, \delta_2, x_{0,2})$ to be two generators. The result of the *meet* operation of G_1 and G_2, *meet*(G_1, G_2), is a reachable generator in which only the common events may occur and in synchrony. Specifically,

$$meet(G_1, G_2) = (X, \Sigma, \delta, x_0)$$

where

$$X = X_1 \times X_2$$

$$\Sigma = \Sigma_1 \cap \Sigma_2$$

$$x_0 = (x_{0,1}, x_{0,2})$$

$$\delta = X \times \Sigma \to X$$

with

$$\delta((x_1, x_2), \sigma) = \begin{cases} (\delta_1(x_1, \sigma), \delta_2(x_2, \sigma)) & \text{if } \delta_1(x_1, \sigma)! \text{ and } \delta_2(x_2, \sigma)! \\ \text{not defined} & \text{otherwise} \end{cases}$$

In *meet*(G_1, G_2) at a given state, event σ can be generated if both generators can generate σ at their respective state.

Consider a generator $G = (\Sigma, X, \delta, x_0)$ and alphabet Σ_1 with $\Sigma_1 \cap \Sigma = \phi$. The ***selfloop*** operation, ***selfloop***(G, Σ_1), constructs a new generator from G by attaching transitions $x = \delta(x, \sigma)$ for all $\sigma \in \Sigma_1$ to every state $x \in X$.

Note that *sync* and *meet* operations have commutative and associative properties. Therefore, the order of operands is not important and brackets can also be ignored. i.e.:

Commutative Property:

$$sync(G_1, G_2) = sync(G_2, G_1)$$

$$meet(G_1, G_2) = meet(G_2, G_1)$$

Associative property:

$$sync(G_1, sync(G_2, G_3)) = sync(sync(G_1, G_2), G_3)$$

$$meet(G_1, meet(G_2, G_3)) = meet(meet(G_1, G_2), G_3)$$

Consider three generators $G1$, $G2$ and $G3$. The synchronous product of $G1$, $G2$ and $G3$, $sync(G_1, G_2, G_3)$, is defined as:

$$sync(G_1, G_2, G_3) = sync(G_1, sync(G_2, G_3)) = sync(sync(G_1, G_2), G_3)$$

Similarly, *meet* operation on $G1$, $G2$ and $G3$ is defined as:

$$meet(G_1, G_2, G_3) = meet(G_1, meet(G_2, G_3)) = meet(meet(G_1, G_2), G_3)$$

sync and *meet* operations on more than three generators can be defined similarly.

Example 2.1:

Two generators, G1 and G2 are shown in Fig. 2.1. The synchronous product of G1 and G2 is depicted in Fig. 2.2-a. Fig. 2.2-b illustrates *meet*(G1,G2).

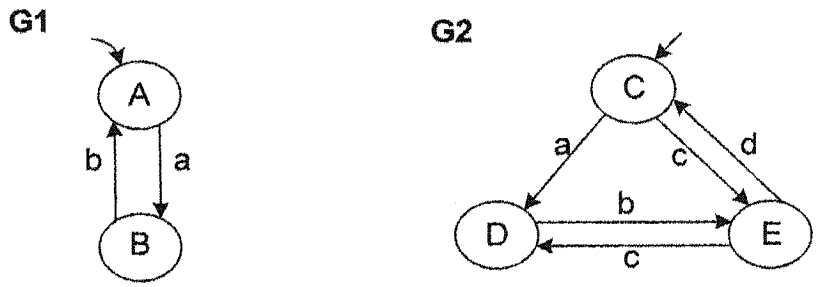
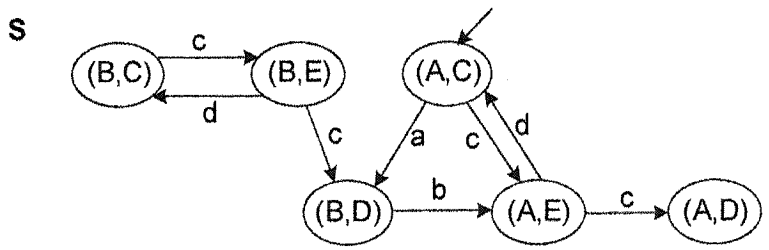
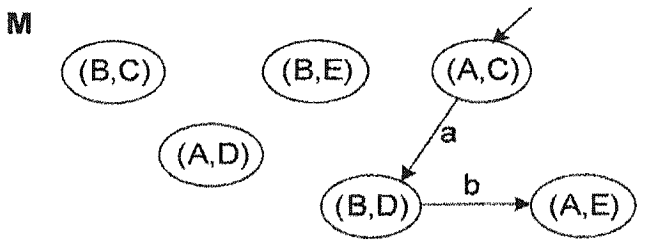


Figure 2.1: Generators of Example 2.1



a) S = *sync*(S1,S2)



b) M = *meet*(S1,S2)

Figure 2.2: *sync* and *meet* operations on the systems in Example 2.1

$$X^{S1} = \{A, B, C.D\} \qquad \Sigma^{S1} = \{a, b, c, d, e\}$$

$$X^{S2} = \{E, F, G, H, I\} \qquad \Sigma^{S2} = \{a, b, d, f, g\}$$

$$X^{S} \subseteq X^{S1} \times X^{S2} = \{(A,E),(A,G),(A,H),(B,F),(C,I),(D,I),(C,G),(D,G),(C,H),(D,H)\}$$

$$\Sigma^{S} = \Sigma^{S1} \cup \Sigma^{S2} = \{a, b, c, d, e, f, g\}$$

$$X^{M} \subseteq X^{S1} \times X^{S2} = \{(A,E),(A,H),(B,F),(C,I),(C,G)\}$$

$$\Sigma^{M} = \Sigma^{S1} \cap \Sigma^{S2} = \{a, b, d\}$$

2.2 Hierarchical Finite State Machines (HFSM)

The finite state machines (generators) are used widely to model discrete event systems. In various applications such as supervisory control, these models are easy to understand and interpret. The main shortcoming of these models is that they are efficient only when the system does not have a "large" state set. In real world systems, many components work together simultaneously. Therefore, the number of system states, in the worst case, grows exponentially in the number of components. This results in very large state sets. In these situations, tools for handling complexity are necessary.

In [Har87], D. Harel introduced a visual framework for representing complex finite state machines. In this visual formalism known as statechart, while preserving many vital features of state machines, he improved their structure by adding powerful features such as hierarchy (depth). Statecharts have become a powerful modeling tool for complex systems.

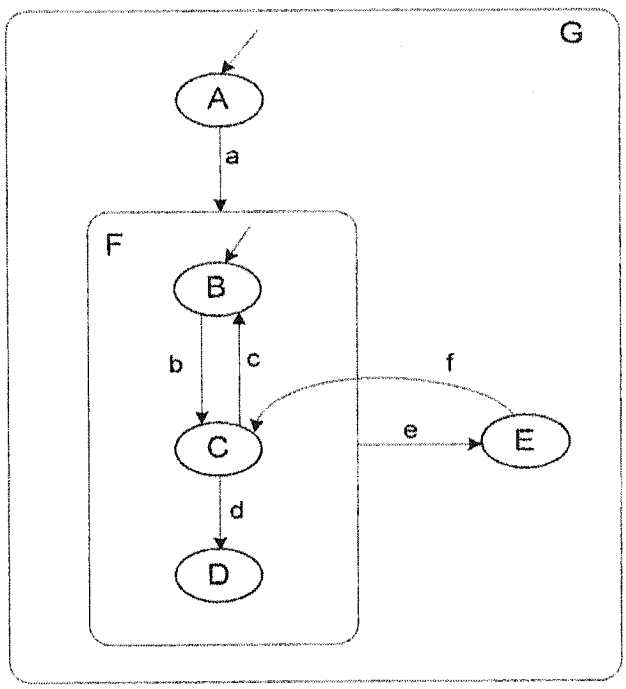
Hierarchical Finite State Machines (HFSM) are a simplified version of statecharts that enhance finite state machine models by adding hierarchy to the system (other features of statecharts are not used). They can be applied for modeling of many complex processes such as communication networks, traffic control systems and manufacturing systems.

HFSMs have been the subject of extensive research. One of the pioneering works on HFSMs is [BH93], in which the authors investigate reachability in HFSMs. They study a class of HFSMs, called asynchronous HFSMs (AHFSMs) and introduce an algorithm for reachability test in systems within this class. They show that the complexity of their method is polynomial in the number of system components while the complexity of the method using the equivalent flat state machine is exponential in the number of components. In the following, we present a brief description of HFSMs.

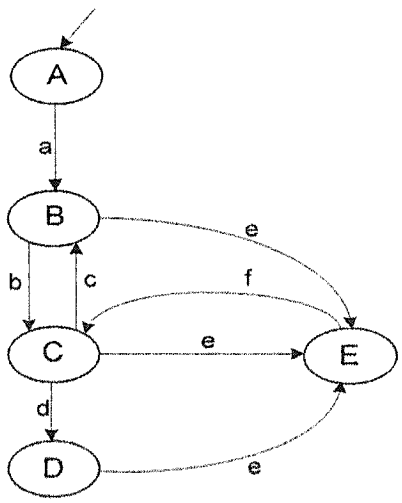
2.2.1 Informal Description of HFSM

In an HFSM, states are represented by boxes. A state (box) may contain other states. This characteristic is used to model depth or hierarchy in the system .We call a state (box) which includes other states a **super state;** otherwise, we refer to it as a **basic state**. The states in each super state are called immediate sub-states. Sub-states can be either super states or basic states. For instance, in Fig. 2.3.a, G is a super state and states A, F and E are its immediate sub-states. B is a sub-state of G but it is not an immediate sub-state of G. State F is also a super state but states A and E are basic states. States B, C and D are basic states and the sub-states of F. It should be noted that we can consider a basic state as a special case of a super state which includes no sub-state. In this thesis, basic states

are represented by circles or ellipses while super states are shown with rounded rectangles.



a) An HFSM G consisting of OR states



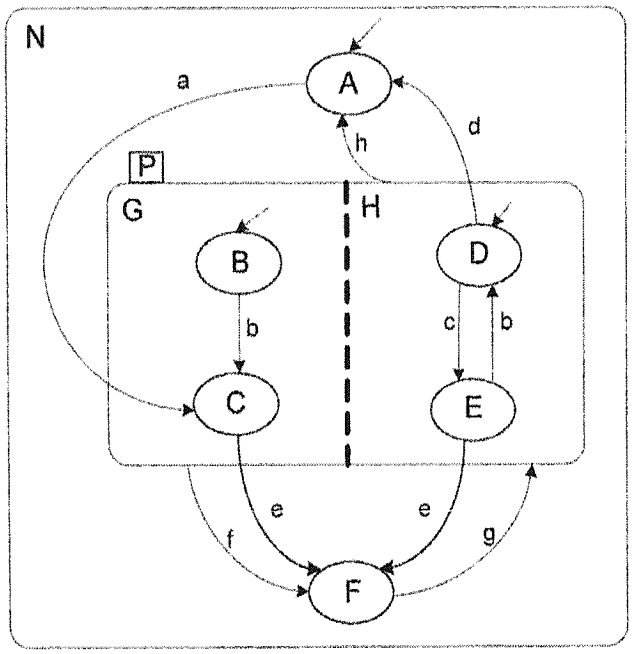
b) The equivalent flat state machine of G

Figure 2.3: An HFSM consisting of OR states and its equivalent flat state machine

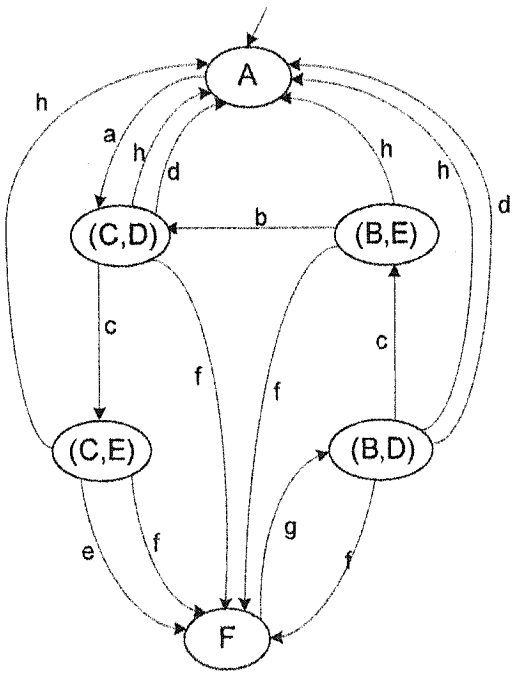
A label or an event is associated with each transition-path (edge or arrow between states). The symbols 'a', 'b', 'c', 'd', 'e' and 'f' in Fig. 2.3.a, represent the events in the system. Arrows with no labels indicate the initial state of each super state. A transition-path to a box represents a transition-path to the initial state of the box (for example, transition 'a' to super state F in Fig. 2.3.a). In fact, arrows attached to initial states remove the need for continuing the event arrows beyond the boundary of the box. It is obvious that if the box represents a basic state, then, the initial state will be the basic state itself. A transition-path from a super state represents transitions from all of the states of that super state. For example, transition-path 'e' from the super state F to E in Fig. 2.3.a represents transitions from B, C and D to E. Fig. 2.3.b shows the equivalent flat model of the HFSM.

We call state F in Fig. 2.3 an **OR-state** since being in F is equivalent to being in B, C or D, but not in more than one state at any time.

Concurrency is represented by **AND** states. A system can evolve concurrently in different sub-states of an AND state. Consider Fig. 2.4.a. P is an AND state consisting of two components, G and H. Being in P is equivalent to being in both G and H at the same time. In other words, while in P, the state of the system is represented by a pair (x, y), where x and y are states in G and H. Note that G and H are OR states themselves. The tuples (B,D), (B,H), and (A) are called **configurations** of the HFSM in Fig. 2.4.a. A configuration is a tuple of states. Specifically, the configurations represent sets of concurrent states which the HFSM can occupy simultaneously. Elements of a configuration can be either basic or super states.



a) An HFSM N consisting of an AND-state.



b) The equivalent (flat) state machine of N

Fig 2.4: An HFSM consisting of an AND state and its equivalent (flat) state machine

A transition-path entering an AND-state, takes the system to the initial states of the AND components. For example, transition 'g' in Fig. 2.4.a takes the system from F to state (B,D), while 'a' takes the system from A to (C,D). We assume that the internal transitions of AND-states are determined by the transition function of the synchronous product of its immediate sub-states. The equivalent (flat) finite-state machine of N in Fig. 2.4.a is given in Fig. 2.4.b.

Figure 2.5 shows the hierarchical structure of the HFSM in Fig. 2.4.a. This diagram is usually called the **hierarchy tree** of the system.

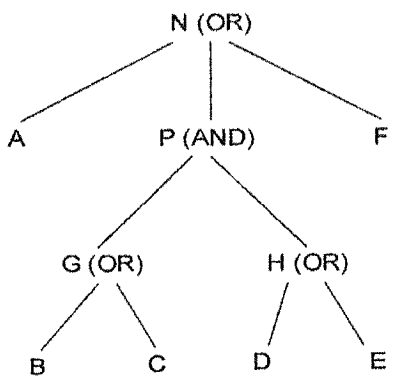


Figure 2.5: The hierarchy tree of the HFSM in Fig. 2.4.a

In general, a transition-path is defined between two configurations. For example, in Fig. 2.4.a, a transition-path labelled 'a' can be represented by the triple ((A), a , (C,D)), where (A) and (C,D) are the **source** and **destination** configurations of this transition-path, respectively. If a configuration only consists of basic states, then it is called a **basic configuration** (for example, configuration (B,D) is basic while (B,H) is not basic). A transition-path from a basic configuration to a basic configuration is called a **basic transition-path**.

Each transition-path t is associated with a unique state of the HFSM. This state is the smallest (in terms of its size) OR-state containing the source and destination configurations of t. For instance in Fig.2.4.a, the transition-path labelled 'b' belongs to the state G and the transition-path labelled 'a' belongs to N.

As pointed out, each OR super state a has an initial immediate sub-state. Let $\bar{\rho}(a)$ denote the initial immediate sub-state of a. We define the initial configuration as follows.

Definition 2.4 [BH93]: The initial (default) configuration of a state a, denoted as $\hat{\rho}(a)$, is defined as the basic configuration obtained inductively as:

1. If a is an AND super state with immediate sub-states $a_1,\cdots,a_k$, $\hat{\rho}(a) = (\hat{\rho}(a_1),\cdots,\hat{\rho}(a_k))$.
2. If a is an OR state with immediate sub-states $a_1,\cdots,a_k$, $\hat{\rho}(a) = \hat{\rho}(a_i)$ *iff* $a_i = \bar{\rho}(a)$.
3. If a is a basic state, then $\hat{\rho}(a) = (a)$. □

In Fig. 2.4.a, $\hat{\rho}(P) = (\hat{\rho}(G), \hat{\rho}(H)) = (B, D)$.

Consider the transition-path $t = ((A), a, (C))$ in Fig. 2.4.a. This transition-path takes the system to (C,D). However, D is not specified in the destination configuration of t. Therefore, (C,D) is the **explicit** destination configuration of t.

For studying an HFSM, it is more convenient to transform the system to an equivalent HFSM in which all the transition-paths are basic with explicit destination configurations. The resulting HFSM is said to be in **canonical form.**[2]

[2] In [BH93], an HFSM is said to be in canonical form if it has an alternating structure. However, we assume this for the system only in chapter 5 when discussing computational complexity.

In the following, we explain how a transition-path can be transformed to one with explicit destination configuration [BH93].

Let $t = (u, \sigma, v)$ be a transition-path belonging to a state a.

i. Backtrack from the elements of v along the hierarchy tree to the level of the state a and mark the strict super states of elements of v.

ii. Add the initial configuration of every unmarked immediate sub-state of a marked AND-state to v.

iii. The resultant tuple is called the explicit destination configuration of the transition-path t.

Since initial configurations are basic, explicit destination configurations are basic too.

For diagnosis purposes, it is easier to study systems having no AND super states. In other words, all AND-states are substituted by the synchronous product of their components. An HFSM in the canonical form with no AND super states is referred to as a **basic HFSM**. Given an HFSM, we can transform it to the equivalent basic HFSM. In the following, we explain how a system M, can be converted to a basic HFSM.

First, we convert the transitions-paths of M to the basic transition-paths with explicit destination configurations (explained earlier). Then, on the hierarchy tree of M, we start from the lowest level on a branch and go up and combine the components of all AND-states on the branch (using the transition function of the synchronous product). We do this for all the branches. In this way, AND-states are replaced with equivalent OR states. Note that some of the OR-states (which are the sub-state of an AND-state of a higher level) are also replaced by their sub-states. For example, if in Fig. 2.4.a, D was an OR-state with basic states, we would replace it with its basic states. The AND states of the

highest level will become OR-states. The resulting system will be an HFSM equivalent to *M*.

It should be noted that after all AND-states have been replaced by OR-states, the basic states of the basic HFSM are the states of the equivalent flat system of *M*. In order to differentiate these states from the original basic states of *M*, we call them **simple states**.

The equivalent basic HFSM of the HFSM in Fig. 2.4.a is shown in Fig. 2.6. Figure 2.7 shows the hierarchy tree of the system in Fig. 2.6.

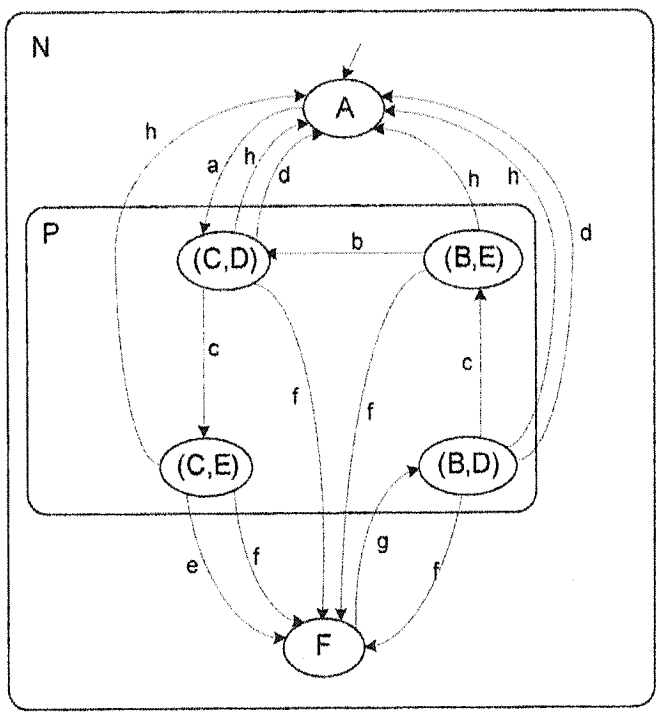


Figure 2.6: The basic HFSM equivalent to the HFSM of Fig. 2.4.a.

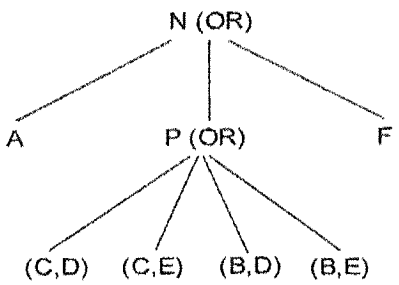


Figure 2.7: The hierarchy tree of the equivalent basic HFSM

2.2.2 Formal Definition of HFSM

In this section, we explain the formal definition of HFSMs. We assume that the HFSM is in canonical form.

Definition 2.5 [BH93]: An **HFSM** is a 5-tuple $M = (A, \Sigma, \Omega, \triangleright, \bar{\rho})$. Here, A is the finite state set of M and is the disjoint union of three subsets: A^+, the set of OR states of M, $A^\perp$, the set of AND states of M, and A^{Basic}, the set of basic states of M. Σ is the set of event symbols and is the disjoint union of an **observable** event set, Σ_o, and an **unobservable** event set, Σ_{uo}. Ω is the set of transition-paths (edges) in the system. The function $\bar{\rho} : A^+ \rightarrow A$ provides the initial state of OR-states. '$\triangleright$' is a binary relation representing the hierarchical structure of M and is called the **hierarchy relation** (on A). The relation '$\triangleright$' satisfies the following conditions:

1) There exists a unique state called the **root state** of M, denoted by $r(M)$, such that for no state $a \in A$, $a \triangleright r$.

2) For every state $a \in A$ and $a \neq r$, there exists a unique state $b \in A$ such that $b \rhd a$. The state b is called the immediate super state of a, whereas a is an immediate sub-state of b.

3) A state $a \in A$ has no immediate sub-state if and only if a is a basic state. In other words, a state $a \in A$ is a super state if there exists a state $b \in A$, such that $a \rhd b$; otherwise a is a basic state. □

The pair $(A, \rhd)$ defines a tree called the **hierarchy tree** of the HFSM.

The transitive closure of $\rhd$, denoted as $\rhd^*$, extends the hierarchy relation as follows. For every $a, b \in A$, $a \rhd^* b$ if b is a (not necessarily immediate) sub-state of a.

Now, we explain the configurations in the system.

Let q be a tuple of basic states.

- A state a is a **super state** of q if every element of q is a sub-state of a. For example, N and P are the super states of (B,D) in Fig. 2.4.a.
- A state a is the **lowest super state** of q if a is a super state of q and for each super state b of q, $b \rhd^* a$. The lowest super state of q is denoted by $LS(q)$. For example, in Fig. 2.4.a, $LS((B,D)) = P$.
- Two states q_1 and q_2 are called **orthogonal** if either $q_1 = q_2$ or, if neither is a super state of the other and $LS((q_1, q_2)) \in A^{\perp}$. A tuple of states q is orthogonal if every pair of states in q is orthogonal. For example, (B,D) is orthogonal but (B,A) and (B,C) are not orthogonal.
- An orthogonal tuple is called a **configuration**. In other words, a configuration is a tuple of states in which all states can be taken simultaneously by the system.

- Let q denote a configuration and a denote a super state of q. q is a **full configuration** of a if it cannot be extended through augmentation with further orthogonal sub-states of a. In other words, for every $b \in A$, if $a \rhd^* b$, it can be concluded that (q,b) is not orthogonal. A full configuration is called a **simple state** of the system. If q is not full, then it is a **partial configuration** of a. For example, in Fig. 2.4.a, (B) is a partial configuration of P, but (B,D) is a full configuration of P. The set of all full configurations of a is denoted as Q_a.
- A **transition-path** $t \in \Omega$ is represented by a 3-tuple $t = (q_1, \sigma, q_2)$, where q_1 and q_2 are two configurations (partial or full) in the system and $\sigma \in \sum$. q_1 is called the source of t. q_2 is called the target of t. □

Definition 2.6: Consider an HFSM $M = (A, \sum, \Omega, \rhd, \bar{\rho})$. We define the function **level** $LV : A \rightarrow \mathcal{N}$ as follows ($\mathcal{N}$ is the set of natural numbers):

- The level of the root super state is zero: $LV(r(M)) = 0$.
- If $a, b \in A$ and $b \rhd a$ then $LV(a) = LV(b) + 1$. □

Consider an HFSM $M = (A, \sum, \Omega, \rhd, \rho)$. The equivalent basic HFSM $H = (X, \sum, \delta, \mapsto, \rho)$ can be constructed from M as follows.

Replace all AND super states with their OR-state equivalents. In the resulting HFSM, H, all basic states will be simple (full configuration).

In H, X is the set of finite states and is the disjoint union of two state sets: X^{Simple} and X^+, where X^{Simple} is the set of simple states and X^+ is given by:

$$X^+ = A^+_{top} \bigcup A^{\perp}_{top},$$

where

$$A^{+}_{top} = \{a \in A^{+} \mid \nexists b \in A^{\perp} : b \rhd^{*} a\}$$

$$A^{\perp}_{top} = \{a \in A^{\perp} \mid \nexists b \in A^{\perp}, b \neq a : b \rhd^{*} a\}.$$

Thus, X^{+} is the set of all OR-states and AND-states in M that do not have an AND super state in M. For example, for the HFSM in Fig. 2.4.a, $A^{+}_{top} = \{N\}$ and $A^{\perp}_{top} = \{P\}$.

$\sum$ is the event set. $\delta : X^{Simple} \times \sum \rightarrow X^{Simple}$ denotes the transition function. $\rho : X^{+} \rightarrow X$ is a function which gives the initial state of an OR super state. Thus,

$$\rho(x) = \overline{\rho}(x) \text{ for } x \in A^{+}_{top}$$

$$\rho(x) = \hat{\rho}(x) \text{ for } x \in A^{\perp}_{top}$$

Note that H has OR super states only. '$\mapsto$' is a hierarchy relation on X. Relation '$\mapsto$' is defined as follows:

i. For $x, y \in X^{+}, x \mapsto y$ if and only if $x \rhd y$

ii. For $x \in X^{+}, y \in X^{Simple}, x \mapsto y$ if and only if $x = LS(y)$. □

The initial state of the system is denoted by x_0 and is the initial state of the root (*i.e.* $x_0 = \rho(r(H))$).

It should be noted that the transition function of H is the same as the transition function of the equivalent flat system.

Definition 2.7: Consider a basic HFSM $H = (X, \sum, \delta, \mapsto, \rho)$. Suppose for $x, y \in X$ and $\sigma \in \sum$, $y = \delta(x, \sigma)$. Then the three tuple $t = (x, \sigma, y)$ is called a **transition** of H. y is called the **target** of t. x is called the **source** of t. □

Transition $t = (x, \sigma, y)$ sometimes is shown as: $x \xrightarrow{\sigma} y$.

T^H denotes the set of all transitions in H.

Definition 2.8: Consider a basic HFSM $M = (X, \Sigma, \delta, \mapsto, \rho)$. We define the function **level** $LV : X \to \mathbb{N}$ as follows ($\mathbb{N}$ is the set of natural numbers):

- The level of the root super state is zero: $LV(r(H)) = 0$.
- If $x, y \in X$ and $y \mapsto x$ then $LV(x) = LV(y) + 1$. □

Definition 2.9: Let $x, y \in X$ and $x \mapsto y$. x is called the **father** of y and y is called a **child** of x.

Definition 2.10: Consider a basic HFSM $H = (X, \Sigma, \delta, \mapsto, \rho)$. Suppose that $x, y, a, b \in X$, $a \mapsto x, b \mapsto y$ and $\sigma \in \Sigma$. H is said to have a **Father-Child connected** (FC connected) form if for any $t = (x, \sigma, y)$, then either $a = b$, or $a \mapsto b$ or $b \mapsto a$. □

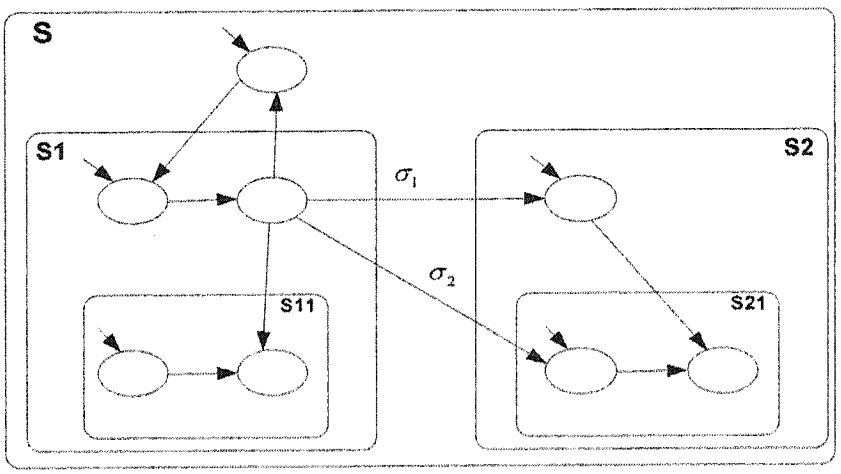


Figure 2.8: An HFSM with a non-FC-connected form

A super state of an FC-connected basic HFSM has only transitions to the states of its children or its father. In our framework, we are concerned with HFSMs with an FC-connected form. For instance, transitions labelled with 'σ_1' and 'σ_2' in Fig. 2.8 are not allowed in our framework.

Theorem 2.1: A basic HFSM which is not FC-connected can be transformed to a basic FC-connected HFSM.

Proof: Let $t = (x, \sigma, y) \in T^H$ be a basic transition of H with $x, y \in X^{Simple}, a \mapsto x, b \mapsto y$. If $a = b, a \mapsto b$ or $b \mapsto a$, then consider other transitions of T^H. Otherwise, replace t according to the following procedure. Let w_0 denote the first common "ancestor" of x and y so that there exist super states $w_1, w_2, \ldots, w_n, w_1', w_2', \ldots, w_m'$ with:

$$w_0 \mapsto w_1, w_1 \mapsto w_2, \ldots, w_n \mapsto x$$

and

$$w_0 \mapsto w_1', w_1' \mapsto w_2', \ldots, w_m' \mapsto y$$

Note that since r is the common ancestor of all states of H, a common "ancestor" w_0 can always be found.

Add dummy states $d_0, d_1, \ldots, d_{n-1}$ to $w_0, w_1, \ldots, w_{n-1}$, respectively. Similarly, add dummy states $d_1', \ldots, d_{m-1}'$ to $w_1', \ldots, w_{m-1}'$, respectively. Clearly, no dummy states has to be added to w_0 if it is the father of x or y (i.e. $w_0 = w_n$ *or* $w_0 = w_m'$).

Add the following dummy transitions to the system.

$$x \xrightarrow{\sigma_n} d_{n-1}$$
$$d_{n-1} \xrightarrow{\sigma_{n-1}} d_{n-2}$$
$$\vdots$$
$$d_1 \xrightarrow{\sigma_1} d_0$$
$$d_0 \xrightarrow{\sigma'_1} d'_1$$
$$d'_1 \xrightarrow{\sigma'_2} d'_2$$
$$\vdots$$
$$d'_{m-2} \xrightarrow{\sigma'_{m-1}} d'_{m-1}$$

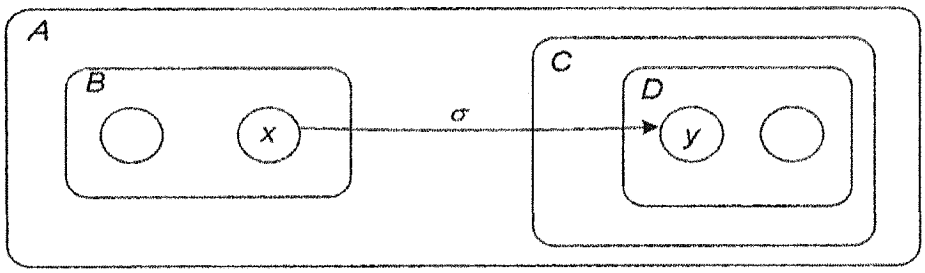
Finally, add $d'_{m-1} \xrightarrow{\sigma} y$.

If σ is observable, then σ_i and σ'_j are assumed observable for all i and j $(1 \le i \le n, \quad 1 \le j \le m)$; otherwise, they are assumed unobservable. We can see that all of the above transitions satisfy the property mentioned in Definition 2.10.

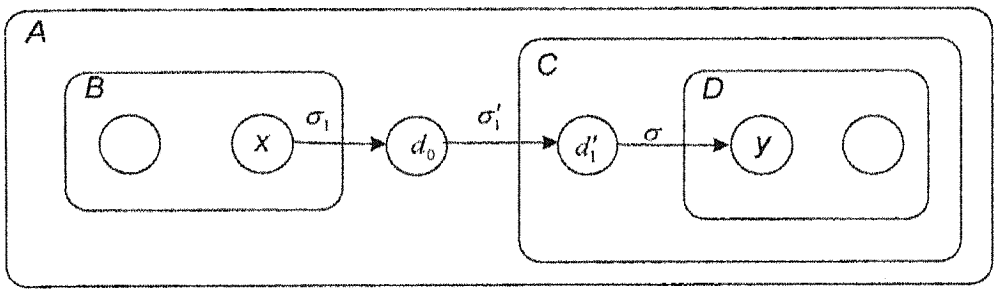
After performing the aforementioned procedure for all $t \in T^H$, the HFSM will have an FC-connected form. □

Example 2.2

The HFSM shown in Fig. 2.9.a does not have an FC-connected form. After applying the above procedure, the HFSM in Fig. 2.9.b is obtained which is FC-connected. In this example, $w_0 = A$ is the first common ancestor of x and y. $w_1 = B, w'_1 = C$ and $w'_2 = D$. Dummy states d_0 and d'_1 along with the dummy transitions $x \xrightarrow{\sigma_1} d_0$ and $d_0 \xrightarrow{\sigma'_1} d'_1$ are added to the system.



a) An HFSM with a non-FC-connected form



b) An HFSM with an FC-connected form

Figure 2.9: Adding dummy states and transitions to an HFSM in order to make it FC-connected.

An HFSM $H = (\sum, X, \delta, \mapsto, \rho)$ is said to be reachable if every $x \in X^{Simple}$ is reachable from x_0 (the initial state of the system).

Definition 2.11: A basic HFSM $H = (\sum, X, \delta, \mapsto, \rho)$ is said to have a **standard** form if:

1) H is reachable.

2) H is FC-connected. □

From now on, all HFSM are standard unless otherwise specified.

In the next chapter, we will present diagnosis in flat DES. We will explain how the diagnoser is constructed for a flat DES using a passive state-based approach. We also review the concept of diagnosability.

Chapter 3

Diagnosis in Flat DES

Fault diagnosis is an important issue in complex systems. Many systematic approaches have been introduced in literature to solve diagnosis problems. Some of these works concentrate on diagnosis of the systems that can be modeled as DES. In this chapter, we will explain a systematic method for designing the diagnoser in DES introduced in [HKW03]. We also discuss the diagnosability of a failure in this framework. It is assumed that the system model does not include any hierarchy. In other words, the system model is assumed to have a flat structure. In the following chapters of the thesis, we will extend this approach to hierarchical and modular DES.

The method presented in [HKW03] is simple and general. Moreover, the designed diagnoser can be transformed into computer code and implemented easily. In [HKW03], outputs generated in the system are employed for diagnosis, but here, we modify the approach so that the diagnoser uses observable events for diagnosis. We think that using observable events simplifies the setup in hierarchical systems. Note that the resulting diagnoser still provides estimates for the system condition and therefore, follows a state-based approach for diagnosis.

In section 3.1, plant and failure modeling are explained. In section 3.2, diagnoser design is discussed. Section 3.3 explains failure diagnosability.

3.1 Plant Model

The plant studied in this chapter is assumed to be a nondeterministic finite state automaton $G = (X, \Sigma, \delta, x_0)$, where X and Σ are the state and event sets respectively. x_0 is the initial state, $\delta : X \times \Sigma \rightarrow 2^X$ is the transition function (2^X denotes the power set of X). It is assumed that the event set Σ can be partitioned to two disjoint subsets, Σ_o and Σ_{uo} ($\Sigma = \Sigma_o \dot{\cup} \Sigma_{uo}$), where Σ_o represents the observable event set and Σ_{uo} consists of unobservable events in the system.

The model describes the system's behaviour in both normal (system functioning properly) and faulty situations. Faulty situations are referred to as **failure modes**. Each failure mode corresponds to some kind of failure or a set of such failures in the system. **Failure events** represent transitions leading to failure modes. Thus, the event set Σ includes failure events. Hence, Σ can be considered as a disjoint union of failure events and non-failure events ($\Sigma = \Sigma_f \dot{\cup} \Sigma_{nf}$). We consider the more challenging case where the failure events are unobservable ($\Sigma_f \subseteq \Sigma_{uo}$); diagnosing failures caused by observable failure events would be very easy.

Suppose that there are p failure modes ($F_1, F_2, \ldots, F_p$) in the system. First, the **single failure scenario** is assumed in the system. It means that at most one of the failure modes may occur at a time. Therefore, the system's condition can be either N (normal) or one of

the p failure modes: $F_1, F_2, \ldots, F_p$. It should be noted that the single failure scenario is different from a single failure mode situation in which the system has only one failure mode; i.e., p =1. Generalization of this approach to the case of simultaneous occurrence of multiple failures is straightforward and will be discussed in Remark 3.1.

Let $\mathrm{K} := \{N, F_1, F_2, \ldots, F_p\}$ be the condition set of the system. It is assumed that the state set X can be partitioned according to the condition of the system: $X = X_N \dot{\cup} X_{F_1} \dot{\cup} \cdots \dot{\cup} X_{F_p}$. The **condition map** $\kappa: X \to \mathrm{K}$ is defined such that for every $x \in X$, $\kappa(x)$ is the condition of the system at the state x: $\kappa(x) = N$ if $x \in X_N$ and $\kappa(x) = F_i$ if $x \in X_{F_i}$ $(i \in \{1, \ldots, p\})$.

The definition of κ is also extended to the subsets of X : $\kappa(z) = \{\kappa(x) \mid x \in z\}$, for any $z \subseteq X$.

In failure detection and isolation, given the observable event sequence ($\sigma_1\sigma_2\sigma_3$ ....), we want to find the condition of the system.

3.2 Diagnoser Design

A diagnoser is a system that detects and isolates failures. In our framework, it is a finite state machine that generates an estimate of the condition of the system as its output using the observable event sequence generated by the system ($\sigma_1\sigma_2\ldots\sigma_n$) (Fig. 3.1). This is done by calculating a set $z_n \subseteq X$ to which x must belong at the time that σ_n was observed; $\kappa(z_n)$ will be the estimate of the system condition. After occurrence of the next observable event (σ_{n+1}), the diagnoser updates z_n.

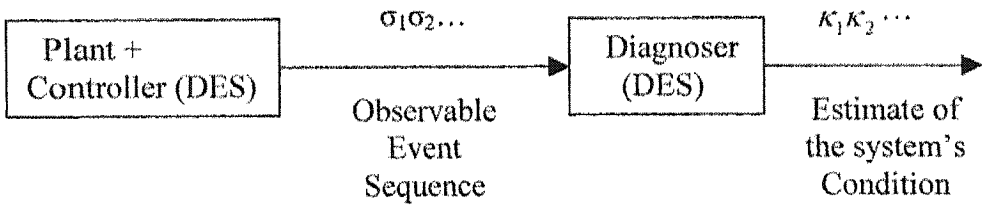


Figure 3.1: System and diagnoser

Before going further, we introduce the concept of observation-adjacency, which we will find useful in diagnoser design.

Definition 3.1: For any two states $x, x' \in X$ in a system, we say x' is **observation-adjacent** to x with respect to $\sigma \in \Sigma_o$ and write $x \overset{\sigma}{\Rightarrow} x'$ if x' can be reached from x using a path along which σ is the only observable event and σ is the last event in the sequence. In cases where the observable event is not important we show the observation adjacency by $x \Rightarrow x'$. This means that x' is observation-adjacent to x with respect to some $\sigma \in \Sigma_o$.□

We define the **diagnoser** to be a finite state Moore machine $D = (Z, \Sigma_o, \xi, z_0, \hat{\mathrm{K}}, \kappa)$, where Z, Σ_o and $\hat{\mathrm{K}} \subseteq 2^{\mathrm{K}} - \{\phi\}$ are the state, event and output sets of D; $z_0 \in 2^X - \{\phi\}$ is the initial state; $Z \subseteq 2^X - \{\phi\}$, and $\xi : Z \times \Sigma \to Z$ represents the transition function; $\kappa : Z \to \hat{\mathrm{K}}$ denotes the output map. Each diagnoser state z is identified with a non-empty subset of X .

The diagnoser state transition $z_{n+1} = \xi(z_n, \sigma_{n+1})$ is given by:

$$z_{n+1} = \{x \mid x \in X \,\&\, (\exists x' \in z_n : x' \overset{\sigma_{n+1}}{\Rightarrow} x)\}, \quad n \geq 1 .$$

z_{n+1} will be the set of states which are observation-adjacent to some of the states of z_n with respect to σ_{n+1}; i.e., each state of z_{n+1} is reachable from a state in z_n using a path along which the only observable event is σ_{n+1} and this event is the last event in the sequence.

z_0 holds the information available about the state of the system at the time that the diagnoser is started. If no information is available at that time, then $z_0 = X$. $z_0 = X_N$ when the system is only known to be in normal condition at the time that the diagnoser is started. The diagnoser is initialized with $z_0 = \{x_0\}$ if it starts at the same time as the system.

Let $z^1, z^2 \in Z$ be two states of the diagnoser such that $\xi(z^1, \sigma) = z^2$ for some $\sigma \in \sum$. Suppose that z^1 is given. z^2 can be computed by calculating for every $x_1 \in z^1$, all x_2 such that $x_1 \overset{\sigma}{\Rightarrow} x_2$. Every $x \in X$ may belong to several states of D. Therefore, computing the set of observation-adjacent states for every $x \in X$ could be an efficient way to decrease the diagnosis computations. For each $x \in X$ a breadth-first search reachability analysis can be done in $O(|X| + |T|)$ time [CLR90], where $|X|$ and $|T|$ denote the cardinalities of X and T (the set of transitions of G). Hence, calculating the set of observation-adjacent states for all states of X can be done in $O(|X|^2 + |X||T|)$ time.

The Reachability Transition System (corresponding to G) is defined to be the transition system $G = (X, R, \sum_o)$, which has $X, \sum_o$ and R as the state, event and transition sets, respectively. $R \subseteq X \times \sum_o \times X$ and $(x_1, \sigma, x_2) \in R$ if and only if $x_1 \overset{\sigma}{\Rightarrow} x_2$. The RTS contains the information about the observation-adjacent states. The diagnoser can be

computed using the RTS. Reachability transition systems may also help in the on-line implementation of diagnosis algorithms. We will discuss this in the following chapter.

3.3 Diagnosability in Flat FSM

Diagnosability of a failure mode F_i in a diagnoser D is the answer to the question of whether or not F_i can always be detected and isolated by the diagnoser. For simplicity, we assume that failure modes are permanent. We also concentrate on the single failure scenario. It means that only one failure mode is possible to occur at a time. Simultaneous occurrence of failures will be discussed later in this section.

Definition 3.2 [HKW03]: A state z of a diagnoser, D, is called **F_i-certain** if from the corresponding estimate of the system's condition, $\kappa(z)$, it can be inferred that the failure F_i has occurred. □

In a single failure scenario, z is F_i-certain iff $\kappa(z) = \{F_i\}$.

Definition 3.3 [HKW03]: A state z of a diagnoser, D, is called **F_i-uncertain** if $\kappa(z)$ indicates that the failure F_i might have occurred, but the occurrence of F_i is not the only possibility. □

In a single failure scenario, z is F_i-uncertain iff $F_i \in \kappa(z)$ but $\kappa(z) \neq \{F_i\}$ .For example, if for some z, $\kappa(z) = \{N, F_i\}$ then z is F_i-uncertain.

Definition 3.4 [HKW03]: A permanent failure mode F_i is diagnosable in a system if F_i can be detected and isolated following the occurrence of at most N_i ($N_i \geq 0$) events in the system after both the occurrence of the failure and initialization of the diagnoser. □

For diagnosability analysis in this framework, the diagnoser can be started either before or after the occurrence of the failure. In other words, the system's condition is assumed to be unknown at the time that the diagnoser is initialized.

Definition 3.5: A path $x_1 \xrightarrow{\sigma_1} \cdots \xrightarrow{\sigma_{n-1}} x_n$ with $n \geq 2$ is a **cycle** if $x_1 = x_n$. □

Definition 3.6: A cycle of F_i-uncertain states of the diagnoser is called an ***F_i*-uncertain cycle**. □

Definition 3.7: A cycle $z^1, \cdots, z^m$ of F_i-uncertain states of a diagnoser D is called ***F_i*-indeterminate** if there exist $l \geq 1$ and $x_1^j, x_2^j, \ldots, x_l^j \in z^j$, for all j with $1 \leq j \leq m$ such that $x_k^j \in X_{F_i}$ for all j and k with $1 \leq j \leq m$, $1 \leq k \leq l$ and $x_1^1, x_1^2, \ldots, x_1^m, x_2^1, \ldots, x_2^m, \ldots, x_l^1, \ldots, x_l^m$ form a cycle in the Reachability Transition System (RTS) of the system. The RTS cycle is called an **underlying faulty cycle** of the F_i-indeterminate cycle. □

Note that not every F_i-uncertain cycle is F_i-indeterminate [HKW03]. Furthermore, the diagnoser can get trapped in an F_i-uncertain cycle if the cycle is F_i-indeterminate.

Theorem 3.1: Assume a single failure scenario in the system and $z_0 = X$. Also assume that there are no cycles of unobservable events or deadlocks in the system. A permanent

failure F_i is diagnosable if and only if there are no F_i-indeterminate cycles in the diagnoser.

We need the following lemma to prove Theorem 3.1.

Lemma 3.1 [HKW03]: Consider a path $z_0 \xrightarrow{\sigma_1} z_1 \xrightarrow{\sigma_2} z_2 \cdots \xrightarrow{\sigma_n} z_n (n \geq 1)$ in the diagnoser. For, any $x_n \in z_n$, there exists $x_i \in z_i (1 \leq i \leq n-1)$ such that $x_i \overset{\sigma_{i+1}}{\Rightarrow} x_{i+1}$ for $(1 \leq i \leq n-1)$. □

Proof of Theorem 3.1 [HKW03]: First, suppose the diagnoser does not have indeterminate cycles. If F_i occurs in the system and a new event is observed, the diagnoser state will be either F_i-certain or F_i-uncertain. Since failures are assumed to be permanent in the system, if the diagnoser state is F_i-certain, then it will remain F_i-certain and F_i is diagnosed. Now assume that the diagnoser state is F_i-uncertain. Clearly, the number of F_i-uncertain states is bounded in the diagnoser. Therefore, assuming that there are no cycles of unobservable events or deadlocks in the system, after observation of a bounded number of events, the diagnoser will reach an F_i-certain state. This is because the diagnoser gets trapped in a cycle of F_i-uncertain states only if the cycle is F_i-indeterminate.

Converesely, if the diagnoser includes an F_i-indeterminate cycle, then there exists a sequence of observable events $\sigma_1, \sigma_2, \ldots, \sigma_n$ that can take the diagnoser on a path $z_0 \rightarrow z_1 \rightarrow z_2 \rightarrow z_n$, where z_n belongs to the F_i-indeterminate cycle. Let $x_n \in z_n$ belong to an underlying faulty cycle in the RTS. By lemma 3.1, there exist states $x_0, \ldots, x_{n-1}$ such that $x_i \overset{\sigma_{i+1}}{\Rightarrow} x_{i+1}$ for $(0 \leq i \leq n-1)$. After reaching x_n, the RTS may remain on the underlying

faulty cycle causing the diagnoser to stay on the F_i-indeterminate cycle indefinitely. Therefore, there exists a trajectory for the system leading to states in X_{F_i} such that the corresponding observable event sequence throws the diagnoser into a cycle of F_i-uncertain states and keeps it there indefinitely. Hence, F_i is not diagnosable. □

Remark 3.1: The results on single failure scenario can be extended to the case of simultaneous failures. For simplicity, we consider a system with only two failure modes. Generalization to any number of failure modes follows similarly.

Assuming two failure modes, there will be three failure scenarios in the system: F_1, F_2 and F_{12}, where F_{12} denotes the simultaneous occurrence of F_1 and F_2. The RTS and the diagnoser are defined and constructed as in section 3.2. However, the condition set of the system changes to $\mathrm{K} = \{N, F_1, F_2, F_{12}\}$. Therefore, the state set of the system can be partitioned according to the system's condition: $X = X_N \dot{\cup} X_{F_1} \dot{\cup} X_{F_2} \dot{\cup} X_{F_{12}}$. In addition, the definition of the condition map $\kappa: X \to \mathrm{K}$ is modified: $\kappa(x) = N$ if $x \in X_N$, $\kappa(x) = F_i$ if $x \in X_{F_i}$ ($i \in \{1,2\}$), and $\kappa(x) = F_{12}$ if $x \in X_{F_{12}}$.

The condition map can be extended to subsets of X according to $k(z) = \{k(x) \mid x \in z\}$ for all $z \subseteq X$.

Now, we discuss the diagnosability of the failure mode F_1. Consider simultaneous occurrence of the failure modes F_1 and F_2 in the system. According to Def. 3.4, a state z of the diagnoser is F_1-certain iff $\kappa(z) = \{F_1\}$ or $\kappa(z) = \{F_{12}\}$ or $\kappa(z) = \{F_1, F_{12}\}$. If z is not F_1-certain and $\kappa(z) \cap \{F_1, F_{12}\} \neq \phi$, then z is F_1-uncertain.

The definition of diagnosability is the same as before. The necessary and sufficient condition for diagnosability of F_1 is the same as that in the single failure scenario except that the definition of F_1-indeterminate cycles has to be modified as follows.

Definition 3.10 [HKW03]: A cycle $z^1,\cdots,z^m$ of F_i-uncertain states of a diagnoser D is called ***F_1*-indeterminate** if there exist $l \geq 1$ and $x_1^j, x_2^j, \ldots, x_l^j \in z^j$, for all $1 \leq j \leq m$ such that $\{x_k^j \mid 1 \leq j \leq m, 1 \leq k \leq l\} \subseteq X_{F_1}$ or $\{x_k^j \mid 1 \leq j \leq m, 1 \leq k \leq l\} \subseteq X_{F_{12}}$ and also $x_1^1, x_1^2, \ldots, x_1^m, x_2^1, \ldots, x_2^m, \ldots, x_l^1, \ldots, x_l^m$ form a cycle in the RTS of the system. □

Chapter 4

D-holon and Diagnosis in HFSM

In this chapter, we propose an approach for diagnosing failures in systems modeled as hierarchical finite state machines (HFSM). HFSM can be used as a powerful tool for modeling complex systems whose structure is organized in a top-down hierarchy. It may be applied, particularly for diagnostic purposes, to many complex systems such as aircrafts, spacecrafts, traffic control and manufacturing systems. We split a HFSM into simpler structures called D-holons. In our approach, a diagnoser is designed for each D-holon and these diagnosers work together to detect the failures in the system. In this way, at any given time, the diagnoser needs only a subset of the transitions of the system. This is particularly useful in the case of complex systems where the entire plant model is too large to be stored in computer Random Access Memory (RAM). In this chapter, we assume that all HFSMs are in the standard form unless otherwise stated. In other words, all HFSMs are reachable, FC-connected with basic transitions. We introduce the concept of D-holon in section 4.1. In section 4.2, first we discuss failure modeling and then we introduce our method of fault diagnosis in HFSM. The problem of diagnosability in our

approach is addressed in section 4.3. We propose a method that uses the hierarchical structure of the system to investigate the diagnosability of a failure mode. We introduce the concept of phase-diagnosability. The discussion in this chapter assumes that at most one failure mode may occur at a time in the system. However, the results can be extended to the case of simultaneous occurrence of two (or more) failure modes in a way similar to that discussed in chapter 3 for flat DES.

4.1 Concept of D-holon

In complex systems, particularly, multi-phase operations, only part of the system model (corresponding to the phase of operation) can perhaps be enough for supervision and diagnosis. For example, in a multiple-phase system, the diagnosability of a specific component failure may be examined only in phases of operation where the component is active.

The HFSM model explained in chapter 2 provides a setup for studying a specific part or level of the system in hierarchical systems. There, the super states describe a part of the operation of the system in terms of the states and transitions among the states within the super state. In this section, we introduce D-holon, a structure which represents a super state with its internal transitions as well as its external transitions to other super states. We will show that D-holon is an appropriate model for diagnostic purposes.

D-holon is a mathematical model for describing the dynamic of discrete event systems. We associate a D-holon with each super state. The D-holon describes the internal transitions of the super state along with its external transitions to higher and lower levels in the HFSM.

The word "holon" which comes from Arthur Koestlirs' 1967 book, "The Ghost in the Machine", was used in [Wan95] for the supervisory control of hierarchical DES. There, the word holon refers to entities in the hierarchical structure which behave partly as independent wholes but at the same time as subordinate parts. The term holon is a combination of the Greek word holos (meaning whole) and the suffix –on (suggesting a particle or part as in proton, neutron). This concept is also used in other works such as [AEM94] and [GHL94] as a modeling tool for physical systems.

The D-holons introduced in this thesis are to some extent similar to the holons in [Wan95]. We have, however, made some modifications to make them suitable for fault diagnosis problems.

Definition 4.1: Consider HFSM $H = (X, \Sigma, \delta, \mapsto, \rho)$. A **D-holon** DH associated with a super state S in H is defined as a 4-tuple:

$$DH_S := (X^{DH_S}, \Sigma^{DH_S}, \delta^{DH_S}, X_0^{DH_S})$$

where

- X^{DH_S} is the state set of DH. It is the disjoint union of $X_I^{DH_S}$ and $X_E^{DH_S}$,i.e., $X^{DH_S} = X_I^{DH_S} \dot{\cup} X_E^{DH_S}$, where $X_I^{DH_S}$ is the internal state set consisting of the simple states of S, i.e., $X_I^{DH_S} = X_S^{Simple}$, and $X_E^{DH_S}$ is the external state set consisting of the immediate simple states of higher and/or lower level super states (father and children super states in HFSMs with an FC-connected form) which are the target of a transition from a simple state of S:

$$X_E^{DH_S} = \{y \mid y \in (X^{Simple} - X_S^{Simple}) \,\&\, (\exists x \in X_S^{Simple}, \sigma \in \Sigma : t = (x, \sigma, y) \in T^H)\}$$

- $X_0^{DH_S} \subseteq X_I^{DH_S}$ is the set of initial states. It is a subset of those elements of X_S^{Simple} which are the target of a transition from a state of a higher level or a lower level super state (father and children in HFSMs with an FC-connected form).
- $\sum^{DH_S}$ is the event set. It is the union of the boundary event set, $\sum_B^{DH_S}$, and the internal event set $\sum_I^{DH_S}$, i.e., $\sum^{DH_S} = \sum_I^{DH_S} \bigcup \sum_B^{DH_S}$. $\sum_I^{DH_S}$ consists of the events associated with transitions among internal states:

$$\sum_I^{DH_S} = \{\sigma \mid \sigma \in \sum^{DH_S} \&(\exists x_1, x_2 \in X_S^{Simple} : t = (x_1, \sigma, x_2) \in T^H)\}$$

$\sum_B^{DH_S}$ includes the events associated with transitions from internal states to external states:

$$\sum_B^{DH_S} = \{\sigma \mid \sigma \in \sum^{DH_S} \&(\exists x_1 \in X_S^{Simple}, x_2 \in (X^{Simple} - X_S^{Simple}) : t = (x_1, \sigma, x_2) \in T^H)\}$$

It should be noted that $\sum_I^{DH_S}$ and $\sum_B^{DH_S}$ are not necessarily disjoint.

- $\delta^{DH_S} : X_I^{DH_S} \times \sum^{DH_S} \rightarrow X^{DH_S}$ is the transition function. $\delta_B^{DH_S}$ and $\delta_I^{DH_S}$ are defined as restrictions of δ^{DH_S} to the internal events and the boundary events respectively, and thus:

$$\delta_I^{DH_S} : X_I^{DH_S} \times \sum_I^{DH_S} \rightarrow X_I^{DH_S}$$

$$\delta_B^{DH_S} : X_I^{DH_S} \times \sum_B^{DH_S} \rightarrow X_E^{DH_S}$$

□

The following example illustrates the concept of D-holon.

Example 4.1:

Figure 4.1 shows an HFSM. The HFSM is not in the canonical form, because it has non-basic transitions (transitions from basic states to super states). Figure 4.2, depicts an

equivalent HFSM in the standard form. There, all non-basic transitions have been substituted by equivalent basic transitions and the AND state S_2 has been replaced by the synchronous product of its components. The D-holons associated with the super states S_0, S_1 and S_2 are shown in Fig. 4.3. The dotted lines shown in D-holons separate the external states from the internal states.

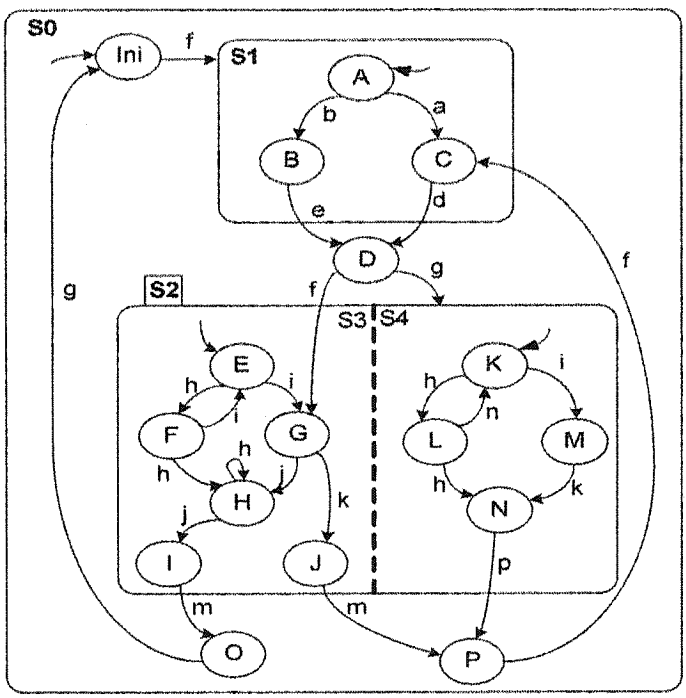


Figure 4.1: The HFSM of Example 4.1

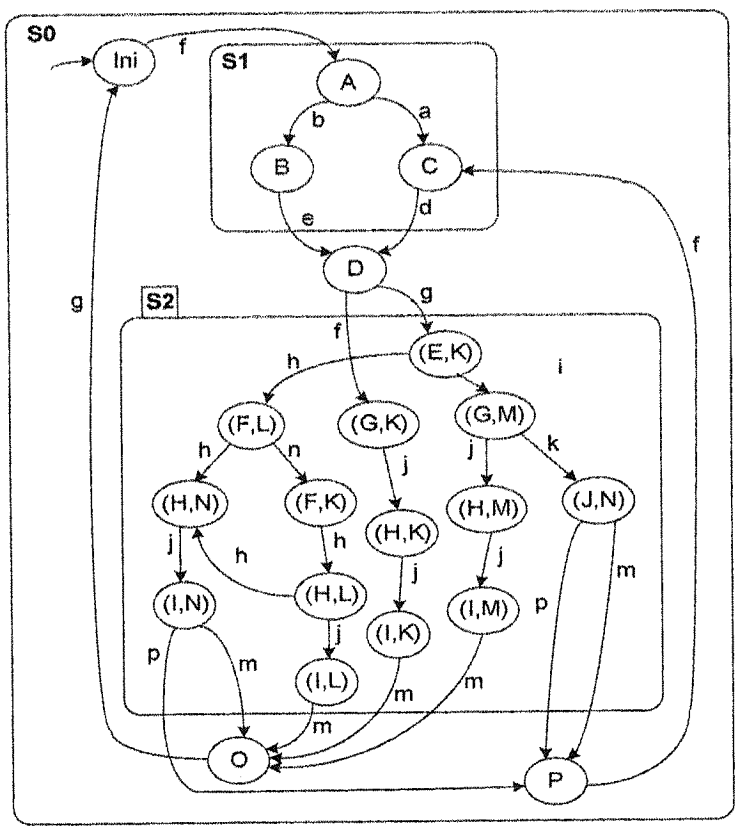


Figure 4.2: The equivalent HFSM of the HFSM in Fig. 4.2

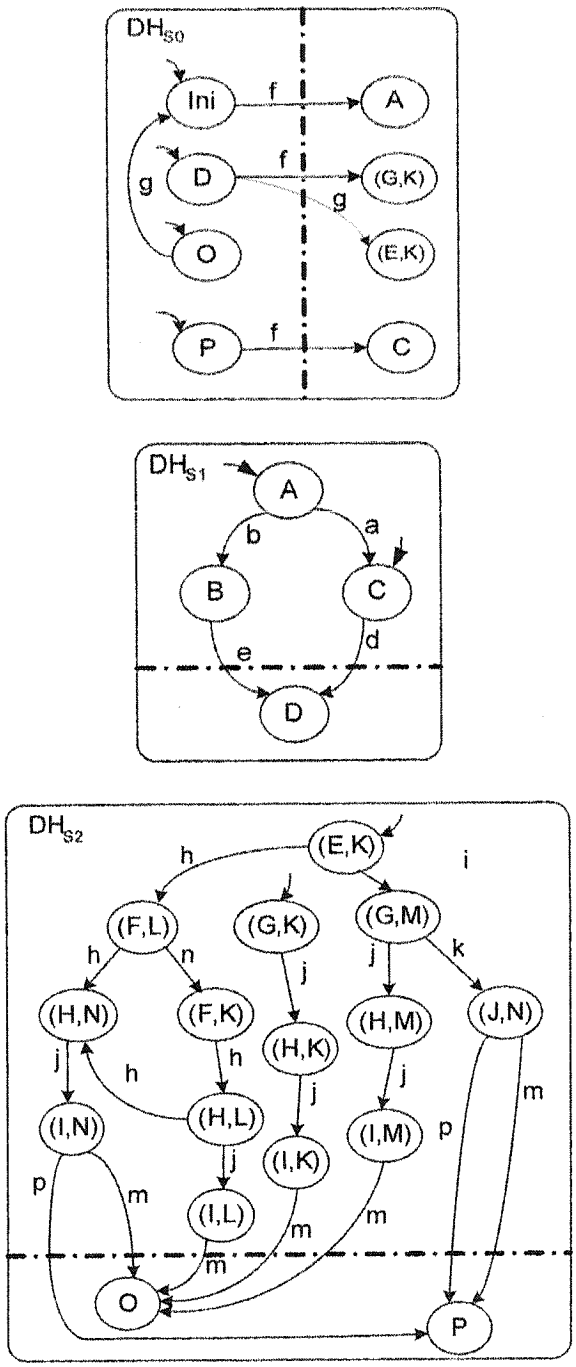


Figure 4.3: D-holons associated with the super states of the HFSM in Example 4.1

In DH_{S0}:

$$X^{DH_{S0}} = \{Ini, A, C, D, O, P, (E,K), (G,K)\}$$

$$X_0^{DH_{S0}} = \{Ini, D, O, P\}$$

$$\sum^{DH_{S0}} = \{g, f\}$$

In DH_{S1}:

$$X^{DH_{S1}} = \{A, B, C, D\}$$

$$\sum^{DH_{S1}} = \{a, b, e, d\}$$

$$X_0^{DH_{S1}} = \{A, C\}$$

In DH_{S2}:

$$X^{DH_{S2}} = \{(E,K),(G,K),(G,M),(J,N),(H,M),(I,M),(H,K),(I,K),$$
$$(F,L),(H,N),(F,K),(H,L),(I,L),(I,N),O,P\}$$

$$\sum^{DH_{S2}} = \{h, i, j, k, m, n, p\}$$

$$X_0^{DH_{S2}} = \{(E,K),(G,K)\}$$

In a hierarchical DES, a D-holon DH can be considered as a part of the system that can be studied (especially for diagnostic purposes) as a distinct system with some states and transitions among the states $(X_I^{DH_S}, \sum_I^{DH_S}, \delta_I^{DH_S})$, and transitions connecting the D-holon to the rest of the system model $(X_E^{DH_S}, \sum_B^{DH_S}, \delta_B^{DH_S})$. The structure of a D-holon is very similar to that of a generator. The only difference is that, in a D-holon DH, in addition to the internal state set, the D-holon has an external state set consisting of those simple states not in $X_I^{DH_S}$ that are the targets of transitions from the states in $X_I^{DH_S}$.

In our model, we do not consider transitions entering the D-holon as a part of the D-holon structure because as we will see later, in fault diagnosis, we require only the information about the target of outgoing transitions of the D-holon.

We observe that an HFSM can be completely defined by its D-holons and its initial state(s). The D-holons describe an HFSM in an organized way and facilitate fault diagnosis by breaking up the whole system model into several sub-models allowing the diagnosis system to focus on the relevant sub-models. As a result, the implementation becomes more efficient in computer memory usage. Later, we will explain this issue in detail.

Definition 4.2: The **level of a D-holon** DH is defined to be equal to the level of the corresponding super state. □

Definition 4.3: Let DH_{S1} and DH_{S2} be the D-holons associated with the super states S_1 and S_2 of an HFSM. Suppose S_1 is the father of S_2. Then, DH_{S1} is called the **father** of DH_{S2}, and DH_{S2} the **child** of DH_{S1}. □

In example 4.1, DH_{S0} is the father of D-holons DH_{S1} and DH_{S2}.

The following condition simplifies diagnoser design.

Condition 4.1: For any D-holon DH, boundary events are observable:

$$\sum_B^{DH} \subseteq \sum_o$$ □

Typically, a D-holon describes a specific phase of operation. Thus, the boundary transitions represent change of the phase of operation. Change of phase commands can

usually be assumed to be observable in an HDES. In fact, they act like bridges between different levels of the hierarchy in the system.

Definition 4.4: We refer to a D-holon which satisfies condition 4.1 as a **standard** D-holon. □

In this thesis, we assume that all D-holons are standard unless otherwise indicated.

Remark 4.1: In the next chapter, the information of individual AND components is used to diagnose failures in HFSMs. There, AND super states are not required to be replaced by the synchronous product of their components. We can associate a D-holon with each OR super state that is a component of an AND-state. The internal states are separated by dashed lines to convey the idea that components are operating synchronously. Figure 4.4, shows the D-holons associated with the AND components S3 and S4 of Example 4.1.

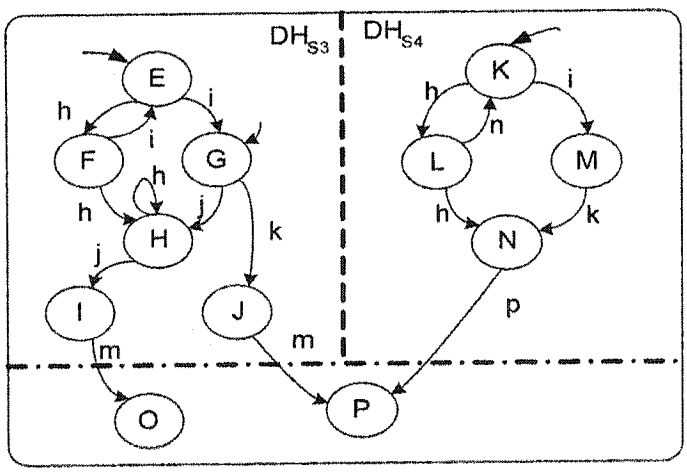


Figure 4.4: D-holons associated with AND components S3 and S4 in Example 4.1

In DH_{S3}:

$$X^{DH_{S3}} = \{E, F, G, H, I, J, O, P\}$$

$$\sum^{DH_{S3}} = \{h, i, j, k, m\}$$

$$X_0^{DH_{S3}} = \{E, G\}$$

In DH_{S4}:

$$X^{DH_{S4}} = \{K, L, M, N, P\}$$

$$\sum^{DH_{S4}} = \{h, i, k, n, p\}$$

$$X_0^{DH_{S4}} = \{K\}$$ □

In the following section we will explain how HFSMs and D-holons are utilized to enhance fault diagnosis process.

4. 2 Fault Diagnosis in HFSM

4. 2. 1 Plant Model

Consider an standard HFSM $H = (X, \Sigma, \delta, \mapsto, \rho)$. We associate a D-holon with each super state of the system. Suppose that DH_S is the D-holon associated with the super state S. As stated earlier, this D-holon is represented by a 4-tuple $DH_S := (X^{DH_S}, \Sigma^{DH_S}, \delta^{DH_S}, X_0^{DH_S})$. We partition the event set Σ^{DH_S} to two disjoint subsets, $\Sigma_o^{DH_S}$ and $\Sigma_{uo}^{DH_S}$, i.e., $\Sigma^{DH_S} = \Sigma_o^{DH_S} \dot{\cup} \Sigma_{uo}^{DH_S}$, where $\Sigma_o^{DH_S}$ represents the set of observable events and $\Sigma_{uo}^{DH_S}$ consists of the unobservable events of DH_S.

Suppose there are P failure modes in the system: $F_1, F_2, \ldots, F_P$. Each failure mode corresponds to some kind of failures or a set of such failures in the system. Failure events may be present at all of the system levels. For instance, a failure in a sensor value (stuck-closed or stuck-open) may occur in low levels of the system. On the other hand, software breakdowns, planning failures and scheduling errors are some kind of failures which usually happen in high levels of the system. The event set $\sum^{DH_S}$ includes failure events of DH_S. It can be considered as a disjoint union of failure events and non-failure events: $\sum^{DH_S} = \sum_f^{DH_S} \dot{\cup} \sum_{nf}^{DH_S}$. Failure events represent those transitions which lead to faulty situations in the system. We assume that failure events are not observable in the system:

$$\sum_f^{DH_S} \subseteq \sum_{uo}^{DH_S}$$

In this chapter, we assume the **single failure scenario** in the system. This means that the system can be in only one of the following P+1 conditions: N (normal), $F_1, F_2, \ldots, F_P$. $\mathrm{K} := \{N, F_1, F_2, \ldots, F_P\}$ will be the condition set of the hierarchical system.

In failure detection and isolation, given the observable event sequence ($\sigma_1\sigma_2\sigma_3$ ....) we want to find the condition of the overall system.

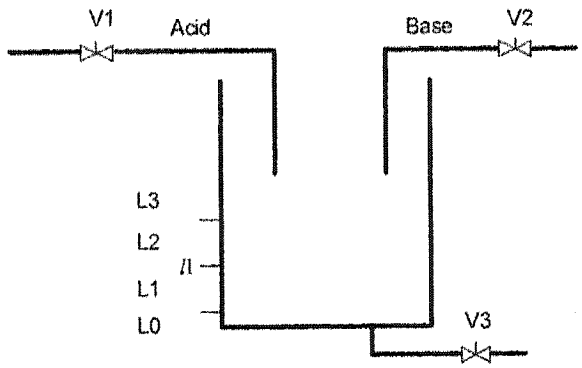


Figure 4.5: Example 4.2. A neutralization Process

Example 4.2: Neutralization process

In a simplified neutralization process (Fig. 4.5), initially, all valves are closed and the tank is almost empty. The process is performed in three phases: *Add-Acid*, *Add-Base* and *Draining*. In *Add-Acid* phase, the controller generates a command to open valve V1 and the reaction tank is filled up to level l_1 with the chemical to be treated (here acid). When the level changes from L1 to L2 this phase ends and the process enters *Add-Base* phase. In this phase, V1 is closed and the neutralizer (base) is added by opening valve V2. When the alkalinity (pH) of the solution reaches the normal range, the process moves to its last phase (*Draining*). In *Draining* phase, V2 is closed and the tank contents are drained through valve V3. After valve 3 is closed by the controller, the cycle of the process is complete. The system goes to the initial state and following this a new cycle can begin.

The neutralization process can be considered as a two level hierarchical system in which the controller generates appropriate commands in each level to guide the process. Table 4.1 shows high and low level events. Sensor measurements are pH and level. pH can be acid (a), normal (n) or base (b), and level can be L0, L1 or L2. Sensor readings and controller commands are observable and the failure events are assumed unobservable.

Here, we only consider one failure mode: valve V1 stuck-open (F). For simplicity, it is assumed that V1 gets stuck-open only when it is open. In our graphs throughout this thesis, failure events are shown with dashed lines.

Figure 4.6 shows a simplified model of neutralization process. A detailed version is presented in Fig. 4.7. Fig. 4.8 shows the D-holons associated with the super states. It should be noted that states AN, AF and B along with the events 'a-added' and 'b-added' are added to the model to make it FC-connected.

Level	Event	Task
Low Level	Oi	Command open valve i
	Ci	Command close valve i
	Lij	Level change from Li to Lj
	n2a	pH change from normal to acid
	a2n	pH change from acid to normal
High Level	add-a	Start Add-Acid phase
	add-b	Start Add-Base phase
	drain	Start Draining phase
	a-added	Phase Add-Acid ended
	b-added	Phase Add-Base ended
	T-empty	Tank is empty

Table 4.1: Example 4.2: Events in a simplified neutralization process

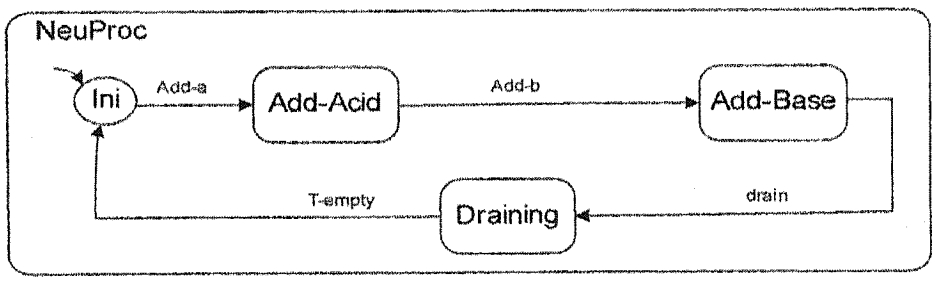


Figure 4.6: A simplified (high-level) description of the neutralization process

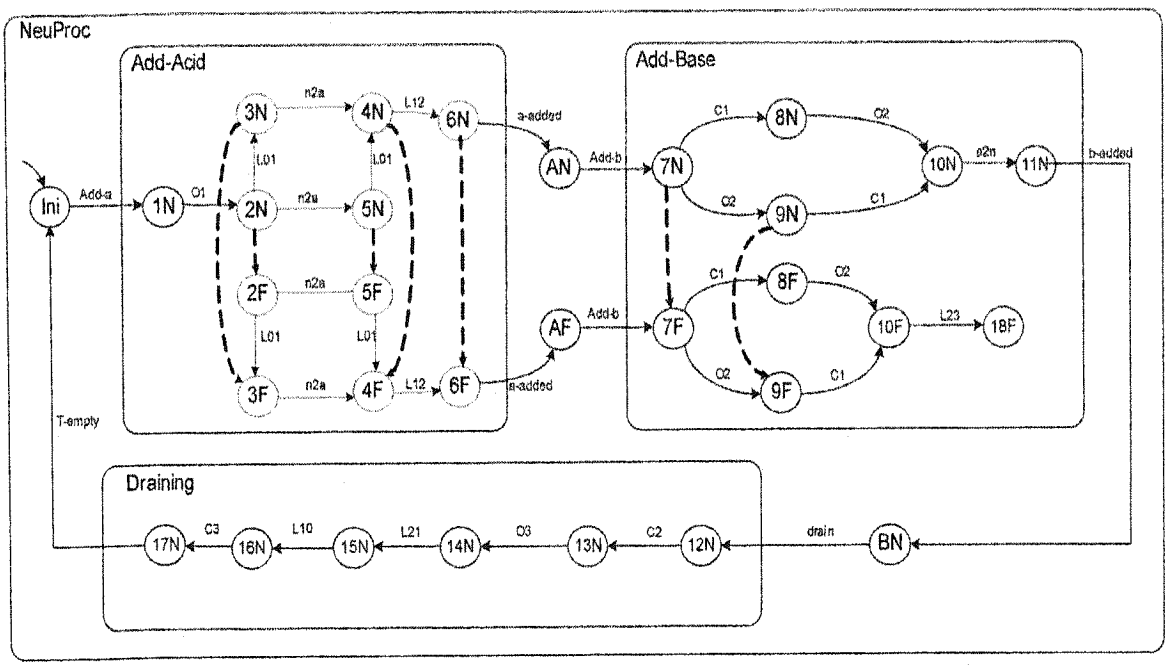
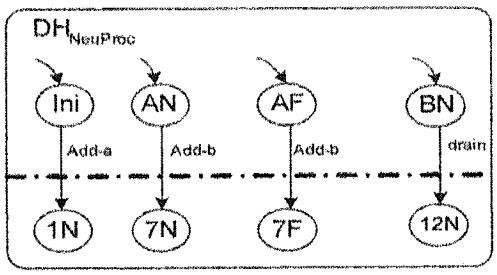
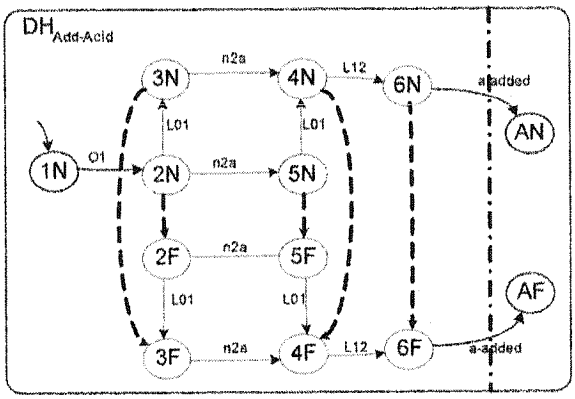


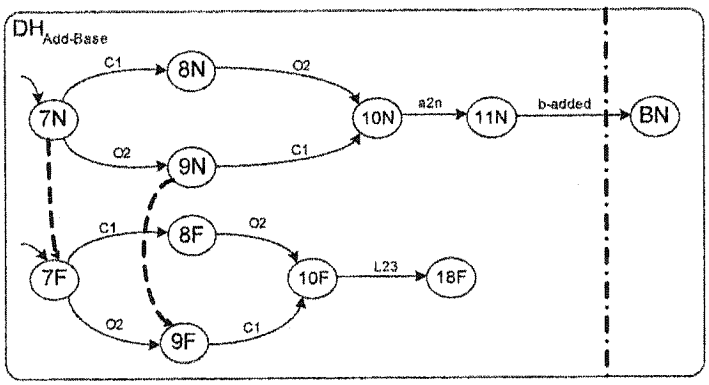
Figure 4.7: A detailed HFSM of the neutralization process



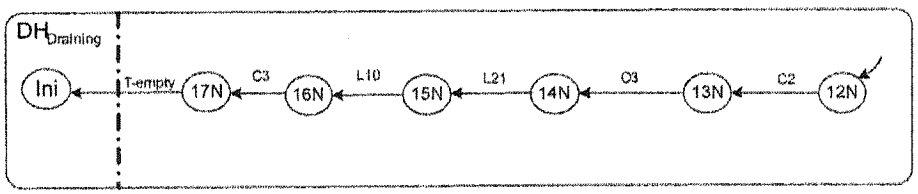
a) $DH_{NeuProc.}$



b) $DH_{Add\text{-}Acid.}$



c) $DH_{Add\text{-}Base}$



d) $DH_{Draining}$

Figure 4.8: D-holons associated with the super states of the neutralization process

The state and event set of each D-holon is as follows.

In $DH_{NeuProc.}$:

$$X = \{Ini, AN, AF, BN, 1N, 7N, 7F, 12N\}$$

$$X_0 = \{Ini, AN, AF, BN\}$$

$$\sum = \{add-a, add-b, drain\}$$

In $DH_{Add\text{-}Acid}$:

$$X = \{1N, 1F, 2N, 2F, 3N, 3F, 4N, 4F, 5N, 5F, 6N, 6F, AN, AF\}$$

$$X_0 = \{1N, 1F\}$$

$$\sum = \{O1, L01, L12, n2a, a-added\}$$

In $DH_{Add\text{-}Base}$:

$$X = \{7N, 7F, 8N, 8F, 9N, 10N, 11N, 18F, BN\}$$

$$X_0 = \{7N, 7F\}$$

$$\sum = \{O2, C1, L23, a2n, b-added\}$$

In $DH_{Draining}$:

$$X = \{12N, 13N, 14N, 15N, 16N, 17N, Ini\}$$

$$X_0 = \{12N\}$$

$$\sum = \{O3, C2, C3, L21, L10, T-empty\}$$

4. 2. 2 Diagnoser Design

Let H be an HFSM modeled by D-holons. Each super state of H is associated with a D-holon describing a particular phase of the system. At the time that the diagnosing process initiates (which may not necessarily be when H starts its operation), a state estimate for

the state of H, z_0, is assumed available. This information usually comes from the sensors and data available in the system (in the worst case, when no information is available, $z_0 = X^{Simple}$). We say a D-holon is *active* if at least one of its internal states belong to z_0. After occurrence of an observable event, the state estimate of the system should be updated. For diagnosis purposes, it may be simpler to use the structure of active D-holons to update z_0. Using the entire model of the system for updating z_0 requires a large amount of memory. In our approach, instead of designing a diagnoser for the equivalent flat system, a diagnoser is constructed for each D-holon of the HFSM. The state estimate of the system will be calculated based on the state estimates provided by its D-holon diagnosers. Initially, the diagnosers of active D-holons are started. The initial state of these D-holon diagnosers is the set of internal states of the D-holon that are present in z_0. The state estimates of the rest of diagnosers will be empty. A D-holon diagnoser is called **active** if its current estate estimate is non-empty. Otherwise, it is called **inactive**. Clearly, several diagnosers may be active concurrently.

After the occurrence of an observable event, only the model of those D-holons whose diagnoser is active is used for updating the state estimate. After the state estimate of a D-holon diagnoser is updated, the new state estimate may include some external states. These states are the internal states of other D-holon. Thus, the name of each external state is passed to the diagnoser of its respective D-holon. This state will be added to the state estimate of the receiving diagnoser. Also, the external state will be removed from the state estimate of the sender diagnoser. If the state estimate provided by a diagnoser gets empty, then it becomes inactive. We see that the state estimate of a diagnoser may be

affected by other diagnosers. Thus, the D-holon diagnosers communicate with each other in order to estimate the system's state and condition.

The following definitions will be found useful.

Definition 4.5: Let DH_1 and DH_2 be two D-holons with DG_1 and DG_2 as their corresponding diagnosers. DG_1 is said to be the **father** of DG_2 if DH_1 is the father of DH_2. In this situation, we refer to DG_2 as the **child** of DG_1. □

Definition 4.6: The **level of a D-holon diagnoser** is defined to be equal to the level of the corresponding D-holon. □

In HFSMs with a standard form and standard D-holons, outgoing transitions from a super state S are observable and enter the super state's father or children only. These transitions along with their target states are included in D-holon DH_S associated with S. Therefore, the diagnoser of DH_S, DG_S, communicates only with its father or its children in order to update its state estimate. This interaction is shown in Fig. 4.9 by arrows.

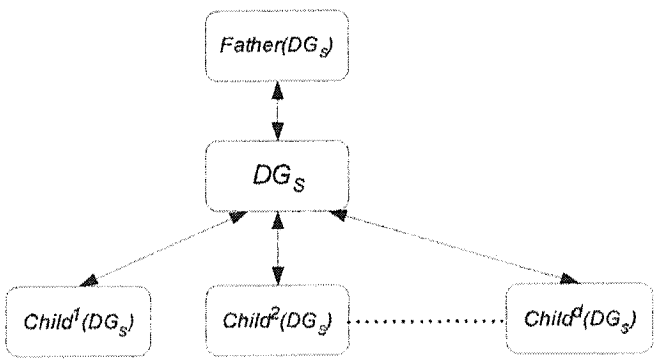


Fig. 4.9: Interaction of father and children on a diagnoser

Figure 4.10 displays a general diagram of the interaction among D-holon diagnosers in an HFSM with an FC-connected form. The diagram is in the form of a tree.

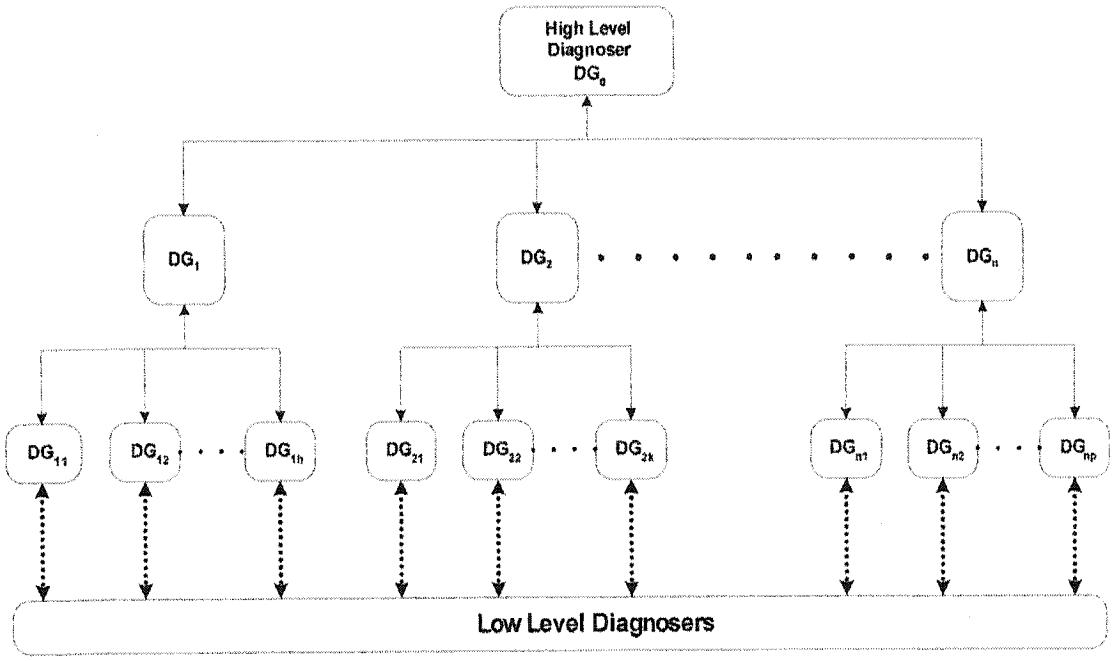


Figure 4.10: Interaction among diagnosers in an HFSM

In the following, we explain the diagnosis process in detail.

Let DH_S be a D-holon in the system and DG_S the corresponding diagnoser. Based on the observed event sequence up to σ_n, DG_S computes a set $z_n^{DG_S} \subseteq 2^{X_I^{DH_S}}$ ($X_I^{DH_S}$ is the internal state set of DH_S). If $z_n^{DG_S} = \phi$, the system can be in $X_E^{DH_S}$. $z_n^{DG_S}$ will be updated after observing σ_{n+1}.

Definition 4.7: DG_S is called **active** if its state estimate $z_n^{DG_S}$ is not empty otherwise it is called **inactive**. □

Let $z_n^{DG_S}$ be the current state estimate provided by DG_S. After the occurrence of a new observable event σ_{n+1}, in order to update the state estimate, a temporary state set $\tilde{z}_{n+1}^{DG_S}$ is calculated as follows:

$$\tilde{z}_{n+1}^{DG_S} = \{x \mid x \in X^{DH_S} \ \& (\exists x' \in z_n^{DG_S} : x' \overset{\sigma_{n+1}}{\Rightarrow} x)\} \ n \geq 0$$

$\tilde{z}_{n+1}^{DG_S}$ contains those states of DG_S which are observation-adjacent to the states of $z_n^{DG_S}$ with respect to σ_{n+1}; i.e., the states that are reachable from the states in $z_n^{DG_S}$ using the path along which the only observable event is σ_{n+1} and this event is the last event in the sequence. It is important to note that σ_{n+1} is the (n+1)th event observed in DH_S from the time that the diagnosis is started (not the (n+1)th observable event generated by the system).

$\tilde{z}_{n+1}^{DG_S}$ can be partitioned according to: $\tilde{z}_{n+1}^{DG_S} = z_{n+1}^{I,DG_S} \dot{\bigcup} z_{n+1}^{E,DG_S}$,where z_{n+1}^{I,DG_S} and z_{n+1}^{E,DG_S} are calculated as follows.

$$z_{n+1}^{I,DG_S} = \tilde{z}_{n+1}^{DG_S} \bigcap X_I^{DH_S}$$

$$z_{n+1}^{E,DG_S} = \tilde{z}_{n+1}^{DG_S} \bigcap X_E^{DH_S}$$

z_{n+1}^{I,DG_S} includes those states of $\tilde{z}_{n+1}^{DG_S}$ that belong to the internal state set of DH_S and z_{n+1}^{E,DG_S} contains the states that belong to the external state set of DH_S. z_{n+1}^{E,DG_S} is used to calculate the states to be sent to other diagnosers. The state set z_{n+1}^{e} passed by DG_S to the diagnoser of another D-holon DH_e is calculated as follows.

$$z_{n+1}^{e} = z_{n+1}^{E,DG_S} \bigcap X_I^{DH_e}$$

z^{e}_{n+1} is a part of the state estimate imported by DH_e. The imported state estimate of a diagnoser DG, is defined as the union of all states passed from the father and (or) children of DG to DG, and it is denoted by $\vec{z}^{DG}_{n+1}$. Thus, the updated state estimate of DG_S, $z^{DG_S}_{n+1}$, will be:

$$z^{DG_S}_{n+1} = z^{I,DG_S}_{n+1} \bigcup \vec{z}^{DG_S}_{n+1}$$

Figure 4.11, demonstrates graphically the updating procedure in a D-holon diagnoser, DG. For simplicity, we have dropped the superscripts DG.

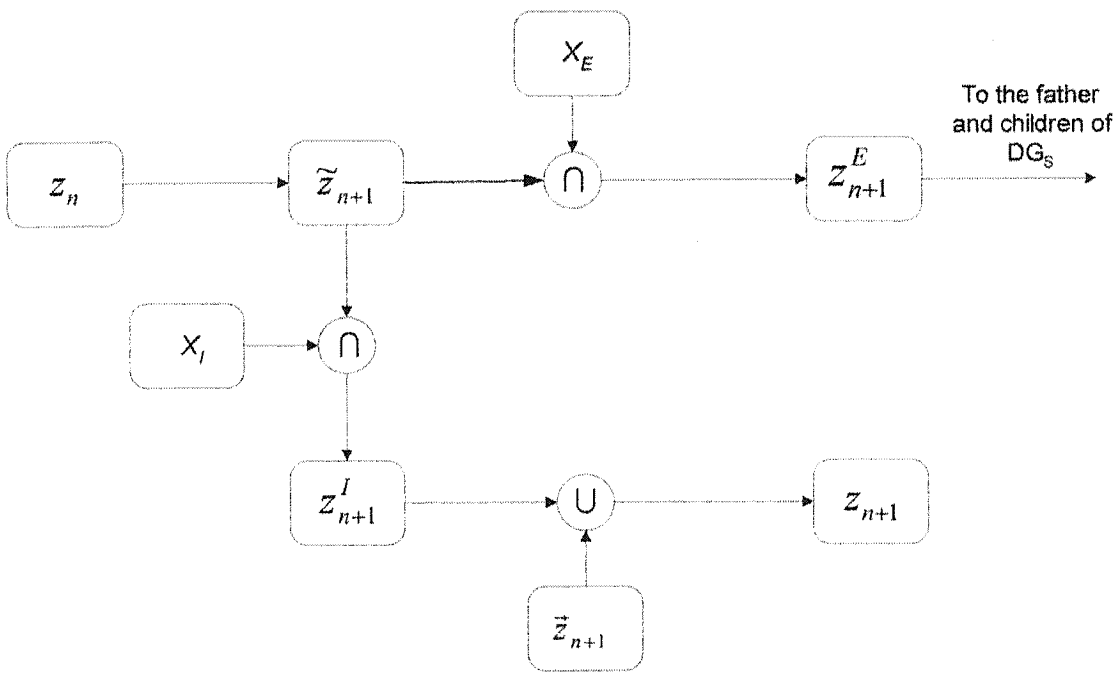


Figure 4.11: The updating procedure in a D-holon diagnoser

Remark 4.2: Suppose that DH_i is one of the D-holons whose diagnoser, DG_i, becomes active when the diagnosis process starts. The initial state of DG_i will be calculated as: $z_0^{DG_i} = X_I^{DH_i} \cap z_0$. The "worst" case would be if $z_0 = X^{Simple}$. In this situation, all D-

holon diagnosers will be initiated with their corresponding D-holon internal state set as the initial state ($z_0^{DG_i} = X_I^{DH_i}$). After the occurrence of the first observable event in the system, only those diagnosers whose updated state estimate is not empty remain active. Furthermore the updated state set of the others may become smaller. The opposite case is when the system and the diagnosis process start simultaneously. In this case, z_0 will contain the initial state of the system only. Therefore, only the diagnoser of the D-holon containing x_0 as an internal state will be initiated in the system. □

As explained in chapter 3, using the RTS of a DES is an economical way to compute the diagnoser of a system. In our approach, we construct an RTS for each D-holon of the system. At any given time, the RTS of the active diagnosers are stored in the memory and used for state and condition estimation.

The RTS of a D-holon DH_S is defined to be a three-tuple $G^{DH_S} = (X^{DH_S}, R^{DH_S}, \sum_o^{DH_S})$, where X^{DH_S} and $\sum_o^{DH_S}$ are the state and observable event sets of DH_S respectively; $R^{DH_S} \subseteq X^{DH_S} \times \sum_o^{DH_S} \times X^{DH_S}$ and $(x_1, \sigma, x_2) \in R^{DH_S}$ if and only if $x_1 \overset{\sigma}{\Rightarrow} x_2$.

Let $H' = (X', \sum', \delta', x_0')$ denote the equivalent flat system of H where $X' = X^{Simple}$ and $\sum' = \sum$ are the state and event sets respectively; $\delta' : X' \times \sum' \to X'$ represents the transition function; x_0' is the initial state of the system. H' is obtained by removing the hierarchical structure of H. There are P failure modes in the system. X' can be partitioned according to the system's condition: $X' = X_N \dot{\cup} X_{F_1} \dot{\cup} \cdots \dot{\cup} X_{F_P}$. Let $DG = (Z, \sum_o', \xi, z_0, \hat{K}, \kappa)$ represent the diagnoser designed for H',

where Z , Σ'_o and $\tilde{\mathrm{K}} \subseteq 2^{\mathrm{K}} - \{\phi\}$ are the state, event and output sets of DG. $z_0 \subseteq 2^{X'} - \{\phi\}$ is the initial state; $Z \subseteq 2^{X'} - \{\phi\}$, and $\xi : Z \times \Sigma'_o \to Z$ represents the transition function; $\kappa : Z \to \tilde{\mathrm{K}}$ denotes the output map of DG. Each diagnoser state z is identified with a non-empty subset of X'. $z_n \subseteq Z$ denotes the state estimate of DG after the occurrence of the observable event σ_n.

Theorem 4.1: The state estimate provided by DG is equal to the union of the state estimates of the D-holon diagnosers:

$$z_n = \bigcup_{j=1}^{d} z_{\sigma_n}^{DG_j},$$

where $z_{\sigma_n}^{DG_j}$ denotes the state estimate provided by DG_j after the occurrence of σ_n, and d is the total number of D-holons .

Proof: Here, we prove the theorem by induction.

Let z_0 be the initial state of the diagnoser. The diagnoser of those D-holons whose internal state share common elements with z_0 becomes active. Assume that DG_i is one of the active diagnosers. Its initial state set is calculated as: $z_0^{DG_i} = X_I^{DH_i} \cap z_0$. The state estimates of the inactive D-holon diagnosers will, of course, be empty. It is clear that $z_0 = \bigcup_{j=1}^{d} z_0^{DG_j}$. Now, assume that: $z_n = \bigcup_{j=1}^{d} z_{\sigma_n}^{DG_j}$. We prove that: $z_{n+1} = \bigcup_{j=1}^{d} z_{\sigma_{n+1}}^{DG_j}$. For this, we show that: $x \in z_{n+1} \Leftrightarrow x \in \bigcup_{j=1}^{d} z_{\sigma_{n+1}}^{DG_j}$ for every $x \in X'$.

Let $x \in z_{n+1}$ be a state of DG. We can conclude that there exist $x' \in z_n$ such that $x \in X'$ and x is observation-adjacent to x' with respect to σ_{n+1} ($x' \overset{\sigma_{n+1}}{\Rightarrow} x$). Thus, $x' \in z_n = \bigcup_{j=1}^{d} z_{\sigma_n}^{DG_j}$. Suppose that $x' \in z_{\sigma_n}^{DG_k}$ for some $k \in \{1,\ldots,d\}$. Since x is observation-adjacent to x', we can say that after observing σ_{n+1}, x either belongs to $z_{(\sigma_{n+1})}^{DG_k}$ or to the state estimate of the father of DG_k or to the state estimate of one of the children of DG_k. Therefore, $x \in \bigcup_{j=1}^{d} z_{\sigma_{n+1}}^{DG_j}$.

Now, assume that $x \in \bigcup_{j=1}^{d} z_{\sigma_{n+1}}^{DG_j}$. Suppose that $x \in z_{\sigma_{n+1}}^{DG_m}$ for some $m \in \{1,\ldots,d\}$. Therefore, there exist x' such that $x' \overset{\sigma_{n+1}}{\Rightarrow} x$ and x' either belongs to $z_{\sigma_n}^{DG_m}$ or to the state estimate of the father of DG_m or to the state estimate of one of the children of DG_m. This implies that $x' \in \bigcup_{j=1}^{d} z_{\sigma_n}^{DG_j}$. From the induction assumption, we can conclude that $x' \in z_n$. Since x is observation-adjacent to x' with respect to σ_{n+1}, it can be inferred that $x \in z_{n+1}$.

□

It follows from the theorem that the condition of the system is:

$$\kappa(z_n) = \kappa\left(\bigcup_{j=1}^{d} z_n^{DG_j}\right) = \bigcup_{j=1}^{d} \kappa(z_n^{DG_j}).$$

Remark 4.3: A D-holon may contain faulty states while the corresponding failure event could have appeared in another D-holon. For instance, in Fig. 4.12, the failure event *f* occurs in super state *S*1. No failure event occurs in *S2* but S2 contains faulty states. □

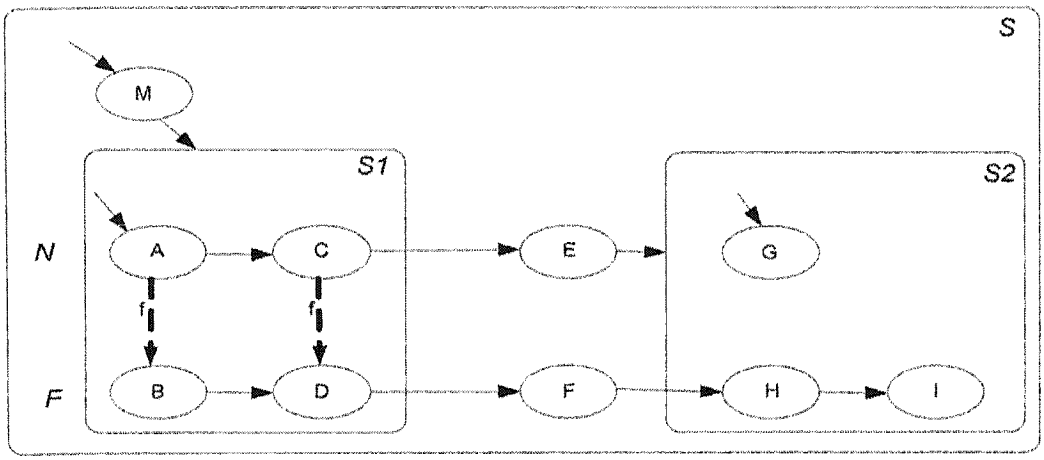


Figure 4.12: Example of a super state containing faulty states corresponding to a failure event occurred previously in another super state.

In our approach, it is possible that several diagnosers be active at the same time in the system. As a result, the state estimate of an active diagnoser may be affected by other diagnosers. In other words, diagnosers in the system have interaction among each other and are "**coupled**". A diagnoser in a standard HFSM only may have interaction with its father or (and) its children during the diagnosis process. Therefore, if a diagnoser is not active simultaneously with its father or any of its children, then it never interacts unless when it invokes its father or children after which it becomes inactive. This could happen if at any given instant, all active diagnosers are in the same level. A set of sufficient conditions for this is that all states in the initial state estimate z_0 belong to the same level

and each level of the hierarchy in the system has a unique event set. To clarify this, we first define the event set of a hierarchy level.

Definition 4.8: The event set of a hierarchy level L is defined to be the set of the events labelling the transitions among states of the D-holons at level L and transitions from states at levels L+1 (one level down) and L-1 (one level up) to states at level L if $L \neq 0$, and transitions from states at level L+1 to states at level L if $L = 0$. □

In a system with levels $0,\ldots,m$, if event sets of the levels are disjoint, then the event set of the system can be partitioned according to the level: $\Sigma = \Sigma_0 \dot{\cup} \Sigma_1 \dot{\cup} \cdots \dot{\cup} \Sigma_m$, where $\Sigma_i (0 \leq i \leq m)$ denotes the event set of level i.

Usually, in real systems, different levels have different physical components and events labelling the transitions among the components are different in different levels. Thus, assuming disjoint level event sets in a hierarchical system could be practically reasonable.

Proposition 4.1: Assuming disjoint level sets in the system and that the states in z_0 belong to the same level, at any given instant, all active diagnosers are from the same level. □

Note that it is possible to have multiple boundary transitions with the same label event in a D-holon resulting in simultaneous activation of multiple children diagnosers. Since, diagnosers only affect the state estimate of their father or children (belonging to different levels), diagnosers of the same level have no interaction among each other. Thus,

diagnosers can be decoupled while working simultaneously provided that they belong to the same level.

In systems with decoupled diagnosers, if a boundary event of a D-holon is observed, the diagnoser of the D-holon becomes inactive and the diagnoser of the D-holon(s) containing the target states as their internal states becomes active. In these cases, the D-holon diagnoser becomes a FSM. This is illustrated in the following example.

Example 4.2: Neutralization Process (Contd.)

The RTS of each D-holon is given in table 4.2. In Fig. 4.13, the high level diagnoser is shown. Fig. 4.14 illustrates the low level diagnosers. Since the D-holon diagnosers have no interaction among each other during their operation, they can be represented by FSM models.

It is assumed that the diagnosis process starts simultaneously with the system (i.e., $z_0 = \{Ini\}$). Therefore, in D-holon diagnosers, the initial state of the diagnosers is a subset of the D-holon's initial state set. When the diagnosing process initializes, the high level diagnoser ($DG_{NeuProc}$) becomes active with the initial state $z_0^{DG_{NeuProc}} = \{Ini\}$. In Fig. 4.13, we have shown five different diagnoser models each corresponding to different initial state sets. Note that the initial state z_0 may be different each time the diagnoser becomes active during the cycle of the system operation. Figure 4.15 demonstrates the activation sequence of D-holon diagnosers in the system.

D-Holon	State	Event (σ)	Observation Adjacent States w.r.t . σ	State	Event (σ)	Observation Adjacent States w.r.t . σ
$DH_{NeuProc}$	Ini	add-a	1N	AF	add-b	7F
	AN	add-b	7N	BN	drain	12N
$DH_{Add\text{-}Acid}$	1N	O1	2N	4N	L12	6N,6F
	2N	L01	3N,3F	4F	L12	6F
	2N	n2a	5N,5F	5N	L01	4N,4F
	2F	L01	3F	5F	L01	4F
	2F	n2a	5F	6N	a-added	AN,AF
	3N	n2a	4N,4F	6F	a-added	AF
	3F	n2a	4F			
$DH_{Add\text{-}Base}$	7N	O2	9N,9F	9N	C1	10N,10F
	7N	C1	8N,8F	9F	C2	10F
	7F	O2	9F	10N	a2n	11N
	7F	C1	8F	10F	L23	18F
	8N	O2	10N	11N	b-added	BN
	8F	O2	10F			
$DH_{Draining}$	12N	C2	13N	15N	L10	16N
	13N	O3	14N	16N	C3	17N
	14N	L21	15N	17N	T-empty	Ini

Table 4.2: Example 4.2: RTS's of the D-holons of the neutralization process.

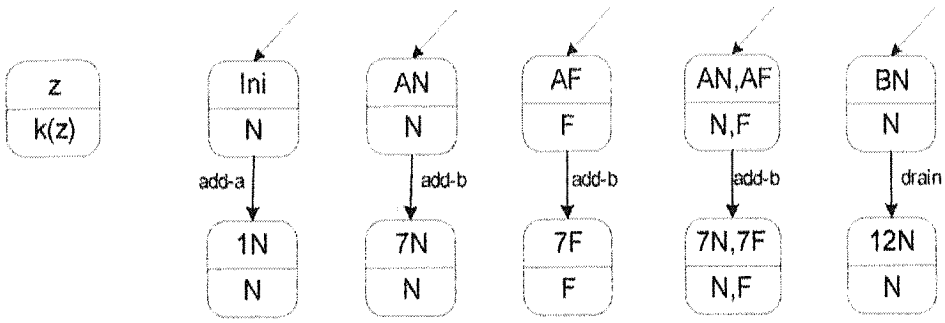


Figure 4.13: High-level diagnoser ($DG_{NeuProc}$)

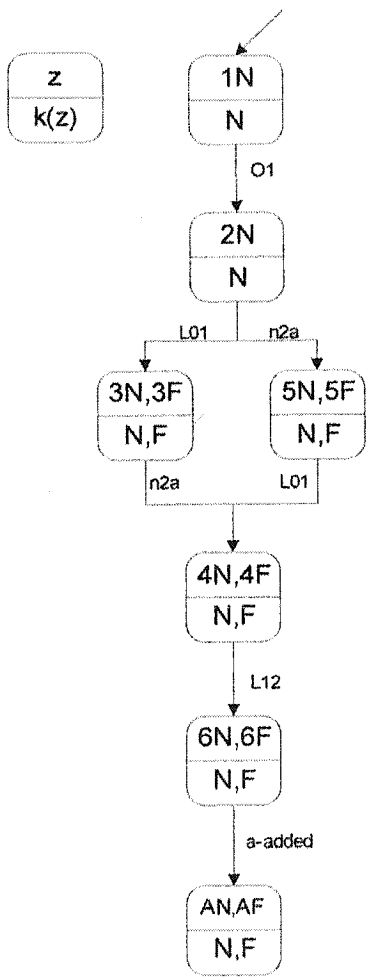


Figure 4.14.a) $DG_{Add\text{-}Acid}$

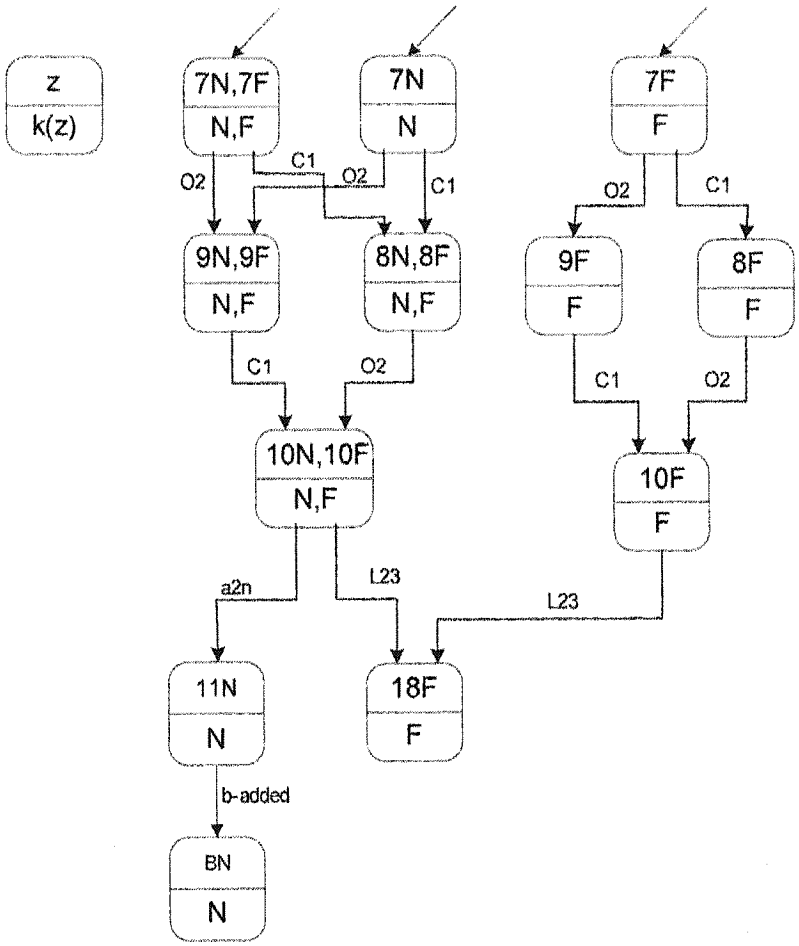
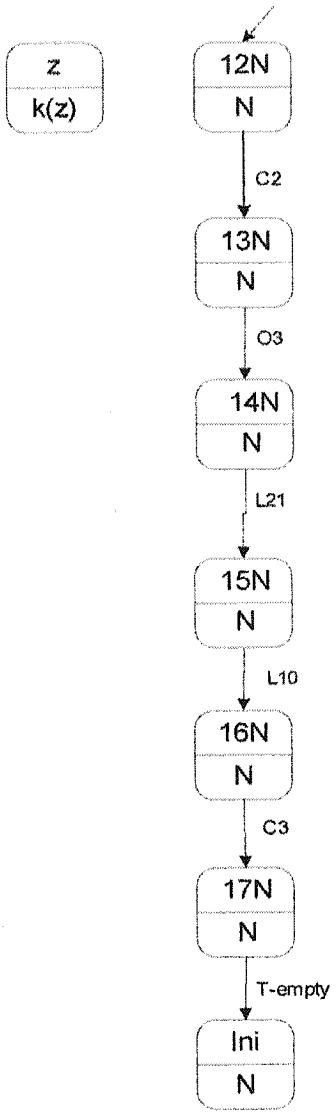


Figure 4.14.b) $DG_{Add\text{-}base}$



c) $DG_{Draining}$

Figure 4.14: Low-level diagnosers

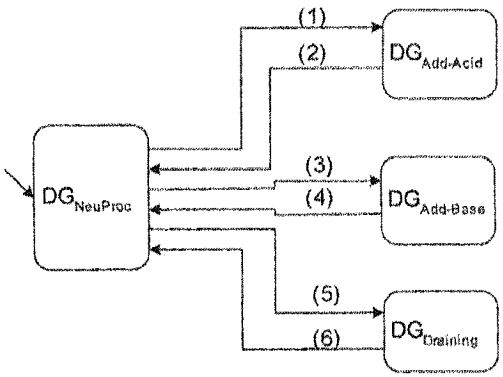


Figure 4.15: Diagram of the activation sequence of diagnosers in Example 4.1

In Fig. 4.15, each box represents a D-holon diagnoser. Initially $DG_{NeuProc}$ becomes active in the system with $z_0 = \{Ini\}$. After the occurrence of the boundary event 'add-a' $DG_{Add\text{-}Acid}$ becomes active with initial state estimate $z_0 = \{1N\}$ and $DG_{NeuProc}$ becomes inactive (transition (1) in Fig. 4.15). The activation sequence of diagnosers can be followed according to the numbers shown on the diagram. After step (6), the sequence starts from (1). Since, at each time only one diagnoser is active in the system, the system's state estimate is equal to the state estimate provided by the active diagnoser. □

It should be noted that the Neutralization Process is used as a simple illustrative example. *Add-Acid*, *Add-Base* and *Draining* are too simple to be considered as separate phases in real systems. The whole process might be considered as a phase in a large complex system.

Remark 4.4: In our framework, each D-holon diagnoser can be automatically translated into computer code. Under operating systems supporting multithreading, the computer code of each diagnoser can be put in a thread. Hence, several diagnosers can be in operation concurrently. The efficiency of our method depends on the amount of

communication among diagnosers. Frequent communication among the diagnosers is undesirable. As pointed out earlier, if the event sets of the levels of the system hierarchy are disjoint and all states in z_0 are at the same level, then the diagnoser will be decoupled and there will be no communications among diagnosers except when a diagnoser invokes another and becomes inactive itself. □

Remark 4.5: Assume all D-holon diagnosers are decoupled in the sense defined earlier. In general, a D-holon diagnoser may become activated with various initial states. This leads to several transition graphs for the diagnoser, each graph corresponding to an initial state. On the other hand, all the information required for computing the diagnosers in the system can be obtained from the RTS of the D-holons. Hence, in our approach, it is more convenient to compute the RTS of D-holons and performing diagnostic computations on-line (as opposed to computing several diagnosers off-line and storing them in computer memory). However, in simple systems with small number of states and decoupled diagnosers, the graphs of the diagnosers may be computed off-line and stored in computer for diagnostic purposes (as in Example 4.2). □

4.3 Diagnosability Using HFSM Structure

In this section we would like to take advantage of the HFSM framework and D-holon model to study the problem of diagnosability in a hierarchical DES.

Consider a component, say a valve, in a system. Suppose that this valve fails stuck-closed while it is closed and no open commands are sent to the valve after the failure. In this case, the diagnoser of the equivalent flat system would not be able to detect the

failure and according to the definition of diagnosability in Chapter 3 the failure is not diagnosable. Since the stuck-closed failure has no effect on the performance of the system, one may ignore the undiagnosability of the failure. However, if the valve is going to be used in another cycle of the operation of the system, ignoring the undiagnosability of the respective failure may cause problem.

The notion of I-diagnosability, introduced in [SSL95], is used to address the above problem. According to [SSL95], the valve failure should be considered undiagnosable only if it cannot be diagnosed following an open command. Otherwise the failure is considered I-diagnosable. Now, suppose that the valve fails stuck-closed while it is open and no commands are issued afterwards. According to the notion of I-diagnosability, the failure can still be considered (I-)diagnosable even if it cannot be detected by the diagnoser. This does not seem to be suitable in practical systems.

The main issue here is that the definition of diagnosability (in Chapter 3) requires that the failure be detected and isolated under all circumstances. However, in cases, where a component becomes faulty while it is not active, it may be too restrictive to expect diagnosability. Therefore, diagnosability requirement should be limited to the phases of operation when the component is active. This leads us to the concept of "phase-diagnosability". The framework proposed in this chapter and in particular, the concept of D-holon, provides a suitable setup for developing the notion of "phase-diagnosability".

In the following, we explain the notion of phase-diagnosability of a failure mode. We assume that the level event sets in the hierarchy are disjoint and therefore, the D-holon diagnosers in the hierarchical system are decoupled. This implies that D-holon diagnosers can be modeled as FSMs. We also assume that we always know which D-holon the

current state of the system is in. In other words, for any $n \geq 0$, there exist $i \in \{1,\ldots,d\}$ such that $z_n \subseteq X_i^{DH_i}$. This can be assured if the initial state estimate z_0 satisfies $z_0 \subseteq X_j^{DH_j}$ for some $j \in \{1,\ldots,d\}$ and for all D-holons, all external transitions with identical label (event) lead to a unique D-holon. These assumptions are not necessarily very limiting. Because the D-holons used in the study of diagnosability describe a complete phase of operation (details will be given later) and we usually know exactly when a phase of operation starts. We also assume that failure modes are permanent. For simplicity, we concentrate on the single failure scenario. The approach can be extended to simultaneous occurrence of failures following a method similar to that explained in chapter 3 for flat DES.

A permanent failure mode F_i is **phase-diagnosable** with respect to a D-holon if F_i can always be detected and isolated before the system leaves the D-holon.

Definition 4.9: A permanent failure F_i is phase-diagnosable with respect to a D-holon if it can be detected and isolated by the diagnoser of the D-holon (with $z_0 = X_0^{DH}$) after the occurrence of at most a bounded number of events following both the occurrence of the failure and initialization of the diagnoser. □

Note that the definition implies that the failure must be diagnosed before the system leaves the D-holon. In general, the initial state of the diagnoser, z_0, is a subset of X_0^{DH} and therefore, the condition $z_0 = X_0^{DH}$ may seem restrictive. In practice, however, the initial state or states of a phase of operation are generally few and the start of each phase

follows an observable event. Thus, $z_0 = X_0^{DH}$ typically holds and is a reasonable condition.

Theorem 4.2: Consider a D-holon *DH* associated with a super state of an HFSM *H*. Assume that there are no cycles of unobservable events or deadlocks in *DH* (states belong to the external state set of *DH* are not considered as deadlocks). Suppose single failure scenario in the system. Let *DG* denote the diagnoser designed for *DH*. A permanent failure F_i is phase-diagnosable with respect to *DH* if and only if:

1- There are no F_i-indeterminate cycles in *DG*.

2- Every diagnoser state, z^{DG}, containing external states ($z^{DG} \subseteq X_E^{DH}$) is not F_i-uncertain.

Proof: Suppose that DG is initialized with z_0^{DG} and no boundary event occurs in the system. In this case, diagnosability of a failure mode in a D-holon is similar to that of a flat DES, because the internal structure of a D-holon is the same as that of a flat DES and only *DG* is active in the system. Therefore, condition (1) is necessary and sufficient condition for diagnosability according to the theorem 3.1. Now, assume that a boundary event occurs. Since diagnosers are assumed to be decoupled, following this event, *DG* becomes inactive and either its father or one (or some) of its children becomes active. It means that F_i is diagnosable if and only if the last state estimate provided by *DG* which will be a subset of X_E^{DH} is not F_i-uncertain. □

Sometimes for a given component, the active phase of operation is described by a set of super states (and the associated D-holons). In these cases, for the purpose of studying

phase-diagnosability, we should first merge the super states and then study diagnosability with respect to the D-holon associated with the new (larger) super state.

Phase-diagnosability is a flexible definition and may allow the accommodation of operational assumptions in the study of diagnosability. For example, Fig. 4.16 shows a system consisting of two phases (super states) S1 and S2.

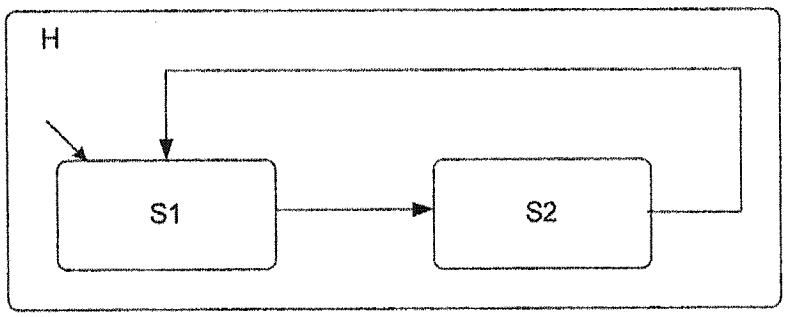


Figure 4.16: A system consisting of two phases (super states)

Suppose that we would like a failure mode F to be diagnosable if the system enters phase S2. This means that as long as the system is in S1, we do not expect to detect and isolate F. To examine diagnosability, we can construct the system in Fig. 4.17 which behaves similar to the original system. Then, we study local diagnosability of the failure with respect to the D-holon associated with H1.

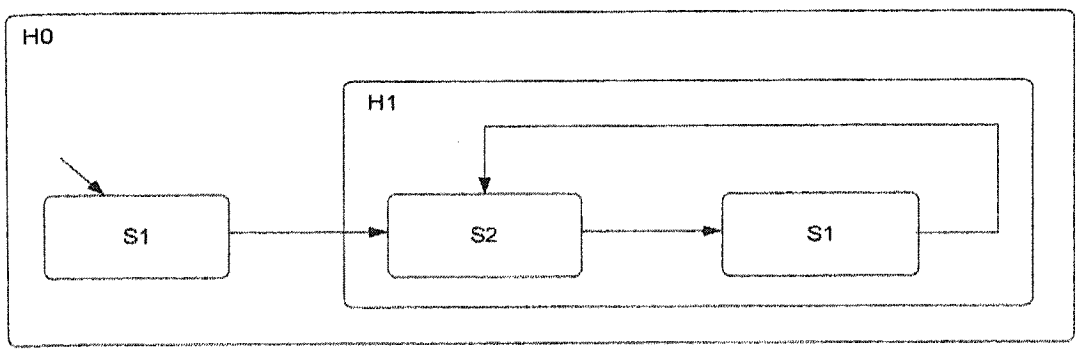


Figure 4.17: Modified system for diagnosability analysis

So far, we have assumed that in the HFSM all AND-states have been replaced by OR equivalent states. In the next chapter, we examine the cases in which the internal transition of AND-states can be expressed using the transition function of the synchronous product of the components of the AND-state. We will show that under certain conditions, the computational complexity of constructing the required RTS and the requirements on computer memory for storing the RTS can be reduced significantly.

Chapter 5

Semi-Modular Fault Diagnosis in Modular Systems and AND D-holons

In the previous chapter, we assumed that AND super-states of the system were substituted with the synchronous product of their components. In other words, in order to apply the diagnosing process discussed in chapter 4, the system with AND super states was converted to its equivalent HDES containing only OR-states. Typically, the number of states of the synchronous product of several FSMs is much larger than the sum of the number of states of the individual FSMs. Therefore, exploiting approaches that, for diagnosis purposes, use the models of individual components, rather than the synchronous product, is usually very useful. This is particularly important where the number of components in the system is very large.

In this chapter, we propose a semi-modular approach in which (if the interactions among components are observable) the updating of the system estimate in the diagnoser can be done using only the RTS of individual components. Therefore, instead of storing the RTS of the combined system in the memory, the RTS of individual components may

be stored and used for diagnosis computations. We also will show that the complexity of computing the required RTS in our approach is polynomial in the number of components, while the complexity of calculating the RTS of the whole system in the conventional method is exponential in the number of components. Furthermore, the sum of sizes of the RTSs in our semi-modular approach will be polynomial in the number of components, while in the conventional, monolithic approach it is exponential. We call our approach semi-modular because we do not design individual diagnosers for each component. The diagnoser that we construct for the total system, however, uses the information of the RTS of individual components to update and compute the state estimate.

In section 5.1, we develop our semi-modular approach for a flat DES which can be represented by the synchronous products of several generators. In section 5.2, we explain how the results in section 5.1 can be applied to D-holon diagnosers in an HFSM. The complexity issues will be discussed in section 5.3.

5.1 Semi-Modular Diagnosis in Flat DES

In this section, we develop a semi-modular approach for diagnoser design in a flat DES which can be represented by the synchronous product of a number of generators.

The plant that we study in this section is a modular finite state machine $H = (X, \Sigma, \delta, x_0)$, where Σ and X are the event and finite state sets respectively; $\delta : X \times \Sigma \to X$ represents the transition function and x_0 denotes the initial state of the system. H consists of $k (k \geq 2)$ components. Each component can also be modeled as a generator $H^i = (X^i, \Sigma^i, \delta^i, x_0^i)$ $(i \in \{1, \ldots, k\})$, where Σ^i and X^i are the event and finite

state sets of the component H^i; x_0^i denotes the initial state of the module H^i and $\delta^i : X^i \times \Sigma^i \to X^i$ is the transition function of the component H^i.

The model of the whole system H can be obtained using the synchronous product of the components. Hence, H can be represented as: $H = Rch(H^1 \| \cdots \| H^k)$, where '||' denotes the synchronous product operator. Each state of H is represented as a k-tuple $x = (x_1, \ldots, x_k)$, where $x_i \in X^i$. The event set and the initial state of H can be calculated as: $\Sigma = \Sigma^1 \cup \cdots \cup \Sigma^k$ and $x_0 = (x_0^1, \ldots, x_0^k)$.

The event set of each component Σ^i can be partitioned into two disjoint subsets Σ_o^i and Σ_{uo}^i ($\Sigma^i = \Sigma_o^i \dot{\cup} \Sigma_{uo}^i$), where Σ_o^i is the set of observable events and Σ_{uo}^i the set of unobservable events in the module H^i.

The event set Σ^i includes the failure events in component H^i (assuming H^i has some failure modes). It can also be partitioned into two disjoint sets Σ_f^i and Σ_{nf}^i ($\Sigma^i = \Sigma_f^i \dot{\cup} \Sigma_{nf}^i$), where Σ_f^i denotes the set of failure events and Σ_{nf}^i represents the set of non-failure events in H^i. It is assumed that $\Sigma_f^i \subseteq \Sigma_{uo}^i$ for any $(i \in \{1, \ldots, k\})$.

H describes the behaviour of the system in both normal and faulty situations. Suppose that there are P failure modes in H: $F_1, F_2, \ldots, F_P$. Let $\mathrm{K} := \{N, F_1, F_2, \ldots, F_p\}$ be the condition set of the system. It is assumed that the state set X can be partitioned according to the condition: $X = X_N \dot{\cup} X_{F_1} \dot{\cup} \cdots \dot{\cup} X_{F_p}$. $\kappa : X \to \mathrm{K}$ denotes the condition map of H.

Similarly, it is assumed that the behaviour of each component can be described in both normal and faulty situations. Considering P_i failure modes in H^i: $F_1^i, F_2^i, \ldots, F_{P_i}^i$ and

$\mathrm{K}^i := \{N, F_1^i, F_2^i, \ldots, F_{p_i}^i\}$ as the condition set of H^i, the state set X^i of each module can be partitioned according to the condition of that module: $X^i = X_N^i \dot{\cup} X_{F_1}^i \dot{\cup} \cdots \dot{\cup} X_{F_{P_i}}^i$. $\kappa^i : X^i \to \mathrm{K}^i$ denotes the condition map of H^i.

First let us assume there are two components ($k = 2$). Generalization to arbitrary number of components will be discussed later.

The details of design were explained in chapter 3. Here, we quickly review the notation.

Let $DG = (Z, \Sigma, \xi, z_0, \tilde{\mathrm{K}}, \kappa)$ be the diagnoser designed for H. Z, Σ and $\tilde{\mathrm{K}} \subseteq 2^{\mathrm{K}} - \{\phi\}$ are the finite state, event and output sets of DG. $z_0 \subseteq 2^X - \{\phi\}$ is the initial state; $Z \subseteq 2^X - \{\phi\}$; $\xi : Z \times \Sigma \to Z$ represents the transition function; $\kappa : Z \to \tilde{\mathrm{K}}$ denotes the output map. Each diagnoser state z is identified with a non-empty subset of X. $z_n \subseteq Z$ is the state estimate of DG after the occurrence of the observable event σ_n.

The diagnoser state transition $z_{n+1} = \xi(z_n, \sigma_{n+1})$ is given by:

- $z_{n+1} = \{(x_1, x_2) \mid \exists x_1', x_2' : (x_1', x_2') \in z_n \text{ and } (x_1', x_2') \overset{\sigma_{n+1}}{\Rightarrow} (x_1, x_2)\}$ if $\sigma_{n+1} \in \Sigma^1 \cap \Sigma^2$
- $z_{n+1} = \{(x_1, x_2) \mid \exists x_1' : (x_1', x_2) \in z_n \text{ and } (x_1', x_2) \overset{\sigma_{n+1}}{\Rightarrow} (x_1, x_2)\}$ if $\sigma_{n+1} \in \Sigma^1 - \Sigma^2$
- $z_{n+1} = \{(x_1, x_2) \mid \exists x_2' : (x_1, x_2') \in z_n \text{ and } (x_1, x_2') \overset{\sigma_{n+1}}{\Rightarrow} (x_1, x_2)\}$ if $\sigma_{n+1} \in \Sigma^2 - \Sigma^1$

Proposition 5.1: Assume $\Sigma_{uo}^1 \cap \Sigma_{uo}^2 = \phi$, then:

1. For $\sigma \in \Sigma^1 \cap \Sigma^2$: $(x_1, x_2) \overset{\sigma}{\Rightarrow} (x_1', x_2')$ if and only if $x_1 \overset{\sigma}{\Rightarrow} x_1'$ and $x_2 \overset{\sigma}{\Rightarrow} x_2'$.
2. For $\sigma \in \Sigma^1 - \Sigma^2$: $(x_1, x_2) \overset{\sigma}{\Rightarrow} (x_1', x_2)$ if and only if $x_1 \overset{\sigma}{\Rightarrow} x_1'$.

3. For $\sigma \in \Sigma^2 - \Sigma^1$: $(x_1, x_2) \overset{\sigma}{\Rightarrow} (x_1, x_2')$ if and only if $x_2 \overset{\sigma}{\Rightarrow} x_2'$.

Proof: We prove part (1). The proof of (2) and (3) will be similar and are omitted for brevity.

Suppose that $(x_1, x_2) \overset{\sigma}{\Rightarrow} (x_1', x_2')$. This means that there exists a sequence of events $s = \sigma_1 \sigma_2 \cdots \sigma_m \sigma$ with $\sigma_1, \sigma_2, \ldots, \sigma_m \in \Sigma_{uo}$ and $\sigma \in \Sigma_o$ such that $\delta((x_1, x_2), s) = (x_1', x_2')$. This implies that $\delta(x_1, P_1 s) = x_1'$ and $\delta(x_2, P_2 s) = x_2'$, where $P_1 : (\Sigma^1 \cup \Sigma^2)^* \to \Sigma^{1*}$ and $P_2 : (\Sigma^1 \cup \Sigma^2)^* \to \Sigma^{2*}$ are the natural projections on to Σ^{1*} and Σ^{2*}, respectively. Therefore, $x_1 \overset{\sigma}{\Rightarrow} x_1'$ and $x_2 \overset{\sigma}{\Rightarrow} x_2'$.

Now, assume that $x_1 \overset{\sigma}{\Rightarrow} x_1'$ and $x_2 \overset{\sigma}{\Rightarrow} x_2'$. Then, there are sequences $s_1 = \sigma_1 \sigma_2 \cdots \sigma_l \sigma$ and $s_2 = \sigma_1' \sigma_2' \cdots \sigma_k' \sigma$ with $\sigma_1, \sigma_2, \ldots, \sigma_l \in \Sigma^1_{uo}, \sigma_1', \sigma_2', \ldots, \sigma_k' \in \Sigma^2_{uo}$ and $\sigma \in \Sigma_o$ such that $\delta(x_1, s_1) = x_1'$ and $\delta(x_2, s_2) = x_2'$. Since modules have no common unobservable events, the sequence, say $s = \sigma_1 \sigma_2 \cdots \sigma_l \sigma_1' \sigma_2' \cdots \sigma_k' \sigma$, can take H from state (x_1, x_2) to (x_1', x_2'). Therefore, $(x_1, x_2) \overset{\sigma}{\Rightarrow} (x_1', x_2')$. □

Assume $\Sigma^1_{uo} \cap \Sigma^2_{uo} = \phi$. Then using proposition 5.1, the state transition function of *DG* can be written as:

$$z_{n+1} = \{(x_1, x_2) \mid \exists x_1', x_2' : (x_1', x_2') \in z_n \text{ and } x_1' \overset{\sigma_{n+1}}{\Rightarrow} x_1 \;\&\; x_2' \overset{\sigma_{n+1}}{\Rightarrow} x_2\} \text{ if } \sigma_{n+1} \in \Sigma^1 \cap \Sigma^2$$

$$z_{n+1} = \{(x_1, x_2) \mid \exists x_1' : (x_1', x_2) \in z_n \text{ and } x_1' \overset{\sigma_{n+1}}{\Rightarrow} x_1\} \text{ if } \sigma_{n+1} \in \Sigma^1 - \Sigma^2$$

$$z_{n+1} = \{(x_1, x_2) \mid \exists x_2' : (x_1, x_2') \in z_n \text{ and } x_2' \overset{\sigma_{n+1}}{\Rightarrow} x_2\} \text{ if } \sigma_{n+1} \in \Sigma^2 - \Sigma^1$$

This means that to update z_n, instead of using the RTS of $H_1 \| H_2$, we can use the RTS's of H^1 and H^2 individually provided that they have no common unobservable events. In systems with more than two modules, the assumption on observable events should be: $\Sigma_{uo}^i \cap \Sigma_{uo}^j = \phi$ for all i, j with $i \neq j$. In other words, unobservable events in one component do not appear in the models of other components. This assumption could be true in systems where the interactions among components are observable.

Failure events are assumed to be unobservable in the system. Therefore, the assumption $\Sigma_{uo}^i \cap \Sigma_{uo}^j = \phi$ $(i \neq j)$ implies that the components have no common failure mode.

Being able to use individual RTS's in place of the RTS of the entire system could reduce computer memory requirements. If each component H^i has p states, the RTS of H, in the worst case, will have p^k states, whereas the sum of the states of individual RTS's in the worst case will be kp. In addition, the (time) complexity of computing the RTS of the system in the worst case will be $O(n^2 k)$ [3], while the complexity of computing individual RTS's in the worst case will be $O(nk^2)$. Therefore, this method reduces both the time complexity of computing the RTS and the memory requirement for on-line implementation of the diagnosis algorithms.

Example 5.1

Figure 5.1 shows a modular system S consisting of two components S1 and S2. There are two failure modes in the system: F1 and F2. The failure mode F1 is the consequence of

[3] Here, we have assumed the order of the number of transitions in the system is linear in the system states p^k.

the occurrence of faulty event f1 in S1. Similarly, the failure mode F2 in S2 is the result of failure event f2. The events 'e' and 'g' are assumed to be unobservable. Unobservable events are shown by dashed lines. Modules S1 and S2 have no common unobservable events. Therefore, for updating the state estimate of the system (used in on-line implementation of the diagnoser), we only need to store the RTS's of the individual components S1 and S2. Therefore, there is no need to store the RTS of S.

The modules' RTSs are given in Table 5.1. The diagnoser of S is shown in Fig. 5.2. It is assumed that the system and the diagnoser are initialized simultaneously.

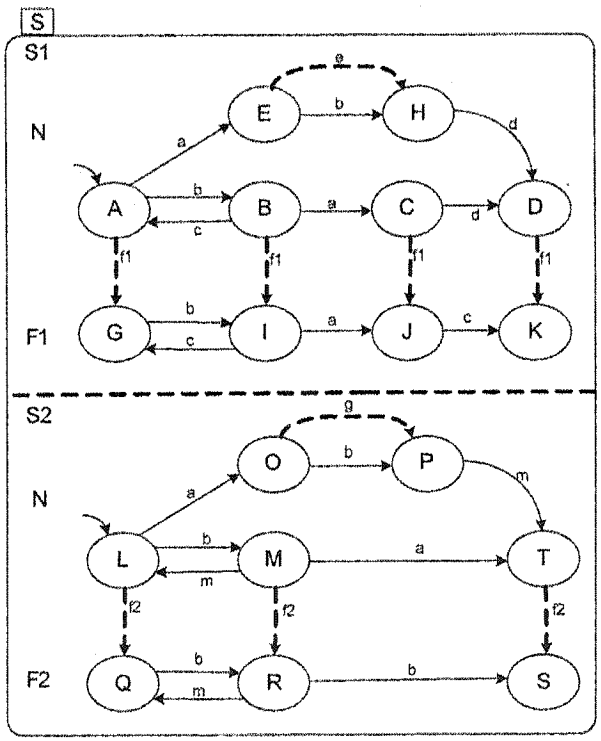


Figure 5.1: The modular system of example 5.1

Module	State	Event (σ_{n+1})	Observation Adjacent States w.r.t . σ_{n+1}	State	Event (σ_{n+1})	Observation Adjacent States w.r.t . σ_{n+1}
S1	A	a	E	E	d	D
	A	b	B,I	H	d	D
	B	a	C,J	G	b	I
	B	c	A,G	I	a	J
	C	c	K	I	c	G
	C	d	D	J	c	K
	E	b	H			
S2	L	a	O	O	m	T
	L	b	M,R	P	m	T
	M	a	T	Q	b	R
	M	b	S	R	b	S
	M	m	L,Q	R	m	Q
	O	b	P			

Table 5.1: Example 5.1: RTS of the system

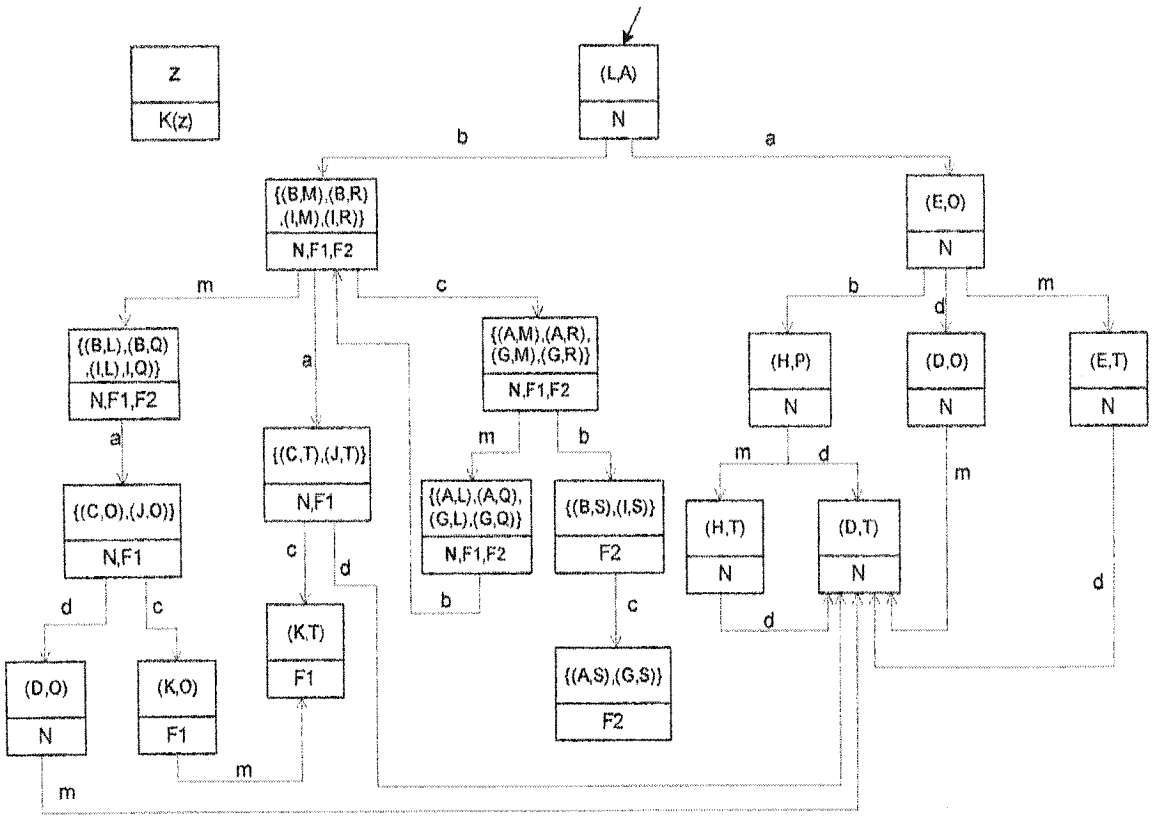


Figure 5.2: The diagnoser designed for the system in Example 5.1

It should be noted that S1 has 10 states and 16 transitions, and S2 has 8 states and 13 transitions. The synchronous product of S1 and S2 has 44 states and 91 transitions. This shows that the diagnosis system using the RTS of the synchronous products of S1 and S2 requires larger amount of memory to store the RTS of *sync*(S1,S2) in comparison with a diagnoser which uses individual RTSs of S1 and S2. Real world systems, typically, have a larger number of components and as a result, the size of the synchronous products of the components would become very larger than the sum of the sizes of individual components.

Remark 5.1: If $\Sigma^i \cap \Sigma^j = \phi$ for all i, j with $i \neq j$, then the system components do not have any interaction. In this case, instead of a monolithic diagnoser, we can build k diagnoser modules (one for each component) and z_n, the system state estimate, can be calculated as: $z_n = z_n^1 \times z_n^2 \times \cdots \times z_n^k$, where z_n^i is the state of the diagnoser module DG^i designed for the module H^i. In this case, diagnosis is completely modular. □

Proposition 5.2: If z_0 is of the form $z_0 = z_0^1 \times z_0^2 \times \cdots \times z_0^k$ ($\Sigma_{uo}^i \cap \Sigma_{uo}^j = \phi$ for all i, j with $i \neq j$), where $z_0^i \subseteq X^i$ then $z_n = z_n^1 \times z_n^2 \times \cdots \times z_n^k$, where z_n^i is the state of the diagnoser module DG^i designed for the module H^i with z_0^i as its initial state.

Proof: The Proof can be simply done by induction. □

Note that Proposition 5.2 provides a set of sufficient conditions for fully modular diagnosis.

As an example, consider Example 5.1. It was assumed that the diagnoser is initialized at the same time with the system. Therefore, $z_0 = \{(x_0^1, x_0^2)\} = \{(L, A)\}$ $= \{L\} \times \{A\} = \{x_0^1\} \times \{x_0^2\}$ and one could design diagnoser modules for S1 and S2 and calculate with $z_0^1 = \{x_0^1\}$ and $z_0^2 = \{x_0^2\}$ in place of the monolithic diagnoser in Fig. 5.2.

5.2 Semi-Modular Diagnosis in HFSM with AND-states

Our approach for semi-modular fault diagnosis can be applied to the AND super-states of a hierarchical FSM. In these systems, a D-holon describing the transition structure of the component within the hierarchical system is associated with the AND component. For

simplicity, we consider an AND super state consisting of two components. Extension of the results to AND super states with multiple components will be straight forward.

Let S be an AND super state having two components: S^i $i \in \{1,2\}$. Suppose that $DH_i := (X^i, \Sigma^i, \delta^i, X_0^i)$ is the associated D-holon with S^i, where Σ^i, X^i and X_0^i represent the event, the finite state and the initial state sets of DH_i respectively; $\delta^i : X^i \times \Sigma^i \to X^i$ denotes the transition function. The transition among the states of S follows the transition law of the synchronous product.

Let DG_S denote the diagnoser designed for S. Based on Proposition 5.1, if $S1$ and $S2$ have no common unobservable events, we can use the RTS's of $S1$ and $S2$, in place of that of S, for updating the state estimates of S. Specifically, after the occurrence of the observable event σ_{n+1} in the system, one can use the RTS of the individual modules to obtain $\tilde{z}_{n+1}$ as follows.

$$\tilde{z}_{n+1}^{DG_S} = \{(x_1, x_2) \mid \exists x_1', x_2' : (x_1', x_2') \in z_n \text{ and } x_1' \overset{\sigma_{n+1}}{\Rightarrow} x_1 \;\&\; x_2' \overset{\sigma_{n+1}}{\Rightarrow} x_2\} \text{ if } \sigma_{n+1} \in \Sigma^1 \cap \Sigma^2$$

$$\tilde{z}_{n+1}^{DG_S} = \{(x_1, x_2) \mid \exists x_1' : (x_1', x_2) \in z_n \text{ and } x_1' \overset{\sigma_{n+1}}{\Rightarrow} x_1\} \text{ if } \sigma_{n+1} \in \Sigma^1 - \Sigma^2$$

$$\tilde{z}_{n+1}^{DG_S} = \{(x_1, x_2) \mid \exists x_2' : (x_1, x_2') \in z_n \text{ and } x_2' \overset{\sigma_{n+1}}{\Rightarrow} x_2\} \text{ if } \sigma_{n+1} \in \Sigma^2 - \Sigma^1$$

$z_{n+1}^{I,DG_S}, z_{n+1}^{E,DG_S}, \tilde{z}_{n+1}^{DG_S}$ and $z_{n+1}^{DG_S}$ are calculated following the procedure discussed in Chapter 4.

It should be noted that if the components of an AND-state include super states as their sub-states, we do not build diagnosers for those super states. However, we use their associated D-holons and construct RTS's for them. In on-line diagnosis computations, the RTS's of these super states would be used (when necessary) for state estimate update.

5.3 Analysis of Computational Complexity

Continuing the discussion of the previous section, here, we compare the computational complexity of calculating the RTS's of the individual components with that of the entire system. We show that the computational complexity of calculating RTS's needed to perform diagnosis in our semi-modular approach is polynomial in the number of components of the AND-states, while that of the RTS of the entire system is exponential. We assume that the HFSM has an alternating structure form defined in the following.

Definition 5.1 [BH93]: An HFSM is said to have an **alternating structure** if the immediate sub-states of OR-states are either AND-states or basic states, and the immediate sub-states of AND-states are OR states. □

In [BH93], alternating structure was assumed for HFSMs. There, it was shown that alternating structures are generic structures in the sense that every HFSM that does not have an alternating structure can be converted into an equivalent HFSM with an alternating structure. Here, for completeness, we bring in the proof in [BH93].

Theorem 5.1 [BH93]: An HFSM which does not have an alternating structure can be transformed to an equivalent HFSM with an alternating structure.

Proof: This can be accomplished using the transformations shown in Fig. 5.3. □

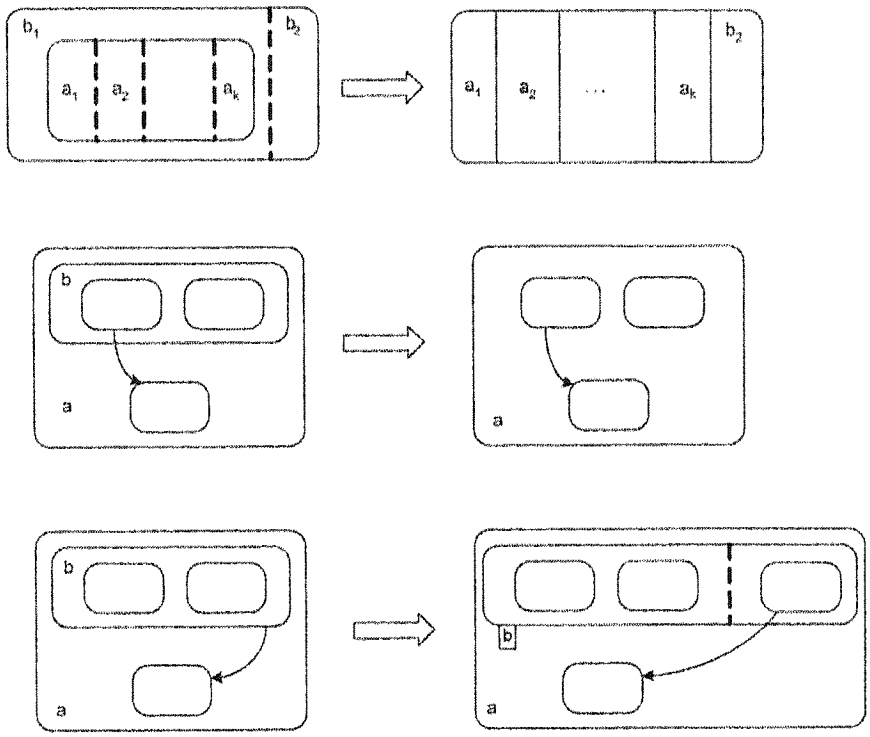


Figure 5.3: Conversions required for the alternating structure

We assume that our system is modeled as an HFSM $M = (A, \Sigma, \Omega, \triangleright, \rho)$ with the canonical form. We also assume that M has an alternating structure with depth of $2m+1$ levels[4], where AND states are at levels $2j$, $0 \le j \le m-1$, OR states are at levels $2j+1$, $0 \le j \le m-1$ and basic states are at levels $2j+1$, $0 \le j \le m-1$ and $2m$. Therefore, $a_{2m} \subseteq A^{Basic}$ and for $0 \le j \le m-1$, $a_{2j} \subseteq A^{\perp}$ and $a_{2j+1} \subseteq A^{Basic} \bigcup A^{+}$ (here a_i denotes states at level i). We also assume that each AND- state has k immediate sub-states, and each OR state contains k immediate AND states and p immediate basic states. We also assume that the number of transitions in each OR-state is linear in the number of its

[4] Similar to condition 4.1, it is assumed that events resulting in change of the level of the system are observable

immediate sub-states, (i.e., $|T^S| = O(k+p), S \in A^+$)[5]. Figure 5.4 illustrates the top level of M. Immediate basic states are denoted by c_j^i. Transitions are not depicted in the graph. Each of AND super states b_j^i has a structure similar to that of M.

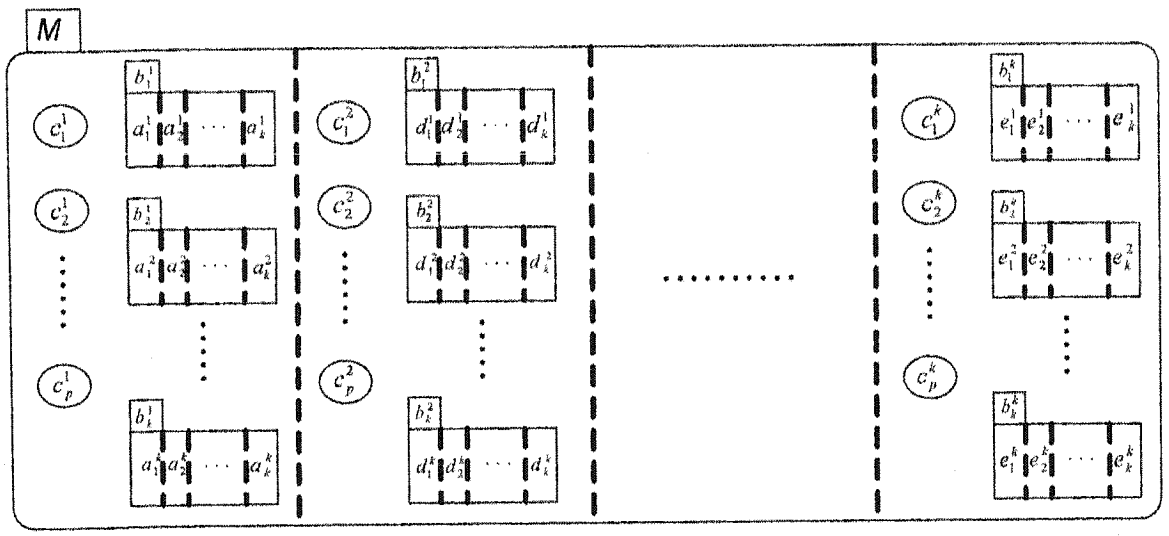


Figure 5.4: The HFSM M

Proposition 5.3: Assume M is a HFSM in the canonical form with an alternating structure. The complexity of calculating the RTS of the D-holons is $O(2nk^2 + 7nkp + 6np^2)$, where $n = k^{2m}$. □

Proposition 5.4: Let M' be the flat equivalent FSM of M with the state set Q. The complexity of calculating the RTS of M' is $O(p^{2\sqrt{n}} k^{2\sqrt{n}/k})$. □

[5] To generalize the results, we may assume $|T^S| = O((k+p)^\alpha)$ with $\alpha \geq 1$. The main result that the computational complexity in the number of components in semi-modular approach is polynomial as opposed to exponential for flat DES, will still be valid.

Comparing the two propositions, one can conclude that the computational complexity of calculating the individual RTS's in our approach is polynomial in n, while the complexity of calculating the RTS of the flat system (i.e., the conventional approach) is exponential in n. n is the order of the total number of components in AND super states (since as will be shown in Lemma 5.1.(i), the number of AND states is of order k^{2m-1}).

Before proving the propositions, we need to establish two important properties of the HFSM M.

Lemma 5.1: Let M be the aforementioned HFSM with parameters k, m and p. Then:

i) The number of AND states is $|A^{\perp}| = \sum_{j=0}^{m-1} k^{2j} = O(k^{2m-1})$.

ii) The number of OR states is $|A^{+}| = \sum_{j=0}^{m-1} k^{2j+1} = O(k^{2m})$.

Proof of lemma 5.1: i) and ii) are inferred from the organization of AND/OR states in the hierarchy. □

Lemma 5.2: Assuming the parameters k, m and p for the HFSM, a lower bound on the size of Q is $p^{\sqrt{n}} k^{\sqrt{n}/k}$, i.e., $|Q| > p^{\sqrt{n}} k^{\sqrt{n}/k}$.

Proof of Lemma 5.2: Let $Q_{S(j)}$ denote the set of the full configurations of the super state S in the level j ($0 \le j \le 2m-1$). $|Q_{S(j)}|$ can be calculated by the following recursive relation:

For each OR state S of level $2m$-1, $|Q_S| = p$ (since each OR state has p immediate basic states). At level $2m$-2, for each AND-state S, $|Q_S| = p^k$ (since each AND state has k

immediate OR states of level $2m$-1). Each OR-state at level $2m$-3 has p immediate basic states and k immediate AND states of level $2m$-2. Thus,

$$\left|Q_{S(2m-3)}\right| = k\left|Q_{S(2m-2)}\right| + p = kp^k + p > kp^k$$

For $j = 2m-4$:

$$\left|Q_{S(2m-4)}\right| = \left|Q_{S(2m-3)}\right|^k = (kp^k + p)^k > k^k p^{k^2}$$

We can see that :

- $\left|Q_{S(j)}\right| = \left|Q_{S(j-1)}\right|^k$ if j is odd.
- $\left|Q_{S(j)}\right| = k\left|Q_{S(j-1)}\right| + p$ if j is even.

Therefore, for levels $2m$-3 to 0, we have (with $3 \le l \le 2m$):

- $\left|Q_{S(2m-l)}\right| > k^{\sum_{i=0}^{\frac{l-1}{2}-1} k^i} p^{k^{\frac{l-1}{2}}}$ if l is odd.
- $\left|Q_{S(2m-l)}\right| > k^{\sum_{i=1}^{\frac{l-2}{2}} k^i} p^{k^{\frac{l}{2}}}$ if l is even.

It is clear that $|Q| = \left|Q_{r(M)}\right|$, where $r(M)$ is the root super-state of M. Since $r(M)$ is an AND-state at level $j = 0$ ($l = 2m$), $|Q|$ can be calculated as:

$$|Q| = \left|Q_{S(2m-l)}\right|_{l=2m} > k^{\sum_{i=1}^{\frac{2m-2}{2}} k^i} p^{k^{\frac{l}{2}}} = k^{\sum_{i=1}^{m-1} k^i} p^{k^m} > k^{k^{m-1}} p^{k^m} = k^{\sqrt{n}/k} p^{\sqrt{n}} \qquad \square$$

Proof of proposition 5.4: As we mentioned in chapter 3, the RTS of a DES can be computed in $O(|X|^2 + |X||T|)$, where $|X|$ and $|T|$ denote the number of states and transitions in the system respectively. We have assumed that $O(|T|) = O(|X|)$; therefore,

the RTS of M' with Q as its state set can be computed in $O(|Q|^2) > O(p^{2\sqrt{n}} k^{2\sqrt{n}/k})$. Therefore, $O(p^{2\sqrt{n}} k^{2\sqrt{n}/k})$ is a lower bound for the complexity of calculating the RTS of M'. □

Proof of proposition 5.3: Assume that DH is a D-holon associated with an OR-state of M. DH has $|X_I| = p$ internal states and $|X_E|$ external states. Since $|X_E| \leq |T^{DH}|$ and that we have assumed that the number of transitions in each OR state is linear in the number of its immediate sub-states, $|X_E| = O(k + p)$. The computation of the DH's RTS can be done in

$$O(\{|X_I| + |X_E|\}^2 + \{|X_I| + |X_E|\}|T^{DH}|) = O((k+2p)^2 + (k+2p)(k+p))$$
$$= O(k^2 + kp + p^2)$$

In the diagnosis process, we only use the RTS of D-holons associated with OR states (diagnosis in an AND-state is achieved using the RTS's of its components). Moreover, according to the Lemma 5.1, the number of D-holons associated with OR states in M is $\sum_{j=0}^{m-1} k^{2j+1} = O(k^{2m}) = O(n)$. Thus, computing the RTS of all D-holons will be fulfilled in $O(nk^2 + nkp + np^2)$. □

In the following we study practical aspect of our framework. We apply the semi-modular method of fault diagnosis to a modular multi-phase ozone generation plant.

5.4 Illustrative Example: Ozone Generation Plant

5.4.1 Overview and Configuration of the Process [Guo02]

Ozone is the second most powerful oxidant in nature. It is widely used in industry particularly for water treatment. It is unstable under atmospheric temperature and pressure and decomposes into oxygen easily. Therefore, it cannot be stored and must be produced on site, just prior to use. Ozone is produced either from dry air or oxygen by a high voltage AC power applied between two concentric electrodes separated by a narrow air gap. The outer tube is stainless steel and the inner one is a glass tube with a thin conducting coating. The oxygen gas passes through the gap and a high-energy discharge produces ozone. A tube type ozone generation element is shown in Fig. 5.5.

The ozone-enriched gas produced by the ozone generation system is then brought into contact with raw water to be treated in a bubble contact chamber. The presence of contact off gas ozone residual is expected in order to make sure that sufficient ozone is dosed to the raw water. Off-gas ozone residual must be decomposed to oxygen before vented to atmosphere.

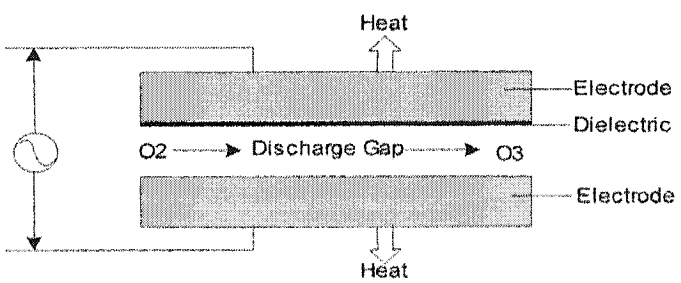


Figure 5.5: A Basic Ozone Generation Element

Oxygen needed for ozone generation can be produced from air on site or purchased and delivered to the plant. The latter method which has only two major components, Liquid Oxygen Storage tank and the Vaporizer, has gained popularity because of its simplicity. Usually, multiple vaporizers are used. In Fig. 5.6, we show the block diagram of a simple water treatment plant with two vaporizers utilized for generating oxygen gas.

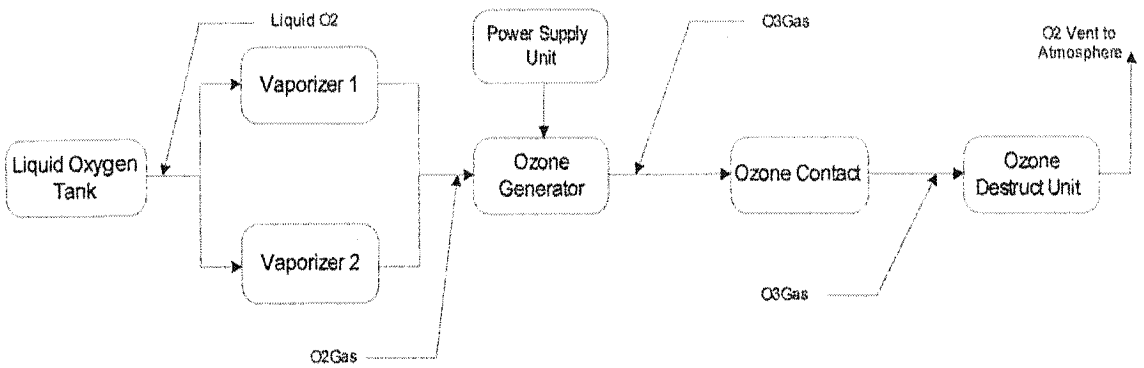


Figure 5.6: Block diagram of a simplified water treatment plant

The whole process of water treatment is very complicated and involves several chemical reactions. In the following, we investigate a simplified ozone generation plant consisting of only two units: Oxygen Supply Unit (OSU) and Ozone Generator Unit (OGU). In order to produce ozone continuously, both units must function concurrently during the normal operation of the system. The system contains both discrete event and continuous variables. However, the continuous parts normally may be considered as discrete at a higher level of abstraction. Figure 5.7 shows the OSU and OGU units of the plant in Fig. 5.6 in more detail.

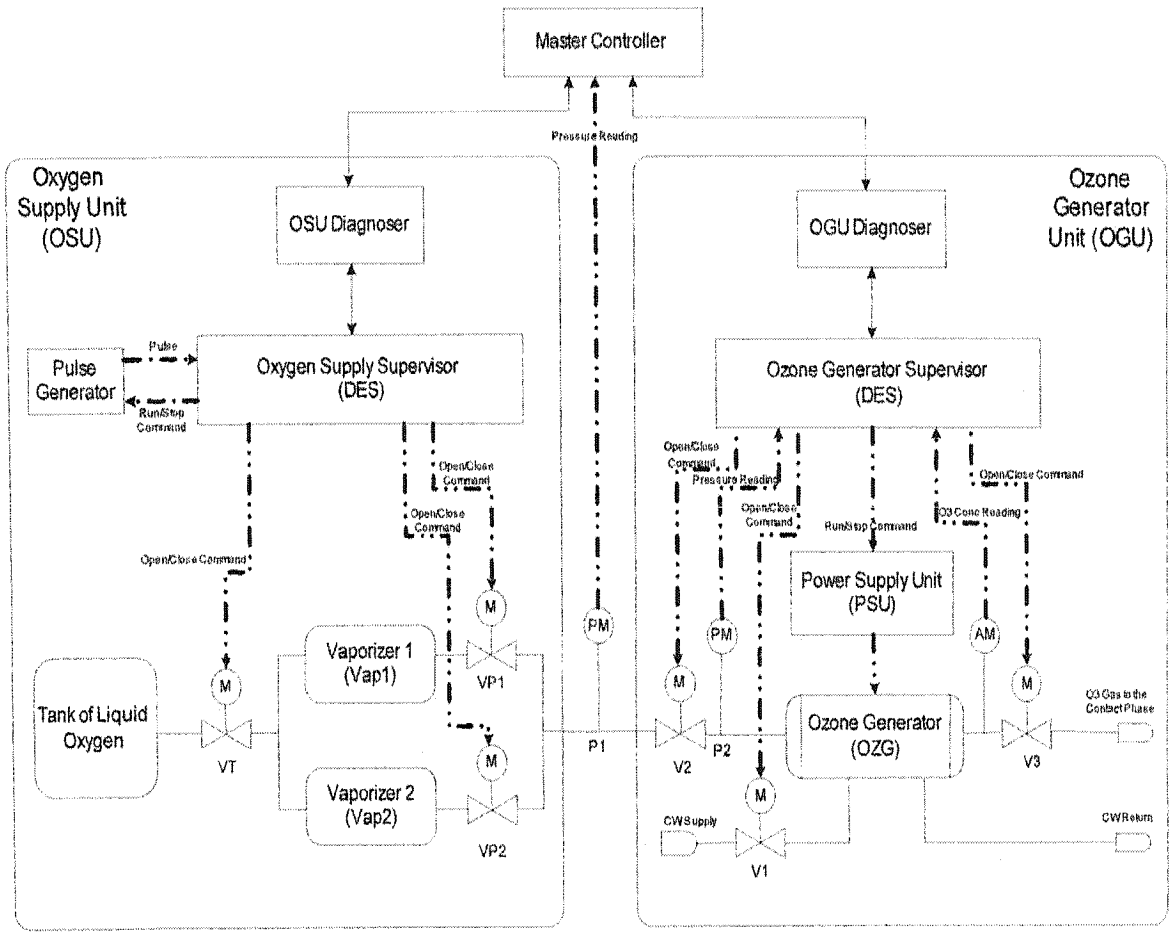


Figure 5.7: Simplified Ozone Generator and Oxygen Supplier units

OSU generates the required oxygen in the system. It consists of one liquid oxygen tank, two vaporizers, two vaporizer outlet valves, a liquid oxygen inlet valve and a pulse generator. During normal operation, one vaporizer is in duty and the other one in standby mode. The vaporizers are rotated in or out of service by opening or closing the vaporizer outlet valve (VPi, $i \in \{1,2\}$), based on a time set-point. Pulse generator produces pulses according to the time set-point. When a pulse is generated, the standby vaporizer is switched on (its valve is opened), and then, the one previously in duty is switched to

standby (the corresponding valve is closed). The flow of liquid oxygen to the vaporizers is controlled by the valve VT.

The OGU is in charge of producing ozone. It contains a Power Supply unit (PSU), a cooling water valve (V1), an oxygen gas inlet valve (V2), an ozone gas outlet valve (V3) and an ozone generation element (OZG).

The OGU is also equipped with several sensors. An ozone concentration analyser (OCA), marked as AM in Fig. 5.7, notifies the system of ozone concentration changes. There are also two sensors measuring the pressure at P1 and P2. Sensor readings at P1 are available to the OSU, the OGU and the master controller while readings at P2 are only monitored in OGU.

Each unit can be modeled by a generator, under the supervision of a local supervisor (controller). Local supervisors are in charge of sending appropriate commands (Run/Stop/Open/Close) to their unit components. There is also a master controller (coordinator) responsible for generating synchronization events (commands) in the system. It basically manages the interlock between the subsystems and the master plant sequence. Figure 5.8 demonstrates the control architecture.

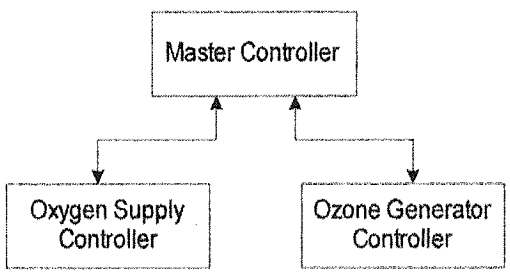


Figure 5.8: Ozone Plant Control System Architecture

In the next sub-section, we will develop a discrete event model for the OSU and OGU units. Diagnoser design is discussed in sub-section 5.4.3.

5.4.2 Discrete-Event Model of the Process

The plant can be modeled as a hierarchical system with an AND super-state at the top level. The components of the AND super-state are OSU, OGU and the master controller. Moreover, the OSU has three operation modes. Each mode can be represented by an OR super state. A simplified HFSM is presented in Fig. 5.9.

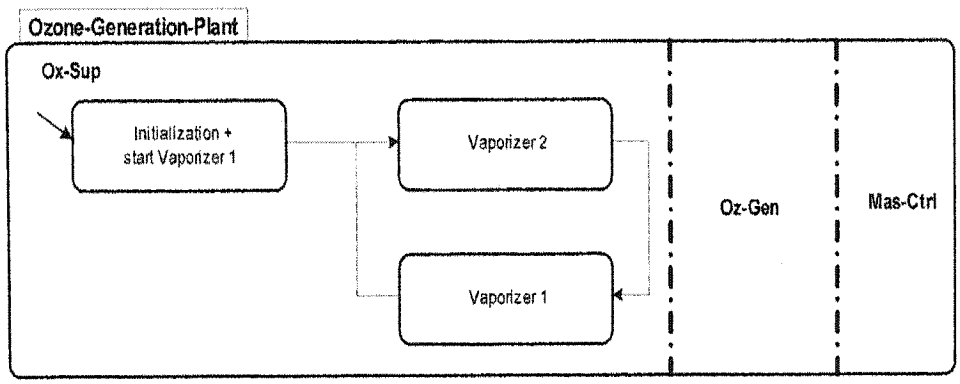


Figure 5.9: Simplified HFSM of the Ozone Generation Plant

In the following sub-sections we explain the model of each unit in more detail.

5.4.2.1 OSU Model

In this unit, one failure mode is considered: Liquid Oxygen Outlet Valve (VT) stuck-closed (Fox). For simplicity, it is assumed that VT becomes stuck-closed only when it is closed. An HFSM modeling the OSU is given in Fig. 5.10. Note that in Fig. 5.10, states XN1, XN2, XF1 and XF2 are fictitious states that have been added to the model to make it FC-connected. Event symbols and their description are given in Table 5.2.

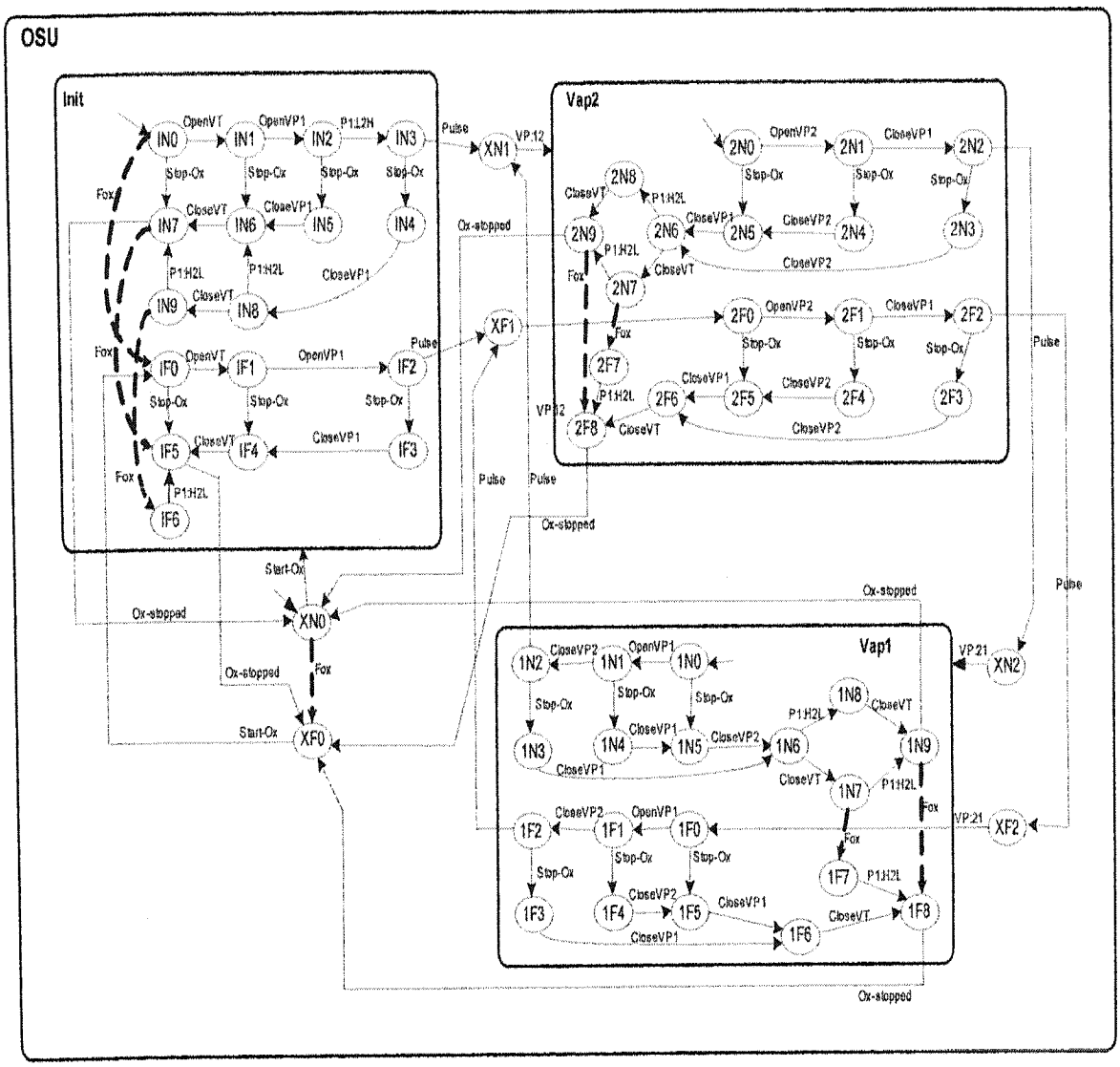


Figure 5.10: The hierarchical discrete-event model of the Oxygen Supply Unit (OSU)

Event	Description
Start-Ox	Command start the OSU
Stop-Ox	Command stop the OSU
OpenVT	Command open valve VT
CloseVT	Command close valve VT
OpenVPi	Command open valve VPi (i =1,2)
CloseVPi	Command close valve VPi(i =1,2)
P1:L2H	P1 changed from Low to High
P1:H2L	P1 changed from High to Low
Pulse	Pulse generator sent a pulse
Fox	Valve VT stuck closed
Ox-Stopped	The OSU stopped

Table 5.2: Events and their description in the OSU

5.4.2.2 OGU Model

In the OGU, one failure mode is considered: oxygen gas inlet valve (V2) stuck-closed (Foz). In this unit we model components as simple generators. The model of the entire unit can be obtained by combining the models of the components, the interaction among the components, and the unit controller.

- **Discrete-Event Model of the components**

1) Cooling Water Valve: V1 (Fig. 5.11.a)

The cooling water valve has two events 'CloseV1' and 'OpenV1' representing the close command and open command. There are two states VC1 and VO1 corresponding to normal-closed and normal-open.

2) Oxygen Gas Inlet Valve: V2 (Fig. 5.11.b)

The oxygen gas inlet valve has three events 'CloseV2', 'OpenV2' and 'Foz' (V2 stuck-closed). There are three states VC2, VO2 and V2F corresponding to normal-closed, normal-open and stuck-closed.

3) Ozone Gas Outlet Valve: V3 (Fig. 5.11.c)

Similar to V1, V3 has two events 'CloseV3' and 'OpenV3' representing the close command and open command. There are two states VC3 and VO3 corresponding to normal-closed and normal-open.

4) Power Supply Unit: PSU (Fig. 5.11.d)

The PSU has two events 'PSU-Run' and 'PSU-Stop' representing the commands for running and stopping the operation of PSU respectively. It has two states: Run and Stop.

5) Oxygen Pressure Sensor at P1 (Fig. 5.11.e)

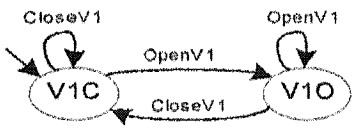
The oxygen pressure at P1 has two states: P1L (pressure low) and P1H (pressure high).

6) Oxygen Pressure Sensor at P2 (Fig. 5.11.f)

The oxygen pressure at P2 has two states: P2L (pressure low) and P2H (pressure high).

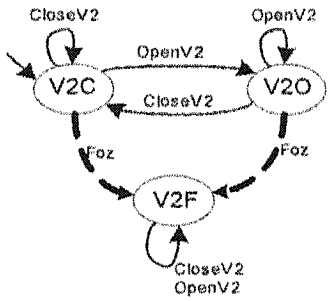
7) Ozone Generator: OzG (Fig. 5.11.g)

The model of the ozone generator has two states: O3L (ozone concentration low) and O3H (ozone concentration high).



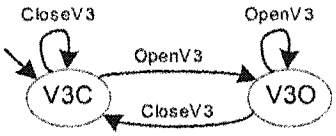
V1

a) Cooling Water Valve



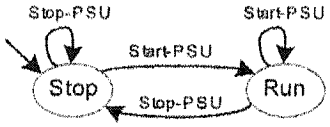
V2

b) Oxygen Gas Inlet Valve



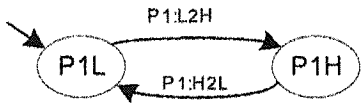
V3

c) Ozone Gas Outlet Valve



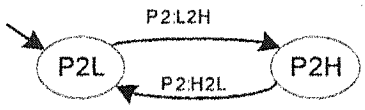
PSU

d) Power Supply Unit



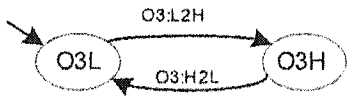
P1

e) Oxygen Pressure Sensor at P1



P2

f) Oxygen Pressure Sensor at P2



OzG

g) Ozone Generator

Figure 5.11: Discrete-event model of the OGU components

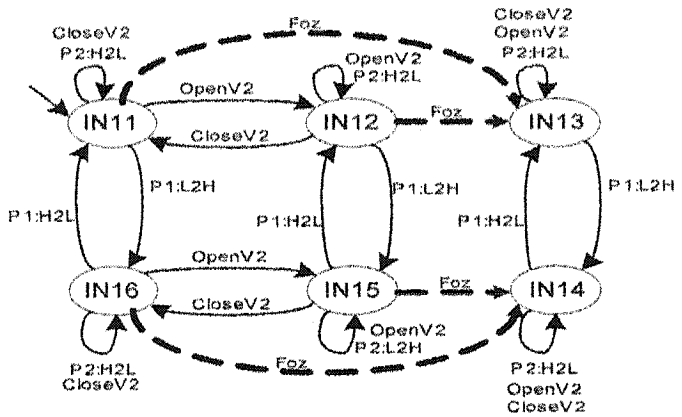
- **Interaction among the Components in OGU**

1) Interaction among the Oxygen Pressure at P2, the Oxygen Pressure at P1 and the state of the Inlet Oxygen Gas Valve (V2) (Fig. 5.12.a)

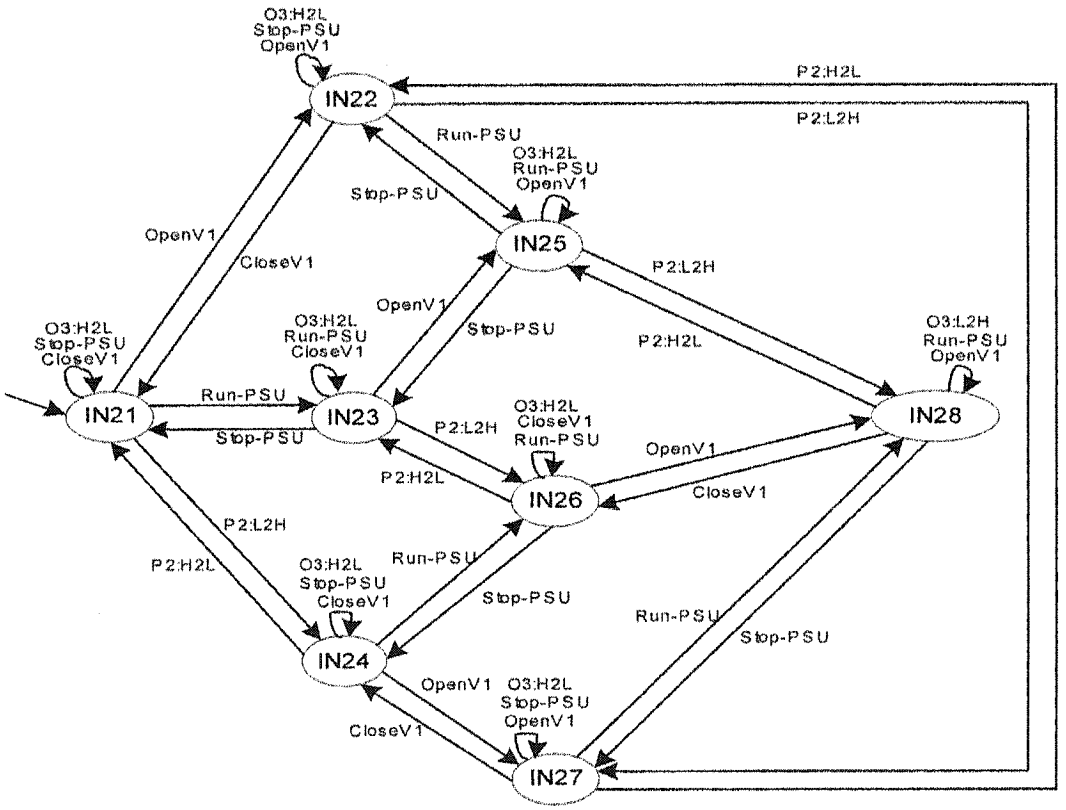
The oxygen pressure at P2 changes from low to high if the oxygen pressure at P1 is high and valve V2 is open (V2 is in state open).

2) Interaction among the Ozone Generator, the PSU and the Cooling Water Valve

Ozone generation is a continuous process described by continuous variable models. However, at a higher level of abstraction, only two major factors count: the effects of the power supply unit and the cooling water supply valve. The ozone concentration in the ozone generator changes from low to high if pressure P2 is high, the power supply unit is running (PSU is in Run state) and the cooling water valve is in open (V1 is in the open state). Figure 5.12.b shows the discrete-event model of this interaction.



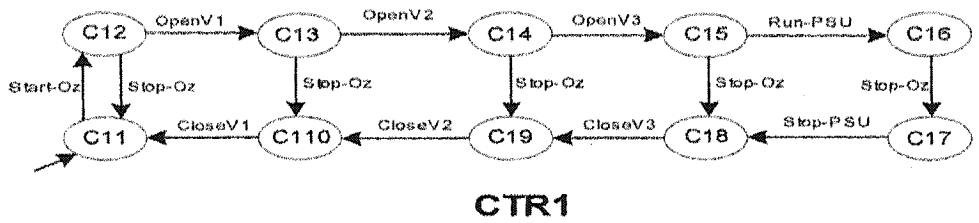
a) Interaction among the pressures at P1 and P2 and the Inlet Oxygen Gas Valve (V2)



b) Interaction among the Ozone Generator, the PSU, Cooling Water Valve and the Oxygen Pressure at P2

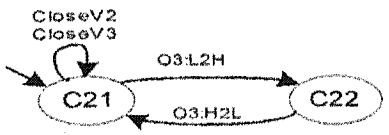
Figure 5.12: Interaction among the components in the OGU

- **OGU Controller**



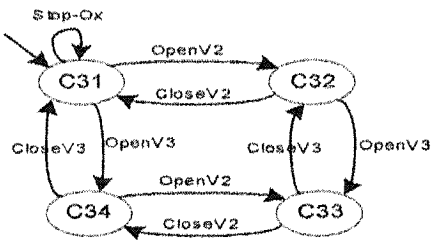
CTR1

a) CTR1



CTR2

b) CTR2



CTR3

c) CTR3

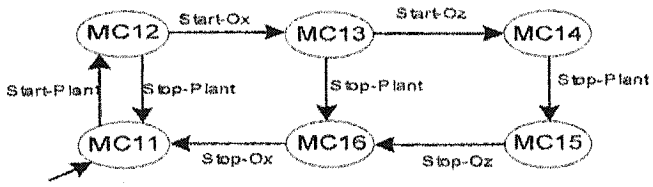
Figure 5.13: Discrete-event model of the controllers in the OGU

Figure 5.13.a shows the controller (CTR1) supervising the start-up and the shut-down sequences of the ozone generator unit. Moreover, in practice, the ozone generator gas valves must not be closed till the ozone concentration is low. This can be enforced by CTR2 shown in Fig. 5.13.b. In addition, for the purpose of safety, the oxygen supply unit can become shut-down if the ozone generator unit is already shut down. In other words, both valves V2 and V3 have to be closed prior to issuing the command 'Stop-Ox' by the master controller. CTR3 in Fig. 5.13.c models this requirement.

Finally, the controller for the ozone generator unit can be obtained by constructing the synchronous product of the three controllers; i.e.:

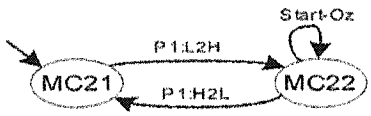
CTR = *sync* (CTR1, CTR2, CTR3)

OGU can be constructed by the synchronous product of the component models, interaction models and the controller.



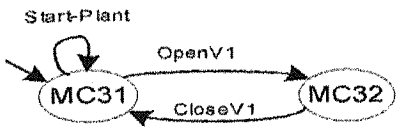
MCtr1

a) MCtr1



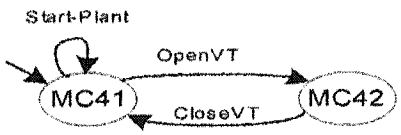
MCtr2

b) MCtr2



MCtr3

c) MCtr3



MCtr4

d) MCtr4

Figure 5.14: Master-Controller

5.4.2.3 Master-Controller Model

The master controller supervises the start-up and shut-down sequence in the plant and synchronizes the shared events in the OSU and the OGU. Figure 5.14.a shows the start-up and shut-down sequence in the plant (MCtr1). Furthermore, in practice, it is desirable that the OGU start functioning if oxygen is ready in the form of the gas for conversion to ozone. In other words, the OGU should become operational if the pressure of oxygen at P1 is high. This requirement can be enforced by the controller MCtr2 shown in Fig. 5.14.b. Note that the plant should only be started when the OGU and the OSU are shut-

down. MCtr3 will insure that the 'Start-Plant' command can only be enabled at the initial state of the OGU. Moreover, MCtr2 will guarantee that the 'Start-Plant' command can only be enabled when all of the phases of the OSU are shut-down. The master controller can be obtained by forming the synchronous product of MCtr1, MCtr2, MCtr3 and MCtr4.

5.4.3 Diagnoser Design

The plant can be considered as the synchronous product of three modules the OSU, the OGU and Master-Controller. The unobservable event sets of these modules are $\{Fox\}$ and $\{Foz\}$. Note that the unobservable event sets of modules are pair-wise disjoint. We also assume that the system and the diagnosis process are initialized simultaneously. Therefore, according to Proposition 5.2, we can build fully modular diagnosers for the system.

Since in this example, we have not considered the failure modes of the Master Controller, we only have to design diagnosers for the OSU and the OGU. For designing diagnoser for the OSU, a D-holon is associated with each super state of the OSU describing a phase of the operation in the OSU. Then, a diagnoser is constructed for each D-holon. Figure 5.15 shows the high level D-holon in the OSU. The low level D-holons are shown in Fig. 5.16.

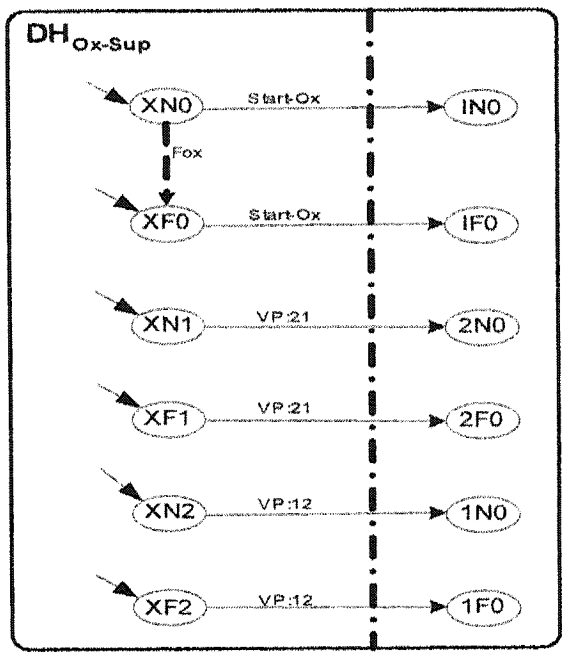


Figure 5.15: High-level D-holon in the OSU

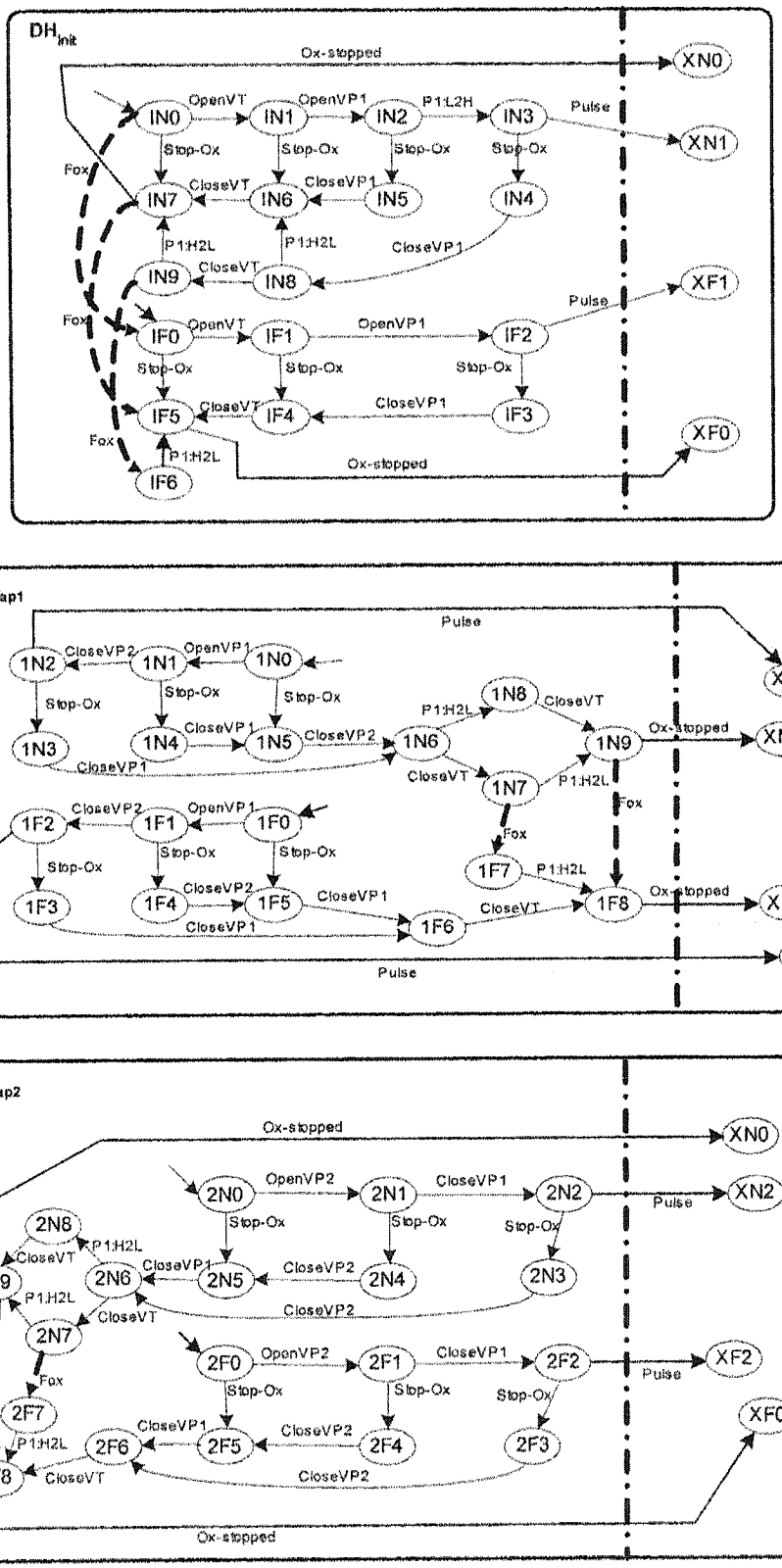


Figure 5.16: Low-level D-holons in the OSU

Figure 5.17 shows the designed modular diagnoser. DG_{Oz-Gen} is the diagnoser for the OGU. DG_{OSU} is the diagnoser for OSU. DG_{OSU} consists of four decoupled D-holon diagnosers, DG_{Ox-Sup}, DG_{Init}, DG_{Vap1} and DG_{Vap2}.

Dotted line in Fig. 5.17 implies that diagnosers DG_{Oz-Gen} and DH_{OSU} function concurrently. When diagnosis process starts, diagnosers DG_{Oz-Gen} and DH_{OSU} become active. Diagnoser DG_{Oz-Gen} remains active during the diagnosis process. After the occurrence of event 'Start-Ox', DG_{Init} becomes active and DG_{Ox-Sup} becomes inactive. The events labeling the arcs in the graph represent transitions among D-holon diagnosers.

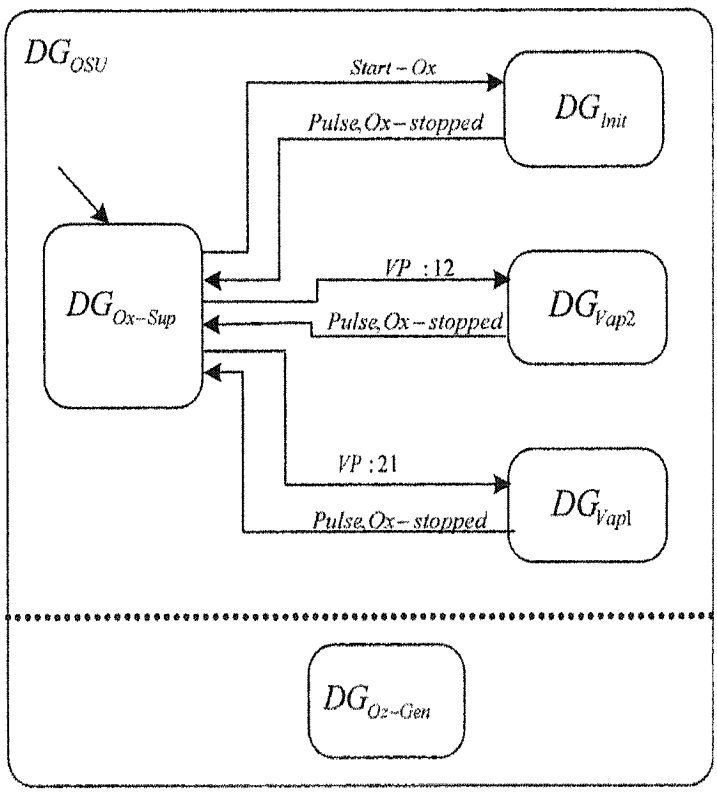


Figure 5.17: Sequence of diagnoser activation in the Ozone Generation Plant

Chapter 6

Conclusion

6.1 Summary

In this thesis, we study fault diagnosis in Hierarchical Finite State Machines (HFSM). A HFSM is a simplified version of a statechart. HFSMs add depth (hierarchy) and orthogonality (parallelism) features to finite state machines and therefore, are more useful for modelling complex systems.

In our approach, we assume that the state set of the HFSM model of the system can be partitioned according to the system's condition (failure status). First, a HFSM is split into a collection of simpler flat substructures called D-holons. Each D-holon describes a phase or a stage of the operation of the system.

Instead of designing a single diagnoser for the entire system (to be more specific, the equivalent flat system), we design simpler diagnosers for the D-holons of the system. A state estimate for the system will be obtained using the state estimates provided by these diagnosers. Using this state estimate, the system condition is determined. The current state estimate determines the D-holons the current state of the system could be in. These D-holons and the corresponding diagnosers are considered active. At any instant,

therefore, only a subset of the diagnosers is needed. This reduces computer Random Access Memory (RAM) requirements.

We also investigate failure diagnosability in HFSMs. In the standard definition of diagnosability, failure diagnosability is required at all phases of the operation. This seems too restrictive. We introduce the concept of phase-diagnosability in which diagnosability of a failure in a component is examined only in the D-holons in which the component is active. A set of necessary and sufficient conditions for phase-diagnosability is provided.

Furthermore, in order to reduce the computational complexity of the diagnosis process, a set of sufficient conditions is provided under which the diagnosis process becomes semi-modular. It is shown that the computational complexity of constructing (time) and storing (space) the transition systems required for diagnosis in the proposed semi-modular approach is polynomial in the number of system components, whereas in the original monolithic approach, the computational complexity is exponential in the number of system components.

We also illustrate our approach by applying our results to an ozone generator plant.

The result of our work can be applied in multi-phase systems with a hierarchical structure and large number of components. For instance, they could be used for systematic diagnosis in systems such as manufacturing systems, batch processes and spacecrafts, which have complex multi-phase models. Certain limitations exist though. In particular, the structure of these systems should meet certain conditions.

6.2 Future Work

The diagnosis process studied in our work uses the observable event sequence generated in the system to detect and isolate failures. In many cases, however, a failure changes only the timing of the observable event sequence (rather than the sequence). In other cases, after the occurrence of a failure, no new observable event is generated in the system. In these situations, using timing information can increase the accuracy and efficiency of our approach. Hence, fault diagnosis in timed hierarchical finite state machines would be an interesting area of research.

In conventional modular fault diagnosis, it is usually assumed that system components are asynchronous, i.e., they do not have common events. In the semi-modular method discussed in chapter 5, we do not make such assumption. There, however, we assume that modular components have no common "*unobservable*" events. This assumption may be limiting in complex systems whose components have strong interaction among each other. It would be useful to have modular diagnosis design techniques with less restrictive conditions on the structure of the components.

In our framework, the entire observable event sequence and the complete system model are used for diagnosis of the failure modes. In many cases, failure events occurring at a system level are different from those of other levels. In these situations a failure mode at a certain level might be detected and isolated using only the system model at that level. This may result in reduced computations. We have not addressed this issue.

In this thesis, we have assumed that transition leaving D-holons are observable. This implies that transitions from level to level (or from phase to phase) are observable. This assumption may be limiting in some systems and restrict the choice of D-holons. In

addition, if as a result of a sensor failure, a transition becomes unobservable, the change from a phase to another may become unobservable in the system. Therefore, a more relaxed condition on the boundary transitions of D-holons may be useful for diagnosis.

Bibliography

[AEM94] J.R. Agre, G. Elsley, D. McFarlane, J. Cheng, B. Gunn, "Holonic Control of a Water Cooling System for a Steel Rod Mill," *Proc. 4th International Conf. on Computer Integrated Manufacturing and Automation Technology*, Oct. 10-12, 1994, Rensselaer Polytechnic Institute, Troy, New York, pp 134 – 141.

[BH93] Y. Brave, M. Heymann, "Control of Discrete-Event Systems Modeled as Hierarchical State Machine," *IEEE Trans. Automat. Contr.*, vol. 38, no. 12, December, 1993, pp. 1803 – 1819.

[CP97] Y.L. Chen and G. Provan, "Modeling and Diagnosis of Timed Discrete Event Systems-A factory Automation Example," *Proc. American Contr. Conf.* , Albuquerque, New Mexico USA, June, 1997, pp 31 – 36.

[CLR90] T.H. Cormen, C.E. Leiserson and R.L. Rivest, *Introduction to Algorithms*. New York: McGraw-Hill, 1990.

[DM02] R. Debouk and R. Malik, "A modular Architecture for Diagnosis of Discrete Event Systems," *Proc. 41th IEEE Conf. on Decision and Contr.*, Las Vegas, Nevada USA, December, 2002, pp 417 – 422.

[Guo02] R. Guo, "Fault Diagnosis in a Water Treatment Plant Using Discrete-event Models," Master of Engineering Project Report, Concordia University, Montreal, Canada, 2002.

[GHL94] L. Gou, T. Hasegawa, P.B. Luh, "Holonic Planning and Scheduling for a Robotic Assembly Tesbed," *Proc. 4th International Conf. on Computer Integrated Manufacturing and Automation Technology*, Oct. 10-12, 1994, Rensselaer Polytechnic Institute, Troy, New York, pp 142 – 149.

[HCK92] W. Hamscher, L. Console and J. de Kleer, Eds., *Readings in Model-Based Diagnosis. San Mateo*, CA: Morgan Kaufmann, 1992.

[Har87] D. Harel, "Statecharts: A Visual Formalism for Complex Systems," Science of Computer Programming, 8 , North, Holland, 1987, pp. 231 – 274.

[HKW03] S. Hashtrudi Zad, R.H. Kwong, W.M. Wonham, "Fault Diagnosis in Discrete-Event Systems: Framework and Model Reduction," *IEEE Trans. Automat. Contr.*, vol. 48, no. 7, pp. 1199 – 1212, July 2003.

[HKW99] S. Hashtrudi Zad, R.H. Kwong, W.M. Wonham, "Fault Diagnosis in Timed Discrete-Event Systems," *Proc. 38th IEEE Conf. on Decision and Contr.*, Phoenix, Arizona USA, December, 1999, pp 1756 – 1761.

[HKW00] S. Hashtrudi Zad, R.H. Kwong and W.M. Wonham, "Fault diagnosis and consistency in hybrid systems", *Proc. 38th Annual Allerton Conference on Communication, Control, and Computing*, University of Illinois at Urbana-Champaign, October 2000, pp. 1135-1144

[Ise97] R. Isermann, "Supervision, Fault-Detection and Fault Diagnosis Methods-An Introduction," Control Eng. Practice, vol. 5, no. 5, pp. 639-652, 1997.

[LY96] D. Lee and M. Yannakakis, "Principles and Methods of Testing Finite State Machines - A Survey," *Proc. IEEE*, vol. 84, no. 8, pp 1090 – 1123, 1996.

[Lev95] N.G. Leveson, *Software: System Safety and Computers*. Reading, Mass.: Addison-Wesley, 1995.

[Lin94] F. Lin, "Diagnosability of Discrete Event Systems and Its Application," Discrete Event Dynamic systems, vol. 4, pp. 197-212, 1994.

[RW82] P.J. Ramadge and W.M. Wonham, "Supervision of Discrete-Event Processes," *Proc. 21th IEEE Conf. on Decision and Contr.*, pp 1228 – 1229, 1982.

[RW89] P.J. Ramadge and W.M. Wonham, "The Control of Discrete Event Systems," *Proceedings of the IEEE*, vol. 77, no. 1, pp. 81-98, 1989.

[RF00] S.L. Ricker, E. Fabre, "On the Construction of Modular Observers and Diagnosers for Discrete-Event Systems," *Proc. 39th IEEE Conf. Decision and Contr.*, Sydney, Australia, December, 2000, pp 2240 – 2244.

[SSL95] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen and D. Teneketzis, "Diagnosability of Discrete-Event Systems," *IEEE Trans. Automat. Contr.*, vol. 40, no. 9, pp. 1555 – 1575, 1995.

[SSL96] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen and D. Teneketzis, "Failure Diagnosis Using Discret-Event Models," *IEEE Trans. Contr. Syst. Technology*, vol. 4, no. 2, pp. 105 – 124, 1996.

[SLT98] M. Sampath, S. Lafortune and D. Teneketzis, "Active Diagnosis of Discrete-Event Systems," *IEEE Trans. Automat. Contr.*, vol. 43, no. 7, pp. 908 – 929, 1998.

[Wan95] B. Wang, "Top-Down Design for RW Supervisory Control Theory," Master's thesis, University of Toronto, Toronto, Canada, 1995.

[Won93] W.M. Wonham: Notes on Control of Discrete Event Systems, Dept. of Electrical and Computer Engineering, University of Toronto, 2003.