Hierarchical Modular Fault Recovery in Discrete-Event Systems

Abdolmajid Moradi

A Thesis

in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of Master of Applied Science at

Concordia University

Montreal, Quebec, Canada

March 2006

Your file Votre référence
ISBN: 0-494-14275-8
Our file Notre référence
ISBN: 0-494-14275-8

ABSTRACT

Hierarchical Modular Fault Recovery in Discrete-Event Systems

Abdolmajid Moradi

This thesis studies the fault recovery problem in discrete-event systems using a two-level modular hierarchical approach. We extend the switching modular solutions for failure recovery problems for flat DESs in the framework of Ramadge and Wonham (RW) for supervisory control, using a two-level hierarchical model for DESs. Although in practice, modular design is found to be effective for a variety of systems, in the case of large-scale systems the state sizes may become too large, and so may the computational complexity of these systems for synthesis purposes. Therefore, a hierarchical approach may largely reduce the complexity of practical systems.

We assume that the plant in our problem can be considered as a standard finite-state automaton, describing both normal and faulty behaviors of the system. Specifications of the system are divided into two main categories. The first category are those that address local (subsystem-level) issues, and for which we design a supervisor at low-level using a switching modular approach, while the second group of specifications address global (system-level) issues. For this group we design a supervisor following a modular hierarchical approach. In this thesis, we assume full events observation. After designing the low-level supervisors to enforce the low-level (subsystem) specifications for normal and recovery modes, the plant under supervision is computed at the low-level. Following this, an abstraction of the plant is computed and accordingly, the high-level supervisors are designed to control the plant in the normal and recovery modes at the high-level.

To illustrate our design approach and also as a motivation for this work, we consider the automatic control of an electrical power network in its normal operation and in response to the faults occurring in the network. The different components of electrical systems are modelled using DESs and the required supervisors will be designed to control the network in both normal and faulty conditions.

To my wife and my parents

Contents

List of Figures

Chapter 1

Introduction

1.1 Preface

Discrete-event systems (DESs) are a useful modelling tool for a variety of systems such as manufacturing systems, electrical power networks, chemical processes and traffic systems. The main property of any DES is that at any time instant, it is recognized by a symbolic *state*, which can change due to the occurrence of an *event*. Thus, the behavior of a DES can be described by the sequences of events occurred in the system, and the states visited. The supervisory control theory introduced by [19] is a general framework for the synthesis of supervisors (controllers) for DESs.

Many industrial systems can be modelled as discrete-event systems, and systematic ways are available for designing supervisors (controllers) using DES techniques. In this thesis, we investigate the fault recovery problem using discrete-event models by introducing a *two-level modular hierarchical* solution to this problem. As a practical example, we will apply our proposed method to the problem of fault recovery in electrical power networks.

1.2 Fault Recovery Problem in Discrete-Event Systems

Failure recovery problem in large-scale industrial systems has been the subject of research for decades. As the complexity of modern controlled systems grows rapidly, the need for having more reliable controllers becomes more and more vital. In case of failure in a system, the controllers (supervisors) should be able to take the necessary actions for recovery. This results in a complex problem for designing recovery procedures. In order to handle this problem in large-scale systems and decrease the complexity of the controllers, a modular hierarchical approach could be used.

Fault recovery problem in discrete event systems has been studied by many researchers (e.g., [2], [20], [5] , [13]). In [20], an integrated approach is proposed for both control and diagnosis problems. Since in this method, the diagnosis and control problems are integrated, in the worst case scenario, we have exponential complexity in the number of plant states.

In [2], DES model of the system is analyzed in order to design a fault tolerant supervisor. Faults and failures are classified as different types of events. Failures are described as the breakdown of a component of the system, while a fault is considered as a malfunction of a component of the system and therefore is tolerable. A fault tolerant supervisor is designed afterwards which can take the system from an initial state to a marked state in the presence of fault. This approach fails to address some issues such as changes in the specification after the occurrence of faults if it happens.

In recent works in this area, [5] discusses the fault recovery problem for a team of robots using discrete-event models. The number of units in a robot team can decrease resulting from failures or off-line commands, so the supervisor has to reconfigure the controlled system in order to satisfy the design specifications. This approach proposes a learning and repair method. When a unit switches to off-line mode, the learning technique deletes the events belonging to the failed units from the supervisor only, and then a repair algorithm is used to reconfigure the control system.

Supervisory control of DES plants under partial observation in the presence of sensor failure is considered in [9]. An algorithm is provided in order to reconfigure the supervisor in response to sensor loss (or recovery) such that the plant under supervision satisfies the design specifications. [9] assumes that any loss of sensor is known before occurring of the next event in the plant.

In [13] and [14], the authors propose a modular switching technique for fault recovery problem in DES, assuming permanent faults in the system. A diagnosis system is assumed to be available with a bounded delay which detects and isolates faults. A finite-state automaton is used to model the diagnosis system and it can be designed according to any diagnosis technique, as long as the bounds for failure detection are available. In this framework, the finite-state automaton can be assumed as an abstraction for the underlying diagnosis system. With this approach, the diagnosis and control problems can be separated with the advantage of having simpler solutions for both problems. The system under control has three modes: normal, transient, and recovery. The specification for each mode in general is different from the other ones. Using the supervisory control theory introduced by [19], a modular switching supervisory control mechanism is proposed to enforce the design specifications in each mode. The nonblocking and admissibility of supervisors are discussed. The complexity of implementing this method is high in normal mode for multiple failure scenarios, because all recovery supervisors should be engaged in the control of the systems in the normal mode, even if there is no detected fault.

Although the initial concepts of discrete-event models seem to be very easy, they suffer from certain drawbacks in real-world applications. For example; the computational complexity is one major issue in systems modelled using DESs. The synthesis of supervisor is polynomial in terms of number of states in a DES model, but the number of states in a real model can be larger than 2^{20}. Modular control and hierarchical control can be considered as great advancements to overcome the above mentioned problem. In literature, on the hierarchical control of DESs, four major approaches are identified:

- Bottom-up design [25] [26]
- Top-down design [22]
- State aggregations [4] [6]
- Interface-based design [10]

In [25], the problem is studied by generating a *hierarchically consistent* abstraction model of a low-level DES. This approach is generally called bottom-up design and is further refined in [26] and [24]. A study which discusses the extension of this approach to a multi-level hierarchical design, is later introduced in [17].

In [22], a top-down design for the hierarchical control of DESs is proposed. A simple DES model is modelled using a hierarchical *State Structure Tree*, where the subsystems can be refined gradually. State aggregation is another method similar to the bottom-up design. In this method, hierarchy is introduced by aggregating states in the state partitions [4], [6]. Finally in [10], a new approach is introduced on the hierarchy control of DESs. Unlike other approaches, the different levels of hierarchy model are strictly decoupled using well-defined interfaces.

As will be discussed later in this chapter, in this thesis we use the framework of [26] to extend the work of [13] to a *hierarchical modular switching* approach for fault recovery. The reason for choosing the approach of [26] is that in our opinion, it fits better with the method of [13]. Furthermore, some of the algorithms of [26] have been implemented in TTCT [21].

1.3 Fault Recovery Problem in Electrical Power Networks

In the area of developing automated support tools for fault diagnosis and recovery in electrical power networks, several attempts have been made. Most of these efforts differ significantly in their technological approaches. Some of these approaches are: neural networks (e.g. [18]), fuzzy logic systems (e.g. [15]), model based diagnosis, reasoning techniques, and DES based techniques (e.g. Petri nets). Neural networks and fuzzy logic systems are particularly suitable for diagnosing non-drastic failures (which require detailed continuous signal analysis). DES models, on the other hand, are suitable for diagnosing drastic failures since for the detection of these failures coarse models may be enough.

We briefly review some of the DES approaches. Different theories based on discrete-event systems, have been studied for fault diagnosis in power systems. In [11] a petri nets based system is introduced and it is shown that by using the proposed method, the processing time can be reduced, and accuracy can be increased. One major drawback for these approaches is that the component models are usually not detailed.

[1] proposes a model-based technique for diagnosis of active large scale discrete-event systems. Their approach was an extension to an earlier work done [12]. The active system which is designed to react to the failure events, is modelled using communicating automata. Unlike other methods, which usually deal with the synchronous product of events and building off-line model of the entire plant, the method proposed in [1] considers the asynchronous events and an on-line reconstruction of the active system using the available observation of the system. The proposed diagnosis technique is then applied to a power transmission network.

[16] considers the designing of a controller for an electrical power transformer station, using SIGNAL language, which has already been developed for precise description of real time systems. They specify both, the physical model of the plant and the control objectives, in SIGNAL language. The SIGNAL compiler translates the SIGNAL program into a Polynomial Dynamical System (PDS). The control objectives can be verified and synthesized by these polynomial relations and operations.

The designed controller should be able to respond to the electric faults on the lines. For example, it has to be able to handle the power interruption, or redirect the supply source in case of failure occurring in the system. The most important objectives of the controller are: ensuring the safety of the equipment, and an uninterrupted electrical power supply. These objectives can be achieved by opening or closing of different circuit breakers in the transformer station, which automatically can be controlled by the controller. Although the authors describe the physical model using a powerful environment, they limit their work to a power transformer station and it is not clear how this method can cope with the huge size of power networks of real-world applications.

In this thesis, we have chosen to use RW control theory of discrete-event systems (DESs) [19], to design the local and global controllers for electrical power systems in response to the electrical faults which could occur in these systems. We start by proposing a modular hierarchical approach, partly motivated by the work of [13], for the fault recovery problem in DESs. It is assumed that the system has two modes of operation: normal and recovery. The design specifications for the operation of the plant in each mode are categorized into two main groups. The first category of specifications are those which address the local (subsystem-level) issues and for those we design a low-level (local) supervisor, while the second group are those concerned with global (system-level) issues for which we design a supervisor by proposing a switching modular hierarchical solution. After designing the relevant (local) supervisors for the first group of the specifications for both normal and recovery modes in the low-level plant, we will use a two-level hierarchical model, introduced by [26], to design the plant model and its required controllers at the high-level to enforce the second group (global) specifications.

The main feature of this approach, in comparison with those we mentioned earlier, is that it uses modular and hierarchical techniques to handle the complexity of the problems and solutions to both problems of failure recovery and automatic control of power networks. The designed supervisor(s) control the plant as the main controller(s) in its normal operation and handle the failures of the system. The modular design gives us the opportunity to divide different tasks of the controller(s). Therefore, the designed supervisor(s) in normal operation mode, can be thought of as supervisor(s) which handle the task of automatic control of the plant whereas, the recovery controllers are involved in controlling of the plant, only if a failure occurs in the system. In addition to modular design, the hierarchical feature also may enable us to overcome the complexity of controller design for large-scale power systems.

1.4 Thesis Outline

An outline of the rest of thesis is as follows.

Chapter 2 gives a background on RW theory of supervisory control. In this chapter, we also review some of the concepts in the hierarchical control of DESs introduced by [26] and [24] with more emphasis.

Chapter 3 covers the problem formulation, modelling and the proposed hierarchical switching modular approach. Specifications for each of the normal and recovery modes are divided into two main groups. For each group we design the proper supervisors. A *two-level hierarchical modular switching solution* is proposed for the fault recovery problem in DESs. We discuss the issues of nonblocking properties of the system under supervision and computational procedures.

To illustrate our approach to the fault recovery problem, in Chapter 4, we investigate the application of method to the automatic control of a simple power generation and distribution network in the presence of electrical faults. In this chapter, we consider short circuit failures. The automated control of a large scale power generation and distribution network using a hierarchical setup, increases the reliability and efficiency of the network.

We consider an electrical power network, consisting of two generation groups, which are located in two different plants. The goal is to redirect the supply sources using a local controller, in order to have no interruption of power supply for the first plant. This needs to be done in case of failure in one of the generating units of the first plant. The global controller is engaged in controlling the network, when we have a total failure for the second plant. Thus, the local controller takes care of the faults that can be dealt with locally, whereas the high-level supervisor functions in case of having a high priority failure in the system which, as we mentioned, can trigger a shut-down in one of the plants.

Finally in Chapter 5, we summarize our work and present some directions for future research.

Chapter 2

Overview of Discrete-Event Systems

The purpose of this chapter is to give an overview of automaton theory and supervisory control of discrete-event systems. In particular we present some basic concepts and fundamental results in RW supervisory control theory, and briefly review the two-level hierarchical control of DESs in [26].

2.1 Languages and Automata

A Discrete-Event System (DES) can be modelled by a 5-tuple automaton:

$$G = (Q, \Sigma, \delta, q_0, Q_m)$$

where Q is the set of states, $\Sigma = \Sigma_c \cup \Sigma_{uc}$ the set of events where Σ_c and Σ_{uc} are *controllable* and *uncontrollable events.* $\delta : Q \times \Sigma \rightarrow Q$ is the partial transition function, $q_0 \in Q$ initial state, and $Q_m \subseteq Q$ the set of marked states [19], [23].

A finite sequence over Σ is a sequence of the form of $\sigma_1\sigma_2\sigma_3...\sigma_n$ where $n \geq 1$ and $\sigma_i \in \Sigma$, $i = 1, 2, 3, ..., n$. We use the notation $\delta(q, \sigma)!$ to show that δ is defined at state q for event σ.

The transition function $\delta : Q \times \Sigma \rightarrow Q$ is a partial transition function and can be recursively extended to operate on Σ^* where $\Sigma^* = \Sigma^+ \cup \{\epsilon\}$. ϵ is empty string and Σ^+ is the set of all possible finite strings. δ can be extended from $Q \times \Sigma$ to $Q \times \Sigma^*$ by the following rules:

$$\delta(q, \epsilon) = \epsilon$$
$$\delta(q, s\sigma) = \delta(\delta(q, s), \sigma)$$

where $s \in \Sigma^*$,$\sigma \in \Sigma$ and $q \in Q$.

The behavior of a Discrete-Event System can be represented by languages. A language over Σ is any subset of Σ^*, with the empty language shown as $\emptyset$. For DES G we define the closed and marked behavior languages as follows:

- **Closed behavior**: The set of strings generated by G or the language generated by G is called the closed behavior of G and can be shown by :

$$L(G) = \{s \in \Sigma^* \mid \delta(q_0, s)!\}$$

- **Marked behavior**: The set of strings that can take DES G from an initial state q_0 to some marked state in Q_m or the language recognized by G is called the marked behavior of G (as the language recognized by G):

$$L_m(G) = \{s \in \Sigma^* \mid \delta(q_0, s) \in Q_m\}$$

It is obvious that $\emptyset \subseteq L_m(G) \subseteq L(G)$ and $\epsilon \in L(G)$.

Next, we define the ***reachable*** and ***coreachable*** state subsets of a DES G denoted by Q_r and Q_{cr}:

$$Q_r = \{q \in Q \mid \exists s \in \Sigma^*, \delta(q_0, s) = q\}$$

$$Q_{cr} = \{q \in Q \mid \exists s \in \Sigma^*, \delta(q, s) \in Q_m\}$$

A DES G is reachable if $Q_r = Q$. Since unreachable states do not affect $L(G)$ or $L_m(G)$, and an equivalent reachable generator can always be constructed, we assume that G is reachable.

A DES G is coreachable if $Q_{cr} = Q$. If a DES G both *reachable* and *coreachable* we call the DES ***trim*** .

- **Nonblocking**: A deterministic DES G is said to be nonblocking if the following holds:

$$\overline{L_m(G)} = L(G)$$

Nonblocking means that the DES can always reach a marked state, in other words every string in $L(G)$ can be completed to a marked sequence in $L_m(G)$.

2.2 Operations on Languages and Automata

In this section we define some useful operations on languages and DES. First, we introduce the **cat** operator.

- **catenation** : The ***catenation*** operator can put strings together according to the following rules :

$$\mathbf{cat} : \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$$

$$\mathbf{cat}(s, \epsilon) = \mathbf{cat}(\epsilon, s) = s, \quad \text{for } s \in \Sigma^*$$

$$\mathbf{cat}(s, t) = st \text{ and } s, t \in \Sigma^+$$

- **Prefix Closure** : If $s = tu$ and $s, t \in \Sigma^*$ we say that t is a prefix of u. The prefix closure of a language $L \subseteq \Sigma^*$ can be shown by $\overline{L}$:

$$\overline{L} = \{s \in \Sigma^* \mid \exists t \in \Sigma^*, (st \in L)\}$$

We call a language $L \subseteq \Sigma^*$ prefix closed if $L = \overline{L}$. From this definition it is clear that $L(G) = \overline{L(G)}$; the closed behavior of DES G is closed.

- M-**Closure**: For two languages, $L, M \subseteq \Sigma^*$, L is $M - closed$ if

$$L = \overline{L} \cap M$$

- **Natural Projection** : Assuming that $\Sigma_o \subseteq \Sigma^*$ we define the natural projection $P_o : \Sigma^* \rightarrow \Sigma_o^*$ according to :

$$P_o(\epsilon) = \epsilon$$

$$P_o(\sigma) = \epsilon \quad \text{if } \sigma \notin \Sigma_o$$

$$P_o(\sigma) = \sigma \quad \text{if } \sigma \in \Sigma_o$$

and

$$P_o(s\sigma) = P_o(s)P_o(\sigma) \qquad \text{for } \sigma \in \Sigma \text{ and } s \in \Sigma^*$$

- **Selfloop** : Consider DES $G = (Q, \Sigma, \delta, q_0, Q_m)$ defined over alphabet Σ. Let Σ' be disjoint from Σ : $\Sigma \cap \Sigma' = \emptyset$. The $G' =$ **selfloop**(G, Σ') is formed from G by attaching a transition $q \xrightarrow{\sigma} q$ at each state q of G, for each $\sigma \in \Sigma'$.

- **Parallel Product (meet)** : Consider two DES G_1 and G_2 shown below :

$$G_1 = (Q_1, \Sigma_1, \delta_1, q_{01}, Q_{m1})$$

$$G_2 = (Q_2, \Sigma_2, \delta_2, q_{02}, Q_{m2})$$

The **meet** product of G_1 and G_2 (denoted by **meet** (G_1, G_2)) is the reachable automaton:

$$G_1 \times G_2 = (Q, \Sigma, \delta, q_0, Q_m)$$

where:

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

$$Q = Q_1 \times Q_2$$

$$Q_0 = (q_{01}, q_{02})$$

$$Q_m = Q_{m1} \times Q_{m2}$$

$$\delta((q_1, q_2), \sigma) = \begin{cases} (\delta_1(q_1, \sigma), \delta_2(q_2, \sigma)) & \text{if } \delta_1(q_1, \sigma)! \text{ and } \delta_2(q_2, \sigma)! \\ \textit{not defined} & \textit{otherwise} \end{cases}$$

In other words, if event $\sigma \in \Sigma_1 \cap \Sigma_2$ is defined and enabled in both G_1 and G_2 it can occur in **meet** (G_1, G_2) and it will be disabled if $\sigma \notin \Sigma_1 \cap \Sigma_2$. We can show that:

$$L(G) = L(G_1) \cap L(G_2)$$

$$L_m(G) = L_m(G_1) \cap L_m(G_2)$$

- **Synchronous Product (sync)**: Let G_1 and G_2 be two DES defined over alphabets Σ_1 and Σ_2 :

  $$G_1 = (Q_1, \Sigma_1, \delta_1, q_{01}, Q_{m1})$$

  $$G_2 = (Q_2, \Sigma_2, \delta_2, q_{02}, Q_{m2})$$

  The synchronous product of G_1 and G_2, **sync** (G_1, G_2) (or $G_1 \| G_2$) is the reachable sub-automaton of:

  $$G_1 \otimes G_2 = (Q, \Sigma, \delta, q_0, Q_m)$$

  where:

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

  $$Q = Q_1 \times Q_2$$

  $$Q_0 = (q_{01}, q_{02})$$

  $$Q_m = Q_{m1} \times Q_{m2}$$

and

$$\delta((q_1,q_2),\sigma) = \begin{cases} (\delta_1(q_1,\sigma),\delta_2(q_2,\sigma)) & \text{if } \sigma \in \Sigma_1 \cap \Sigma_2 \text{ and } \delta_1(q_1,\sigma)!, \delta_2(q_2,\sigma)! \\ (\delta_1(q_1,\sigma),q_2) & \text{if } \sigma \in \Sigma_1 - \Sigma_2 \text{ and } \delta_1(q_1,\sigma)! \\ (q_1,\delta_2(q_2,\sigma)) & \text{if } \sigma \in \Sigma_2 - \Sigma_1 \text{ and } \delta_2(q_2,\sigma)! \\ \textit{not defined} & \textit{otherwise} \end{cases}$$

Both **meet** and **Sync** products are associative. Therefore, for example,

$$\textbf{sync}(G_1, G_2, G_3, \ldots, G_n)$$

is well-defined. The **meet** and **Sync** operators are usually used to build models for DES from models of their DES components.

2.3 Supervisory Control

Ramadge-Wonham [RW] supervisory control theory provides a framework for the control of discrete-event systems (DES). These systems are modelled as automata which can generate formal languages.

Given a DES $G = (Q, \Sigma, \delta, q_0, Q_m)$ as plant, the legal marked behavior is specified by a language $L_m(E) \subseteq L_m(G)$. DES G satisfies the specification of the desired behavior if $L_m(E) \subseteq L_m(G)$, otherwise we have to modify the system using supervisor so that the closed loop system satisfies the specification. The system under control will be denoted by: S/G.

In the RW framework, the supervisor monitors the sequences of events generated by the plant and by disabling a subset of controllable events at each point may restrict the system behavior so that the system under supervision satisfies the specification:

$$L_m(S/G) \subseteq L_m(E)$$

Figure 2.1 shows the closed-loop system. As plant G generates the string $s = \sigma_1\sigma_2\ldots\sigma_n$, the supervisor S tells the plant what (controllable) events are allowed in the next step, and in this way the plant behavior is restricted by S. As mentioned earlier, the plant under supervision will be shown by: S/G.

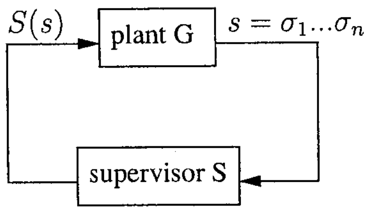


Figure 2.1: Closed Loop System

The supervisor S can be regarded as a map $S : L(G) \rightarrow 2^{\Sigma}$, such that for $s \in L(G)$, $S(s)$ is the set of events permitted by the supervisor S to happen generated by the plant G. For the system under supervision, $L(S/G)$ the language generated by the system under supervision (S/G) will be defined as follows:

- $\epsilon \in L(S/G)$
- $s \in L(S/G)$, $s\sigma \in L(G)$ and $\sigma \in S(s) \Rightarrow s\sigma \in L(S/G)$
- no other string belongs to $L(S/G)$.

It is obvious that $L(S/G) \subseteq L(G)$, and according to above definition we can easily show that $L(S/G)$ is closed:

$$L(S/G) = \overline{L(S/G)}$$

The marked behavior of S/G is defined according to:

$$L_m(S/G) = L_m(G) \cap L(S/G).$$

$L_m(S/G)$ consists of those sequences in the marked behavior of plant G that survive after supervision by S.

A supervisor S is called a *nonblocking supervisor* if the following holds:

$$\overline{L_m(S/G)} = L(S/G).$$

If a supervisor does not attempt to disable any uncontrollable event, it will be called an *admissible* supervisor. It can be shown that S is admissible if and only if $L(S/G)$ (or $L(S)$) is controllable.

- **Controllability**: Consider a DES $G = (Q, \Sigma, \delta, q_0, Q_m)$ with event set Σ partitioned into *controllable* and *uncontrollable* events: $\Sigma = \Sigma_c \cup \Sigma_{uc}$. Let $K \subseteq \Sigma^*$, language K is said to be *controllable* with respect to the plant G if:

$$\overline{K}\Sigma_{uc} \cap L(G) \subseteq \overline{K}$$

Given a DES $G = (Q, \Sigma, \delta, q_0, Q_m)$ as plant and legal marked behavior E we define $C_{in}(E)$ to be the set of all sublanguages of E which are controllable with respect to G:

$$C_{in}(E) = \{K \subseteq E \mid K \text{ is controllable wrt. } G\}$$

The *supremal controllable sublanguage* of E exists and is the union of all controllable sublanguages of E [19]:

$$supC_{in}(E) = \bigcup\{K \mid K \in C_{in}(E)\}$$

Assuming E is $L_m(G)$-closed and regular, a trim DES S that marks $supC_{in}(E)$ is the realization of a (minimally restrictive) supervisor and the system under supervision S/G is represented by $\mathbf{meet}(S, G)$.

Modular Control

In designing controllers using supervisory control theory, a modular approach is chosen mainly for the following advantages: a modular approach is usually easier for design and implementation purposes compared to a centralized approach; modules may easily be modified; and a modular approach can reduce the computational complexity significantly.

Given a DES G (with m states) as plant and legal marked behavior E (with n states), the computational complexity of designing an admissible supervisor based on $SupC_{in}(E)$ will be $O(mn)$. This complexity can be reduced using a modular approach and it can be justified as follows.

The modular control is referred to the situation where the control action of a supervisor S is given by some combination of the control actions of two or more supervisors. Given admissible supervisors S_1 and S_2 each defined for DES G (with m states), we define S as the joint product of S_1 and S_2:

$$S{=}\mathbf{meet}(S_1, S_2)$$

If we consider design specifications as SPEC1 (n_1 states) and SPEC2 (n_2 states), for S_1 and S_2, then the computational complexity for the synthesis of S_1 and S_2 will be

$O(n_1m)$ and $O(n_2m)$, respectively. Therefore we will have computational complexity of $O(\max(n_1, n_2)m)$ following a modular approach for synthesis of S. This computational complexity for supervisor synthesis using the monolithic approach would be $O(n_1n_2m)$[3]. If S_1 and S_2 have p_1 and p_2 states respectively, implementing S_1 and S_2 in a modular form requires computer memory for storing $p_1 + p_2$ states (and their transitions), whereas implementing S would require p_1p_2 states.

2.4 TTCT Program

TTCT is a program which is used for the synthesis of supervisory controls for DES. The TTCT procedures which are used in this thesis will be explained during the discussions. For a complete list of procedures, the reader is referred to [21]. Here we give only a brief review of this program. In this program, generators are represented by a five tuple:

[size, init, mark, voc, trans]

Size is the number of states (state set is 0, ..., size-1), init is the initial state (always 0), mark is the set of marked states, voc is the set of vocal states, and trans is the transitions. Vocal states will be pairs shown by [q,t] meaning that we have output t (positive integer) at state q. A transition is a triple [j,e,k] representing a transition from state "j"(exit) to state "k"(entrance) and having "e" as event label. Depending on whether the event is controllable or uncontrollable, "e" will be an odd or even nonnegative integer number respectively.

- **Computation of Supervisor**

Consider a DES $G = (Q, \Sigma, \delta, q_0, Q_m)$ as plant, and the design specification in the form of a language $L(SP) = L_m(SP) \subseteq \Sigma^*$ where SP is an automaton $E =$

$(Y, \Sigma, \eta, y_0, Y)$. According to RW supervisory control theory, the proper supervisor can be obtained by finding the supremal controllable sublanguage of the marked legal language: $L_m(G) \cap L_m(SP) = L_m(E)$.

In TTCT this can be computed using the **supcon** procedure of this program. Let: $S=$ **supcon** (G, E), the **supcon** procedure forms a trim recognizer S such that:

$$L_m(S) = supC_{in}(L_m(G) \cap L_m(SP))$$

and

$$L(S) = \overline{L_m(S)}.$$

- **Verification of Supervisor**

Given a DES $G = (Q, \Sigma, \delta, q_0, Q_m)$ as plant and its corresponding supervisor $S = (Y, \Sigma, \eta, y_0, Y)$, we like to know if S is a *proper* supervisor or not. For a supervisor S to be *proper* the following conditions should hold:

(i) **Admissability**

This condition holds if S does not attempt to disable any uncontrollable event. In other words, we need to check if $L_m(S) = L(S)$ is controllable with respect to $L(G)$ and Σ_{uc}.

In TTCT this can be verified by using the **condat** procedure. **condat** returns control data DAT for the supervisor S of the controlled system. The **condat** will display a list of events that must be disabled at each state of S.

SDAT=**condat**(G,S)

If all events to be disabled are controllable, $L(S)$ is controllable, and S admissible.

(ii) **Nonblocking**

For this property to hold we need to show that the supervisor S is a nonblocking supervisor. In other words we have to check the nonblocking property of the plant under the supervision (S/G).This can be verified in TTCT using **nonconflict** procedure:

nonconflict(G,S)=true

If the above equation holds S will be a nonblocking supervisor, and the plant under supervision (S/G) is nonblocking.

(iii) **Trim**

For S to be a trim supervisor it needs to be both reachable and coreachable. This can be checked in TTCT by using two procedures: **trim** and **isomorph**. By applying **trim** procedure to S, we get a trim generator for S. If **trim** (S) and S are identical up to renumbering of states verified by **isomorph** procedure, then we call S a **trim** supervisor. This can be shown by following:

TRIMS=**trim**(S)

isomorph(TRIMS,S)=identical

2.5 Hierarchical Control of DESs

In this section, we briefly review the hierarchical control of DESs introduced in [25] and [26] and later developed in [24]. A *two-level hierarchical control structure* is studied and some important results relevant to our later discussions will be discussed.

2.5.1 Two-Level Controlled DES

A two-level model is shown in Figure 2.2. A low-level plant G^{lo} is controlled by a high-level supervisor S^{hi} and a low-level operator S^{lo}. The abstraction of plant is called G^{hi}, which is controlled by a high-level supervisor called S^{hi}.

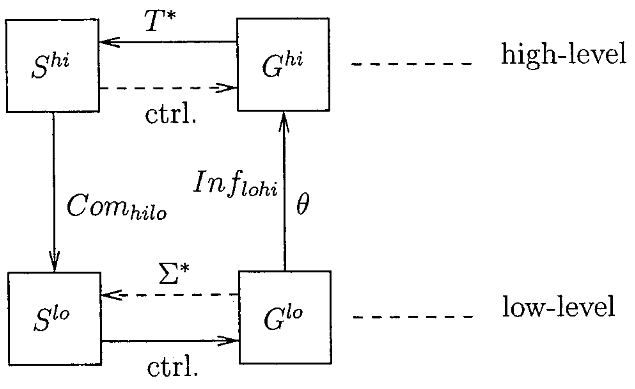


Figure 2.2: two-level hierarchical control structure

The events at the high level are reported through an information channel from low-level plant. In this hierarchical structure the control of S^{hi} is virtual, since G^{hi} is an abstract model driven from low-level plant via information channel (Inf_{lohi}). The control commands of S^{hi} is transmitted to S^{lo} through command channel (Com_{hilo}), while G^{lo} is actually controlled by S^{lo}. The behavior of G^{hi} is controlled by S^{hi} via control channel, and the behavior of G^{hi} is determined by the behavior of G^{lo} through information channel (Inf_{lohi}).

The plant at the low level can be thought of as a standard DES: $G^{lo} = (Q, \Sigma, \delta, q_0, Q_m)$, where $\Sigma = \Sigma_c \cup \Sigma_{uc}$. The set of events at the high-level is denoted by T, which is a nonempty set of labels of "significant" events. T can be considered as events which are entered into the description of the high-level plant model G^{hi}.

The high-level plant G^{hi} is defined using a causal map $\theta : L(G^{lo}) \rightarrow T^*$, with the following properties:

$$\theta(\epsilon) = \epsilon,$$

$$\theta(s\sigma) = \begin{cases} \text{either} \;\; \theta(s) & \\ \text{or} \;\; \theta(s)\tau, & \text{for some } \tau \in T \end{cases}$$

where $s \in \Sigma^*$ and $\tau \in T$ and ϵ denotes the empty string.

The high-level plant can be visualized if we consider a *vocalized* state set for G^{lo}. Let $\tau_0 \notin T$ denote "silent" event symbol and $T_0 = \{\tau_0\} \cup T$. To this end, now we are ready to define the *output function map* $\omega : Q \rightarrow T_0$ such that $\omega(q_0) = \tau_0$ and for any other state q reachable via the string $s\sigma$ we will have the following:

$$\omega(q) = \tau \text{ if } \theta(s\sigma) = \theta(s)\tau, \quad \omega(q) = \tau_0 \text{ if } \theta(s\sigma) = \theta(s).$$

In other words ω outputs the silent output τ_0 if θ outputs nothing new, and outputs fresh symbol τ otherwise. Events which are not silent are called "vocal" events. In simple words we can say that each state of G^{lo} contains a high-level event which is sent to output when the state is reached. Figure 2.3 shows a simple low-level plant G^{lo} along with its high level realization using above method. The silent event τ_0 is ignored by the high-level plant.

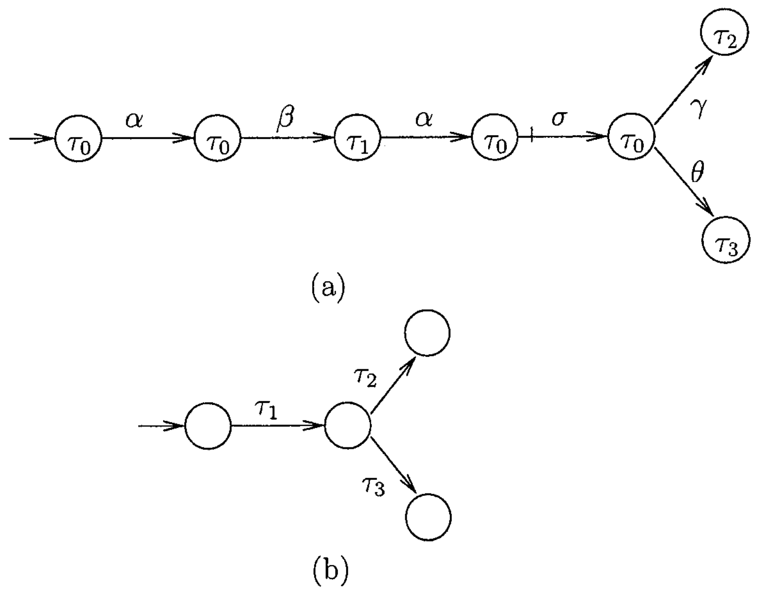


Figure 2.3: Low-level plant G^{lo} (a), and its derived high level model G^{hi} (b). States of G^{lo} are shown by the output of function $\omega(q)$.

At this point we can denote the low-level plant with the following unified description:

$$G^{lo} = (Q, \Sigma, T_0, \delta, \omega, q_0, Q_m)$$

with $Q_m = Q$.

The closed behavior of high-level system G^{hi} is shown by $L(G^{hi})$ and it will be the image of the low-level language $L(G^{lo})$ under θ :

$$L(G^{hi}) = \theta(L(G^{lo})) \subseteq T^*$$

2.5.2 High-Level Control Structure

In order to define a supervisor for the high-level system we need to partition the high-level alphabet T to controllable and uncontrollable events: $T = T_c \cup T_{uc}$. However, it may not be possible to derive this information easily from the low-level alphabet Σ. For example consider the high and low-level plants shown in Figure 2.4.

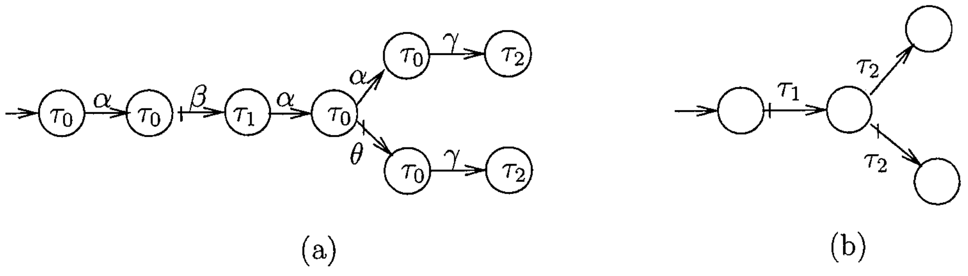


Figure 2.4: (a) G^{lo} with $\Sigma_c = \{\beta, \theta\}$, and $\Sigma_{uc} = \{\alpha, \gamma\}$; (b) G^{hi}

It is clear from this example that event τ_2 in high-level alphabet T, can be considered as both controllable and uncontrollable event.

To overcome the above-mentioned problem the notion of *output control consistency* is introduced: (G^{lo}, θ) is *output control consistent* if for any $\tau \in T$, it will be unambiguous if τ is controllable or uncontrollable.

If a hierarchical system (G^{lo}, θ) is not output control consistent, it can be converted to an output control consistent DES over a possibly new alphabet from T [26], [23]. For this any ambiguous event τ will be replace with new controllable τ_c and uncontrollable τ_{uc} events, and θ will be adjusted accordingly and as a result, the system becomes output control consistent. In TTCT the **vocalize** procedure is used to vocalize a low-level system and procedure **outconsis** will be used to compute an output control consistent system from a low-level system[1]. The equivalent high-level system can be computed using **higen** procedure. The following result, describes the effect of the hierarchical control scheme.

Theorem 2.5.1 *[26] Consider the high-level system G_{hi} and a prefix closed specification language shown by $E^{hi} \subseteq L(G^{hi})$, assumed to be controllable with respect to $L(G^{hi})$. Let $E^{lo} = \theta^{-1}(E^{hi})$ and assume that the low-level system G^{lo} is output control consistent. Then there exists a high-level supervisor S^{hi}, where implementation in the low-level system by a supervisor S^{lo} will result in having: $L(S^{lo}/G^{lo}) = supC_{in}(E^{lo})$.* $\square$

It can be shown that $\theta(supC_{in}(E^{lo})) \subseteq E^{hi}$. In other words, only a subset of high level specifications can be realized in the low-level plant due to the low-level control. If the (virtual) specifications in the high-level can be achieved in the low-level plant, the system is called *hierarchically consistent*, i.e., $\theta(supC_{in}(E^{lo})) = E^{hi}$. *Strict output control consistency is a sufficient condition for having hierarchical consistency.* The low-level system is strictly output control consistent if it is output control consistent and it allows us to enable or disable any controllable high-level event without affecting other high-level events. In TTCT the procedures **hiconsis** and **higen** directly compute low-level and high-level systems which are hierarchically consistent.

[1]If G^{lo} has n states and m transitions, the complexity of the combined algorithms (for output-control consistency and for strict output-control consistency) is $O(n^3m + n^2m^2)$[23].

Chapter 3

A Modular Hierarchichal Approach to the Fault Recovery Problem in DESs

In this chapter, we study the synthesis of fault recovery procedures using discrete-event models. The problem will be formalized and a hierarchical modular solution will be presented. Specifications for each mode (normal, recovery), will be divided into two main groups depending on the scope of the specifications. In each mode, we consider the first group of the specifications as those for which we can find a solution at low-level, while the second group are those for which we can design a supervisor following a hierarchical approach. For each category of specifications, the appropriate supervisors will be designed to satisfy the design requirements.

For designing the low-level supervisors, a modular approach, similar to the one introduced by [13] for the fault recovery problem, will be followed. Here we assume full event observation. Therefore, failure events are considered observable and no diagnosis system is included in modelling of the plant. Following this, a two-level modular hierarchical solution for the fault recovery problem in DESs will be proposed in order to design the high-level supervisors of the plant. Throughout this chapter examples will be accompanied with the discussions to illustrate the ideas.

3.1 Introduction

As modern control systems become more and more complex, the need for having more reliable supervisors (controllers) has been increased significantly, especially in the event of failures happening in the systems. The supervisors are required to take the necessary actions for the system's safe operation in the presence of failures. Therefore, the controllers may need to take some recovery actions and this clearly shows why failure recovery is considered as an important issue for any control system.

For example, consider an electrical generator which supplies a power network through a circuit breaker. If for any reason a fault occurs for the generator, the breaker should be opened by its controlling relay. In case of having this failure in the system, the supervisor may decide to take the recovery action of redirecting the supply from a different generating unit by closing another circuit breaker. As a result of this recovery action, there will not be any shortage of electrical power in the related network.

We assume the system which needs to be controlled can be modelled as a finite-state automaton (G) and the failure recovery problem can be considered as a control problem in the RW supervisory control framework. The supervisors' design will be discussed for plant's normal operation and also for recovery from the failures in both low-level and high-level models.

The plant model includes failure events. We assume any failure in the system can be modelled as an uncontrollable, but observable event. If a fault occurs in the plant, without any delay or considerably small delay (ignorable), the system will enter a *faulty-state* and recovery actions are to be initiated (recovery mode). The state transition graph of the plant is shown in Figure 3.1.

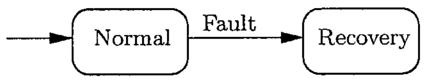


Figure 3.1: State transition graph of the plant to be controlled.

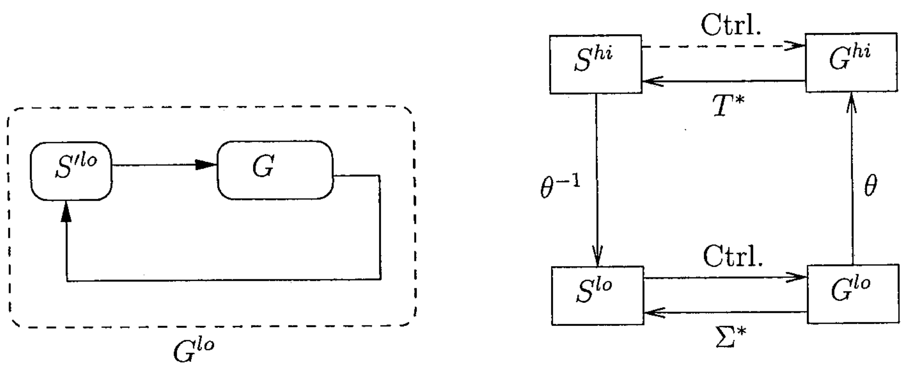
The specifications of each mode (normal, recovery), as mentioned earlier, can be classified into two main categories according to their scopes. The first category are considered as those for which we can design a low-level supervisor. Intuitively, these specifications concern local issues. On the contrary, the second category of specifications are considered as those depending on their scope, for which we can find a simplified high-level solution; thus a hierarchical solution. Intuitively, these specifications concern system-wide (global) issues. In extreme cases one mode may only have specifications from one category only.

By following the above mentioned approach, the second group of the specifications can be dealt with in a higher level using an abstracted model of the plant, in order to simplify the synthesis of the required supervisors. The main reason for this simplicity of controllers' design at high-level is the filtering of the irrelevant information from low-level plant, which are not needed and used in the high-level system model. This results in a much simpler problem at high-level. In other words, an abstraction of the plant at high-level can largely decrease the behavior complexity for designing supervisors.

As mentioned before, the plant under supervision can operate in two modes: normal and recovery. For each mode of operation we can categorize the specifications into two groups such that each group can be dealt with using a low or high-level supervisor. For the first category of the design specifications, we propose a modular switching supervisory scheme (S'^{lo}) to enforce the desired specifications in both normal and recovery modes at low-level. In this case, if a fault occurs, the system will enter the recovery mode and a low-level recovery supervisor will be in charge of taking necessary actions. After computing the plant under supervision at low-level (G^{lo}), the high-level model of the supervised plant (G^{hi}) and also the high-level versions of the second category of the specifications will be derived. Accordingly, normal and recovery supervisors will be designed and a modular switching supervisory control scheme (S^{hi} along with low-level operator S^{lo}), will be developed for G^{hi} (and G^{lo}). Therefore, the conjunction of S'^{lo} and S^{hi} (along with S^{lo}), from two feedback loops, with S'^{lo} in the inner loop and S^{hi} in the outer loop. The plant and supervisors are shown in Figure 3.2.

Therefore, a *two-level modular switching supervisory control scheme* is proposed for the total system under supervision. In general modular control is preferred to monolithic supervisory control because:

1. Supervisor's design in a modular fashion is usually easier, compared to the centralized case.



(a) (b)

Figure 3.2: (a) Inner feedback loop; (b) Outer feedback loop.

2. Modular design are usually easier to maintain and modify.

3. For the problem of fault recovery, where the number of recovery modes could be high, a modular approach can reduce the computing requirements for implementation significantly.

The only problem with modular approach , is the problem of interference among different supervisors, which needs to be taken into the account. Specifically the overall system under supervision may fail to be nonblocking. This issue will be addressed later in the chapter.

Moreover, a two-level hierarchical approach is chosen to reduce the complexity of the controller design. This especially is noticeable in systems with a large number of subsystems and failures, where even the conventional modular approach, may also fail to deal adequately with the complexity and computational problems of control system design. In other words, in addition to taking advantage of modular approach for flat systems, the problem will be further broken down by proposing a solution which uses vertical modularity. It should be noted that nothing will stop us from extending this two-level hierarchy to a multi-layer modular hierarchy model [17]. Therefore, giving more flexibility to the designers to deal with the complexity of the control problems in general.

3.2 Problem Formulation

We start this section by modelling the plant. Models of the plant for both normal and recovery modes will be discussed and the system to be controlled will be introduced and formalized. Following this, the control objectives will be specified.

3.2.1 Plant Model

We assume that the original plant can be modelled as a DES $G = (Q, \Sigma_G, \delta, q_0, Q_m)$, where Q is the set of states, $\Sigma_G = \Sigma_{G,c} \dot{\cup} \Sigma_{G,uc}$ the set of controllable and uncontrollable events ($\dot{\cup}$ is disjoint union), $\delta : Q \times \Sigma_G \rightarrow Q$ is the partial transition function, $q_0 \in Q$ initial state, and $Q_m \subseteq Q$ is the subset of marked states. Both, normal and faulty situations can be described by this model. The event set of the plant (Σ_G) can be partitioned: $\Sigma_G = \Sigma_p \dot{\cup} \Sigma_f$. Σ_f is the set of failure events: $\Sigma_f = \{f_1, \ldots, f_m\}$. The failure events are uncontrollable. Σ_p will contain the remaining events of the plant. Thus there are m failure (recovery) modes: $R_1, \ldots, R_m$, with $m \geq 1$. It is also assumed that all failures in the system are permanent. In other words, in case of occurrence of any failure in the plant, it will remain in the faulty condition. Furthermore, the *single-failure scenario* is assumed, meaning that only one failure mode may occur during the plant's operation. Therefore, the plant can operate in $m + 1$ modes: N (normal), and $R_1, \ldots, R_m$ as recovery (failure) modes. Accordingly, we can partition the state set of the plant: $Q = Q_N \dot{\cup} Q_R$. Where, Q_R further can be partitioned into the following: $Q_R = Q_{R_1} \dot{\cup} \ldots \dot{\cup} Q_{R_m}$. [1] These blocks are shown in Figure 3.3.

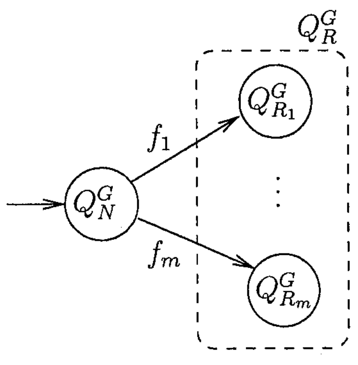


Figure 3.3: State partition diagram of the plant to be controlled.

Finally in this part, we define different sub-generators of the system according to the above partitions. Let G_N be the sub-generator of G correspond to Q^G_N. G_N will describe the plant's behavior in its normal mode. G_{NR} will correspond to $Q^G_N \dot{\cup} Q^G_R$. Also we define sub-generators G_{NR_i} to correspond to the states $Q^G_N \dot{\cup} Q^G_{R_i}$ $(i = 1, \ldots, m)$. Note that, G_N and G_{NR_i} are considered as low-level models of the plant, for which a low-level supervisor can be designed to enforce the relevant low-level specifications.

3.2.2 Fault Recovery Problem

It is assumed that design specifications are available in the form of legal languages $E_N \subseteq L_m(G_N)$ and $E_{R_i} \subseteq L_m(G_{NR_i})$ for the modes N, $R_1, \ldots, R_m$. The specifications

[1] In this thesis full event observation is assumed; therefore upon occurrence of a failure, the failure event is immediately observed and the system can enter recovery mode. If events were unobservable failure and we considered a bounded detection delay for failure events, we would have to include a transient mode for our model as well. In that case, the system would enter the recovery mode with a bounded delay.

for each mode can further be categorized into two groups according to: $E_N = E_N^{\prime lo} \dot{\cup} E_N^{lo}$ and $E_{R_i} = E_{R_i}^{\prime lo} \dot{\cup} E_{R_i}^{lo}$ for $(i = 1, \ldots, m)$. The first group are those for which we can design a supervisor at the low-level, whereas the second group are those specifications for which a hierarchical solution can significantly simplify the design procedures to obtain a high-level supervisor. Low-level specifications $E_N^{\prime lo}$ and $E_{R_i}^{\prime lo}$ typically are related to subsystems (local) issues and the second group of specifications (E_N^{lo} and $E_{R_i}^{lo}$) for which we design high-level supervisors, describe system-level (global) behaviors. (In extreme cases, one mode may have specifications from one group of specification only.) The reason for this partitioning will be discussed in details in the next section. The normal specification (E_N) does not include any failure events, ($E_N \subseteq \Sigma_P^*$). For recovery specifications we have: $E_{R_i} \subseteq (\Sigma_P \dot{\cup} \{f_i\})^*$.

Given the DES G and the set of specifications in the form of legal languages E_N and E_{R_i}, for $i = 1, \ldots, m$, we like to design an admissible supervisor S such that it can enforce the required design specifications for normal and recovery modes and the system under supervision (closed-loop system) is nonblocking in both modes. Specifically, the following conditions should hold:

(1a) $L_m(S/G_N) \subseteq E_N$

(1b) $L_m(S/G_{NR_i}) \subseteq E_{Ri} \qquad (i = 1, \ldots, m)$

(2a) $\overline{L_m}(S/G_N) = L(S/G_N)$

(2b) $\overline{L_m}(S/G_{NR_i}) = L(S/G_{NR_i}) \qquad (i = 1, \ldots, m)$

Note that during recovery procedure, the supervisor knows which fault has occurred and for each failure mode a different recovery strategy may be adopted. In the following section we will propose a *hierarchical modular switching* solution to solve the fault recovery problem. First, we design the required supervisors at low-level to enforce the first group of the specifications (E'^{lo}_{N}, $E'^{lo}_{R_i}$) by using a modular switching scheme. A *two-level modular hierarchical* solution will be presented to enforce the second group of the specifications (E^{lo}_{N}, $E^{lo}_{R_i}$) afterwards.

3.3 Supervisor and Recovery Procedure Design

The design procedures and proposed solution for the fault recovery problem will be described in this section. First, we consider a plant with only one permanent failure event. The multiple failure case will be discussed later in the section. A computational procedure for the general case of fault recovery problem using proposed solution will also be presented. Finally, we will finish this chapter by an illustrative example.

3.3.1 Plant with one Permanent Failure Event

Figure 3.4 shows a plant with one permanent failure mode. f_1 is considered as the only failure event of the system ($m = 1$).

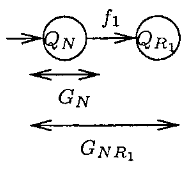


Figure 3.4: Plant G with one failure event f_1 .

Note that for simplicity we have dropped the superscript G in state sets. Let $E_N = E'^{lo}_{N} \dot{\cup} E^{lo}_{N}$ denote the specification for normal mode and $E_{R_1} = E'^{lo}_{R_1} \dot{\cup} E^{lo}_{R_1}$ the design specifications for the recovery mode R_1. We propose a two-step solution for the problem of supervisor design. First we design a supervisor (denoted by $S'^{lo}_{NR_1}$) to enforce E'^{lo}_N and $E'^{lo}_{R_1}$. Next, we construct the closed-loop plant $G_{lo} = S'^{lo}_{NR_1}/G$. Following this, we use a hierarchical approach to design a high-level controller S^{hi} to enforce the second group of the specifications E^{lo}_N and $E^{lo}_{R_1}$. For the low-level plant, we design the following supervisors to enforce the first group of the specifications E'^{lo}_N and $E'^{lo}_{R_1}$:

(1a) S'^{lo}_N: a nonblocking supervisor designed for G_N to enforce E'^{lo}_N;

(1b) $S'^{lo}_{R_1}$: a nonblocking supervisor designed for G_{NR_1} to enforce $E'^{lo}_{R_1}$.

In the proposed solution, initially in the normal mode, both S'^{lo}_N and $S'^{lo}_{R_1}$ are in the feedback loop. The reason for involving $S'^{lo}_{R_1}$ in controlling of the plant from beginning is that (in case of failure) to be able to meet recovery specifications, since some control actions may be required from the normal mode. Once failure f_1 occurs, the normal supervisor S'^{lo}_N is taken out of the loop and the recovery supervisor $S'^{lo}_{R_1}$ takes control of the plant G (Figure 3.5). A simple method to capture the switching of S'^{lo}_N is as follows.

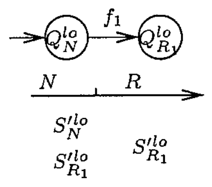


Figure 3.5: Low-level supervisors S'^{lo}_N and $S'^{lo}_{R_1}$ used in controlling of the plant G.

We construct a supervisor $\tilde{S}'^{lo}_N$ with the following method: we will add a marked dump state x_d to S'^{lo}_N; from every state q in S'^{lo}_N, we add a transition (q, f_1, x_d) to the dump state x_d. We also attach a transition (x_d, σ, x_d) for all $\sigma \in \Sigma_p$. The control action of $\tilde{S'}^{lo}_N$ will be identical to S'^{lo}_N during the normal mode and once f_1 occurs, $\tilde{S'}^{lo}_N$ enters state x_d) and will be effectively disabled and therefore it will be out of the control loop. Figure 3.6 shows the modified supervisor $\tilde{S'}^{lo}_N$.

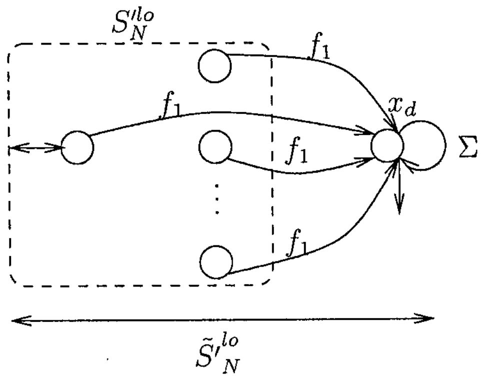


Figure 3.6: Modified supervisor $\tilde{S'}^{lo}_N$.

Let $S'^{lo}_{NR_1}$=**meet**($\tilde{S'}^{lo}_N$, $S'^{lo}_{R_1}$). $S'^{lo}_{NR_1}$ models the joint operation of $\tilde{S'}^{lo}_N$ and $S'^{lo}_{R_1}$ at low-level. The system under supervision of the low-level supervisors will be: G^{lo}=$S'^{lo}_{NR_1}$/ G_{NR_1}=**meet**($S'^{lo}_{NR_1}$,G_{NR_1}). This completes the first step of the design.

Next we design a hierarchical control system to enforce the specification $E^{lo}_N \subseteq \Sigma^*_P$ and $E^{lo}_{R_i} \subseteq (\Sigma_P \dot{\cup} \{f_i\})^*$, for $(i = 1, \ldots, m)$, with G^{lo} as the low-level plant. The first step is to vocalize G^{lo} and eventually, build the high-level model G^{hi}.

The vocalization procedure may vary depending on the problem and specified design requirements. We consider all important events in the low-level which need to be signalled to the high-level plant model as "significant" events. Some general rules which can be considered in selecting these events in the low-level, used in our proposed hierarchical solution, are as follows:

1. We consider all failure events of the plant (at low-level) as "significant" events, which have to be signalled to the high-level model. This means that for each failure event at low-level, we have at least a corresponding output which needs to be considered in our hierarchical model.

2. The rest of the "significant" events of the plant, which need to be signalled to the high-level model, are chosen according to the second group of the specifications expected to be satisfied.

According to the above-mentioned vocalization rules, for the case of the plant with only one failure event (f_1), all states of the low-level plant which are the destination of a f_1 transition, are vocalized by a high-level event, say τ_1.

In general, if we have $\Sigma_F = \{f_1, \ldots, f_m\}$ as the set of failure events of the system, then we define the following high-level alphabet set: $T = \{\tau_1, \ldots, \tau_m, \ldots, \tau_n\}$ ($n \geq m$), with $T = T_P \dot{\cup} T_F$ where $T_P = \{\tau_{m+1}, \ldots, \tau_n\}$ and $T_F = \{\tau_1, \ldots, \tau_m\}$ (high-level vocalized failure events). With the above sets of alphabet, we have: $E_N^{hi} \subseteq T_P^*$ and $E_{R_i}^{hi} \subseteq (T_P \dot{\cup} \{\tau_i\})^*$ $(i = 1, \ldots, m)$. Each failure scenario has a different recovery procedure and therefore, we may have a different recovery specification in each mode. Note that the target states of f_i transitions may have to be split in cases where the state is the destination of non-f_i transitions. This process is shown in Figure 3.7.

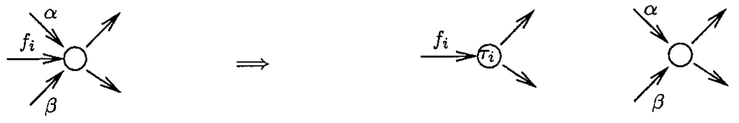


Figure 3.7: Vocalization of a target state with f_i and non-f_i transitions.

Going back to our discussion of the case with single failure, suppose G^{lo} has been vocalized. This will be our starting point for the hierarchical control design. Accordingly, the *output-control-consistent* high-level model G^{hi} and sub-generators, G^{hi}_N and $G^{hi}_{NR_1}$, are constructed by applying the proper algorithm ([23] and [25]) to get a high-level consistent model. These high-level sub-generators (G^{hi}_N and $G^{hi}_{NR_1}$) are shown in Figure 3.8.

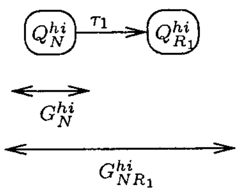


Figure 3.8: High-level plant G^{hi} with one failure event τ_1 (f_1).

Given E^{lo}_N and $E^{lo}_{R_1}$, the high-level specifications $E^{hi}_N = \theta(E^{lo}_N)$ and $E^{hi}_{R_1} = \theta(E^{lo}_{R_1})$ can be computed using high-level alphabet T and causal map θ. Finally the following supervisors will be designed for the high-level plant model:

(2a)S^{hi}_N: a nonblocking supervisor designed for G^{hi}_N to enforce E^{hi}_N;

(2b)$S^{hi}_{R_1}$: a nonblocking supervisor designed for $G^{hi}_{NR_1}$ to enforce $E^{hi}_{R_1}$.

Figure 3.9 shows the control sequence of the plant from a high-level point of view. Initially, the conjunction of S^{hi}_N and $S^{hi}_{R_1}$ controls the plant. If a failure event f_1 (corresponding to the high-level event τ_1) occurs, S^{hi}_N will be disabled and $S^{hi}_{R_1}$ will be engaged in controlling of the plant (in addition to $S'^{lo}_{R_1}$). Note that S^{hi}_N and $S^{hi}_{R_1}$ are high-level supervisors which are designed to satisfy the second group of the specifications for normal and recovery modes (N and R_1).

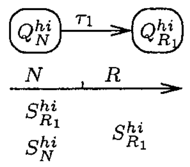


Figure 3.9: Supervisors S_N^{hi} and $S_{R_1}^{hi}$ used in controlling of plant G^{hi}.

The control mechanism for the high level will be the same as what we had for the low-level. Similar to the low-level controllers, with some modifications to S_N^{hi}, we can construct a single automaton to represent the modular switching control at the high-level. First, we construct $\tilde{S}_N^{hi}$ by using a process similar to that shown in Figure 3.6. $\tilde{S}_N^{hi}$ will behave similar to S_N^{hi} during normal mode. We define the joint supervisor $S_{NR_1}^{hi}$=**meet** $(\tilde{S}_N^{hi}, S_{R_1}^{hi})$.

3.3.2 Plant with Multiple Failure Events

For the general case ($m \geq 1$), we will have m failure modes for the system, as shown in Figure 3.3. In this case, the required controller design procedure will be as follows:

Low-level: A nonblocking supervisor $S_N^{\prime lo}$ will be designed similar to the case of $m = 1$ to enforce $E_N^{\prime lo}$. We also design m different recovery supervisors $S_{R_1}^{\prime lo}, \ldots, S_{R_m}^{\prime lo}$, for the low-level recovery modes to enforce the recovery specifications $E_{R_1}^{\prime lo}, \ldots, E_{R_m}^{\prime lo}$, respectively, and to satisfy the nonblocking. Specifically, we can say that $S_{R_i}^{\prime lo}$ $(1 \leq i \leq m)$ is a supervisor designed for G_{NR_i} to enforce $E_{R_i}^{\prime lo}$, while satisfying the nonblocking requirement of $S_{R_i}^{lo}/G_{NR_i}$. In the normal mode, $S_{NR_i}^{\prime lo}$ =**meet**($S_N^{\prime lo}$, $S_R^{\prime lo}$), will control the plant, where $S_R^{\prime lo}$=**meet**($S_{R_1}^{\prime lo}, \ldots, S_{R_m}^{\prime lo}$). If a failure event f_i occurs, plant will enter the recovery mode R_i and only $S_{R_i}^{\prime lo}$ will remain in the feedback loop. Similar to the case of one failure, we can modify S'^{lo}_N and $S'^{lo}_{R_i}$ $(i = 1, \ldots, m)$ to obtain $\tilde{S}'^{lo}_N$ and $\tilde{S}'^{lo}_{R_i}$ such that the single generator S^{lo}_{NR}=**meet**($\tilde{S}'^{lo}_N, S'^{lo}_R$), with S'^{lo}_R=**meet**($\tilde{S}'^{lo}_{R_1}, \ldots, \tilde{S}'^{lo}_{R_m}$), represent the low-level supervisor. The procedures to obtain $\tilde{S}'^{lo}_N$ and $\tilde{S}'^{lo}_{R_i}$ are shown in Figures 3.10 and 3.11. Plant under supervision will be G^{lo}=**meet**(S^{lo}_{NR}, G) at low-level. By computing the closed loop system (G^{lo}) at low-level, we will be ready to apply our hierarchical approach to control the plant.

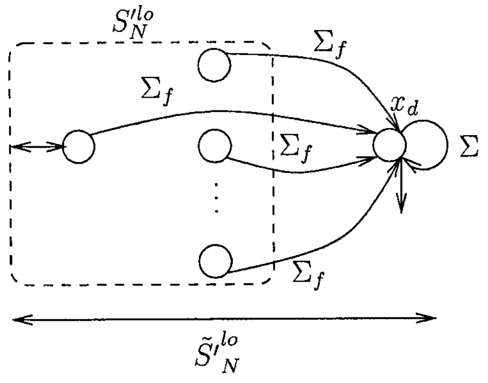


Figure 3.10: $\tilde{S}'^{lo}_N$.

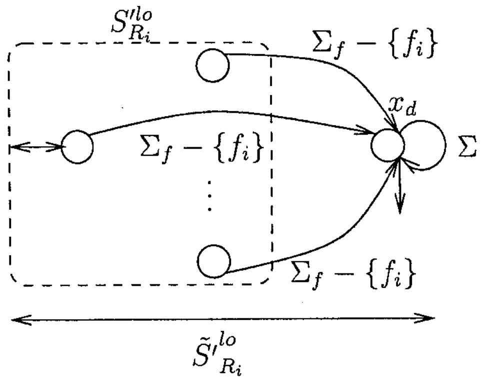


Figure 3.11: $\tilde{S}'^{lo}_{R_i}$.

High-level: We start modelling of our high-level system with G^{lo} as our starting point for the hierarchy control. We define the vocal states, using our proposed method for vocalization procedure and by selecting "significant" events of the plant. A consistent high-level model of the system will be derived by applying the proper algorithm proposed in [26] and as explained in Chapter 2. Similar to the case with one failure mode, given E_N^{lo} the high-level version of this specification $E_N^{hi} = \theta(E_N^{lo})$ will be computed using a proper causal map. It should be noted that initially the design specifications may be given in their high-level forms (E_N^{hi}, $E_{R_i}^{hi}$), in terms of sequences in T^*. Accordingly, we design S_N^{hi} to enforce E_N^{hi}.

The second category of the design specifications, $E_{R_1}^{lo}, \ldots, E_{R_m}^{lo}$, for all recovery modes are those for which the high-level recovery supervisors[2], $S_{R_1}^{hi}, \ldots, S_{R_m}^{hi}$, are designed to enforce the corresponding high-level recovery specifications $E_{R_1}^{hi}, \ldots, E_{R_m}^{hi}$, and also to satisfy the nonblocking requirements of each case. $E_{R_1}^{hi}, \ldots, E_{R_m}^{hi}$ will be the high-level versions of the low-level specifications: $E_{R_1}^{lo}, \ldots, E_{R_m}^{lo}$, which can be computed using a proper causal map θ ($E_{R_i}^{hi} = \theta(E_{R_i}^{lo})$). Thus $S_{R_i}^{hi}$ is a supervisor designed for $G_{NR_i}^{hi}$ to enforce $E_{R_i}^{hi}$, while ensuring the nonblocking property of $S_{R_i}^{hi}/G_{NR_i}^{hi}$ $(1 \leq i \leq m)$.

To have a single generator to represent the high-level switching controller, we apply a procedure similar to that depicted in Figures 3.10 and 3.11 to construct $\tilde{S}_N'^{hi}$ and $\tilde{S}_{R_i}^{hi}$ $(1 \leq i \leq m)$. Let S_R^{hi}=**meet**$(\tilde{S}_{R_1}^{hi}, \ldots, \tilde{S}_{R_m}^{hi})$, and S_{NR}^{hi}=**meet**$(\tilde{S}_N^{hi}, S_R^{hi})$. Thus S_{NR}^{hi} represent the modular switching high-level controller.The combined operation of low-level and high-level supervisors can be described as follows.

In the normal mode, the joint operation of $S_{NR}'^{lo}$ and S_{NR}^{hi} (and its low-level operator

[2] m, number of high-level recovery supervisors

S^{lo}_{NR}) will control the plant. If a failure event f_i corresponding to the high-level event τ_i occurs, the plant will enter the recovery mode R_i and a high-level supervisor $S^{hi}_{R_i}$ (and its low-level operator $S^{lo}_{R_i}$), along with $S'^{lo}_{R_i}$, will be the only supervisors which control the plant during the recovery procedures.

3.3.3 Computational Procedure Using TTCT

We conclude this section by discussing a computational procedure for implementation in TTCT which summarizes our proposed solution. The following algorithm lists all steps required to be taken, in order to apply our design methodology for the problem of the fault recovery using DES models. It should be mentioned that each problem has its unique features and following computational method may require to be considered by some adjustments for different problems and also specific design requirements. The idea is to list all general procedures which have been motivated by our proposed solution.

It is assumed that the plant has m permanent failure modes. We also assume that the design specifications are available in the form of legal languages and can be partitioned into two groups for normal and all recovery modes. The algorithm is stated in the following:

1. We obtain models of the different components of the plant. We refer to these components as $G_1, \ldots, G_n$. They will represent the basic description of system's components in both normal and faulty conditions.

2. The complete plant model (to be controlled) can be constructed using the synchronous product of plant and a *single-failure scenario* (SFS) generator. The SFS

for the general case is shown in Figure 3.12. SFS enforces the single failure assumption.

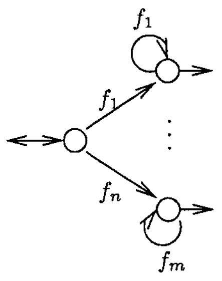


Figure 3.12: Single-failure scenario (SFS).

Using the **sync** procedure, we will obtain the general description of the plant G=**sync**$(G_1, \ldots, G_n, SFS)$ and also sub-generators $G_N, G_{NR_1}, \ldots, G_{NR_m}$.

3. Using the **supcon** procedure, we compute low-level supervisors $S_N^{\prime lo}$ and $S_{R_1}^{\prime lo}, \ldots, S_{R_m}^{\prime lo}$ to enforce the design specifications $E_N^{\prime lo}$ and $E_{R_1}^{\prime lo}, \ldots, E_{R_m}^{\prime lo}$, respectively, for $(G_N, G_{NR_1}^{lo}, \ldots, G_{NR_m}^{lo})$.
4. Using the **edit** procedure, modified supervisor generators $\tilde{S}_N^{lo}$ and $\tilde{S}_{R_i}^{lo}$ are constructed (as shown in Fig. 3.10 and 3.11 in Sec. 3.3.2).
5. The closed loop system will be computed at low-level: GLO=$S_{NR}^{\prime lo}/G$=**meet**$(G, S_{NR}^{\prime lo})$. Here $S_{NR}^{\prime lo}$=**meet**$(\tilde{S}_N^{\prime lo}, S_R^{\prime lo})$, and $S_R^{\prime lo}$=**meet**$(\tilde{S}_{R_1}^{\prime lo}, \ldots, \tilde{S}_{R_m}^{\prime lo})$.
6. After defining vocal states using **vocalize** procedure, closed loop system will be vocalized according to the proposed vocalization procedure and by selecting "significant" events of the system: VGLO=**vocalize**(GLO).
7. By applying the **hiconsis** and **higen** procedures, a high level consistent model of the plant is constructed: GHI=**higen**(**hiconsis**(VGLO)). Note that, if the plant

is already a *strictly output-control-consistent* plant then, we can directly use higen command of TTCT: GHI=**higen**(VGLO).

8. The high-level versions of the design specifications, E_N^{lo} and $E_{R_1}^{lo}, \ldots, E_{R_m}^{lo}$, will be derived by using a causal output map θ. E_N^{hi} and $E_{R_1}^{hi}, \ldots, E_{R_m}^{hi}$ are considered as high-level specifications for the normal and recovery modes. In some problems, the high-level specifications are available from the start and the mapping (θ) may not be required.

9. Using the **supcon** procedure, we compute S_N^{hi} and $S_{R_1}^{hi}, \ldots, S_{R_m}^{hi}$, to enforce the desired specifications E_N^{hi} and $E_{R_1}^{hi}, \ldots, E_{R_m}^{hi}$ for G_N^{hi}, $G_{R_1}^{hi}, \ldots, G_{R_m}^{hi}$, respectively.

10. The modified recovery supervisors $\tilde{S}_N^{hi}$ and $\tilde{S}_{R_k}^{hi}$ $(k = 1, \ldots, m)$ are constructed, using the **edit** procedure.

11. The closed loop high-level system will be computed: CGHI=S_{NR}^{hi}/GHI=**meet**(GHI, S_{NR}^{hi}). Here, S_{NR}^{hi}=**meet**($\tilde{S}_N^{hi}, S_R^{hi}$), and S_R^{hi}=**meet**($\tilde{S}_{R_1}^{hi}, \ldots, \tilde{S}_{R_m}^{hi}$).

3.4 Nonblocking Property

In this section, we address the issue of nonblocking property of the plant under the supervision of the designed controllers.

The system under supervision should satisfy the design specifications in both levels. In addition, we need the plant under supervision to satisfy the nonblocking property as well. In other words, the system should be nonblocking in normal mode and in case of any failure, remain nonblocking during the recovery procedures. By assumption, $\tilde{S}'^{lo}_N$ and $\tilde{S}^{hi}_N$ are considered nonblocking for G^{lo}_N and G^{hi}_N, respectively. It is also assumed that $\tilde{S}'^{lo}_{R_i}$ is nonblocking for G_{NR_i} $(i = 1, \dots, m)$, and $\tilde{S}^{hi}_{R_i}$ is nonblocking for $G^{hi}_{NR_i}$ $(i = 1, \dots, m)$.

In the normal mode at the low-level, the plant is controlled by S'^{lo}_{NR}=**meet**$(\tilde{S}^{lo}_N, S'^{lo}_R)$, where S'^{lo}_R=**meet**$(\tilde{S}'^{lo}_{R_1}, \dots, \tilde{S}'^{lo}_{R_m})$. If $\tilde{S}'^{lo}_{R_i}$ for $(i = 1, \dots, m)$ and $\tilde{S}'^{lo}_N$ are admissible and nonblocking supervisors for G_N and G_{NR_i} and $L_m(\tilde{S}'^{lo}_N/G_N)$ and $L_m(\tilde{S}'^{lo}_{R_i}/G_N)$ are nonconflicting languages, then **meet**$(\tilde{S}'^{lo}_N, S'^{lo}_R)/G_N$ will be nonblocking [23]. This will guarantee that plant under supervision of the low-level supervisor S'^{lo}_{NR} in normal mode (G^{lo}_N) will be nonblocking.

To investigate the nonblocking property in the recovery mode, first we consider the low-level plant and the case of one failure event in the system $(m = 1)$. Let $S'^{lo}_{NR_1}$=**meet**$(\tilde{S}'^{lo}_N, \tilde{S}'^{lo}_{R_1})$. If we consider $S'^{lo}_{NR_1}$ as the supervisor which controls the plant from normal to recovery mode at the low-level in the inner feedback loop, then the nonblocking condition for the recovery mode will be equal to have: $S'^{lo}_{NR_1}/G_{NR_1}$ a nonblocking system. This condition can be formulated as: $\overline{L_m}(S'^{lo}_{NR_1}/G_{NR_1}) = L(S'^{lo}_{NR_1}/G_{NR_1})$.

Proposition 3.4.1 *Assume $\tilde{S}_{R_1}^{\prime lo}$ is a nonblocking supervisor for G_{NR_1}. If the system under supervision of* **meet**$(\tilde{S}_N^{\prime lo}, \tilde{S}_{R_1}^{\prime lo})$ *or $S_{NR_1}^{\prime lo}$ is nonblocking in normal mode, then the system under supervision of $S_{NR_1}^{\prime lo}$ will be nonblocking in recovery mode, i.e., $\overline{L_m}$ $(S_{NR_1}^{\prime lo}/$ $G_{NR_1})= L(S_{NR_1}^{\prime lo}$ $/G_{NR_1})$.*

Proof. The sequences in $L(S_{NR_1}^{\prime lo}/$ $G_{NR_1})$ can be divided into two groups: (i) the sequences that keep the plant in the normal mode, and (ii) the sequences that take the plant to the recovery mode and therefore contain at least one failure event.

(i) Normal mode: By assumption the system under supervision in normal low-level mode is nonblocking, in other words $S_{NR_1}^{\prime lo}/G_N$ is nonblocking. Therefore, from any normal state in $S_{NR_1}^{\prime lo}/G_{NR_1}$, there will be a sequence which could take the plant to some marked state in the normal mode.

(ii) Recovery mode: Note that the following condition holds: $L(S_{NR_1}^{\prime lo}/G_{NR_1}) \subseteq L(\tilde{S}_{R_1}^{\prime lo}/G_{NR_1})$. Now, assume that "$s$" is a string which includes at least one failure event and therefore, it will take the plant to the recovery mode. In other words $s \in L(S_{NR_1}^{\prime lo}/G_{NR_1})$. Since $L(S_{NR_1}^{\prime lo}/G_{NR_1}) \subseteq L(\tilde{S}_{R_1}^{\prime lo}/G_{NR_1})$, we can conclude that $s \in L(\tilde{S}_{R_1}^{\prime lo}/G_{NR_1})$. By assumption, $\tilde{S}_{R_1}^{\prime lo}/G_{NR_1}$ is nonblocking and therefore, there exist s' such that the sequence ss' leads to some marked state. But since in recovery mode R, the supervisory action of $\tilde{S}_{R_1}^{\prime lo}$ and $S_{NR_1}^{\prime lo}$ are the same, s' is also defined in $S_{NR_1}^{\prime lo}/G_{NR_1}$. This means that $S_{NR_1}^{\prime lo}/G_{NR_1}$ must be nonblocking in the recovery mode as well. From (i) and (ii) we can conclude that $S_{NR_1}^{\prime lo}$ is a nonblocking supervisor for G_{NR_1}. In other words, $\overline{L_m}(S_{NR_1}^{\prime lo}/G_{NR_1}) = L(S_{NR_1}^{\prime lo}/G_{NR_1})$. □

For the multiple failure case ($m \geq 1$), consider failure mode k. Consider the combined low-level supervisor S'^{lo}_{NR}=**meet**$(\tilde{S}'^{lo}_{N}, \tilde{S}'^{lo}_{R_1}, \ldots, \tilde{S}'^{lo}_{R_m})$. By this formulation the nonblocking condition for recovery mode R_k at low-level will be: $\overline{L_m}(S'^{lo}_{NR}/G_{NR_k}) = L(S'^{lo}_{NR}/G_{NR_k})$. The following will be an extension of the previous proposition.

Proposition 3.4.2 *Assume $\tilde{S}'^{lo}_{R_k}$ is a nonblocking supervisor for $G^{lo}_{NR_k}$ $(1 \leq k \leq m)$. If the system under supervision of S'^{lo}_{NR}=**meet**$(\tilde{S}'^{lo}_{N}, \tilde{S}'^{lo}_{R_1}, \ldots, \tilde{S}'^{lo}_{R_m})$ is nonblocking in normal mode, then the system under supervision of S'^{lo}_{NR} will be nonblocking in all recovery modes.*

The above proposition guarantees that G^{lo} is nonblocking in both normal and all recovery modes. Note that the important property is the nonblocking property of the plant under supervision of the conjunction of the low-level and high-level supervisors. Nevertheless the nonblocking property of inner loop (i.e., G^{lo}) may be desirable since dealing with blocking in the low-level (subsystem) using the high-level (system-level) controllers may not be desirable.

Our staring point for vocalization and hierarchical design, is the plant under the supervision at low-level (G^{lo}). Up to this point, we have proved that the supervised plant will satisfy the nonblocking property at low-level. Assuming that we design admissible nonblocking supervisors for the high-level plant in the normal and recovery modes, we investigate the nonblocking issue at the high-level in the following.

In case of having one permanent failure f_1 , we need to satisfy the following conditions at the high-level:

Suppose E^{hi}_{N} and $E^{hi}_{R_1}$ are the design specifications of the plant G^{hi} for the normal and recovery modes at high-level and let the required supervisors to enforce this behaviors will be $\tilde{S}^{hi}_N$ and $\tilde{S}^{hi}_{R_1}$ for G^{hi}_N and $G^{hi}_{NR_1}$, respectively. Therefore, if $\tilde{S}^{hi}_N$ and $\tilde{S}^{hi}_{R_1}$ to be designed as nonblocking supervisors for their relevant high-level sub-generators G^{hi}_N and $G^{hi}_{NR_1}$, then by following an approach similar to what we discussed for the low-level plant, they can guarantee our nonblocking requirements at the high-level. Namely, if nonblocking property holds for normal mode, it will be guaranteed for the recovery modes.

Extension for multiple failures is similar to the case of low-level and omitted here for brevity.

Finally the important question is that whether the plant G under the supervision of the conjunction of the low-level supervisor S'^{lo}_{NR} and the implementation of the high-level supervisor S^{hi}_{NR} (through its low-level operator S^{lo}_{NR}) is nonblocking. Let $K^{hi}_{NR} = L(S^{hi}_{NR}/G^{hi})$ denote the high-level closed behavior of the plant under supervision. By assumption, G^{lo} is output-control-consistent and therefore by Theorem 2.5.1, the closed behavior of the plant under supervision of the conjunction of the low-level and high-level supervisor, S, will be:

$$K^{lo} = SupC_{in}(\theta^{-1}(K^{hi})) = L(S/G)$$

Then the nonblocking properties for normal and recovery modes will be:

$$\overline{L_m}(S/G_N) = \overline{K^{lo} \cap L_m(G_N)} = K^{lo} \cap L(G_N) = L(S/G_N)$$
$$\overline{L_m}(S/G_{NR_i}) = \overline{K^{lo} \cap L_m(G_{NR_i})} = K^{lo} \cap L(G_{NR_i}) = L(S/G_{NR_i}) \ (1 \leq i \leq m).$$

In the following section we illustrate our hierarchical approach by applying it to an example.

3.5 Example

To illustrate our proposed method, we provide an example in which, we have a plant with two permanent failures. In the following discussions, first we design the required supervisors for the low-level specifications. After computing the plant under supervision in the low-level and obtaining the high-level versions of the specifications, we use the high-level abstraction of the plant to design the normal and recovery supervisors for the high-level model.

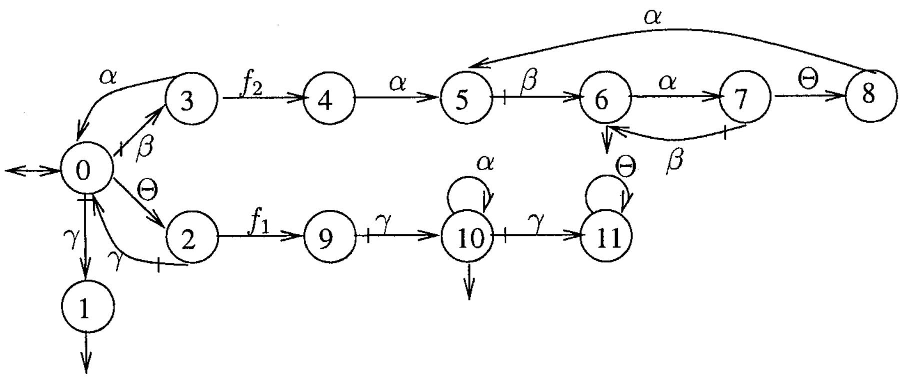
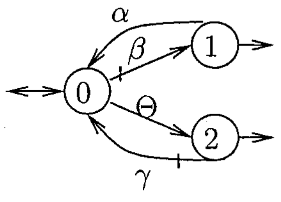


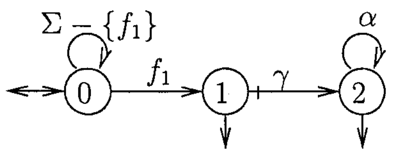
Figure 3.13: $G = G_{NR}$

Consider a plant with two permanent failures f_1 and f_2. Figure 3.13 shows the plant to be controlled $G = G_{NR}$. We assume $\Sigma_G = \Sigma_p \dot{\cup} \Sigma_f$, where $\Sigma_p = \{\alpha, \beta, \gamma, \Theta\}$, $\Sigma_f = \{f_1, f_2\}$, $\Sigma_{G,c} = \{\beta, \gamma\}$, and $\Sigma_{G,uc} = \{\alpha, \Theta, f_1, f_2\}$. The state sets of interest in the plant are: $Q_N^G = Q_N = \{0, 1, 2, 3\}$, $Q_{R_1}^G = Q_{R_1} = \{9, 10, 11\}$, $Q_{R_2}^G = Q_{R_2} = \{4, 5, 6, 7, 8\}$, and $Q_R^G = Q_{R_1}^G \dot{\cup} Q_{R_2}^G$. Finally, the marked states of the plant are: $Q_m = \{0, 1, 7, 10\}$.

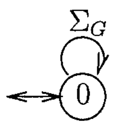
Specifications for the normal mode is given by $E_N = E_N^{\prime lo} \dot{\cup} E_N^{lo}$. For the recovery modes R_1 and R_2, we will have: $E_{R_1} = E_{R_1}^{\prime lo} \dot{\cup} E_{R_1}^{lo}$ and $E_{R_2} = E_{R_2}^{\prime lo} \dot{\cup} E_{R_2}^{lo}$ as the desired behaviors of the plant, respectively. Figure 3.14 shows generators for the design specifications $SP_N^{\prime lo}$, $SP_{R_1}^{\prime lo}$, $SP_{R_2}^{\prime lo}$. The corresponding legal behaviors are $E_N^{\prime lo}$=**meet**$(G_N, SP_N^{\prime lo})$, $E_{R_1}^{\prime lo}$=**meet**$(G_{NR_1}, SP_{R_1}^{\prime lo})$, and $E_{R_2}^{\prime lo}$=**meet**$(G_{NR_2}, SP_{R_2}^{\prime lo})$. We assume that $E_{R_1}^{lo} = L_m(G_N^{lo})$ and $E_{R_1}^{lo} = L_m(G_{NR_1}^{lo})$. Sometimes it is more convenient to express the second group of the specifications directly at high-level using generators defined over the high-level alphabet. In this example, the high-level version of $E_{R_2}^{lo}$, normally, $E_{R_2}^{hi}$ will be provided later in the discussion of high-level supervisors.



(a)$SP_N^{\prime lo}$



(b)$SP_{R_1}^{\prime lo}$



(c)$SP_{R_2}^{\prime lo}$

Figure 3.14: (a) $SP_N^{\prime lo}$, (b) $SP_{R_1}^{\prime lo}$, and (c) $SP_{R_2}^{\prime lo}$.

In order to enforce the design specifications at low-level, $S_N^{\prime lo}$, $S_{R_1}^{\prime lo}$, and $S_{R_2}^{\prime lo}$ are designed for sub-generators of the plant in normal mode (G_N) and in recovery modes $(G_{NR_1}$ and G_{NR_2}), respectively. We use a modular approach to design the supervisors. First, we consider the plant without any failure, and design $S_N^{\prime lo}$. Following that, considering G_{NR_1} and G_{NR_2}, we design the recovery supervisors $S_{R_1}^{\prime lo}$ and $S_{R_2}^{\prime lo}$. **Supcon** procedure of TTCT can be used to compute all three supervisors.

$S_N^{\prime lo}$, $S_{R_1}^{\prime lo}$, and $S_{R_2}^{\prime lo}$ are shown in Figure 3.15. Note that, $S_N^{\prime lo}$, $S_{R_1}^{\prime lo}$ and $S_{R_2}^{\prime lo}$ are all nonblocking supervisors for G_N, G_{NR_1}, and G_{NR_2}. $S_N^{\prime lo}$ disables γ at state 0 of the plant G and prevents it from entering the forbidden state 1 of the plant G. Moreover, the recovery supervisor ($S_{R_1}^{\prime lo}$) disables event γ at state 10, therefore preventing it from entering the deadlock 11. While $S_{R_2}^{\prime lo}$ will only follow the plant's operation in both normal and recovery modes, since the specification $SP_{R_2}^{\prime lo}$ does not put any restriction on plant behavior.

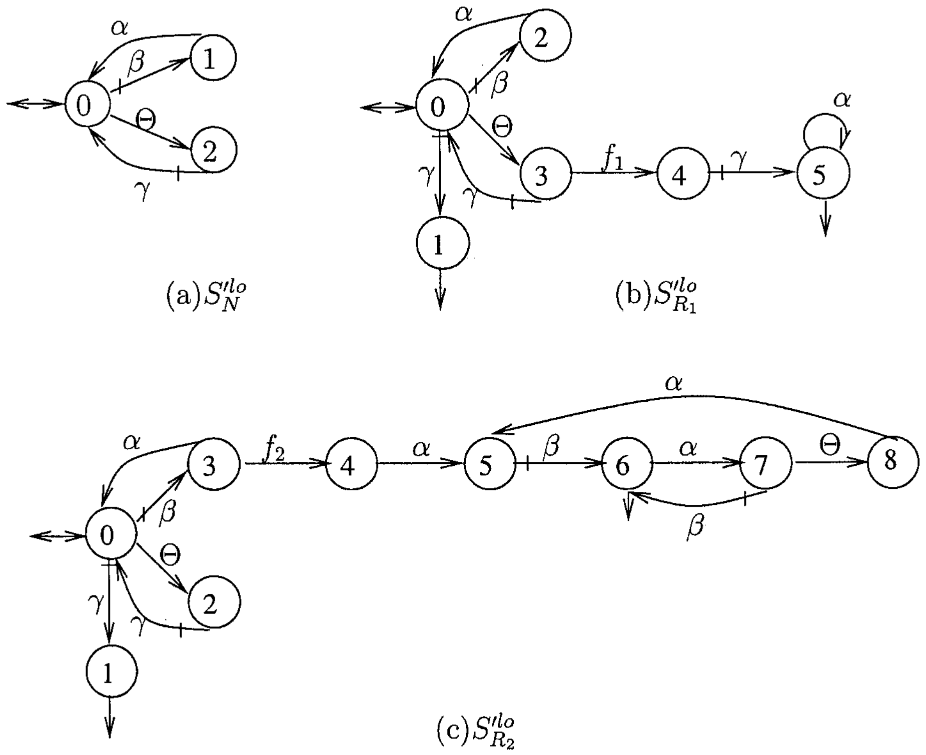


Figure 3.15: (a) $S_N^{\prime lo}$, (b) $S_{R_1}^{\prime lo}$, (c) $S_{R_2}^{\prime lo}$.

As shown in Figure 3.15, it is obvious that $S_N^{\prime lo}$ is identical with $E_N^{\prime lo}$. The plant under supervision of $S_N^{\prime lo}$, $S_{R_1}^{\prime lo}$, and $S_{R_2}^{\prime lo}$ (i.e., $S_{NR}^{\prime lo}$= **meet**($\tilde{S}_N^{\prime lo}, \tilde{S}_{R_1}^{\prime lo}, \tilde{S}_{R_2}^{\prime lo}$)) is denoted by:

G^{lo} ($G^{lo} = S_{NR}^{\prime lo}/G$). G^{lo} is shown in Figure 3.16. It can be seen that the system under supervision ($S_{NR}^{\prime lo}/G$) is nonblocking in normal and recovery modes and supervisors do not attempt to disable any uncontrollable event.

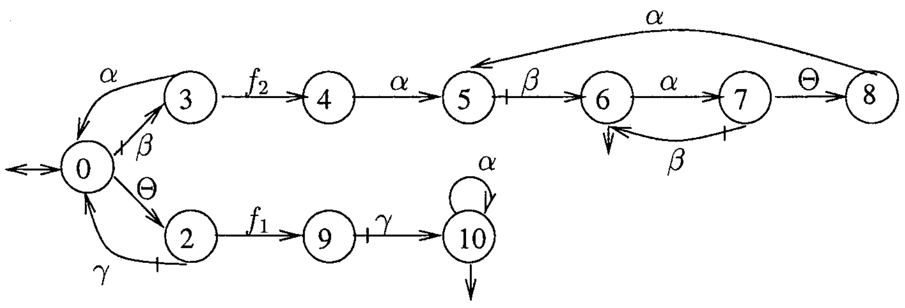


Figure 3.16: System under supervision in the low-level: $G^{lo} = (S_{NR}^{\prime lo}/G)$.

At this point, we are ready to develop our hierarchical controllers to enforce the second group of specifications. The starting point for our hierarchy design will be G^{lo}. In the following we will discuss different procedures used for designing the required supervisors for both normal and recovery modes, step by step.

As first step of the hierarchical design, we define the vocal states of the plant at low-level G^{lo}. This process starts by choosing the "significant" events at the low-level to be signalled to the high-level. To this end, the states that are the destinations of failure events f_1 and f_2 are given outputs τ_1 and τ_2. The rest of the low-level events which need to be signalled to the high-level model will be Θ and β, corresponding to high-level events τ_3 and τ_4. This can be done in TTCT by using **vocalize** procedure. The vocalized plant is shown in Figure 3.17. All silent states are assigned the silent event τ_0.

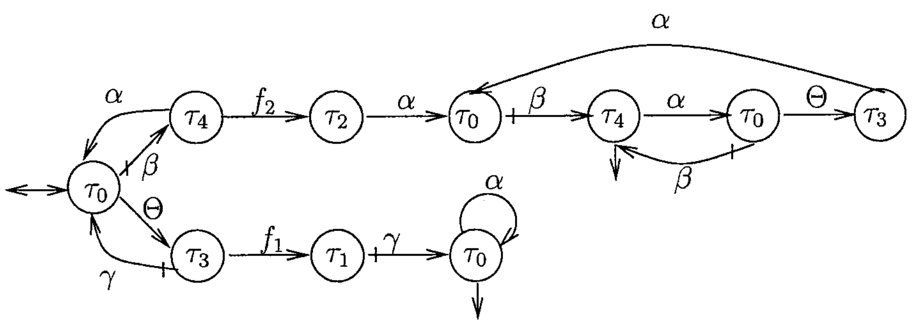


Figure 3.17: Vocalized G^{lo}.

A hierarchically consistent plant can be computed directly by using **hiconsis** procedure of TTCT, if the plant is not already output control consistent. We can also compute an output consistent low-level plant and then by applying **higen** command, an abstraction of the plant can be realized. The reason for applying this procedure is the requirement of having a hierarchically consistent structure for the high-level plant model, as mentioned in the theory of hierarchy control. Therefore, the extra vocal state(s) may need to be created and the causal map (refer to Chapter 2) may need to be adjusted, accordingly. As we discussed earlier, this can be done automatically by TTCT.

By inspection it can be seen that all vocal events in the vocalized plant model (Figure 3.17) are unambiguously controllable or uncontrollable. Therefore, G^{lo} is output-control-consistent, and we can compute the high-level plant by applying **higen** procedure and applying to the output-control-consistent model of G^{lo}. The resulting plant is displayed in Figure 3.18(b). Figure 3.18(a) shows the normal part of model.

The required specifications for both normal and normal-recovery models at high-level are given by their corresponding generators: SP^{hi}_{N}, $SP^{hi}_{R_1}$, and $SP^{hi}_{R_2}$ as shown in

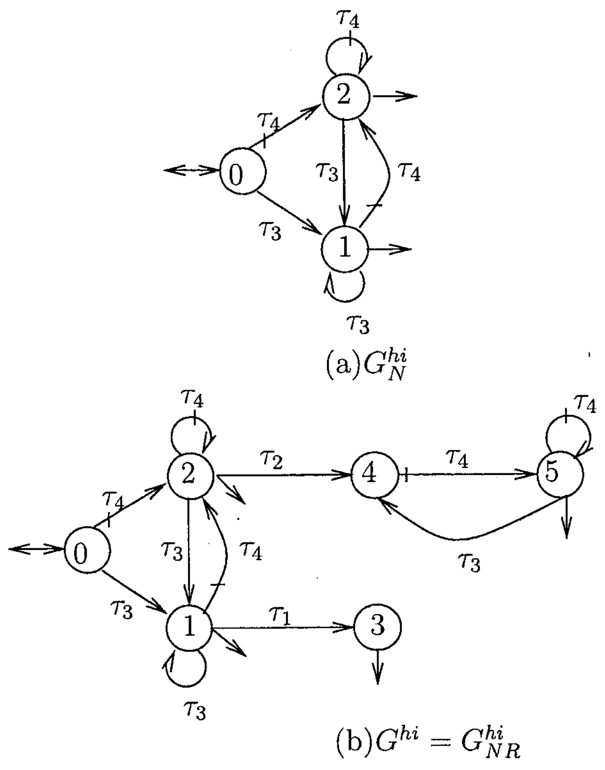


Figure 3.18: High level consistent models G_N^{hi} and $G^{hi} = G_{NR}^{hi}$.

Figure 3.19. The specifications SP_N^{hi} and $SP_{R_1}^{hi}$ (corresponding to legal behaviors E_N^{hi} and $E_{R_1}^{hi}$) do not restrict the behavior of plant G^{hi}. Therefore we do not require to design any supervisor for them.

As we discussed earlier, sometimes it is more convenient if the required specifications for the high-level model directly shown by a high-level generator over the high-level alphabet T. A possible high-level specification for recovery mode R_2 can be $SP_{R_2}^{hi}$ as displayed. $SP_{R_2}^{hi}$ may be interpreted as follows: "if two consecutive event τ_3 occur after having a failure τ_2 (f_2), allow event τ_4 to occur only once". The legal behavior is $E_{R_2}^{hi}$=$\mathbf{meet}(G_{NR_2}, SP_{R_2}^{hi})$.

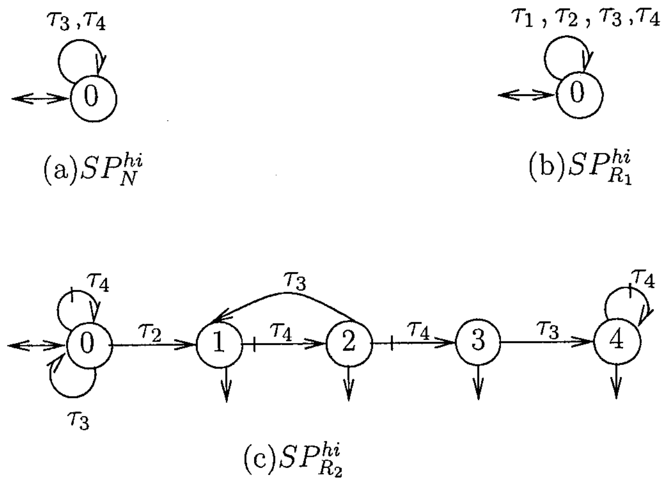


Figure 3.19: Specifications: (a)SP_N^{hi}, (b)$SP_{R_1}^{hi}$, and (c)$SP_{R_2}^{hi}$.

The resulting supervisor for high-level model can be computed using **supcon** procedure. We will have the following:

$$S_{R_2}^{hi}=\textbf{supcon}(\ G_{NR_2}^{hi},\ E_{R_2}^{hi})$$

The high level supervisor is sown in Figure 3.20. Note that, $S_{R_2}^{hi}$ is a nonblocking supervisors for $G_{NR_2}^{hi}$. $S_{R_2}^{hi}$ disables high-level event τ_4 at state 5 in the high-level plant G^{hi}. The system under supervision of $\tilde{S}_{R_2}^{hi}$, $\tilde{S}_{R_2}^{hi}/G^{hi}$, can be computed as $\tilde{S}_{R_2}^{hi}/G^{hi}$=meet$(G^{hi}, S_{NR_2}^{hi})$, and is displayed in Figure 3.21. It can be seen that, the system under supervision at high-level ($\tilde{S}_{R_2}^{hi}/G^{hi}$) is nonblocking in both normal and recovery modes.

Now let us compute the low-level implementation of the entire plant under the supervision of the high and low-level supervisors. The result is shown in Figure 3.22.

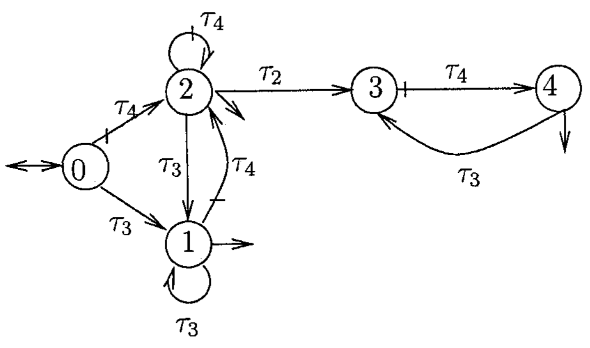


Figure 3.20: high level supervisors: $S^{hi}_{R_2}$.

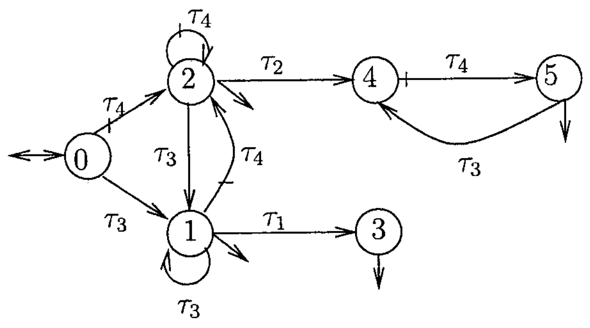


Figure 3.21: High level system under supervision: $(S^{hi}_{NR_2}/G^{hi}_{NR})$.

Comparing the plant under supervision with the original plant G, we can see that forbidden transitions $0 \xrightarrow{\gamma} 1$, $10 \xrightarrow{\gamma} 11$, and $7 \xrightarrow{\beta} 6$ are now disabled, resulting in the satisfaction of all design specifications.

For the purpose of comparison, we also compute the low-level equivalent of the designed high-level controller $S^{hi}_{R_2}$. Figure 3.23 shows the low-level equivalent of $S^{hi}_{R_2}$, if we had considered only the low-level plant G^{lo} in designing the supervisor. By comparing $S^{hi}_{R_2}$ and $S^{lo}_{R_2}$, we notice that number of states of $S^{hi}_{R_2}$ is 5 less than its low-level implementation which has 8 states. It is not difficult to verify that in G (Figure 3.13) we had

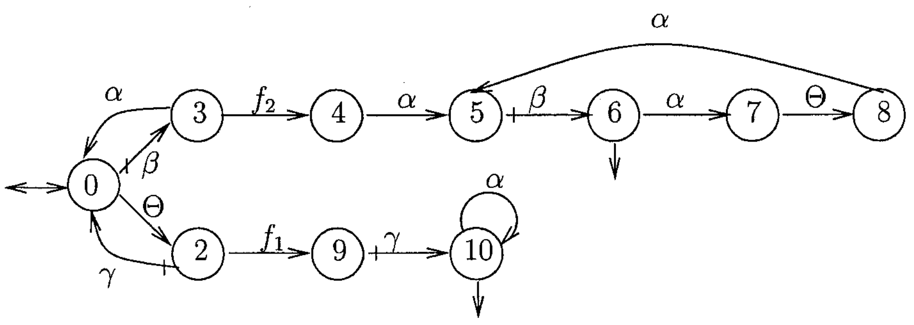


Figure 3.22: The entire plant under the supervision of the low-level and high-level supervisors.

"n" consecutive transitions from state 6 to 7, then the high-level supervisor $S^{hi}_{R_2}$ would still have 5 states, whereas the equivalent low-level $S^{lo}_{R_2}$ (Figure 3.23) would have 8+n states. This difference in the number of states is the result of the abstraction that takes place in constructing high-level model G^{hi} resulting in reduction in the complexity of the controller design procedure which could be important in large-scale systems. This was the main motivation of this work and the proposed solution.

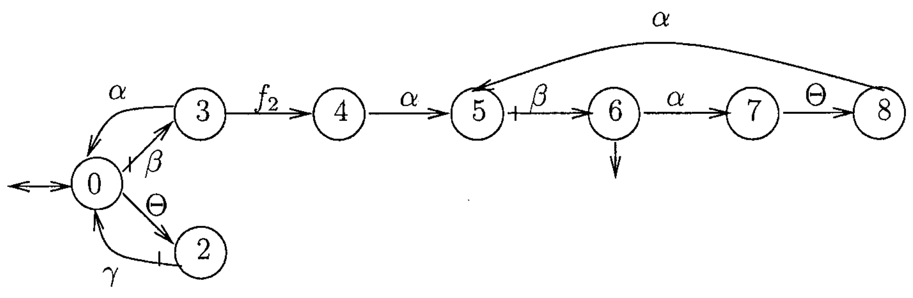


Figure 3.23: $S^{lo}_{R_2}$.

In this example, for a plant with two permanent failures, we designed low and high-level supervisors to enforce the required specifications. In addition, we showed that by choosing a hierarchical approach, we can have a much simpler plant model to control, and therefore we have less complexity for design procedures. As this example shows, the state set in high-level model may be considerably smaller than its low-level counterpart.

3.6 Conclusions

In this chapter, we developed our switching modular hierarchical approach of the fault recovery problem in DESs. We started with an introduction to the problem. Then, we formulated the problem by categorizing the specifications of the system into two main groups for each of the normal and recovery modes. After designing the required supervisors for the first category of the specifications, we computed an abstraction of the plant under supervision at the low-level. For our high-level models, a *modular hierarchical solution* for designing the supervisors for the second class of the specifications have been proposed. For each level, we considered normal and recovery modes. A *modular switching supervisory control* scheme was proposed for designing the required supervisors at each level to control the plant in fault recovery problems.

We developed our method for the case in which we had only one failure mode in the system. We proposed our solution for the general case by considering multiple failure events in the plant. A computational procedure for the general case of the fault recovery problem in DESs using modular hierarchical method was introduced, and we discussed the nonblocking issue.

We provided a simple example with two permanent failures to show the different procedures of the proposed solution. In the following chapter, we will investigate the effectiveness of our proposed solution for a failure recovery problem in electrical power networks.

Chapter 4

Fault Recovery in Electrical Power Networks

In this chapter, we apply our proposed approach for failure recovery problem using hierarchical DESs to the automatic control of a power generation and distribution network. We design local and global controllers to respond to electrical failures which could occur in the system. The controllers (supervisors) may redirect a supply source, in case of loss of supply from a generating unit, or simply to isolate a faulty part of the network from the rest, so that the network will be able to operate in a safe condition.

After a brief description of power generation and distribution networks, we model a typical power plant and its different components using DES models. We compute the automata representing the individual models and also the entire model of the plant in normal and faulty conditions. We also design the required supervisors to satisfy the specifications in both normal and recovery modes. To illustrate our modular hierarchical solution to the fault recovery problem, we consider a simpler model of the network just to simplify our computations and illustrations. Finally, we design local and global controllers to respond to the failures in the network. We start by giving an introduction to these systems.

4.1 Introduction

In order to avoid service interruptions resulting from failures in any power generation and distribution network, we need to separate and isolate the faulty parts and components as soon as possible. For this purpose, a number of protection components are distributed over the network. These protection systems are in charge of disconnecting a component such as a line, a transformer, or a generation unit from other parts of the network in case of having them in a dangerous condition (short circuit). They also have to keep other non-faulty components in operation, in order to avoid a black-out for the whole system. This can be achieved by controlling different circuit-breakers, each associated with one of the protection systems [12].

Currently protective relays shut down the power lines or generating groups by triggering a related circuit-breaker if a fault, say a short circuit, occurs and possibly causes overheat. This relays rely only on local information and may disconnect a line or component by mistake. For example, a local sensor failure may result in disconnection of a line unnecessarily. Therefore, a comprehensive control system which is capable of detecting sensor failures can significantly increase the reliability of the power systems. A better communication between different parts of the whole network and specifically among protection systems also needed to have a more reliable and stable power distribution network. Ideally, a master controller capable of detecting important failures of the system could also serve as a global supervisor (controller) for the failure recovery purposes. In fact, this requires an improved control method especially designed to automatically isolate the faulty spots from rest of the network, and also to make a balance between the amount of electricity generation and consumption. Many attempts have been made in the area of developing automated support tools for electrical power networks [8], [18], [15], [1], and [16].

The main objectives for having such an automatic control method for the power networks are very simple: best uninterrupted power supply service and safety of the equipment. Safety of the equipment and material can be achieved by triggering different circuit-breakers and isolating the troubled parts of the network, whereas the best uninterrupted and a more reliable power supply service can be achieved by minimizing the number of consumers affected by a power outage and also by reestablishing the power as soon as possible in case of a failure. The functionality of the proposed control method is to handle the task of automatic control of the plant in normal mode, and also to be able to perform appropriate actions so that the system can recover from failures. For example, it has to be able to redirect the supply sources in plant's normal operation or in case of failure in the generation groups. It also needs to be able to reestablish the power supply following an interruption in the system, or to isolate the faulty parts in order to avoid a black-out in the entire network [16].

To this end, the requirement for having an automated control tool for normal operation and also for the failure recovery of electrical power networks, is considered as an important need for these systems. We start our discussions by giving a brief description of power distribution networks in the following section.

4.2 Overview of Electrical Power Systems

In this section, we give a general overview of power generation and distribution networks as well as different components of these systems.

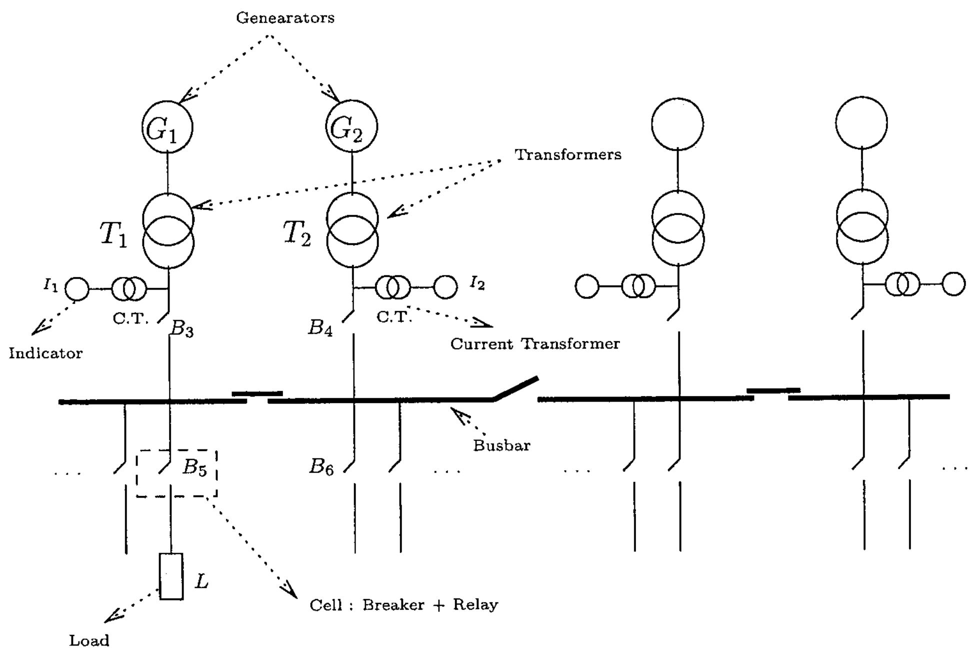


Figure 4.1: A typical model of an electrical power network.

Figure 4.1 shows a simplified diagram of a typical power generation and distribution network. Any electrical power network consists of several high voltage stations. Generation groups of these stations are linked to medium voltage parts of the network via transmission lines. In order to protect the equipment and also for maintenance purposes circuit-breakers are used. Plants linked to each other through a central busbar and together they form the whole network. In the following, we will introduce some important components of electrical power systems.

Electrical power systems are composed of three main components: the generation units, the transmission lines, and the distribution system. Generating stations are usually hydro stations, steam or nuclear powered stations. Transmission lines connect power generation units and distribution system. They also could make the connections to other power systems through interconnections to form larger networks. Distribution system can deliver the power from transmission lines to the customers. Transformers are used in different parts of the network to provide the desired levels of the voltage.

Generating stations are considered as the main sources of the power supply in electrical systems. Usually after power is generated, the level of the generated voltage is increased by using a step-up transformer. In order to reduce the loss of electrical energy in long transmission lines, these step-up transformers are being placed in generating units immediately after the generators.

Distribution systems are usually divided into two main parts: primary and secondary. The primary part includes circuit and devices which carry a voltage higher than average voltage of the system, whereas the secondary distribution system includes the range of voltages at which the ultimate customers are able to use the electrical energy. There are also distribution transformers between these two parts of the network. For simplicity, we assume a single distribution systems in our plant model. Distribution systems contain busbars, transformers, circuit-breakers, and different types of relays. In our work, we study a simplified system and therefore, only discus those components which are included in our model for the failure recovery problem.

4.2.1 Power Network

We call a group of generating units together with their transmission lines and distribution systems, an **electrical power network**. The local failures are considered as any type of fault which can occur inside one of the individual plants of a network and they need to be handled by using a local controller. For example, if a load becomes faulty it needs to be isolated by its related breaker. The following components can be considered for a typical power network.

Generator

Generators are considered as the main sources of electrical energy in power networks. Different types of generators can be found in electrical systems: steam turbine generators, gas turbine generators, nuclear powered generators, and etc.

Transformer

Transformers are used in many parts of the electrical systems to generate different voltage levels at different locations depending on the requirements of the network. Three main groups of power transformers are: step-up, step-down, and isolating transformers.

Cell

In Figure 4.1, the combination of a circuit-breakers and a relay is called a cell. Circuit breakers are considered as protection devices in power distribution systems. As we mentioned earlier, in case of failure, they must separate the faulty parts of the network from the rest of the system. They are usually controlled with relays (specifically, over-current relays). In our problem, we consider the combination of a relay, and a circuit-breaker as a single device of the system.

Load

In electrical power systems load is considered as the customer. The ultimate goal of any power system is to carry the electrical energy to the customers. Each load is protected by combination of a breaker and an over current-relay (cell).

4.3 Modelling and Control of Electrical Power Networks Using DES Models: Local (Subsystem) Supervision

Figure 4.2 shows the schematic diagram of a plant which we want to model using DES models.

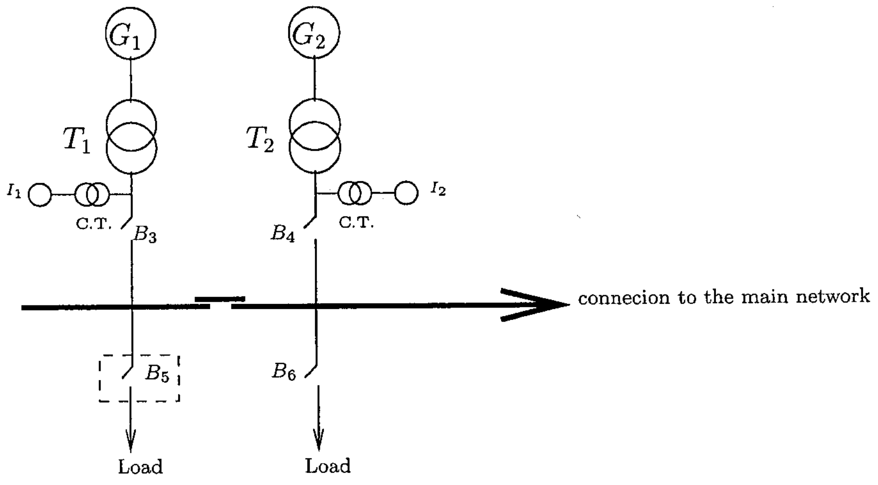


Figure 4.2: A simple electrical power network.

In the following section, we discuss the modelling and control of a simple power network consisting of two generators. The modular switching supervisor designed will address local (subsystem) specifications of the system. In Section 4.4. we consider a power system consisting of two power networks (subsystems). We will apply the modular switching hierarchical method proposed in this thesis to design supervisor to address both subsystem-level and system-level design specifications.

In normal operation of the plant we consider only one of the generating groups of the plant is in service (T_1 and G_1). In this mode, the controller's task is only to control the start-up and shut-down sequences which can be triggered by operator's commands. In the recovery mode, we assume that, the current flowing in power line containing T_1 and B_3 can reach to a value higher than its nominal value, due to the failure in transformer (T_1). Therefore, for the recovery procedures, we need to isolate this section of the network by opening B_3. For maintaining the functionality of the network, the recovery supervisor also needs to close B_4, which initially is assumed to be in the open position.

We start by creating models for the individual components of the plant and their interactions. After obtaining individual models, the resulting DES models are presented for both normal and normal-recovery modes. Generators for the design specifications of operation of the plant in normal and recovery modes will be discussed, and finally we will design the required supervisors to control the plant using our modular approach for the failure recovery problem in DESs.

4.3.1 Individual Models of the Plant in Normal Mode

Transformers

Each transformer of the plant shown in Figure 4.2 can be modelled in its normal mode using a two state automaton (Figure 4.3(a)). The state of each transformer is related to its current. Events $i0$, $i2$ $(i = 1, 2)$ represent current changes and are uncontrollable.

Circuit-Breakers

Breakers are considered as protection devices and they have to separate their lines in case of failures in them. They can be modelled by their status (open or close). We use two controllable events for each breaker to show their open and close commands. In Figure 4.3(b) DES model of breakers B_j $(j = 3, 4, 5, 6)$ are shown in their normal mode.

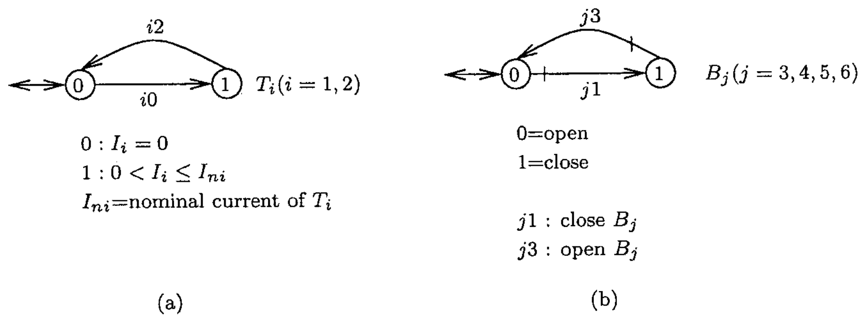


Figure 4.3: DES models of the transformers and breakers in normal mode.

Operator

the operator can issue start-up or shut-down commands for the entire system at any time. Operator's DES model (OP) is shown in Figure 4.4.

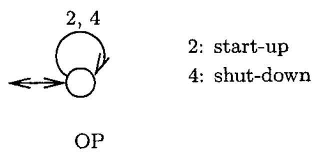
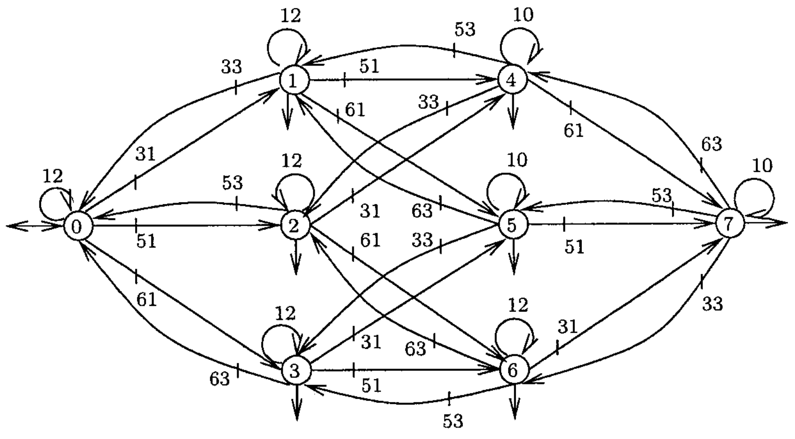


Figure 4.4: DES model of the operator.

Interaction

In normal mode, interaction between different components of the plant will be modelled by using (INTERN) automaton and which is shown in Figure 4.5. INTERN basically describes how changes in the state of circuit breakers affect the current. For instance, starting from initial state 0, upon the execution of the sequences 31 and 51 (breakers B_3 and B_5 open), the event 10 (current flow through T_1 can occur). Note that, in normal mode only T_1, B_3, B_5, and B_6 will be in operation mode.



INTERN

Figure 4.5: Interaction in normal operation of the plant.

Following the convention of TTCT, all odd events are controllable and all even events are uncontrollable. The alphabet set Σ={2,4,10,12,14,31,33,41,43,51,53,61,63}, is considered for the plant in normal and normal-recovery modes. Event {14} is considered as the failure event of the system.

4.3.2 Component Models in Failure Mode

We consider the failure scenario in which the value of the current flowing through the line containing T_1 goes higher than the nominal value (I_{n1}), due to an internal problem of the transformer (T_1). For example, this could be the result of an internal short-circuit happened inside the transformer's coils. The models of some of the plant components will be affected from the failure. In our problem, these components are T_1 and B_3. Figure 4.6 shows the complete T_1 and B_3 describing both normal and faulty operations. In this figure, event 14 is compound event representing short-circuit followed by breaker B_3 tripping.

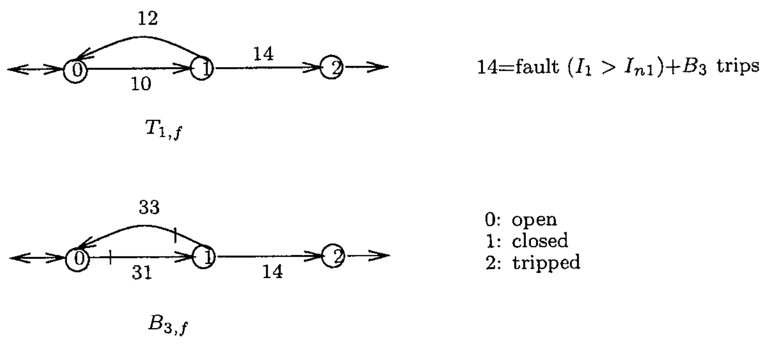


Figure 4.6: $T_{1,f}$ and $B_{3,f}$.

We also need to model the new interaction between the components of the plant to account for both normal and faulty modes. Therefore, a new interaction model is constructed. Figure 4.7 shows the complete model of interaction between the components of the plant (INTERF). Note that, we consider I_1 is the only current which can increase in the network due to the failure in T_1; in other words failure in T_2 are not considered.

In the following, we will define the design specifications for the normal and recovery

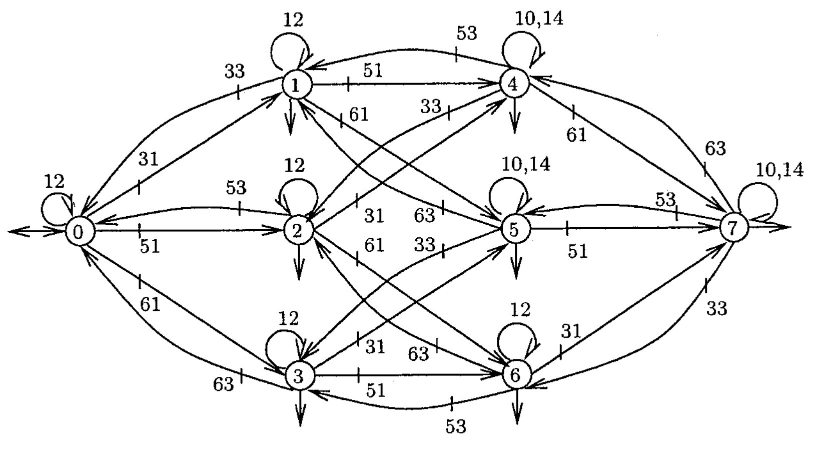


Figure 4.7: Interaction in the failure mode.

operations of the plant.

4.3.3 Specifications for the Normal Operation

Assume that the plant is in normal mode. In this case the normal operation of the plant are as follows:

(i) If the operator issues a start-up command (2), B_3 must be closed followed by B_5 and B_6; The operator's shut-down commands before completing start-up cycle (EN_1) are ignored.

(ii) If the operator issues a shut-down command (4), B_5 and B_6 have to be opened before B_3 is opened; The operator's start-up commands before completing shut-down cycle (EN_2) are ignored.

(iii) The system under supervision should be nonblocking.

The above-mentioned requirements are captured by EN_1 and EN_2 shown in Figure 4.8.

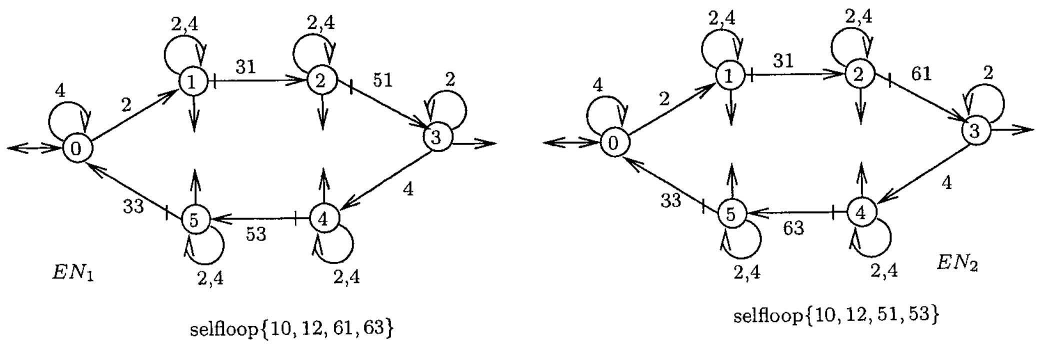


Figure 4.8: Specifications for normal mode.

4.3.4 Specifications for the Recovery Mode

For the recovery mode (following the occurrence of failure for T_1 and B_3 entering trip state), we need to close B_4 to maintain the network in operation. In this mode, we also must satisfy the requirements for the safe operation of the plant in case of a shut-down command. Therefore, we have the following design specifications for this mode:

(i) If B_3 goes to the tripped state, B_4 needs to be closed by controller; B_5 also needs to be opened before B_4, upon receiving a shut-down command (ER_1).

(ii) If B_3 goes to the tripped state, B_4 needs to be closed by controller; B_6 has to be opened before B_4, upon receiving a shut-down command (ER_2).

(iii) The system under supervision must be nonblocking.

Figure 4.9 shows the generators of the required specifications in the recovery mode.

4.3.5 Supervisor Design

Finally we design the required supervisors for controlling the plant in normal and recovery modes to enforce all specifications, while satisfying the nonblocking property of the plant under supervision.

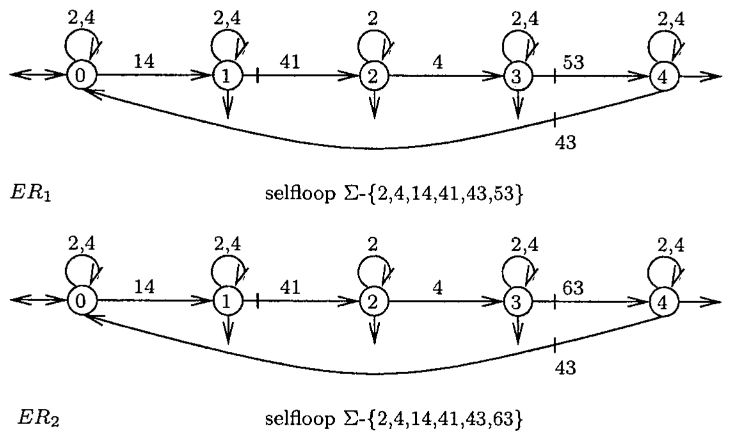


Figure 4.9: Specifications for the recovery mode.

Normal Supervisor

The following procedures of TTCT will be used to compute the supervisor in normal mode and also to verify if it is a proper supervisor.

PLANTN=**sync**($T_1, B_3, B_5, B_6, OP, INTERN$); (16,88); Blocked events = None.

EN=**meet**(EN_1, EN_2); (18,90)

EN = CONDAT(PLANTN, EN) ; CONTROLLABLE.

true = Nonconflict(PLANTN, EN)

The results of **condat** and **nonconflict** procedures for EN show that EN is a proper supervisor for the plant in normal mode. Therefore, we can consider the normal supervisor as following:

SUPERN=EN=**meet**(EN_1, EN_2); (18,90)

The resulting supervisor has 18 states and 90 transitions.

Recovery Supervisor

Similar to the normal case we can compute the recovery supervisor as described below:

PLANTR=**sync**($T_{1,f}, B_{3,f}, B_4, B_5, B_6, OP, INTERF$); (40,254); Blocked events = None .

ER=**meet**(ER_1, ER_2); (7,63)

SUPERR=**supcon**(PLANTR, ER); (29,111); Blocked events = None.

SUPERR = Condat(PLANTR, SUPERR) ; Controllable.

true = Nonconflict(PLANTR, SUPERR)

As we can see, the recovery supervisor is a proper supervisors with 29 states and 111 transitions. Due to the large size of the computed supervisors and the plants under supervision, we have not showed the diagrams of the supervisors and their related plants.

In the following section, we apply the switching modular hierarchical method proposed in this thesis to a power network. The problem has been kept simple enough to allow to display and describe the operation of the supervisor.

4.4 Fault Recovery in Electrical Systems Using Hierarchical Design: Local and Global Supervision

Figure 4.10 shows a simple power system consisting of two different plants linked together using an interconnection (B_3) and forming a network. Each plant has its own generation group, transmission line, and distribution system. In normal operation of the plant, we consider that only one generating unit of the plant1 (P_1) is in service, which supplies the network via (B_1). Also in this mode we assume that two separate power plants are connected by the breaker B_3. If as a result of a failure in plant1 (P_1) (for example the temperature in transformer T_1 becomes higher than normal), and B_1 goes to a trip mode, in order to isolate T_1 from rest of the network, then the local controller should close B_2, so that we could have one of the generating stations (T_2) still in service to supply the network.

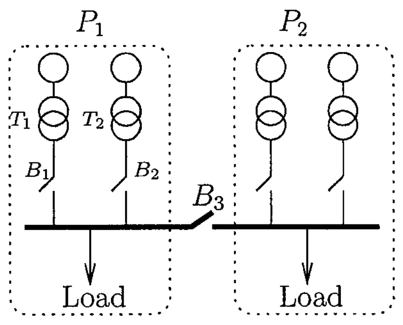


Figure 4.10: Simplified model.

If the sub-system (P_2) becomes faulty, to prevent a total black-out of the whole network we need to separate P_2 from rest of the network. This needs to be done by designing a global (system-level) controller, which will not interfere with the functionality of the local controllers of both plants P_1 and P_2. In the following, we model the components of plants using DES models. Generators for required specifications for the operation of the plant in both normal and recovery modes will be presented, and finally we will design the supervisors for both low-level and high-level models. As in the previous section, controllable (resp. uncontrollable) events are designated by odd (resp. even) numbers.

4.4.1 Component Models

We create models for individual components of the system, their interactions (if applicable), and the resulting DES model.

Breakers

As we discussed, circuit-breakers are used as protection devices in electrical systems. We model them by using DES having two controllable events for opening and closing. In case of failure (short circuit) in their related lines, they go to a tripped state. Here for simplicity, we only consider faulty mode for the line controlled by B_1. Figure 4.11 shows the DES model of the breakers of system in normal mode. Note that, B_1 could enter the

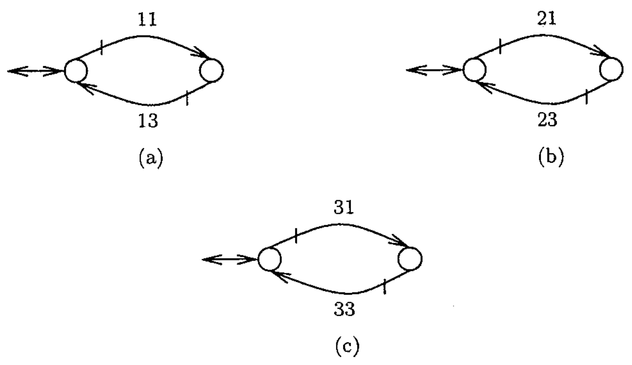


Figure 4.11: Breakers: (a) B_1, (b) B_2, and (c) B_3 shown in normal mode.

faulty mode only when it is closed. This is shown in Figure 4.12 using uncontrollable event 10.

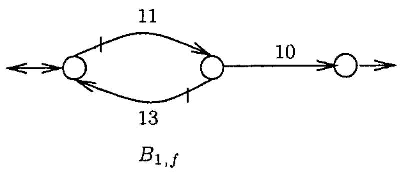


Figure 4.12: Complete model of B_1.

Plant P_2

We consider an abstract model for the second plant P_2. We assume that P_2 can enter a failure mode f_2, if an uncontrollable event (failure event 20) occurs in the system. Failure f_2 (20) is by assumption a major failure which would require the isolation of P_2 from the rest of network to prevent black-out in the entire network. Figure 4.13 shows the model of P_2.

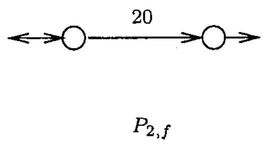


Figure 4.13: Plant P_2.

At this stage, we will have the following sets of alphabet for the low level plant: $\Sigma_p = \{11, 13, 21, 23, 31, 33\}$, $\Sigma_G = \{10, 11, 13, 20, 21, 23, 31, 33\}$, with: $\Sigma_{G,c} = \{11, 13, 21, 23, 31, 33\}$ and $\Sigma_f = \Sigma_{G,uc} = \{10, 20\}$.

4.4.2 Local (Subsystem-Level) Supervisor

First we present the model of the system at the low-level and after discussing the specifications, we design the required subsystem-level (local) supervisors to enforce them.

System to be Controlled

All of the individual models of the system can be generated in TTCT software. The complete model of the plant can be obtained using the **sync** procedure. The system which needs to be controlled has a normal and two recovery modes. The required models G^{lo}_N, $G^{lo}_{NR_1}$, $G^{lo}_{NR_2}$, and G_{NR} are constructed as follows:

$$\begin{gathered} G_N{=}\mathbf{sync}(B_1, B_2, B_3) \\ G_{NR_1}{=}\mathbf{sync}(B_{1,f}, B_2, B_3) \\ G_{NR_2}{=}\mathbf{sync}(B_1, B_2, B_3, P_{2,f}) \\ G = G_{NR}{=}\mathbf{sync}(B_{1,f}, B_2, B_3, P_{2,f}, SFS). \end{gathered}$$

For brevity, we will only show the diagram of the plant under supervision after the design is complete. SFS is a generator to enforce the *single failure scenario* assumption (Figure 3.12, Section 3.3.3).

Specifications

For the entire plant, we have normal and recovery specifications. Specifications for the normal mode are given by $E_N = E^{\prime lo}_N \dot\cup E^{lo}_N$. For the recovery modes R_1 and R_2, we will have: $E_{R_1} = E^{\prime lo}_{R_1} \dot\cup E^{lo}_{R_1}$ and $E_{R_2} = E^{\prime lo}_{R_2} \dot\cup E^{lo}_{R_2}$ as the desired behaviors of the plant, respectively. The required specifications for operation of the plant in normal and recovery modes in addition to nonblocking property are defined in the following:

(1) First group (subsystem-level) specifications:

(i) In normal operation of the plant we need B_3 to be closed after B_1 is closed. It is also required that B_3 to be opened before B_1 is opened ($E^{\prime lo}_N$). The corresponding generator $SP^{\prime lo}_N$ is shown in Figure 4.14(a). Thus $E^{\prime lo}_N{=}\mathbf{meet}(G_N, SP^{\prime lo}_N)$.

(ii) For recovery mode (R_2), in case of having a failure situation for B_1, the desired action is to close B_2. The generator $SP'^{lo}_{R_2}$ (Figure 4.14(a)) represents this specifications: $E'^{lo}_{R_2}$=**meet**$(G_N, SP'^{lo}_{R_2})$.

(iii) In case of a failure in P_2, there is no subsystem-level specification. i.e., $E'^{lo}_{R_2} = L_m(G_{NR_2})$.

(2) Second group (system-level) specifications:

(i) Normal operation: no restrictions.

(ii) Recovery mode (R_1): no restrictions.

(iii) Recovery mode (R_2): In case of a total failure in second plant P_2, this part of the network is required to be separated from the rest of the system by opening B_3.

Sometimes it will be more convenient if the generators representing the second group of the specifications directly computed at high-level using the high-level alphabet. In this problem, the required generator for capturing $E^{lo}_{R_2}$ will be directly computed at high-level.

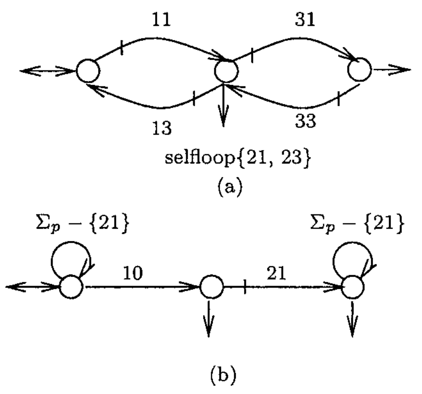


Figure 4.14: Specifications for normal and recovery modes: (a) SP'^{lo}_{N}, (b) $SP'^{lo}_{R_2}$.

Low-Level (Subsystem-Level) Supervisors

We design the required supervisors using the RW framework, to enforce the design specifications for normal and recovery modes. Since in this thesis we assume all events to be observable events, using the **supcon** procedure, we can compute the relevant supervisors. Therefore we will have the following:

$$S_N^{\prime lo} = \mathbf{supcon}(G_N^{lo},\ E_N^{\prime lo})$$

$$S_{R_1}^{\prime lo} = \mathbf{supcon}(G_{NR_1}^{lo},\ E_{R_1}^{\prime lo})$$

Figure 4.15 shows the resulting supervisors $S_N^{\prime lo}$, and $S_{R_1}^{\prime lo}$ for normal and recovery modes. Note that, no low-level supervisor for recovery mode R_2 is designed since G_{NR_2} is nonblocking and no low-level design specification has been stated in problem formulation.

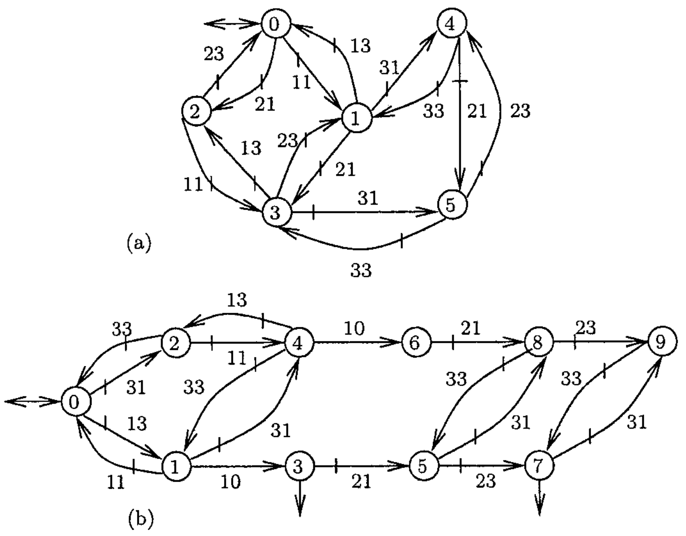


Figure 4.15: Supervisors for normal and recovery modes: (a) $S_N^{\prime lo}$, and (b) $S_{R_1}^{\prime lo}$.

The result of **condat** procedure for $S_N^{\prime lo}$ and $S_{R_1}^{\prime lo}$, shows that all supervisors are controllable and therefore, they are admissible supervisors for plant in recovery and normal modes. Also the result of **nonconflict** procedure is *true* and this shows that they are nonblocking supervisors for their relevant sub-generators.

Supervised Plant

In normal mode plant will be controlled by: S'^{lo}_{NR}=**meet**$(\tilde{S}'^{lo}_{N}, \tilde{S}'^{lo}_{R_1})$. In recovery mode R_1, $\tilde{S}'^{lo}_{N}$ will be disabled and $\tilde{S}'^{lo}_{R_1}$ will remain in the feedback loop. The system under supervision in both modes ($G^{lo} = S'^{lo}_{NR}/G$) is shown in Figure 4.16. Note G^{lo} is non-blocking in N, R_1, and R_2.

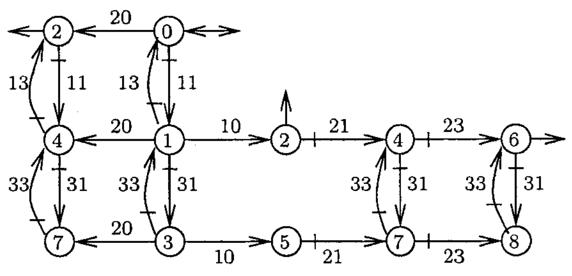


Figure 4.16: System under supervision at low-level: G^{lo}.

4.4.3 Global (System-Level) Supervisor

At this point, we need to compute the model of the system at high-level in order to apply our hierarchical design procedures for the fault recovery problem. Accordingly, the required controllers will be designed to enforce the design specifications. First, we explain how to obtain the high-level plant model.

System to be Controlled at High-Level

The starting point of the hierarchical control design is G^{lo} (Figure 4.16). Figure 4.17 shows the vocalized plant. Note that, the states of the plant which need to be vocalized are shown in this figure in terms of the high-level alphabet $T = \{\tau_1, \tau_2, \tau_3\}$. The silent states are shown by silent event τ_0. τ_1 and τ_2 signify failure events f_1 and f_2, and τ_3 signify opening of B_3 (in R_2 mode). The triples $[* \xrightarrow{10} \tau_1]$, $[* \xrightarrow{20} \tau_2]$, and $[* \xrightarrow{33} \tau_3]$ have been considered as the triples which need to be vocalized.

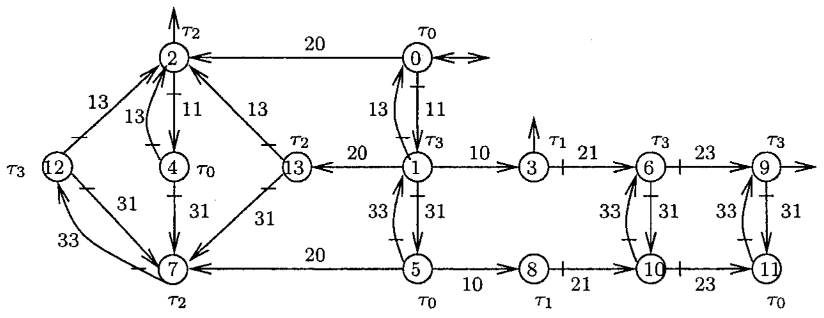


Figure 4.17: Vocalized G^{lo}.

The foregoing adjustments can be made using **edit** and **vocalize** procedures of TTCT program. Note that, state 4 has been split to new states 12 and 13. By applying **hiconsis** and **higen** procedures to the vocalized model of G^{lo}, as explained in the proposed design algorithm, we can compute the high-level plant model. The result is shown in Figure 4.18. The new high-level events (τ_{2c}, τ_{2u}, τ_4, and τ_5), have been added to the high-level alphabet T by **hiconsis** command in order to drive a hierarchically consistent model.

Note that Figure 4.18 is displaying both normal and normal-recovery models of the plant at high-level. Now that we have obtained the high-level models of the plant, we are ready to define the required behaviors of the plant using our high-level specifications. In the following after defining the specifications, we will design the relevant supervisors of the plant to enforce the required behaviors.

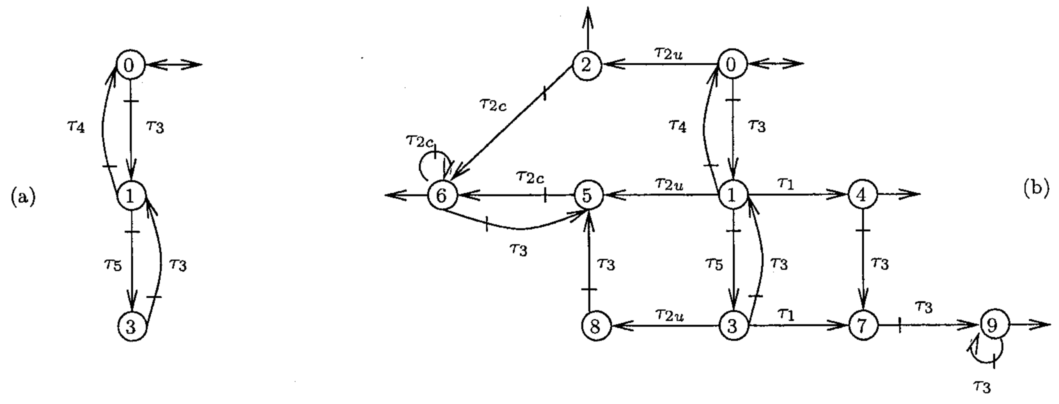


Figure 4.18: (a) G_N^{hi}, and (b) G^{hi}.

High-Level Specifications and Supervisors

At this point we calculate the high-level versions of the second group of the specifications for the hierarchical solution. E_{R2}^{hi} is the high-level version of E_{R2}^{lo}. As we already mentioned E_{R2}^{hi} will be directly captured by a generator which uses only the high-level alphabet set T. The required specifications for recovery models at high-level, is given by the generators E_{R2}^{hi} shown in Figure 4.19.

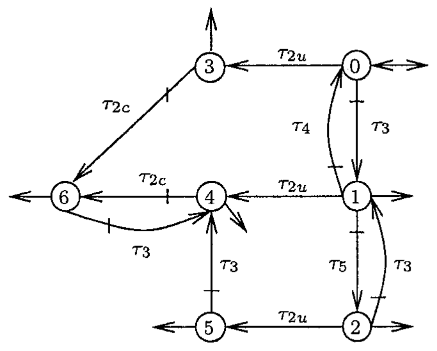


Figure 4.19: High-level specification $E_{R_2}^{hi}$.

The resulting supervisors for high-level plant can be computed by using **supcon** command of TTCT and is shown in Figure 4.20.

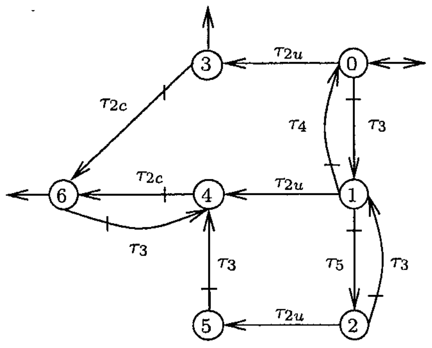


Figure 4.20: High-level supervisor: $S^{hi}_{R_2}$.

$$S^{hi}_{R_2}=\textbf{supcon}(\ G^{hi}_{NR_2},\ E^{hi}_{R_2})$$

Note that, $S^{hi}_{R_2}$ is a nonblocking supervisor for $G^{hi}_{NR_2}$.

System Under Supervision at High-Level

The high-level plant will be controlled by: $S^{hi}_{NR}=\tilde{S}^{hi}_{R_2}$. The system under supervision at the high-level S^{hi}_{NR}/G_{NR} is shown in Figure 4.21.

It can be seen that, the system under supervision at high-level S^{hi}_{NR}/G^{hi} is nonblocking in both normal and normal-recovery modes. Also the supervisor does not attempt to disable any uncontrollable event and therefore, $S^{hi}_{R_2}$ is an admissible supervisor. This results can also be verified using **condat** and **nonconflict** procedures of TTCT program.

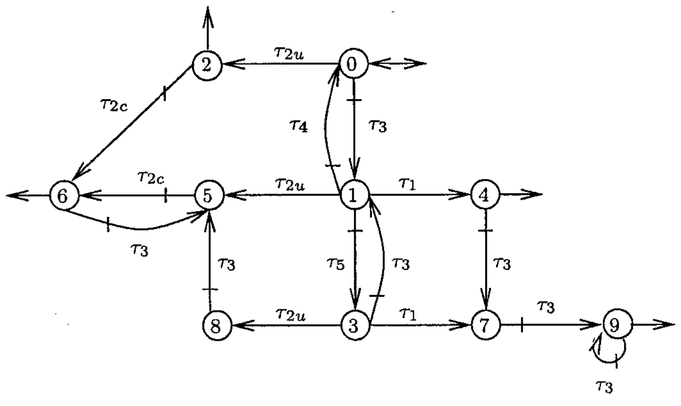


Figure 4.21: S^{hi}_{NR}/G^{hi}.

4.5 Conclusions

In this chapter, we considered the modelling of electrical power systems using DES models for the failure recovery purposes. We started by giving an overview of these systems and their main components. We showed that DES models can effectively be used as a modelling device for analysis of these networks by designing controllers for normal and recovery operations of the plant. Although our answer is not considered as a unique solution for design procedures, but the main objective was to show that DESs can be considered as an important modelling tool for supervisor design and synthesis for this kind of industrial systems.

Following that, we illustrated our switching modular hierarchical approach to the problem of fault recovery by applying this method to the automatic control of a simple power plant. We designed the controllers for both low and high-level models to enforce subsystem-level (local) and system-level (global) specifications.

Chapter 5

Conclusions and Future Research

5.1 Conclusions

In this thesis we proposed a two-level hierarchical modular approach to the problem of fault recovery in DESs. After giving a brief overview to the theory of discrete-event systems and also RW supervisory control theory in Chapter 2, basic concepts of a two-level hierarchical DES have been reviewed to derive an abstraction of the plant and to design a higher level supervisor. The modelling is carried out in a bottom-up fashion, while the control is applied using a top-down approach.

A finite-state automaton is assumed to model the plant in both normal and recovery modes. We assume full events observation, and thus there is no delay in the detection of failures in the system. Therefore, we model the entire plant in normal and normal-recovery modes, without using any diagnosis system. This assumption is merely to simplify the discussion. Extension of our setup to the case of control under partial observation is not difficult. In the problem with partial observation, we have to consider diagnosis.

Specifications of the system are divided into two main categories for both normal and recovery modes. The first category are those that address low-level (local, subsystem-level) issues for which we design a supervisor at the low-level using a modular approach, while the second group of specifications are those addressing global (system-level) issues and for this group we design a supervisor following a modular hierarchical approach. In our design methodology, faults are assumed to be permanent and that means if a fault occurs in the system, the plant will remain in the faulty condition. Moreover we assume not more than one failure in the system at any time of its operation (single-failure scenario). First, we design the required controllers at the low-level to enforce the design specifications of the plant for the first category of the specifications in both normal and recovery modes. After designing the supervisors for both normal and recovery modes, the system under supervision is computed at the low-level.

The starting point for applying hierarchical control theory is the system under supervision at the low-level. After defining the "significant" events and the vocal outputs for the plant at the low-level, we compute a consistent abstraction of the plant (high-level model) for the higher level using the approach in [26]. For the high-level model the required behaviors of the model are defined in the form of legal languages over the high-level alphabet set. We design the required supervisors for both normal and recovery modes at the high-level to enforce the high-level specifications, and finally the plant under supervision is computed at high-level. A computational design procedure, presented in Chapter 3 for the general case, summarizes the whole design methodology. We also investigate the nonblocking property of the plant under supervision.

In our framework, a modular approach is chosen mainly for the following advantages: a modular approach (divide and conquer) is usually easier for design and implementation purposes compared to a centralized approach; modules may easily be repaired or modified; and for the fault recovery problem in particular, a modular approach can reduce the computational complexity significantly (where the number of the controllers sometimes could be very high).

In addition to the modular design for the supervisors of the plant, we decided to use a two-level hierarchical approach. Use of a hierarchical modular can be justified as follows. Studies of real-world systems often show that the designers have to deal with the large number of states which usually leads to complex models and supervisors for these applications. An alternative way to manage complexity in these realistic systems when faced with state explosion, is to use an abstraction of the plant (high-level model) instead of the plant itself to simplify the complex behavior.

To illustrate the application of our design methodology for the failure recovery problem in Chapter 4, we considered the automatic control of an electrical power network in normal and faulty modes. We started by briefly reviewing electrical power generation and distribution networks. We described different components of these networks using DES models. A simple electrical power network consisting of generators, transformers, and different circuit-breakers was chosen to be modelled using DES models. We defined the required behaviors of the power network in terms of legal languages and accordingly the relevant supervisors were designed. We showed that DES models in general and modular supervisory control can be used as a good modelling tool for synthesis of controllers in electrical power networks.

To simplify the design procedures for the failure recovery and automatic control of electrical power networks, we further applied our hierarchical method to these networks and designed supervisors to control the plant. By applying supervisory control theory to the automatic control and fault recovery problem of power networks, we can combine these two important tasks into an integrated problem and finally, solution. Therefore, we will be able to give a compact solution to these problems. In the following section, we outline some directions for future research.

5.2 Future Research

This research can be continued in many directions, including the following.

- In our design methodology, we did not consider simultaneous failures, and single failure scenario was considered through-out the thesis. However, in practice it is very common for failures to occur simultaneously. Therefore, a possible future work is to extend our framework to deal with this problem using DES models.
- We used un-timed DES models to solve the supervisory control problem for the failure recovery in our models. Timed DES can be used as another possible extension.
- We assumed that there was no delay in detecting the failures in the plant (failure events were considered observable). Therefore we did not include any diagnosis system in our model. Another possible extension is to include a diagnosis system in our model. Furthermore, we may investigate recovery of the transient failures

of the system.

- To construct our two-level hierarchical model, the full event observation was assumed. However, we can extend our design methodology by considering unobservable events of the system and then constructing the hierarchical model. In this case, we may consider the *H-observability* concept in our hierarchical model to ensure the consistency between low and high-level models [7].

- Another interesting challenge is to extend the two-level hierarchical model to a multi-level hierarchical approach. This may provide further simplifications for the designers to deal with the complex behaviors of the real-world applications.

- In our particular application of designing automatic controllers for the failure recovery problem in electrical power networks, we used an abstraction of the components of the system. It will be more challenging if we use a more detailed model of these components using DESs. Also, we considered only short circuits as the main failures of these system. Another topic is to consider other types of faults which can occur in these networks. To tackle more detailed model, we need to develop more capable software to implement various hierarchical design algorithms.

Bibliography

[1] P.Baroni, G.Lamperti, P.Pogliano, and M.Zanella. Diagnosis of a class of distributed discrete-event systems.*IEEE Transactions on Systems, Man and Cybernetics, Part A,* 30(6), Nov. 2000 Page(s):731 - 752

[2] K. H. Cho and J. T. Lim. Synthesis of fault-tolerant supervisor for automated manufacturing systems: A case study on photolithographic process. *IEEE Trans. on Robotics and Automation*, 14(2), Pages:348 351, April 1998.

[3] C. G. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems.* Kluwer Academic Publishers, Norwell, Massachusetts, USA, 1999.

[4] P. Caines and Y. Wei. The hierarchical lattices of a finite machine. *Systems and Control Letters*, Pages: 257 - 263, July 1995.

[5] D. Gordon and K. Kiriakidis. Reconfigurable robot teams: Modeling and supervisory control. *IEEE Trans. on Control Systems Technology*, 12(5), Pages:763 - 769, Sept. 2004.

[6] P. Hubbard and P.E. Caines. Dynamical consistency in hierarchical supervisory control *IEEE Trans. Automat. Control*, 47 (1), Jan. 2002, pp. 37-52.

[7] S.-G.Kim, K.-H.Cho, and J.-T Lim. Hierarchical supervisory control of discrete event systems based on H-observability. *IEE Proceedings on Control Theory and Applications*, 150(2), March 2003 Page(s): 179 - 182

[8] M.Kezunovic, I.Rikalo, and D.J.Sobajic. Neural network applications to real-time and off-line fault analysis. *Int. Conf. on Intelligent System Application to Power Systems*, Montpellier, France, 1994, page(s):29 - 36.

[9] J. Liu and H. Darabi, Control reconfiguration of discrete event systems controllers with partial observation, *IEEE Trans. Syst. Man Cyber.* B, vol. 34, no. 6, pp. 2262 - 2272, Dec. 2004.

[10] R. J. Leduc. Hierarchical Interface-based Supervisory Control. PhD thesis, Department of Electrical and Computer Engineering, University of Toronto, 2002.

[11] K.L. Lo, H.S. Ng and J. Trecat, Power System Fault Diagnosis Using Petri nets. *IEE Proc.-Gener. Transm. Distrib.*, 1997, 144(3), Pages: 231 - 236.

[12] G. Lamperti and P. Pogliano. Event-based reasoning for short circuit diagnosis in power transmission networks. *IJCAI*, Nagoya, Japan, 1997.

[13] M. Moosaei and S. Hashtrudi Zad. Fault recovery in control systems: A modular discrete-event approach. *Intern. Conf. on Electrical and Electronics Engineering (CINVESTAV / IEEE)*, Page(s): 445 450, 2004.

[14] M. Moosaei and S. Hashtrudi Zad. Modular fault recovery in timed discreteevent systems: Application to a manufacturing cell. *Proc. IEEE Conference on Control Applications*, Toronto, Canada, pp. 928-933, Aug. 2005.

[15] H. Monsef, A.M. Ranjbar and S. Jadid. Fuzzy Rule-Based Expert System for Power System Fault Diagnosis, *IEE Proc.-Gener. Transm. Distrib.*, 144(2), 1997, Pages: 186 - 192.

[16] H.Marchand, M.Samaan. Incremental design of a power transformer station controller using a controller synthesis methodology. *IEEE Transactions on Software Engineering,* 26(8), Aug. 2000 Page(s):729 - 741

[17] K. Q. Pu. Modeling and control of discrete-event systems with hierarchical abstraction. Masters thesis, Department of Electrical and Computer Engineering, University of Toronto, 2000.

[18] M.A.P. Rodrigues, J.C.S.Souza, M.T. Schilling, and M.B.Filho. Fault Diagnosis in Electrical Power Systems Using Artificial Neural Networks. *Int. Conf. on Electrical Power Engineering*, 1997, Budapest.

[19] P.J.Ramadge and W.M.Wonham. Supervisory control of a class of discrete event processes. *SIAM J. Control and Optimization*, 25(1), Pages:206-230,1987.

[20] M. Sampath, S. Lafortune and D. Teneketzis. Active diagnosis of discrete-event systems. *IEEE Trans. Automat. Contr.*, 43(7), Pages: 908 - 929, 1998.

[21] TTCT software. Available at http://www.control.toronto.edu/DES/.

[22] B. Wang. Top-down design for RW supervisory control. Masters thesis, Department of Electrical and Computer Engineering, University of Toronto, 1995.

[23] W. M. Wonham. Notes on supervisory control of discrete-event systems. Can be down-loaded from http: //www.control.toronto.edu/DES/, 2004.

[24] K.C. Wong and W.M. Wonham. Hierarchical control of discrete-event systems. *Discrete Event Dynamic Systems: Theory and Applications*, 6 (3) July 1996, pp. 241-273.

[25] H. Zhong and W. M. Wonham. On hierarchical control of discrete-event systems. *In Proceedings of the 22nd Annual Conference on Information Sciences and Systems*, Pages: 64 - 70, Princeton, New Jersey, USA, March 1988.

[26] H. Zhong and W. M.Wonham. On the consistency of hierarchical supervision in discrete-event systems. *IEEE Transactions on Automatic Control*, 35(10), Pages: 1125 - 1134, 1990.