Fault Recovery In Discrete-Event Systems With Intermittent And Permanent Failures

Ganesh Koraginjala

A Thesis
In
The Department
Of
Electrical & Computer Engineering

Presented in Partial Fulfillment of the Requirements
for the Degree of Master of Applied Science
(Electrical & Computer Engineering) at
Concordia University
Montréal, Quebec, Canada

April 2009

© Ganesh Koraginjala, 2009

Library and Archives Canada

Published Heritage Branch

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque et Archives Canada

Direction du Patrimoine de l'édition

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence
ISBN: 978-0-494-63170-6
Our file Notre référence
ISBN: 978-0-494-63170-6

NOTICE:

The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distribute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

The author retains copyright ownership and moral rights in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

AVIS:

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, publier, archiver, sauvegarder, conserver, transmettre au public par télécommunication ou par l'Internet, prêter, distribuer et vendre des thèses partout dans le monde, à des fins commerciales ou autres, sur support microforme, papier, électronique et/ou autres formats.

L'auteur conserve la propriété du droit d'auteur et des droits moraux qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

Canada

ABSTRACT

Fault Recovery in Discrete-Event Systems with Intermittent and Permanent Failures

Ganesh Koraginjala

As systems grow more complex to cater to demanding operational requirements, they tend to suffer from increasing component failures. It is important to minimize the effect of these failures on the overall performance of these systems. In this thesis, fault recovery using discrete event systems theory is studied.

It is assumed that the plant can be modeled as a finite state automaton, and that is prone to failures. For this study all events are assumed observable and the extension to the case of partial observation is left for future research. The problem of the synthesis of fault recovery procedures is studied. In particular, the cases are studied in which the plant may return to normal operation. This could be either because the failures are intermittent or because the plant has the capacity to repair or reset. Both of the above cases are studied in this thesis.

It turns out that the problem is an instance of the problem of robust nonblocking supervisory control for countably infinite number of plants. The objective of the thesis is to obtain maximally permissive solution for the above problem. It is shown that the

desired supervisor can be obtained as the maximally permissive solution of a robust control problem involving a bounded number of plants. Furthermore, an iterative procedure is provided to solve the original problem involving an infinite number of plants. The procedure is guaranteed to converge in a bounded number of steps. Several examples are provided to illustrate the proposed procedures.

Acknowledgments

I, sincerely, appreciate the help and support of my supervisor, Dr. Shahin Hashtrudi Zad and thank his continuous support and encouragement throughout this research. I also would like to thank my friends, Mohsen Azizi, Mani Mesgarpourtousi and Nathalia Parra for helping me in carrying out this research through completion.

Table of Contents

List of Figures

List of Tables

Abbreviations

DES : Discrete Event System

LQR : Linear Quadrant Regulator

IH : Infinite Horizon

FH*l* : Finite Horizon *l*

Chapter 1

Introduction

1.1 Introduction

As systems grow more complex to cater to demanding operational requirements, they suffer from increasing component failures. It is important to minimize the effect of these failures on the overall performance of these systems. In this thesis, we study fault recovery using discrete event systems theory. We consider a specific case wherein the systems are capable of returning to normal operation from failure modes of operation.

The task of designing and implementing control policies for these large systems under various failure scenarios is very complex and computationally expensive. Large manufacturing systems, space systems, and communication systems are all examples of discrete event systems (DES). A DES model of a spacecraft propulsion system may have

millions of states to deal with. Failure scenarios are defined by number of failures, types of failures, and times of occurrences of these failures[1].

A failure can be an event occurring at any level of system dynamics and cause the system to malfunction or show degraded performance or shutdown completely. A failure can be permanent or nonpermanent. An example for a permanent failure can be a broken shaft in a motor system wherein a nonpermanent failure can be a short circuit in power system network. A permanent failure keeps the system in failure mode indefinitely whereas a nonpermanent failure keeps the system in failed mode for some duration only and then it may disappear.

These large systems are usually controlled for reasons such as: to

i. Avoid unwanted behaviour (safety requirements)
ii. optimize use of resources
iii. obtain optimum performance
iv. provide reconfiguration and/or alternative operational paths in case of failures

For example, in a manufacturing system, a controller (supervisor) can be used to prevent buffer overflow/underflow. The effect of a controller is, in general, restrictive on the system's behaviour. The system or process to be controlled is called the 'plant.' We assume that this plant can be modeled as a state machine and that this plant contains finite number of states.

[1] In this thesis, a failure and a fault carry same meaning and are used interchangeably.

In DES theory, the events that happen in the plant are represented as symbols and the set of all possible sequences of symbols generated by a particular plant is called the language of the plant. In the theory of supervisory control of Ramadge-Wonham (RW), a DES supervisor is capable of restricting the plant behaviour by restricting the language generated by the plant within pre-specified limits. These limits are called specifications and are set forth by the safety and performance requirements.

The following control loop depicts the control structure where the supervisor S reads the sequence generated by the plant ($s = \sigma_1\sigma_2...\sigma_n$) and then enables only events to be generated by the plant ($S(s)$).

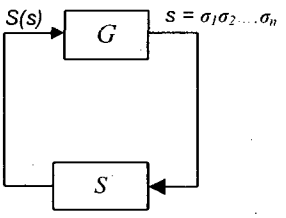

Fig. 1.1: General control structure for plant G and supervisor S

In the above control structure, the supervisor S is in a feedback loop with the plant G. The supervisor monitors the event dynamics in the plant and then sends commands (in terms of enabled events) to the plant to ensure that the plant generates only those event strings stipulated by the specifications.

Generally, the specifications are given in terms of legal and illegal event strings. However, it is possible to represent these specifications in terms of safe and unsafe states.

In that case, the supervisor can work in a state feedback fashion and limits plant behaviour only to the safe states stipulated by state specifications.

A return-to-normal event is defined in the recovery mode of the plant. When this event occurs, the plant moves back from recovery mode to normal mode. This event effects the dynamics of the plant directly. The plant actually moves from a state in the recovery mode states to a state in normal mode states. After moving back to normal states, the plant may experience a different settling of normal mode states and exhibit different normal behaviour. A simple example is presented in Chapter 3. Designing the DES supervisor to effectively implement the control policies considering these changes in dynamics is a complicated task. The problem considered in this thesis can be summed up as, "Designing a supervisor to control a plant when the plant is capable of returning to normal mode from recovery mode."

The return-to-normal event may occur in two cases. The first is when the failures are intermittent and the return-to-normal events correspond to "failure-correct" events. "Failure-correct" events are uncontrollable (i.e., can not be disabled or enabled by the supervisor). The second case is when the plant has the capacity to either repair the faulty components (or replace them) or "reset" itself to normal operation. In this case, return-to-normal events are "recovery-action" events, and controllable (i.e., can be enabled or disabled by the supervisor). Due to the difference in the controllability properties of return-to-normal events as mentioned above, the nonblocking specification (i.e., the ability to prevent deadlocks and livelocks) of the resulting control problem will be different.

In the next section, we review some relevant articles in the literature.

1.2 Literature Review

There are certain frameworks and techniques used for fault recovery in discrete event systems as reported in literature. These frameworks consider permanent failures and do not expect the plant to return to normal mode from recovery modes. There were no studies reported in open literature about how to compute supervisors for plants which are capable of returning to normal modes of operation from recovery modes through well defined return-to-normal events.

In [17], a model-based programming method is provided to perform reconfiguration of fault recovery. This method tracks the present state of the system, diagnoses the fault and provides reconfiguration. A version of this framework is used in a deep space probe (Deep Space One, DS1) by NASA. In this setup, the *most likely* present state of the plant is continuously estimated (by a 'model-based executive,' called Titan). This program, then generates a sequence of control actions to move the plant to more desired state dictated by specifications.

In [6], adaptive supervisor synthesis algorithms are presented. The algorithm does two parts: learning and repairing. A supervisor is computed that is capable of reconfiguring itself in supervising a team of robots. In particular, robots switch offline due to failure events and the algorithms learn and repair the supervisor to control the new set of robots. When a robot goes offline, the learning algorithm deletes the events which belong to the

failed unit alone from the previous supervisor, and then the repair algorithm restores the fractured automaton. However, the article does not deal with the case when new units dynamically join the team. The authors report this aspect as possible future research.

In [16], a weaker notion of fault-tolerance is considered to be sufficient in some applications. As such, following any fault, the system is guaranteed to reach a recovery state from where the subsequent behaviours are subsumed by those that are possible from a nonfaulty state. Necessary and sufficient conditions for existence of a weakly fault-tolerant supervisor are given. However, this particular supervisor will not provide a viable solution for mission critical applications such as spacecraft. Also, the category of applications for which this supervisor can be used effectively is not provided.

[5] deals with systems particularly with multiple resources failures. This article considers supervisory control for deadlock-free resources allocations, in manufacturing systems. This reference is of particular interest since, in the setup considered, the controller must guarantee that a set of resource failures does not propagate through blocking to stall other portions of the system. It is considered in such a way that the plant can continue operating without the need for the failed resource. However, it will not be the case for many practical applications. Also, a procedure to successfully separate the failed resource from the rest of the plant has to be developed.

In [10], fault recovery problem in discrete event systems is considered. A diagnostic system which can detect and isolate the faults with bounded delay is assumed to be available. In this framework, the diagnostic system can be designed using any technique

as long as the lower and upper bounds of the delay in detecting and isolating the fault are available. This improves the flexibility of the design. However, this framework results in a more conservative supervisor for the fault recovery. The authors proposed a modular switching supervisory scheme. An extension of this framework to timed DES is dealt with in [19].

[14] mainly deals with the fault recovery in discrete event systems using observer-based supervisors. A modular switching scheme of supervisors is proposed for fault recovery. However, such cases where recovery to normal operation is possible, are not considered. Two solutions are provided, one in which the recovery supervisor is in feedback loop when the system is started in its normal mode, and another in which the recovery supervisor is engaged only when a failure is detected and isolated.

In [12], the problem from [10] is transformed into an equivalent robust nonblocking supervisory control problem under partial observation. A set of necessary and sufficient conditions for the existence of a solution for the fault recovery problem are proposed. In this framework, the fault events are unobservable but can be diagnosed. One of the main assumptions in this framework is that the exact model of the plant in any of its operational mode (normal, transient, and recovery) is known. Here it is assumed the faults are permanent and as a result, the corresponding robust control problem involves a finite set of plant models (each corresponding to a mode of operation). Here it is assumed that the permanent fault occurs once, following which recovery actions are taken.

In this thesis, we study the problem of fault recovery when return to normal operation is possible either because the faults are intermittent or because the plant has the capacity of repair or reset.

1.3 Thesis Contributions

The contributions of this thesis can be summarized as follows.

The problem of fault recovery in DES systems is studied when return to normal operation is possible. Return-to-normal can be either because the faults are intermittent or because the plant has the capacity to repair or reset.

We considered two cases of supervisory control problem, one with controllable return-to-normal (recovery-action) events and the second one with uncontrollable return-to-normal (fault-correct) events. In each case, the resulting supervisory control problems will be instances of Robust Nonblocking Supervisory Control Problem of (*countably*) *infinite* number of plants. The solution for robust control of *finite* number of plants is given in [2],[13]. Here we extend some of those results to the case of infinite number of plant models. Specifically, we looked for the *maximally permissive* supervisor (i.e., a supervisor that does not disable an event unless it has to, in order to meet the design specifications). We show that the above optimal (maximally permissive) solution can be obtained as the maximally permissive solution of a problem involving a bounded number of plants. Furthermore, we propose an iterative procedure for obtaining such solution. The procedure is guaranteed to converge in a bounded number of steps.

Three case study problems are provided to illustrate the solution procedure.

1.4 Thesis Outline

In Chapter 2, we review the background material for discrete event systems and few concepts used in this thesis. We study robust non-blocking supervisory control problem. An example is given to illustrate computation of supervisors in standard (non-robust) RW-based procedures.

Chapter 3 formulates the problem that is studied in this thesis. The problem of finding a robust supervisor for failure recovery when the plant is capable of returning to normal from recovery modes is given in detail. The significance of the return-to-normal events is discussed.

In Chapter 4, we solve the fault recovery problems posed in Chapter 3. As mentioned before, the problems are instances of Robust Nonblocking Supervisory Control of countably infinite number of plants. We also propose iterative procedures for solving the problems.

In Chapter 5, three application examples are solved using the design algorithm presented. TTCT [15] has been used to carry out the computations. The simplified propulsion system (SPS) and extended propulsion system (EPS) have uncontrollable return-to-normal events. The automatic resource allocator (ARA) has a controllable return-to-normal event.

Finally we conclude the thesis with conclusions and future research in Chapter 6.

1.5 Conclusion

In this chapter, Discrete Event Systems (DES) are introduced and the general nature of the problem explained. The existing literature is reviewed. We reviewed the thesis contributions and thesis outline is presented.

Chapter 2

Background Review

Formally, a DES can be considered as a dynamic system equipped with a discrete state space and a state transition structure. It is a system that is asynchronous or event driven. In this Chapter, we review some basic concepts as well as supervisory control in DES. We also review state-based approach in DES and provide an example.

2.1 Introduction

A framework to model supervisory controllers for DES was introduced in [18] by Ramadge and Wonham. It will be referred to as RW. RW framework is automaton-based. The following are some salient points of this framework:

i. The framework proposes procedures that take the finite state automata models of the plant and the design specifications and generate control structures, guaranteed to satisfy the design specifications and eliminate blockings.

ii. Both the plant and controller are treated separately in this framework.

iii. The framework can characterize the solutions of the control problem in terms of naturally definable control concepts and properties, such as controllability, and observability.

One of the challenges that this framework faces is the computational complexity for large-scale systems. One way to deal with this complexity is to exploit horizontal and vertical modularity. For more details, the reader is referred to [19] and [11].

2.2 Automata

Finite State Automaton

Consider a plant modeled as a finite state automaton G, with

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

where Q is the set of states, Σ is the event set. δ is the partial transition function $\delta : QX\Sigma \to Q$, q_0 is the initial state, and $Q_m \subseteq Q$ is the set of marked states. The transition function defines how the plant moves from state to state driven by the events. The marked states represent those states of plant that are significant for reasons mostly to do with plant operation.

Let Σ^* denote the set of all sequences of events Σ and the empty sequence ε. A language over Σ is any subset of Σ^*, i.e., an element of the power set Pwr(Σ^*). Then the definition includes both the empty language ϕ, and Σ^* itself. For language $L \subseteq \Sigma^*$,

$\overline{L}$ denotes the prefix-closure (or simply closure) of L. L is closed if $L = \overline{L}$. For two languages $L, M \subseteq \Sigma^*$, L is M-closed if $L = \overline{L} \cap M$.

Let $L(G) = \{s \in \Sigma^* \mid \delta(s, q_0) \text{ is defined}\}$ be the uncontrolled language generated by G over alphabet Σ. $L(G)$ represents the set of all possible event sequences that take G from the initial state to a reachable state. The set of all possible event sequences which take G from the initial state to some marked state in G is represented by $L_m(G)$. While $L(G)$ represents the closed behavior of the plant, $L_m(G)$ represents the marked behavior of the plant G.

Synchronous and Parallel Products

Complex controlled DES are directly modeled as product structures of simpler components. Two operations that are often used in modeling the joint operations of DES are 'product' and 'Synchronous product.'

Let G_1 and G_2 represent two DES with alphabets Σ_1 and Σ_2. In product, the two DES synchronize the occurrence of their common events: $\sigma = \Sigma_1 \cap \Sigma_2$ occurs if it is defined and enabled in both G_1 and G_2. Events that are not in $\Sigma_1 \cap \Sigma_2$ are disabled.

In synchronous product, the DES synchronize on the common events (similar to product). However, each may execute its own private events without any synchronization; that is events in $(\Sigma_1 - \Sigma_2)$ and $(\Sigma_2 - \Sigma_1)$. The reader is referred to [18] for more details.

TTCT is a software program that is used for analysis, synthesis, and verification of DES and supervisory control in DES. [18] gives us a review of TTCT procedures.

The TTCT **meet** procedure computes the reachable part of product of G_1 (with event set Σ_1) and G_2 (with event set Σ_2) to create G_3 with event set $\Sigma_3 = \Sigma_1 \cap \Sigma_2$.

$$G_3 = \text{meet}\ (G_1, G_2)$$

The **sync** procedure forms the synchronous product of G_1 and G_2 to create G_3. The event set of G_3 will be $\Sigma_1 \cup \Sigma_2$.

$$G_3 = \text{sync}\ (G_1, G_2)$$

Nonblocking Automata

G is nonblocking if for any string $t \in L(G)$, there is at least one string 's' such that $ts \in L_m(G)$. This means that from every reachable state in G, there is a path to a marked state. This DES is said to be nonblocking if $\overline{L_m(G)} = L(G)$, where $\overline{L_m(G)}$ represents the prefix-closure of $L_m(G)$. Alternatively, let $R(G)$ be the set of states of G *reachable* from the initial state q_0.

$$R(G) = \{ q \in Q \mid \exists s \in L(G),\ q = \delta(q_0, s) \}.$$

The set of *coreachable* states of G is defined according to

$$CR(G) = \{ q \in Q \mid \exists s \in \Sigma^*,\ \delta(q, s) \in Q_m \}.$$

Thus G is nonblocking if and only if $R(G) \subseteq CR(G)$.

2.3 Supervisory Control

Consider a plant $G = (Q, \Sigma, \delta, q_0, Q_m)$. It is assumed that the event set Σ can be partitioned into controllable and uncontrollable events, i.e., $\Sigma = \Sigma_c \bigcup \Sigma_{uc}$. Controllable events can be disabled or enabled. In this thesis, we assume all events are observable.

Let $E \subseteq Q$ denote the set of "legal" (safe) states of the plant *G*. In supervisory control theory, we want to design a supervisor *S* to ensure that the plant never leaves *E*, and the plant under supervision is nonblocking. A supervisor monitors the sequence of observable events generated by the plant and restricts the behavior of the plant to the "legal" states by disabling and enabling of controllable events.

The assumption that the specification is given in terms of "legal" (safe) states is not limiting. Problems involving "legal event sequences" can be transformed into equivalent problems involving "legal" states as specifications by adding suitable automata (modifiers) to capture the "history" of event sequences [18]. More detail on modifiers is provided at the end of this section.

Based on $s = \sigma_1 .... \sigma_k \in \Sigma^*$ generated by the plant (under supervision), and observed by the supervisor, the supervisor can determine the state of the plant. The supervisor can be defined as a state-feedback map $S : Q \to \Gamma$ where $\Gamma := \{\Sigma' \subseteq \Sigma | \Sigma' \supseteq \Sigma_{uc}\}$. At a state *q*, *S(q)* is the set of events enabled by the supervisor. *S* interacts with *G* to form the closed-loop system. The traditional control loop in DES is shown in Fig. 2.1. Note that *S* only

disables controllable events. A supervisor that never disables uncontrollable events is called *admissible* (controllable).

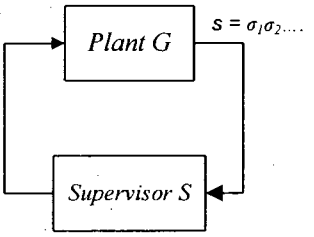

Fig. 2.1: Traditional control loop

Let $L(S/G)$ be the language generated by the plant G under supervision of $S(S/G)$, and $L_m(S/G)$ be the marked behavior. The closed behavior $L(S/G)$ is defined inductively as follows [Wonham]:

- $\varepsilon \in L(S/G)$
- If $s \in L(S/G)$, $\sigma \notin S(\delta(q_0, s))$, and $s\sigma \in L(G)$, then $s\sigma \in L(S/G)$
- No other strings belong to $L(S/G)$.

We then have $L(S/G) \subseteq L(G)$, and according to the above definition, $L(S/G)$ is closed. The marked behavior of S/G is defined as

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

Thus $L_m(S/G)$ consists of sequences in the marked behavior that can be generated in system under supervision. Note that in *S/G*, marking is still determined by the plant *G*.

Abusing the notation, we let *R(S/G)* and *CR(S/G)* be the reachable and coreachable states of the plant under supervision:

$$R(S/G) = \{ q \in Q \mid \exists s \in L(S/G) \ : \ q = \delta(q_0, s) \}$$

$$CR(S/G) = \{ q \in Q \mid \exists s, s' : s \in L(S/G), ss' \in L(S/G), q = \delta(q_0, s) \text{ and } \delta(q, s') \in Q_m \}.$$

In the supervisory control problem, the objective is to find an admissible supervisor S such that

(i) $R(S/G) \subseteq E$;

(ii) $R(S/G) \subseteq CR(S/G)$.

The former condition confines G stay inside the set of desirable states, and the latter ensures that S/G is nonblocking. Assuming such S exists, then $R(S/G)$ is a controllable, nonblocking predicate, and vice versa, if a subset of E that is controllable and nonblocking exists, then the supervisory control problem is solvable [18].

A supervisor S is *maximally permissive* (*optimal*) if it only disables an event when it has to. Note that in the case of full observation as in this thesis, we can always construct an optimal supervisor [5]. However, an optimal supervisor may not exist for the case of control under partial observation.

The supervisory control problem discussed earlier is referred as "state-based" supervisory control problem since the design specification is given in terms of safe (legal) states (E). An alternative, equivalent formulation of the supervisory control problem follows a linguistic approach in which the design specification is given in terms of a legal language (i.e., legal event sequences).

The solutions to the linguistic supervisory control problem can be characterized in terms of controllable and $L_m(G)$-closed languages.

Definition 2.3.1. [11] A language K is called *controllable* with respect to the plant G if

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

In the linguistic approach, the supervisor is considered as a map $S : \Sigma^* \to \Gamma$.

Let $E \subseteq L_m(G)$ be the legal language (design specification). The supervisory control under partial observation is to find an admissible supervisor such that

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

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

It can be shown that [8], [4] for $K \subseteq E$, there exists an admissible supervisor S such that

(i) $L_m(S/G) = K$

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

If and only if

K is controllable and $L_m(G)$-closed.

Thus the class of controllable and $L_m(G)$-closed languages characterizes the set of solutions to the supervisory control problem.

Using state-modifiers, any language-based supervisory control problem can, without any loss of generality, be converted into a state-based problem.

State-Modifiers

A state-modifier refines the state transition graph of the plant so that event sequences leading to every state are either all legal or all illegal, creating a partition of states into safe and unsafe states.

We compute the modifiers from their respective mode specifications models. To compute a modifier, the following steps have to be followed:

1. Take the final specification for a given mode;
2. Add a dump state to it and connect each state in the plant with the dump state using event transitions that are not defined at that particular state in the plant;
3. Finally, mark the dump state.

The plant transition structure can be modified by using the **meet** function from TTCT [15]

Modified Plant = meet(plant, state-modifier)

The next section shows an example of supervisory control.

Example

Let us consider the following plant model:

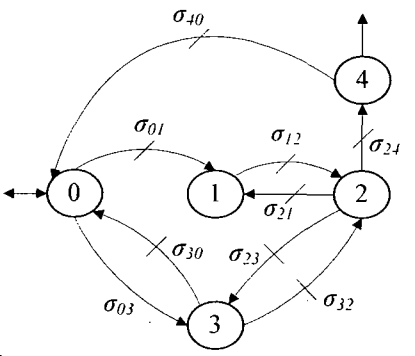

Fig. 2.2: Plant Model

The plant has 5 states and 9 event transitions. Event σ_{03} is an uncontrollable event and rest of the events are all controllable events. Throughout this thesis, marked states are shown with an outgoing arrow. Also controllable events are crossed (⟋). It can be seen that states 0 and 4 are marked. Let E be the state-based specification denoting state 2 as the unsafe state. We construct the supervisor as follows.

State 2 can be reached from states 1 and 3 through the event transitions σ_{12} and σ_{32}. The supervisor disables these two events resulting in the following model:

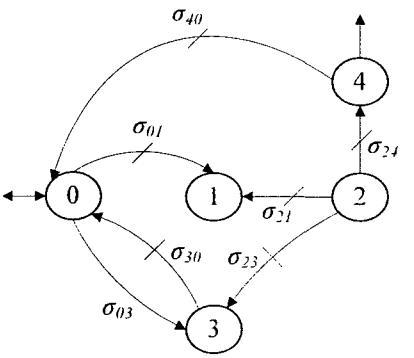

Fig. 2.3: Partial model of the supervisor

It can be seen from Fig. 2.3 that the state 2 became unreachable. The following model shows the model for the supervisor with state 2 removed:

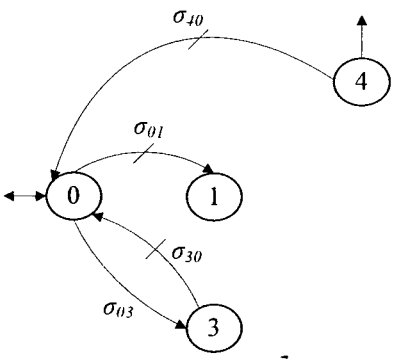

Fig. 2.4: Partial model of the supervisor

It can be seen that the above model shows state 4 as unreachable. Further, we need to remove state 4 and event σ_{40} resulting in the following model:

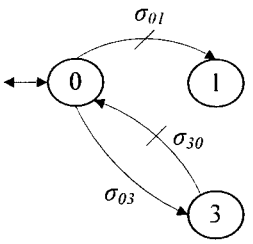

Fig. 2.5: Partial model for the supervisor

The above model shows all reachable states; however, we face the problem of blocking in this model. Once the plant reaches state 1, it is blocked. To solve this problem, we need to disable event σ_{01} and remove state 1 as well. The following model shows the final supervisor S:

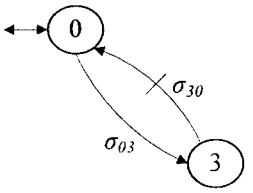

Fig. 2.6: Supervisor *S*

The above design procedure can be done in a systematic way [18].

2.4 Robust Supervisory Control

In this thesis, we study fault recovery problems in cases where return-to-normal operation after a fault is possible. As we will see in Chapter 3, the problem will become an instance of robust control of an *infinite* number of plants. Here we review some relevant results from literature on robust control of *finite* number of plants.

Consider a finite set of plants $G_1,...,G_n$, each a finite-state automaton $G_i = (Q_i, \Sigma_i, \delta_i, q_{0_i}, Q_{m_i})$. Let $E_i \subseteq L_m(G_i)$ be a set of legal languages. Let $\Sigma = \bigcup_{i=1}^{n} \Sigma_i$. It is assumed that the plant models agree on controllability of events. That is $\sigma \in \Sigma_i \bigcap \Sigma_j$ is controllable if only if σ is controllable in both G_i and G_j and otherwise it is uncontrollable in both G_i and G_j. The Robust Nonblocking Supervisory Control Problem is find a supervisor S such that

(i) $L_m(S/G_i) \subseteq E_i$

(ii) $\overline{L_m}(S/G_i) = L(S/G_i)$

Let G be a finite state automaton with $L(G) = \bigcup_{i=1}^{n} L(G_i)$ and $L_m(G) = \bigcup_{i=1}^{n} L_m(G_i)$. Also define $E = \bigcap_{i=1}^{n} (E_i \bigcup (\Sigma^* - L_m(G_i))) \bigcap L_m(G)$.

Then S solves the robust nonblocking supervisory control problem if and only if there exists $K \subseteq E$ such that [1], [13]:

1. K is controllable with respect to G

2. K is $L_m(G)$ -closed

3. K is G_i -nonblocking $(\overline{K \cap L_m(G_i)} = \overline{K} \cap L(G_i))$

The following two lemmas will be useful.

Lemma 1. [13] Suppose G_1 and G_2 are DES over the alphabet Σ with $L(G_1) \subseteq L(G_2)$. For a supervisor S:

$$L(S/G_1) = L(S/G_2) \cap L(G_1).$$

Lemma 2. [1] Suppose G_1 and G_2 are DES over the alphabet Σ with $L(G_1) \subseteq L(G_2)$ and $L_m(G_1) \subseteq L_m(G_2)$. For a supervisor S:

$$L_m(S/G_1) = L_m(S/G_2) \cap L_m(G_1).$$

2.5 Conclusion

In this chapter, we presented various details about Discrete Event Systems relevant to this thesis. We explained concepts such as supervisory control and robust supervisory control. In the next Chapter, the problem studied in this is formulated.

Chapter 3

Problem Formulation

In this chapter, we formulate the supervisory control problem for failure recovery when the plant can return to normal mode from recovery modes. As mentioned in Chapter 1, return-to-normal is possible in case where either the failure is intermittent or some mechanism exists to fully repair (replace) the faulty components. The general structure of the plant to be controlled is described in Section 3.2. In Section 3.3, the problem statement is established. A mathematical example is presented in Section 3.4.

3.1 Introduction

In complex control systems that comprise thousands of components, design and implementation of control policies become extremely difficulty to manage. This is partly due to the large size of the problems. For example, let us consider a spacecraft. The task at hand is to design control policies that will successfully launch the spacecraft into a particular orbit. This example is considered in detail in [17]. As an example consider a

spacecraft containing one science camera, and two twin engines. One of the design specifications could be:

> "Heat up both engines (standby mode). Meanwhile, turn the camera off, in order to avoid plume contamination. When both tasks are accomplished, thrust one of the two engines, using other engine as backup in case of primary engine failure"

The control policy required to achieve this specification must be able to turn on various heating elements, valve drives, open sets of valves and record and interpret various sensor readings often in a predefined order. This spacecraft, as a discrete-event plant would have millions of states, and a very large set of design specifications. Designing and implementing supervisory control policies for this set up would be complex.

In addition to the intrinsic complexity posed by the large number of components, many components are prone to failures. An example of a failure is for an inlet valve to a particular engine becoming stuck-closed, and consequently, the task of firing and achieving thrust from that particular engine becomes impossible. Supervising a complex system such as a spacecraft with possible component failures poses considerable difficulties in developing control policies. As a result, the development of systematic methods for control systems has been the subject of extensive research.

Generally, failures can be **permanent** or **intermittent**. A valve becoming stuck-closed could be an example of permanent failure and an electronic circuit becoming open

as a result of heating and high temperature is an example of intermittent failure. Depending on the severity of failures, the plant either can recover (through repair or replacement of faulty components) or continue to function at lower performance or can be completely shut down (for safety reasons).

In this thesis, we study fault recovery in systems that can be modeled as Discrete-Event Systems (DES). In a DES framework, occurrences of failures are modeled by (uncontrollable) failure events. We assume that a fault diagnosis system is available to detect and isolate the faults and report them to the supervising control system. The combination of plant and diagnoser system can be regarded to be in one of the three different modes: 1) **Normal mode**, 2) **Transient mode**, and 3) **Recovery mode**. As depicted in Fig. 3.1, the system initiates in normal mode. Once a failure event occurs, the system enters into the transient mode before the failure is detected by the diagnosis system. Once the failure is detected, and isolated, the system enters into recovery mode.

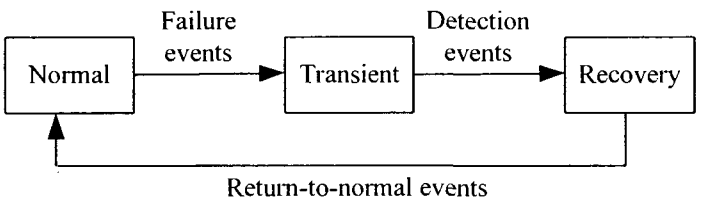

Fig. 3.1: Different modes of the system (plant and diagnoser)

The responsibility of a DES supervisor is to ensure that the plant works in accordance with pre-determined set of design specifications (concerned with the safety of the entities involved such as the equipment, personnel, and environment). These specifications

normally restrict the plant behaviour in all three modes. A supervisor should ensure that the plant adheres to these specifications both in normal, transient, and recovery modes.

As an example, let us consider a fuel pump system feeding a furnace, which has two primary pipes (P1, P2) and one auxiliary pipe (AUX). The system also has valves to shut/open the flow and individual sensors to read the flow rate in each pipe. Let us assume another flow meter reads the fuel flow rate reaching the furnace. Fig 3.2 below shows the setup:

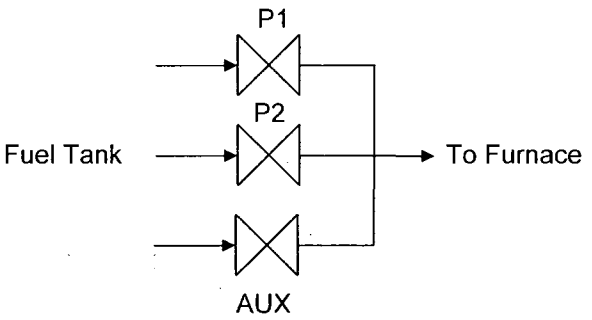

Fig. 3.2: Furnace System with three pipes

In normal mode of the operation, both primary pipes allow the fuel flow and the auxiliary pipe is used as a backup in case of a failure in any of the primary pipes. For simplicity, we assume that the failure can occur only in one pipe at a time. Example specification in normal mode for this system would be as following:

'The flow rate can be between 10 – 20 sccm (sccm- Standard Cubic Centimetres per Minute).'

An intermittent failure in this example can be complete blockage in one of the primary pipes. This failure is assumed to be detected by a sensor rather quickly. Upon

occurrence of this failure, the system enters into recovery mode. The recovery mode specifications would be:

'The flow rate should not fall below 8 sccm, if it
falls below 3 sccm; the system has to be shutdown.'

Recovery mode specifications are generally less restrictive than those of normal mode (Here, in terms of required flow rate). If the flow rate falls below 8 sccm, the recovery action is to open the auxiliary pipe to boost the flow rate. If the flow rate falls below 3 sccm, the system has to be shutdown for safety reasons so not to allow the furnace on very low fuel input. When the flow rate falls below 3 sccm, opening the auxiliary pipe is not an option since the furnace faces dry run conditions and needs to be shutdown before restarting again with full flow rate.

In this thesis, we study the design of supervisory control systems for plants that are subject to failures. We assume that the plant may return to normal mode of operation either because some of the failures are intermittent or because full repair (or replacement) of faulty components is possible.

Typically some of the events are unobservable, for example, failure events are usually unobservable. In this thesis, as a first step towards solving problems involving return-to-normal events, we will assume for simplicity that all events are observable. The assumption of observability of failure events means the diagnosis system can detect and isolate failure events before the next event occurs in the plant. The extension of the

solutions developed in this thesis to the case of control under partial observation is left for future research.

As a result of the assumption of observability of failure events, the transient mode never occurs (Fig.3.1) and after a failure event the system enters the recovery mode. We will base our solution on the Ramadge-Wonham (RW) theory of supervisory control. Fig. 3.3 shows the control loop in which a plant G is supervised by supervisor S. S monitors the events unfolding in the plant and at any given step, based on the sequence generated by the plant (s), makes a decision about the disablement and enablement of controllable events of the plant ($S(s)$).

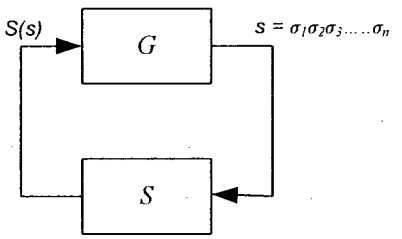

Fig. 3.3: Control loop (G: Plant; S: Supervisor)

3.2 Plant Model

We assume that plant G can be modeled as a finite-state automaton $G = (Q, \Sigma_G, \delta_G, q_0, Q_m)$ where Q represents the set of states and Σ_G represents the set of events in the plant. Both the normal and recovery modes are included in this model.

The set of states can be divided into two main categories -- **normal**, and **faulty**. The states that represent the normal modes of the plant belong to the normal state set, denoted by 'Q_N.' The states that represent the recovery (faulty) modes of the plant belong to the

recovery (faulty) state set, denoted by Q_R and $Q_R = \bigcup_{r=1}^{p} Q_R$. The recovery states are reached when different failure events occur. Fig. 3.4, below, depicts the state sets, with failure event (f_i) and return-to-normal (Σ_{r_i}) events, assuming single failure scenarios (i.e., no simultaneous occurrences of failures). Throughout this thesis, we will consider only single-failure scenarios.

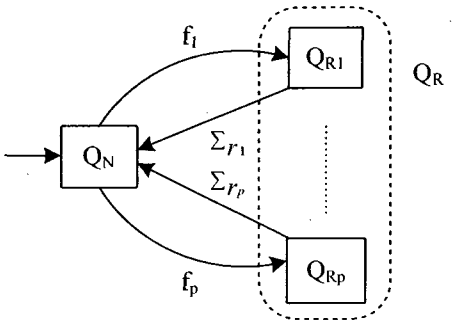

Fig. 3.4: Normal and Recovery state sets

The event set Σ_G contains two most important event sets: (1) **Failure events** (Σ_f); and (2) **Return-to-normal events**(Σ_r). A failure event takes the plant from normal mode of operation to recovery (failure) mode of operation and is an uncontrollable event. In this thesis, all of the events including failure events are assumed to be observable. We consider both intermittent and permanent failure events. We define return-to-normal events for both kinds of failure events.

The return-to-normal events are those events that take the plant from recovery modes (Q_R) to normal mode of operation (Q_N).

There are two types of return-to-normal events: (a) **Failure-correct events**, and (b) **Recovery-action events**. The set of failure-correct events is denoted by Σ_{fc}. A failure-correct event can be understood as an event that models the disappearance of a particular intermittent failure. These events are uncontrollable and their occurrence needs no intervention from the system, in other words, no action is required that is external to the failed component. An example of intermittent failure can be an electronic circuit that becomes open-circuit once it heats up. After it becomes open, it cools down, the failure disappears and the circuit resumes operation.

Recovery-action events set is denoted by Σ_{ra}. These are the actions that the plant takes (such as repair or replacement of faulty parts) to return to normal mode of operation. Opening an auxiliary valve in response to a stuck-closed primary valve is an example of a recovery-action event. These events are assumed to be controllable events. The disjoint union of Σ_{fc} and Σ_{ra} is denoted by $\Sigma_r (\Sigma_r = \Sigma_{fc} \dot{\cup} \Sigma_{ra})$. Σ_r denotes the set of all return-to-normal events.

The event set Σ_G can be partitioned into controllable and uncontrollable event sets, i.e. $\Sigma_G = \Sigma_{G,C} \dot{\cup} \Sigma_{G,UC}$. The event set Σ_G can also be expressed as the union of two subsets: $\Sigma_G = \Sigma_p \cup \Sigma_f$, where $\Sigma_f = \{f_1 ... f_p\} (p \geq 1)$ represents the set of failure events, and Σ_p represents the set of all other plant events (excluding the failure events). Obviously, $\Sigma_r \subseteq \Sigma_p$.

We finish this section with a note on recovery to normal events. While the consequence of a return-to-normal event is mainly the plant moving from recovery mode to normal mode, the dynamics of the plant are more affected because of this event. Specifically, after a recovery event, the plant may enter a normal state that it would not have entered if it had not experienced failure and then recovery. In other words, some normal states are only reachable after return-to-normal event.

To elaborate on this, let us revisit the furnace system we considered in the introduction (Section 3.1). In this example, we considered a pump system with two primary pipes and an auxiliary pipe. When a failure (blocking of a primary pipe) occurs, the plant enters into recovery mode. To enforce the recovery specification, we can open the auxiliary pipe. But when the intermittent failure is rectified, i.e., blockage is cleared in the primary pipe due to a failure-correct event and when the plant moves back to normal mode, now in the normal mode of the plant, all three pipes are in open condition and allowing a flow above the level required for the normal operation of the furnace.

3.3 Problem Statement

The problem considered in this thesis can be stated as "**Designing a supervisor to control a plant when the plant might return to normal modes from recovery modes.**" The design of such supervisor has to consider the plant dynamics involved, when the plant moves from recovery mode to normal mode. The two main aspects that the supervisor has to take into consideration are:

a) **The plant under supervision has to follow the designated specifications for each of the plant modes;** and

b) **The plant under supervision should be nonblocking in all of the plant modes.**

Let us elaborate on these two aspects:

Safety requirements: These requirements ensure that the plant operates safely in all of its operational modes. The safety is achieved by restricting the plant behaviours to specifications in all modes. The plant initially starts in normal mode. The plant under supervision in normal mode should abide by the specifications and should not enter into the forbidden states in normal mode. Upon occurrence of a failure, the plant enters into recovery (faulty) mode. Consequently, the supervisor must be able to limit the plant state only to safe states as per the specifications in recovery mode. Also this supervisor must be able to restrict the plant only to safe normal states upon return to normal states from recovery mode.

Nonblocking requirements: The supervisor must be able to control the plant in such a way that nonblocking is guaranteed in all the modes. At first, the plant under supervision in normal mode should be nonblocking. Upon occurrence of a failure, the plant enters into recovery mode. The plant under supervision in this mode should also be nonblocking. At this juncture, it is important to note that if a blocking situation in normal mode can, subsequently, be cleared only by a failure, the situation in normal mode still has to be considered as blocking. This is because we cannot count on a failure to occur to come out of blocking. Similarly, when the plant is in recovery mode, if only an uncontrollable return-to-normal event (failure-correct event) can clear any existing

blocking, the situation has to be considered as blocking, since uncontrollable return-to-normal events in the case of intermittent failures are not guaranteed to occur and should not be relied on for getting out of deadlocks and livelocks.

The controllability nature of return-to-normal events in a mode *i* presents us with two cases:

Case 1: The return-to-normal event set Σ_{ri} contains only failure-correct events, i.e., $\Sigma_{ri} = \Sigma_{fci}; \Sigma_{rai} = \phi$. (Here Σ_{fci} and Σ_{rai} are the failure-correct and recovery-action event sets in mode *i*)

Case 2: The return-to-normal event set Σ_{ri} contains only recovery-action events, i.e., $\Sigma_{ri} = \Sigma_{rai}; \Sigma_{fci} \neq \phi$.

Cases in which Σ_{ri} includes both failure-correct and recovery-action events can be dealt with using a modified version of case 2.

The need to enforce the nonblocking conditions in every recovery mode depends on whether the mode belongs to case 1 or 2. In case 1, since the return-to-normal events set only contains the failure-correct events and by definition these events are uncontrollable, the nonblocking requirement has to be enforced in every recovery mode of the plant. On the other hand, in case 2 problems the return-to-normal events set contains a few recovery-action events, which are controllable. In this case, the nonblocking requirement does not have to be enforced as the recovery-action events can be controlled and be made to occur to bring the plant from the recovery mode to normal mode.

Let us assume for now that the plant has a single failure mode with one failure event (p = 1). The extension of results to arbitrary p will be discussed later. Fig 3.5 below depicts both modes of the plant:

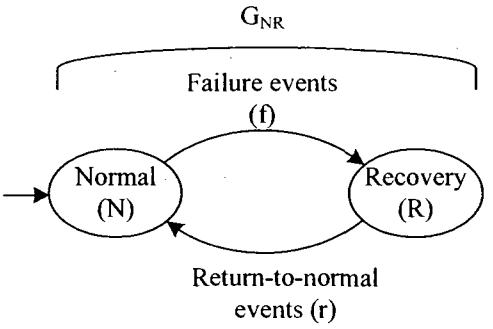

Fig. 3.5: Plant G assuming one failure mode (p = 1)

The plant initially starts in normal mode. In this mode, all of the components exhibit normal behaviour of operation. When the failure event (f) occurs, the plant enters into failure (recovery) mode (state set Q_R). The algorithm that is proposed in Chapter 4 is capable of computing a supervisor for a given single-failure mode. However, a given plant may have multiple failure modes. The algorithm proposed in Chapter 4 can be extended to tackle these more complex problems.

As discussed, the plant continuously moves between normal modes and recovery modes as failure events and return-to-normal events occur. In order to state the control problem, let us consider the unraveled, expanded state transition graph of the plant G, G_{ex} shown in Fig. 3.6 (which is similar to a reachability tree).

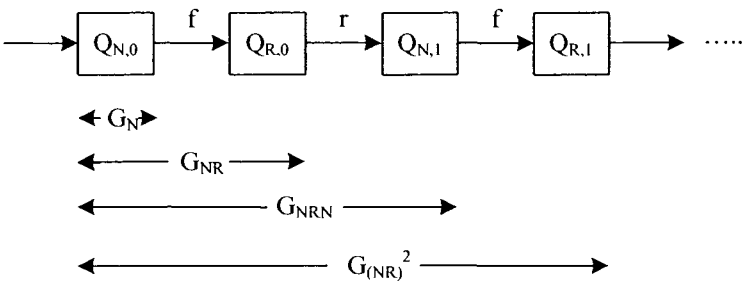

Fig. 3.6: Expanded transition graph G_{ex}

In this figure, $Q_{N,i}$ and $Q_{R,i}$ (i = 0,1,...) denote the set of normal and recovery (faulty) states reached after the i^{th} return-to-normal event. G_N and G_{NR} are the subautomata containing states $Q_{N,0}$ and $Q_{N,0} \cup Q_{R,0}$. $G_{(NR)^i N}$ and $G_{(NR)^{i+1}}$ are the subautomata containing states $\bigcup_{j=1}^{i}(Q_{N,j} \cup Q_{R,j}) \cup Q_{N,i}$ and $\bigcup_{j=1}^{i+1}(Q_{N,j} \cup Q_{R,j})$ (i = 0,1,2,..).

To construct the automaton G_{ex} in Fig.3.6, formally, we can proceed as follows. First introduce a counter for return-to-normal events (and failure events) as shown in Fig. 3.7. Let us call the counter C. Then G_{ex} can be formed as the product of G and C:

$$G_{ex} = \text{meet}\,(G,C)$$

$$\text{with } Q_{N,i} = Q_N \text{ x } \{i\} \text{ and } Q_{R,i} = Q_R \text{ x } \{i'\}$$

Similarly, subautomata of G_{ex}, namely, $G_{(NR)^i N}$ and $G_{(NR)^{i+1}}$ (i = 0,1,...) can be formed using the product of G and subautomata of C containing states $\{0,0',....i\}$ and $\{0,0',...i,i'\}$.

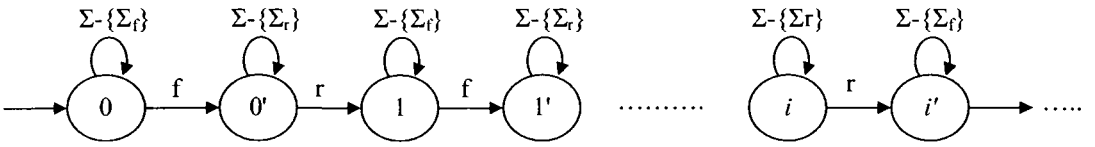

Fig. 3.7: Expanded states showing the events counter

With the notation given, we present the mathematical definitions for the two cases 1 and 2 discussed earlier. Let $E_N \subseteq Q_N$ and $E_R \subseteq Q_R$ be the set of legal states in normal and recovery (faulty) modes (design specifications). The first problem, wherein the return-to-normal events are uncontrollable (only failure-correct events Σ_{fc}) can, mathematically, be defined as follows:

"Design a supervisor S such that

$$\left.\begin{array}{ll} a) & R(S/G_N) \subseteq E_{N,0} \\ b) & R(S/G_N) \subseteq CR(S/G_N) \end{array}\right\} \text{ normal mode;}$$

after a failure event:

$$\left.\begin{array}{ll} a) & R(S/G_{NR}) \cap Q_{R,0} \subseteq E_{R,0} \\ b) & R(S/G_{NR}) \subseteq CR(S/G_{NR}) \end{array}\right\} \text{ recovery mode;}$$

after a return-to-normal event:

$$\left.\begin{array}{ll} a) & R(S/G_{NRN}) \cap Q_{N,1} \subseteq E_{N,1} \\ b) & R(S/G_{NRN}) \subseteq CR(S/G_{NRN}) \end{array}\right\} \text{ normal mode;}$$

after another failure:

$$\left.\begin{array}{ll} a) & R(S/G_{NRNR}) \cap Q_{R,1} \subseteq E_{R,1} \\ b) & R(S/G_{NRNR}) \subseteq CR(S/G_{NRNR}) \end{array}\right\} \text{ recovery mode;}$$

and so on. More compactly, for $i \geq 0$, we want the following:

$$\left.\begin{array}{l} a)\ R(S/G_{(NR)^i N}) \cap Q_{N,i} \subseteq E_{N,i} \\ b)\ R(S/G_{(NR)^i N}) \subseteq CR(S/G_{(NR)^i N}) \end{array}\right\} \text{normal mode;}$$

and

$$\left.\begin{array}{l} a')\ R(S/G_{(NR)^{i+1}}) \cap Q_{R,i} \subseteq E_{R,i} \\ b')\ R(S/G_{(NR)^{i+1}}) \subseteq CR(S/G_{(NR)^{i+1}}) \end{array}\right\} \text{recovery mode.}$$

In the above description, $R(S/G_N)$ is the reachable part of the plant under supervision in normal mode; $CR(S/G_N)$ is the co-reachable part of the plant under supervision. $E_{N,i} = E_N x\{i\}$ represents the normal mode legal states after the i^{th} return-to-normal event (after one failure event). Similarly, $E_{R,i} = E_R x\{i\}$ represents the recovery mode legal states after i^{th} return-to-normal event and $(i+1)^{th}$ failure event. The mathematical description of the problem presents us with two sets of conditions. The first set, set 'a' conditions, restrict the reachable part of the plant under supervision in normal and recovery modes to the respective state specifications, thus ensuring safety in all modes. The second set, set 'b' conditions, enforce nonblocking by ensuring that the reachable part of the plant under supervision is also co-reachable, so that the plant under supervision can reach a marked state from any state in all modes.

The above description represents the problem when the return-to-normal events are uncontrollable ($\Sigma_r = \Sigma_{fc}$). When some of the return-to-normal events are controllable, i.e., $\Sigma_{ra} = \Sigma_r; \Sigma_{fc} \neq \phi$, we do not enforce the nonblocking requirements in recovery modes since we can enable the return-to-normal events and expect the plant to be able to move

to normal mode. So the problem deals with the enforcing of both safety and nonblocking requirements in normal modes and only safety requirements in recovery modes. The following is the mathematical description:

"Design a supervisor S such that

$$\left.\begin{array}{l} a)\ R(S/G_N) \subseteq E_{N,0} \\ b)\ R(S/G_N) \subseteq CR(S/G_N) \end{array}\right\} \text{ normal mode;}$$

after a failure event:

$$\left.\begin{array}{l} a)\ R(S/G_{NR}) \cap Q_{R,0} \subseteq E_{R,0} \end{array}\right\} \text{ recovery mode;}$$

after a return-to-normal event:

$$\left.\begin{array}{l} a)\ R(S/G_{NRN}) \cap Q_{N,1} \subseteq E_{N,1} \\ b)\ R(S/G_{NRN}) \subseteq CR(S/G_{NRN}) \end{array}\right\} \text{ normal mode;}$$

after another failure:

$$\left.\begin{array}{l} a)\ R(S/G_{NRNR}) \cap Q_{R,1} \subseteq E_{R,1} \end{array}\right\} \text{ recovery mode;}$$

and so on. Alternatively, for $i \geq 0$,

$$\left.\begin{array}{l} a)\ R(S/G_{(NR)^i N}) \cap Q_{N,i} \subseteq E_{N,i} \\ b)\ R(S/G_{(NR)^i N}) \subseteq CR(S/G_{(NR)^i N}) \end{array}\right\} \text{ normal mode;}$$

and

$$\left.\begin{array}{l} a')\ R(S/G_{(NR)^{i+1}}) \cap Q_{R,i} \subseteq E_{R,i} \end{array}\right\} \text{ recovery mode.}$$

We can see the problems in both cases 1 and 2 are instance of the robust control problem for an *infinite* number of plant models $G_N, G_{NR}, G_{NRN}, \ldots\ldots$ In the following Chapter, we offer solutions for the problems.

3.4 Example

Let us consider the following example to illustrate the details of the problem. Fig. 3.8, below, shows the DES model of a plant (G) that contains both failure events and return-to-normal events. Here both failure events (f) and return-to-normal events (r) are uncontrollable and assumed to be observable.

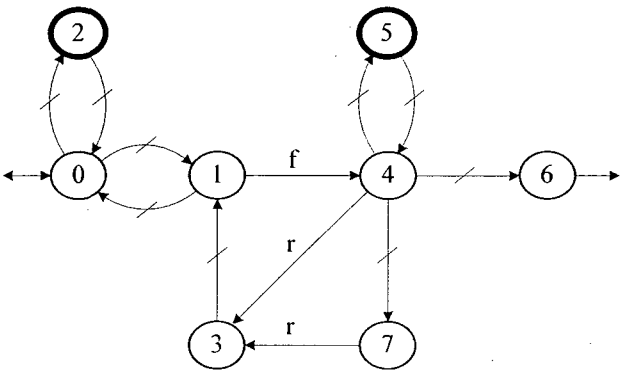

Fig. 3.8: Plant (G)

The plant G is prone to failure at state 1. The return-to-normal event is defined both at state 4 and 7. The following state sets show us both the safe state specifications and plant states for both the normal and recovery modes: The unsafe states are shown in bold.

$$E_N = \{0,1,3\}, \qquad Q_N = \{0.1.2.3\};$$

$$E_R = \{4,6,7\}, \qquad Q_R = \{4,5,6,7\};$$

The above problem is solved based on standard (non-robust) supervisory problems and the plant under supervision (S/G) is shown below in Fig. 3.9.

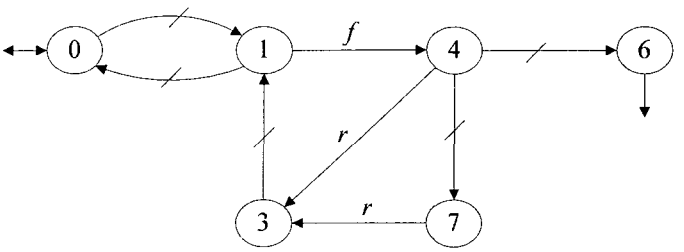

Fig. 3.9: Plant under supervision of a standard supervisor (S_{Std}/G)

It appears that the plant under supervision is nonblocking. However, note that at the recovery (faulty) state 7, to reach a marked state (0 or 6) the occurrence of failure-correct event r is necessary. The return-to-normal event is uncontrollable and uncontrollable events are not be guaranteed to occur. Thus state 7 should be considered as deadlock. So the solution based on standard supervisory methods does not provide nonblocking guarantee. That is because, in computing this solution, the standard methods did not consider the actual dynamics of the plant with respect to the uncontrollable failure-correct events. Fig. 3.10 below presents the plant under supervision as per the robust control solution computed based on the algorithm presented in Chapter 4.

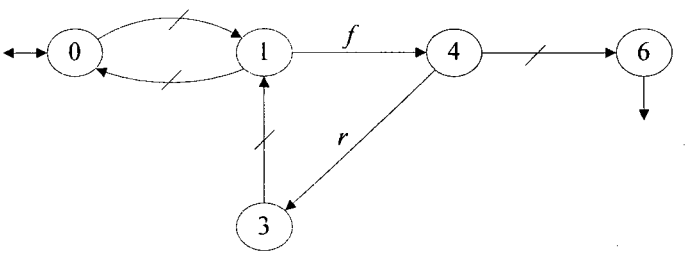

Fig. 3.10: Plant under supervision of a robust supervisor (S_{Robust}/G)

It can be seen that our solution actually provides a robust supervisor which guarantees nonblocking both in recovery and normal modes of the plant operation. This plant observes no blocking after return to normal mode of operation under supervision by

disabling the controllable transition from state 4 to state 7. The complete solution to this example is presented in Chapter 4.

3.5 Conclusion

In this chapter, the problem of fault recovery when return to normal operation is possible, is formulated. We presented two cases of the problems considered. In Chapter 4, we present the solution along with the design algorithm.

Chapter 4

Solution to Fault Recovery Problem

In this Chapter, we will present a solution to the fault recovery problem and then develop a computational procedure. The problem formulation is reviewed and some useful definitions are introduced in Section 4.1. In Section 4.2, the solution for finding an optimal (maximally permissive) solution is presented. Section 4.3 develops a computational algorithm and provides an example.

4.1 Introduction

In Chapter 3, a supervisory control problem with return-to-normal events was presented. Depending on the controllability of return-to-normal events, we identified two cases:

1) Return-to-normal events are failure-correct events and thus uncontrollable: $\Sigma_r = \Sigma_{fc} \subseteq \Sigma_{uc}$. This is the case we encounter in dealing with intermittent failures, and

2) return-to-normal events are recovery action events and assumed controllable:

$\Sigma_r = \Sigma_{ra} \subseteq \Sigma_c$.

We saw that the above problems are instances of robust control problem. In this Chapter, we present a solution for the first problem. The solution for the second problem is similar and omitted for brevity. Therefore, from now on, we will assume $\Sigma_r = \Sigma_{fc} \subseteq \Sigma_{uc}$. In Chapter 3, the statement of the problem was given in terms of subautomata of the expanded plant G_{ex}. If the total number of fault and return-to-normal returns cycles is finite, then G_{ex} will have a finite state set and the problem posed in Chapter 3 will be a robust control problem involving a finite number of automata. The solution to this problem (at least in a linguistic setup) is available [13]. Therefore, in this chapter, we will assume that an unbounded number of fault and return-to-normal returns cycles is possible in the plant (in the case of intermittent failures that is typically the case). As a result, we have to solve a robust control problem with an infinite number of plant models.

The problem is to find a supervisor S such that for $i = 0,1,2,\ldots$.

$a)\ R(S/G_{(NR)^i N}) \cap Q_{N,i} \subseteq E_{N,i}$

$b)\ R(S/G_{(NR)^i N}) \subseteq CR(S/G_{(NR)^i N})$

$a')\ R(S/G_{(NR)^{i+1}}) \cap Q_{R,i} \subseteq E_{R,i}$

$b')\ R(S/G_{(NR)^{i+1}}) \subseteq CR(S/G_{(NR)^{i+1}})$

We refer to the following problem as the Infinite Horizon (IH) problem. We will show that the optimal (maximally permissive) solution to the IH problem can be obtained as the limit of the sequence formed from solutions to the following Finite Horizon problems.

Finite-Horizon-l (FHl) problem: Find a supervisor S such that for $0 \le i \le l$

$$a)\ R(S/G_{(NR)^i N}) \cap Q_{N,i} \subseteq E_{N,i}$$
$$b)\ R(S/G_{(NR)^i N}) \subseteq CR(S/G_{(NR)^i N})$$

and for $0 \le i \le l-1$

$$a')\ R(S/G_{(NR)^{i+1}}) \cap Q_{R,i} \subseteq E_{R,i}$$
$$b')\ R(S/G_{(NR)^{i+1}}) \subseteq CR(S/G_{(NR)^{i+1}})$$

Finite-Horizon-l' (FHl') problem: Find a supervisor S such that for $0 \le i \le l$

$$a)\ R(S/G_{(NR)^i N}) \cap Q_{N,i} \subseteq E_{N,i}$$
$$b)\ R(S/G_{(NR)^i N}) \subseteq CR(S/G_{(NR)^i N})$$

$$a')\ R(S/G_{(NR)^{i+1}}) \cap Q_{R,i} \subseteq E_{R,i}$$
$$b')\ R(S/G_{(NR)^{i+1}}) \subseteq CR(S/G_{(NR)^{i+1}})$$

FHl considers l cylces of fault and return-to-normal events whereas FHl' considers l such cycles and the $(l+1)^{\text{th}}$ failure. From now on, for convenience we use G_l and $G_{l'}$ to denote $G_{(NR)^l N}$ and $G_{(NR)^{l+1}}$.

4.2 Maximally Permissive Solution

In this section, we show that the infinite horizon problem has a maximally permissive solution in the form of state feedback. Furthermore, this solution can be obtained as the limit of maximally permissive state feedback solutions for the finite horizon problem (as the length of horizon increases). We will show that the aforementioned sequence terminates in a bounded number of steps.

We begin by exploring some of the properties of the solutions of IH problem.

Proposition 4.1. A supervisor S solves IH problem if and only if S solves FHl and FHl' problems for $l = 0,1,2,\ldots..$

Proof follows immediately from the statement of IH, FHl and FHl' problems. The following lemmas state that legal sublanguages that posses the three conditions of controllability, $L_m(G)$ -closure and G -nonblocking characterize the solutions of IH problem (this is similar to the case of finite number of plants in [12]).

Lemma 4.2. Let S be a solution of IH problem. Then $K = L_m(S/G)$ will be

1. controllable with respect to G
2. $L_m(G)$ - closed
3. G_l-nonblocking and $G_{l'}$ nonblocking for all $i = 0,1,2,\ldots.$

Proof: Define $K_l = L_m(S/G_l)$ and $K_{l'} = L_m(S/G_{l'})$ for $l \geq 0$. From lemma [1], $K_l = K \bigcap L_m(G_l)$ and $K_{l'} = K \bigcap L_m(S/G_{l'})$. Also note that $L_m(G_{ex}) = L_m(G)$ and $L(G_{ex}) = L(G)$.

By Prop.4.1, S is a solution of FHl and FHl' ($l \geq 0$). Therefore, K_l (resp. $K_{l'}$) is controllable with respect to G_l (resp. $G_{l'}$), $L_m(G_l)$-closed (resp. $L_m(G_{l'})$-closed) and G_l-nonblocking ($G_{l'}$-nonblocking). Using the above, we can show $K = \bigcup_{l=0}^{\infty}(K_l \cup K_{l'})$ is controllable with respect to G, $L_m(G)$-closed and G_l-nonblocking and $G_{l'}$-nonblocking

($l \geq 0$). The proof is identical to that provided in part (2) of the proof of Thm. 9 of [13]. The proof given in [13] is for a finite number of plant models but all of derivations hold for a countably infinite number of plants. □

Lemma 4.3. Let $K \subseteq L_m(G)$ and $K \neq \phi$ be a sublanguage of $L_m(G)$ containing legal strings for IH problem. If K has the following properties

1. K is controllable with respect to G
2. K is $L_m(G)$-closed
3. K is $G_{(NR)^l N}$ -nonblocking and $G_{(NR)^{l+1}}$ -nonblocking ($i = 0, 1, ..$)

Then there exists a supervisor S that solves IH problem with $K \subseteq L_m(S/G)$.

Proof: It follows from (1) and (2) that there exists a supervisor S such that $K = L_m(S/G)$ and $\overline{L_m}(S/G) = L(S/G)$. We show S solves FHl ($l \geq 0$).

First note that

$$L_m(S/G_l) = L_m(S/G) \cap L_m(G_l)$$

$$= K \cap L_m(G_l)$$

Thus $L_m(S/G_l)$ contains legal strings of $L_m(G_l)$. Regarding nonblocking property

$$\overline{L_m(S/G_l)} = \overline{L_m(S/G) \cap L_m(G_l)} \qquad \text{(By Lemma [1])}$$

$$= \overline{K \cap L_m(G_l)} \qquad (K \text{ is } G_l\text{- nonblocking})$$

$$= \overline{K} \cap L(G_l)$$

$$= L(S/G) \cap L(G_l)$$

$$= L(S/G_l)$$

Therefore S solves FHl. Similarly, S solves FHl', and then by Prop. 4.1, S solves IH problem. □

Since controllability, $L_m(G)$ -closure and G -nonblocking properties are closed under union operation ([19],[13]), then assuming IH is solvable, it has a maximally permissive solution.

Proposition 4.4. If the infinite horizon (IH) problem has a solution, then it has a maximally permissive solution.

Proof. Follows from lemmas 4.1 and 4.2 and that controllability, $L_m(G)$-closure and G-nonblocking properties are closed under union operation. □

Consider FHl problem. This is a control problem formalized in a state-based framework. The solution of this problem [19] is a state feedback $S: Q_l \to \Gamma$. Each state Q_l is a pair (q,i) with $q \in Q$ is the state of plant, i the number of failure and return-to-normal cycles $(0 \le i \le l)$. Similarly, the solutions to FHl' are of the form $S: Q_{l'} \to \Gamma$ with $Q_{l'}$ consisting of pairs (q,i') with $q \in Q$ and $0 \le i' \le l$. The next result shows that even though the feedback law for FHl and FHl' depend on q and i, the feedback law for the maximally permissive solution for the infinite horizon (IH) problem depends only on q (not i). This result resembles the control law in Linear Quadratic Regulator (LQR) problem in which the solution for finite time horizon depends on the state and time but

the solution for infinite horizon problem depends on state only. In the following, the state feedbacks with domain Q (the state set of G) designated by V to differentiate them from S whose domain is subsets of Q_{ex} (the state set of G_{ex}).

Proposition 4.5. If IH problem has a solution, then the maximally permissive solution will be a state feedback law $V_\infty^* : Q \to \Gamma = \{\Sigma' | \Sigma' \supseteq \Sigma_{uc}\}$. In other words, if $S_\infty^* : Q_{ex} \to \Gamma$ denotes the maximally permissive solution, then $S_\infty^*((q,i)) = V_\infty^*(q)$.

Proof: We have to show that for two states (q, i) and (q, j) the control patterns for the maximally permissive solution, that is $S_\infty^*((q,i))$ and $S_\infty^*((q,j))$ are the same. Since the design specifications for IH are in state-based form, at any given state (q, i), the control pattern only depends on the subautomaton of G_{ex} reachable from the state (q, i). Now note that the subautomata reachable from (q, i) and (q, j) are isomorphic. Furthermore, the states of the above mentioned can be related by the following isomorphism which preserves the legality and the original label of the state from $\eta((q',k)) = (q', k+j-i)$. Therefore, $S_\infty^*((q,i)) = S_\infty^*((q,j))$. □

Loosely speaking, the control action in IH problem depends on the possible future behaviors of the plant and not on how many times in the past the fault and return-to-normal cycle has occurred.

Now that it is established that the desired maximally permissive supervisor is in the form of a state feedback law $V^* : Q \to \Gamma$, let us study the subsets of the solutions of FHl and FHl' that have the same form. So let

$$\mathcal{V}_l = \{V : Q \to \Gamma \mid V \text{ solves FHl}\};$$

$$\mathcal{V}_{l'} = \{V : Q \to \Gamma \mid V \text{ solves FHl'}\};$$

$$\mathcal{V}_\infty = \{V : Q \to \Gamma \mid V \text{ solves IH}\}.$$

The next proposition shows that $\mathcal{V}_l$ and $\mathcal{V}_{l'}$ (if nonempty) posses maximally permissive elements. Note that if IH is solvable, then V_∞^* exists (by Prop. 4.4 and 4.5) and therefore $\mathcal{V}_\infty \neq \phi$ and $\mathcal{V}_l$ and $\mathcal{V}_{l'}$ are nonempty (by Prop. 4.1).

Another important issue is that the total number of state feedback controllers $V : Q \to \Gamma$ is less than or equal to $|Q| \text{x} 2^{|\Sigma_c|}$ since at any state q, there are at most $2^{|\Sigma_c|}$ possibilities for the enablement of controllable events. Therefore $|\mathcal{V}_l|$, $|\mathcal{V}_{l'}|$, $|\mathcal{V}_\infty| \leq |Q| . 2^{|\Sigma_c|}$.

Proposition 4.6. If $\mathcal{V}_l$ (resp. $\mathcal{V}_{l'}$) is nonempty, then it has a maximally permissive element $\mathcal{V}_l^*$ (resp. $\mathcal{V}_{l'}^*$).

Proof: Consider two supervisors $V_l^1, V_l^2 \in \mathcal{V}_l$. Without any loss of generality we assume that

$$V_l^1(q) = \Sigma_{uc} \text{ for } q \notin R(V_l^1 / G_l)$$

$$V_l^2(q) = \Sigma_{uc} \text{ for } q \notin R(V_l^2 / G_l)$$

This means that the supervisor V_l^1(resp. V_l^2) disables all events at states that can not be reached under its supervision. We can assume this without loss of generality since it does not affect the closed-loop systems V_l^1 / G_l and V_l^1 / G_l.

Now define the merger of V_l^1 and V_l^2 according to:

$$V_l = \text{merge}\,(V_l^1, V_l^2): Q \to \Gamma$$

$$V_l(q) = V_l^1(q) \bigcup V_l^2(q)$$

It follows from the above definition that $R(V_l / G_l) = R(V_l^1 / G_l) \bigcup R(V_l^2 / G_l)$ and more generally, $R(V_l / G_i) = R(V_l^1 / G_i) \bigcup R(V_l^2 / G_i)$ for $0 \le i \le l$, and

$R(V_l / G_{i'}) = R(V_l^1 / G_{i'}) \bigcup R(V_l^2 / G_{i'})$ for $0 \le i \le l-1$. Since V_l^1 and V_l^2 solve FHl, then $R(V_l^1 / G_i)$, $R(V_l^2 / G_i)$ and $R(V_l^1 / G_{i'})$, $R(V_l^2 / G_{i'})$ are controllable, nonblocking predicates [19]. Therefore, $R(V_l / G_i)(0 \le i \le l)$ and $R(V_l / G_{i'})(0 \le i \le l-1)$ are controllable and nonblocking. Therefore $V_l \in \mathcal{V}_l$. Now let V_l^* be the merger of all elements of $\mathcal{V}_l$. It immediately follows that V_l^* will be a maximally permissive state feedback of the form $V: Q \to \Gamma$. The proposition can be similarly proved for $V_{l'}^*$ □

The next result shows how IH problem can be obtained by solving a finite horizon problem.

Theorem. 4.1: For $l \geq |Q|-1$, $V_l^* = V_\infty^*$ and $V_{l'}^* = V_\infty^*$.

Proof: Since $\mathcal{V}_\infty \subseteq \mathcal{V}_l$, we only need to show $V_l^* \in \mathcal{V}_\infty$ (i.e., V_l^* solves IH problem). Since $R(V_l^* / G) \subseteq Q$, V_l^* / G will have at most $|Q|$ states. Therefore , every state in V_l^* / G can be reached by a sequence of a length less than $|Q|$. Therefore, with in l-1 fault and return-to-normal cycles all of the reachable states of the plant $R(V_l^* / G)$ can be visited once. In other words, for every $q \in R(V_l^* / G)$, there exists $0 \leq j \leq l$ such that $(q, j) \in R(V_l^* / G_l)$. Since V_l^* solves FHl, then all states $(q, j) \in R(V_l^* / G_l)$ and therefore all $q \in R(V_l^* / G)$ or equivalently all $(q, i) \in R(V_l^* / G_{ex})$ must be legal (and satisfie the (a) specifications in IH problem). Since V_l^* solves FHl, $R(V_l^* / G_i)$ is a nonblocking predicte for $0 \leq i \leq l$, and so is $R(V_l^* / G_{i'})$ for $0 \leq i \leq l-1$.

Now consider $R(V_l^* / G_i)$ with $i > l$, and $(q,k) \in R(V_l^* / G_i)$. If $(q,k) \in R(V_l^* / G_l)$, then as mentioned above it must be coreachable using a sequence through legal states. If $(q,k) \notin R(V_l^* / G_l)$, then $k \geq l+1$. But there exist (q, j), $0 \leq j \leq l$ such that $(q, j) \in R(V_l^* / G_l)$. Since V_l^* solves FHl, then there exists a sequence in G_l starting from (q, j), consisting of legal states, and leading to a marked state: $(q, j) \rightarrow (q_1, j_1) \rightarrow (q_2, j_2) \rightarrow .....(q_n, j_n)$ with $q_n \in Q_m$. Since V_l^* is a state feedback law based on q only, therefore the following sequence of legal states also exists in $V_l^* / G_i : (q,k) \rightarrow (q_1, j_1 + k - j) \rightarrow (q_2, j_2 + k - j) \rightarrow .... \rightarrow (q_n, j_n + k - j)$. Thus

$R(V_l^* / G_l)$ is nonblocking. Similarly, $R(V_l^* / G_{l'})$ is nonblocking. Therefore V_l^* solves IH problem. Similarly, we can show $V_{l'}^*$ solves IH problem. □

Based on the above results, instead of finding a maximally permissive solution to IH problem, we can find maximally permissive state feedback solutions FHl or FHl' for $l \geq |Q| - 1$. The set of state feedback solutions for FHl (resp. FHl'), $\mathcal{V}_l$ (resp. $\mathcal{V}_{l'}$) are finite sets and can be searched recursively for desired maximally permissive answer. Note that G_l contains (in the worst case) $(l+1)|Q|$ states. So for $l = |Q| - 1$, G_l will have in the worst case $|Q|^2$ states. Instead of solving FHl (for $l \geq |Q| - 1$), we can obtain maximally permissive sequence of solutions (for FH0, FH0', FH1, FH1',....): $V_0^*, V_{0'}^*, V_1^*, V_{1'}^*, \ldots\ldots$ We know for $l \geq |Q| - 1$, $V_l^* = V_{l'}^* = V_\infty^*$. It is possible that the sequence terminates faster. So if for $0 \leq k \leq |Q| - 1$, $V_k^* = V_{k'}^*$, (or $V_k^* = V_{(k-1)'}^*$), then the sequence has possibly terminated. To check this, we obtain $R(V_k^* / G)$ and if for every $q \in R(V_k^* / G)$, there exists $(q, i) \in R(V_k^* / G_k)$, then, similar to the discussion in the proof of Thm. 4.1., $V_k^* = V_\infty^*$ and the sequence has terminated. In practice, k would be as small as 1. In the next section, we will discuss a simple example.

Remark: In our discussion, in this section, we assumed the plant has one failure mode (p = 1). The discussion and solution can be extended to the case of plants with more than one failure mode ($p > 1$). Assuming, single-failure scenario (i.e., one failure at a time), as shown in Fig. 3.4., the problem and solution will be similar except that G_l as introduced

in Sec. 4.1, correspond to the plant model with l cycles of fault and return-to-normal events, covering sequences of l faults from the set fault events $\Sigma_f = \{f_1, ... f_p\}$. $G_{l'}$ is defined similarly. To solve the IH problem, we can construct the sequence of solutions to FH0, FH0', ...: $V_0^*, V_{0'}^*, V_1^*, V_{1'}^*, ...$

4.3 Example

In the previous section, we arrived at an algorithm for obtaining a maximally permissive solution for IH problem. The algorithm consists of constructing the sequence of maximally permissive state-feedback solutions for FHl and FHl': $V_0^*, V_{0'}^*, V_1^*, V_{1'}^*, ...$ If for some k, $V_k^* = V_{k'}^*$, (or $V_k^* = V_{(k-1)'}^*$) and $R(V_k^* / G) = \{q | (q,i) \in R(V_k^* / G_k)$ for some $0 \le i \le k$ or $(q,i') \in R(V_k^* / G_k)$ for some $0 \le i \le k-1\}$ then $V_k^* = V_\infty^*$. The convergence is guaranteed for $k = |Q| - 1$.

In the following, we consider a simple example. In Chap. 5 a more detailed example will be discussed.

Consider the plant G given in Fig.4.x1.

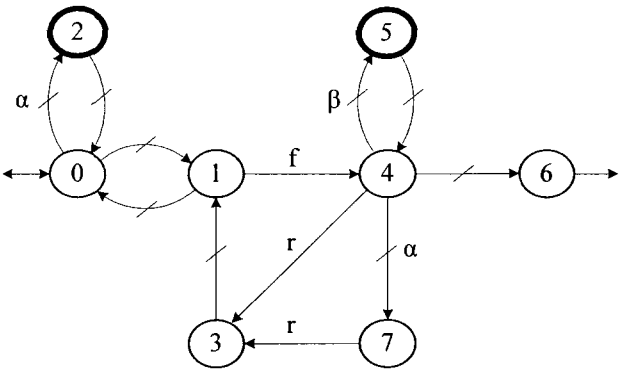

Fig. 4.x1: Plant G

Note that only few transitions are labeled with the corresponding events that matter most to the discussion. The unsafe states are bolded.

In G_I the normal states are $Q_N = \{0, 1, 2, 3\}$, and the faulty (recovery) states $Q_R = \{4, 5, 6, 7\}$. Suppose states 2 and 5 are the unsafe (illegal) states. Therefore,

$$E_N = \{0,1,3\}\,;\;\; E_R = \{4,6,7\}\,.$$

Also note the $Q_R = \{0, 6\}$ are the marked states.

We would like to solve the IH problem. First we solve FH0 for $G_N = G_0$ shown in Fig. 4.x2, with $E_{N,0} = \{(0,0),(1,0)\}$.

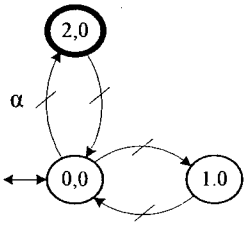

Fig. 4.x2: $G_N = G_0$

By inspection, we see that disabling controllable event α at state (0,0) will solve the problem. Therefore, the following state feedback law will be a maximally permissive solution:

$$V_0^*(q) = \begin{cases} \Sigma - \{\alpha\} & q = 0 \\ \Sigma & \textit{otherwise} \end{cases}$$

Next we solve FH0' for $G_{NR} = G_{0'}$ (Fig. 4.x3), with $E_{R,0} = \{(4,0'),(6,0'),(7,0')\}$.

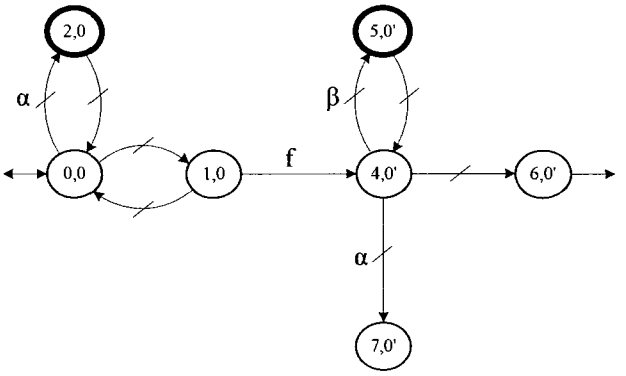

Fig. 4.x3: $G_{NR} = G_{0'}$

To avoid entering illegal states (2,0) and (5,0'), events α at (0,0) and β at (4,0') must be disabled. Also, by inspection the deadlock state (7,0') should be avoided and thus α at (4,0') has to be disabled. Hence a maximally permissive state feedback supervisor is:

$$V_{0'}^*(q) = \begin{cases} \Sigma - \{\alpha\} & q = 0 \\ \Sigma - \{\alpha, \beta\} & q = 4 \\ \Sigma & \textit{otherwise} \end{cases}$$

Note that $V_{0'}^*$ is more restrictive than V_0^*.

Next we solve FH1 for $G_{(NR)N} = G_1$ (Fig. 4.x4)

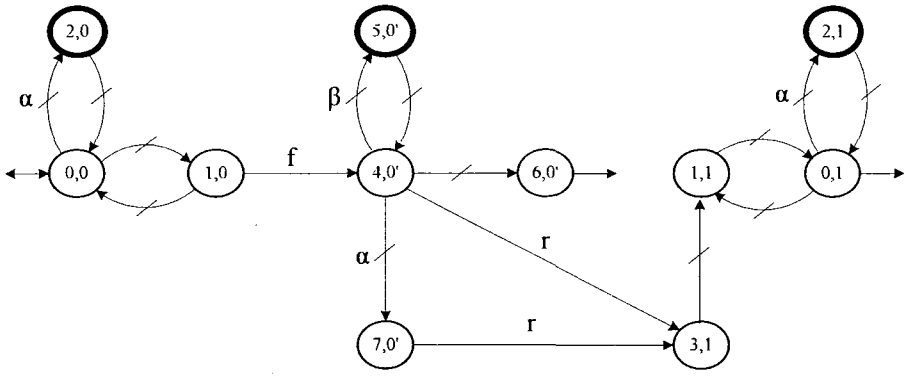

Fig. 4.x4: $G_{(NR)N} = G_1$

To avoid entering illegal states (2,0), (5,0') and (2,1), α has to be disabled at $q = 0$ and β should be disabled at $q = 4$. Furthermore to prevent entering (7,0') which is a deadlock state for $G_{NR} = G_{0'}$, α should be disabled at 4. Thus a maximally permissive solution will be

$$V_1^*(q) = \begin{cases} \Sigma - \{\alpha\} & q = 0 \\ \Sigma - \{\alpha, \beta\} & q = 4 \\ \Sigma & \textit{otherwise} \end{cases}$$

We note that $V_{0'}^* = V_1^*$. In Fig. 4.x5 and 4.x6, the closed-loop systems V_1^* / G and V_1^* / G_1 are shown.

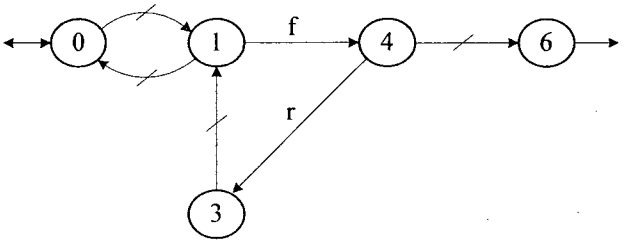

Fig. 4.x5: System under supervision V_1^* / G

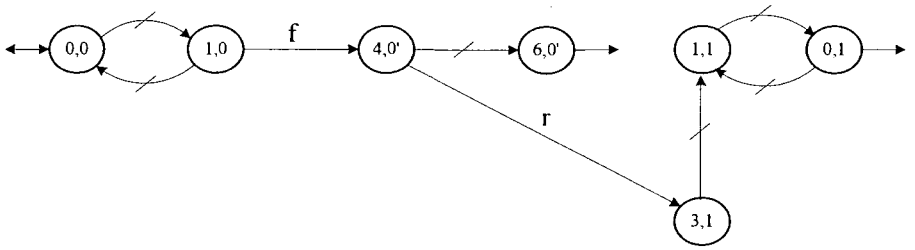

Fig. 4.x6: V_1^* / G_1

We note that $R(V_1^* / G) = \{0,1,3,4,6\}$

$$= \{ q | (q,i) \in R(V_1^* / G_1) \text{ for } i = 0, 1 \} \cup$$

$$\{ q | (q,0') \in R(V_1^* / G_1) \}.$$

As in all supervisory control problems, the automaton in Fig. 4.x5 can be used as an implementation of the supervisor as well.

4.4 Conclusion

In this chapter, an iterative solution for the problem considered in Chapter 3 is presented. The solution has been found to be converging (in a bounded number of steps) providing a robust supervisor. In Chapter 5, three application examples are provided.

Chapter 5

Application Examples

In this chapter, we consider three physical examples and solve them using the algorithm proposed in Chapter 4. We consider a Simplified Propulsion System (SPS) in Section 5.2. In Section 5.3, we solve an extended version of SPS: Extended Propulsion System (EPS). In Section 5.4, we solve Automatic Resource Allocator (ARA) problem.

5.1 Introduction

Discrete Event Systems theory lends itself as a better tool to solve the higher level supervisory control problems. One of the prominent areas where DES is used is spacecraft systems. [17] deals with few aspects of DES applications to robotic space explorers.

Three physical examples are considered in this Chapter: i) Simplified Propulsion System (SPS), ii) Extended Propulsion System (EPS), and iii) Automatic Resource Allocator (ARA). The plant dynamics for the propulsion system examples are taken from actual propulsion systems used in spacecraft [5, page: 129]. The dynamics for the component interaction is taken from their operational characteristics from real propulsion systems. [7] is referred for constructing some of the operational procedures considered for the examples. [13, page: 164] describes various steps that are involved in operating a spacecraft propulsion system. These steps are considered in designing the event sequences for our propulsion system examples. In both propulsion system examples, the return-to-normal events are uncontrollable events.

The third physical example considered is an automatic resource allocator, which can resemble any system with two users and two resources. In this example, the return-to-normal event is a controllable event. Also the solution can be extended to multiple users/ multiple resources systems.

5.2 Simplified Propulsion System (SPS)

In this example, we consider a spacecraft propulsion system in its simplified version. An extended version is considered in section 5.4. We call this simplified version: SPS. SPS represents a miniature version of a large spacecraft propulsion system such as the one used in Galileo spacecraft [3]. SPS is a mono-propellant system. Spacecraft propulsion systems produce the thrust required to accomplish tasks such as controlling the direction

of the spacecraft, inserting the spacecraft into orbit and for reactive control systems (RCS) of spacecraft [17].

5.2.1 Setup of SPS

The simplified propulsion system (SPS) contains the following components:

1) Pilot: To initiate start and stop commands to start and stop the whole operation.
2) Pyro-valve: A pyro-valve is a regular valve operated using pyro (temperature) techniques. Also a pyro-valve can only be operated once. The pyro-valve in SPS is initially open and is closed to stop the flow of the propellant when required.
3) Valve: A regular valve which can be opened or closed to control the flow of the propellant to the engine.
4) Thrust Engine: An engine that burns the propellant and produces thrust.

The following diagram shows the setup of the components for SPS:

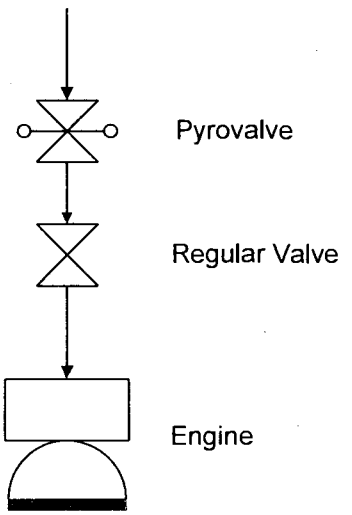

Fig. 5.1: Setup for SPS

Note that the diagram does not include the pilot. The normal operation of the plant is as follows: initially the pyro-valve is open and the regular valve is closed and the engine is off. Upon receiving a start command from the pilot, the regular valve is opened to allow the propellant to reach the engine. Subsequently, the engine is turned on by starting the pre-heating and ignition processes. We consider all the steps involved in turning the engine on as one event. In this state the plant generates thrust. Once the desired mission is accomplished, the pilot sends a stop command. Upon receiving that command, the regular valve is closed followed by turning off the engine. The pyro-valve is not operated in the normal mode of the operation.

The failure event (f) is defined as the regular valve getting stuck in open position thus creating a situation where the propellant is allowed to reach the engine continuously. When this event occurs, the plant enters into failure mode which is also recovery mode. In this mode of operation, the propulsion system keeps producing the thrust even after a stop command is received from the pilot. We define a return-to-normal (failure-correct) event (r) which can bring the failed valve back into normal state (open position) from where it can be closed when a stop command is received from the pilot. Both the failure and failure-correct events are considered to be uncontrollable as the supervisor has no control over them. However, if this failure-correct event does not happen, we provide an alternative way to stop the flow of the propellant when the stop command is received. We use the pyro-valve to close the flow path for the propellant and then subsequently turn the engine off. However, once the pyro-valve is closed, the plant is not reusable. In propulsion systems used in spacecraft, usually, multitudes of SPS are used to create

sufficient redundancy in the mission operation. The next section presents the DES models for individual components and then plant models are computed. A more complex problem is studied in Section 5.3.

5.2.2 DES Models

In our modeling of the components of the simplified spacecraft propulsion system, we consider the general dynamics of the components and their interaction taken from the operational specifications of real spacecraft propulsion systems [3],[7],[9]. The following remarks apply to all DES models in this chapter:

1) Uncontrollable events are even numbered;
2) Controllable events are odd numbered.

The following table describes all the events used in this example:

Event	Tag	Description
1	V_open	Valve opens
2	V_fail	Valve stuck open (failure event)
3	V_close	Valve closes (failure-correct)
4	V_return	Return-to-normal event (r)
5	PV_close	Pyro-valve closes
6	Start	Start command from the pilot
7	E_on	Turns the engine on
8	Stop	Stop command from the pilot
9	E_off	Turns the engine off

Table 5.1: Event list for SPS

The following diagram shows the model for the regular valve and includes the failure and return-to-normal (failure-correct) events.

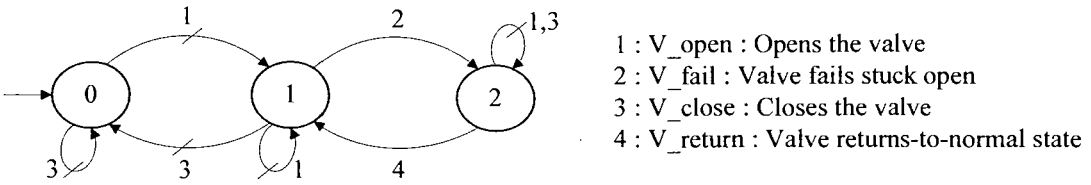

Fig. 5.2: Valve (SPS) (Full Mode)

The valve is prone to failure in its open position (at state 1) leading to failed state 2. At this state, the return-to-normal event (4) is defined. This event returns the failed valve to normal open position (state 1). When the failure event occurs in the valve, the plant enters into recovery mode and returns to normal mode when the return-to-normal event occurs. As stated before, the failure and return-to-normal (failure-correct) events are uncontrollable.

The following diagram shows the model for the pilot with uncontrollable events: 6 and 8. Due to its nature, the pilot is not controlled by the supervisor. Event 6 is a start command to start the whole operation while event 8 is a stop command to signal completion of the operation and initiate shutting down the propulsion system.

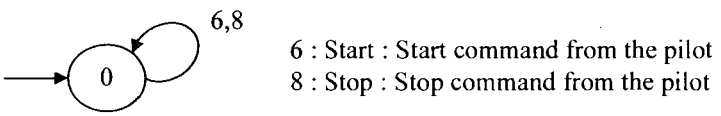

Fig. 5.3: Pilot (SPS)

The next model represents the pyro-valve with a single event: 5. The pyro-valve is initially open and event 5 closes the pyro-valve. This operation is irreversible as pyro-valves can only be operated once.

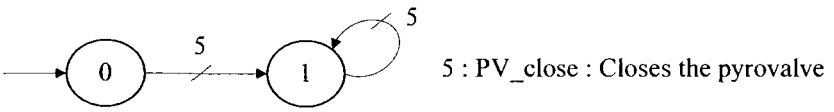

Fig. 5.4: Pyro-valve (SPS)

The model for the engine is given next. Events 7 and 9 turn the engine on and off respectively. This is a very simplified model of a real engine used in propulsion systems.

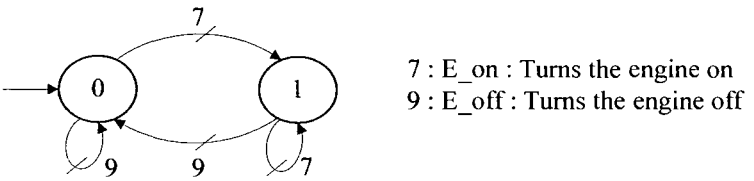

Fig. 5.5: Engine (SPS)

5.2.3 Plant Models

We use TTCT [15] to compute the plant model as shown in Fig 5.6 below. States 0 and 8 are marked corresponding to states when SPS is off. At state 0, engine is off, regular valve is closed and pyro-valve is open. At 2, engine is off and both regular and pyro-valves are closed, and at 8, engine is off, regular valve is stuck-open but the pyro-valve is closed

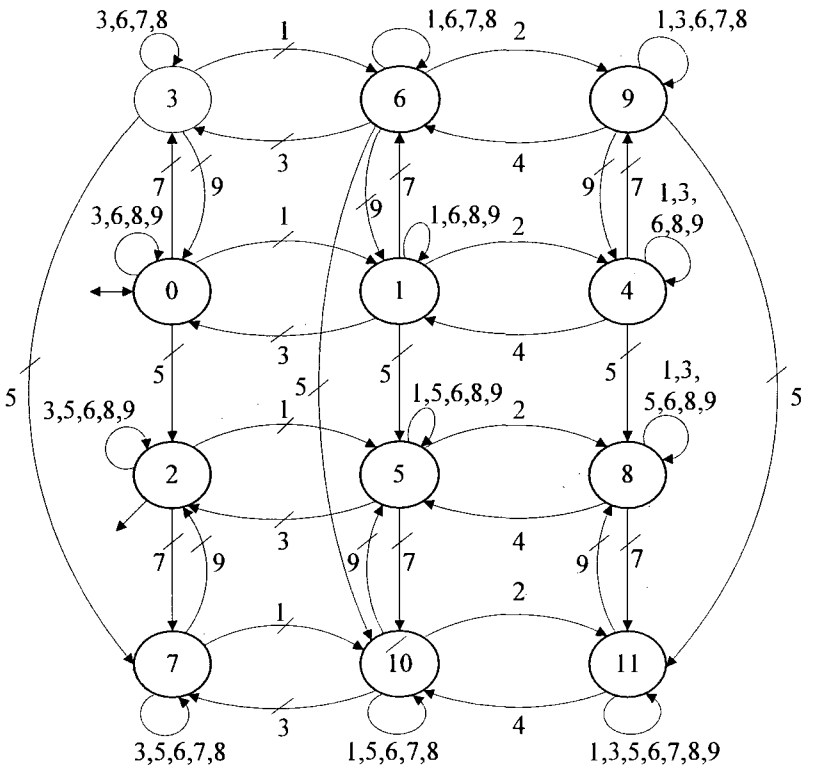

Fig. 5.6: Complete model of SPS (G^L)

This plant model G^L represents the full model of plant including both the normal mode as well as the recovery mode. The subscript letters denote the modes of operation (N: Normal, R: Recovery) and the superscript L denotes the linguistic nature of this model. It is possible to deduct the plant in normal mode only by removing the failure, return-to-normal events and the faulty (recovery) states. The model of plant in normal mode corresponds to the subautomaton of G^L containing states {0,1,2,3,5,6,7,10}. These linguistic based plant models are converted into state based models in Section 5.2.5.

5.2.4 State Specifications

First, we define the recovery mode specifications and subsequently deduce the normal mode specifications. We present the specifications in simple linguistic sequences. This simplifies the understanding of the restrictions on the plant behaviour. However, we present DES models for these specifications in state-based models as explained in Chapter 4. The following list outlines the normal mode specifications:

Ni) The operation starts when the pilot gives a start command, pyro-valve is open, and the engine is off.

Nii) The regular valve is opened before the engine is turned on.

Niii) The termination of the operation is initiated by stop command from the pilot and when the engine is on.

Niv) The valve is closed before the engine is turned off.

Nv) Pyro-valve should not be used in normal mode.

The following represent the specifications for recovery mode operation:

Ri) The termination of the operation is initiated by stop command from the pilot and when the engine is on.

Rii) The pyro-valve is closed before the engine is turned off.

The DES model SPEC for both normal and recovery mode specifications is shown in Fig. 5.7 below.

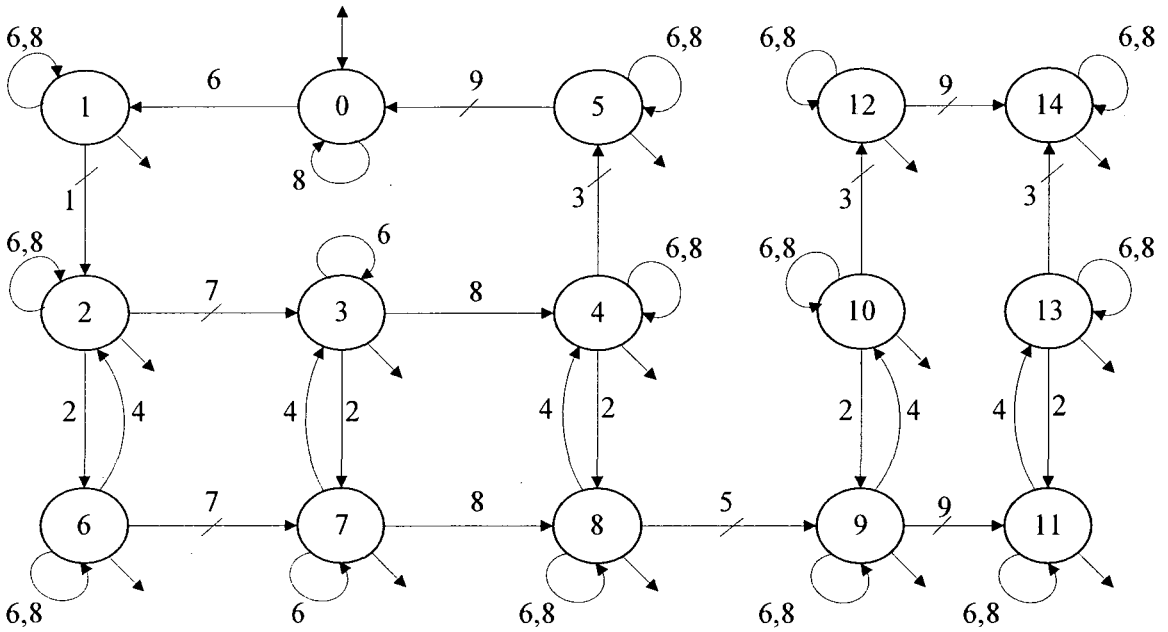

Fig. 5.7: SPEC: Specifications (SPS)

Note that at state 8 in SPEC, if the return-to-normal event (4) does not occur, then the pyro-valve is closed. However, this action renders SPS not reusable.

5.2.5 Plant Modifiers

The plant modifier can be obtained by converting specification model SPEC to an automaton with a total transition function. This is done by adding a marked dump state 'd' to SPEC and adding the transitions from all states to the dump state so that the resulting automaton has a total transition. We use TTCT [15] to carry the computations and the following formula is used:

Modifier = {Mark the additional state of [Complement of (Complement of (Specification model))]}.

5.2.6 Modified Plant Models

First, we modify the plant model in normal mode. We use TTCT [15] to do the following operation to find the new model:

$$G_N = \text{Meet}\,(G_N^L, \text{Modifier});$$

G_N represents the state based plant model for normal mode. The following diagram shows the state based plant model in normal mode. The states of G_N are of form (q, q') with q being a state of G_N^L and q' a state of modifier. If q' = d (dump state) then (q, q') is an illegal state. Therefore, E_N = {(0,0),(0,1),(1,2),(6,3),(6,4),(3,5)}. The states of G_N are renamed and shown in Fig. 5.8b.

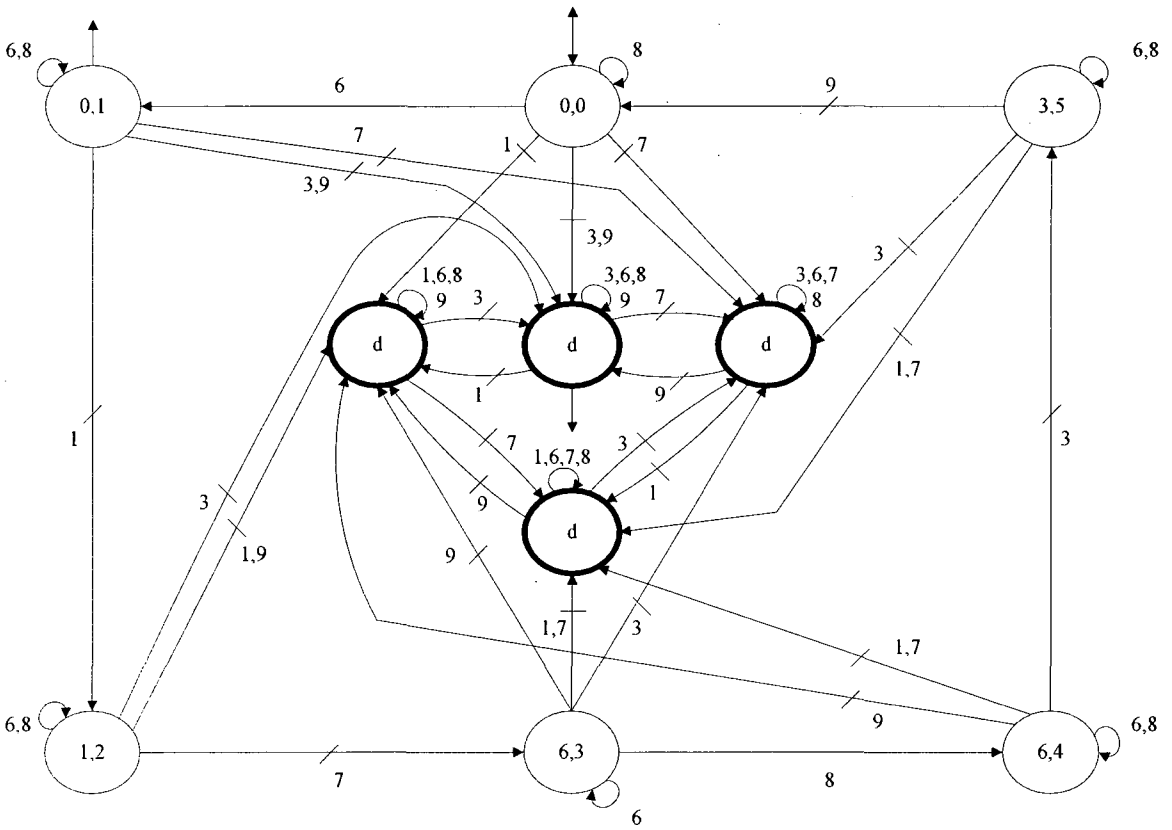

Fig. 5.8a: G_N (State-based plant in normal mode)

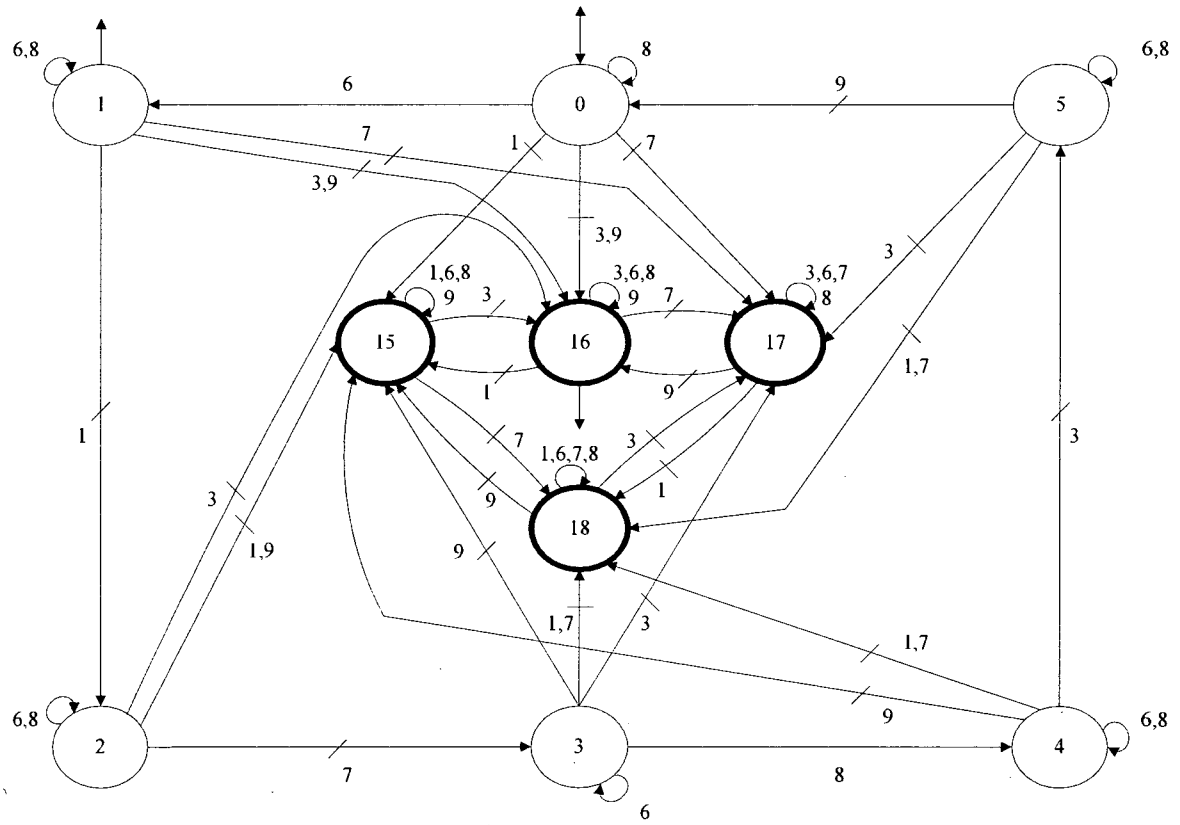

Fig. 5.8b: G_N (States renamed)

We compute the state based plant model in recovery mode using similar procedure. We use TTCT to meet the plant in linguistic model in recovery mode and the modifier. The resulting model: G is shown in Fig. 5.9 below. The model G contains 25 states and 175 transitions. Note that only partial model is shown with important states named.

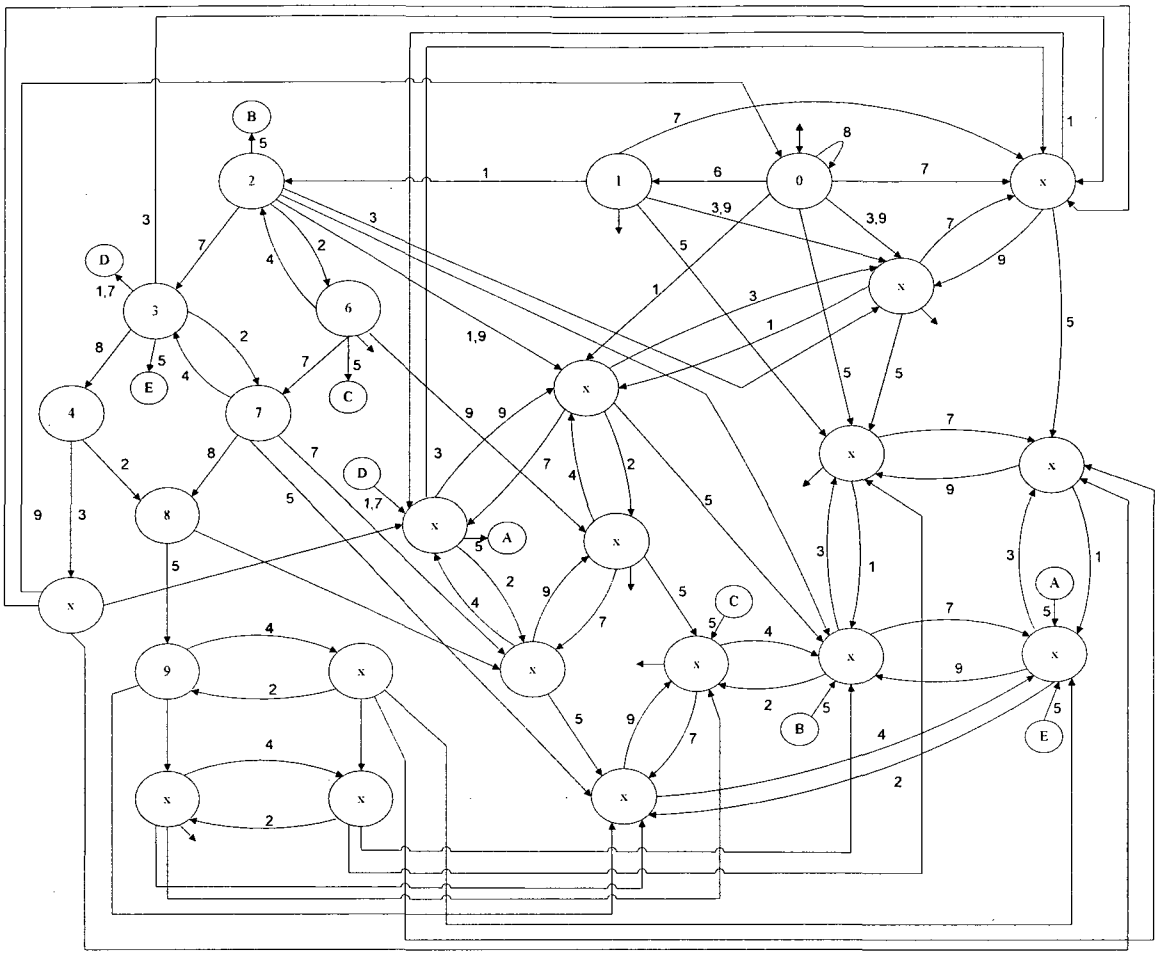

Fig. 5.9: Partial model for G

5.2.7 Supervisors

Following our algorithm presented in Chapter 4, we compute the sequence of supervisors $V_0^*, V_{0'}^*, \ldots$ for SPS.

The following diagram shows the supervisor: V_0^*.

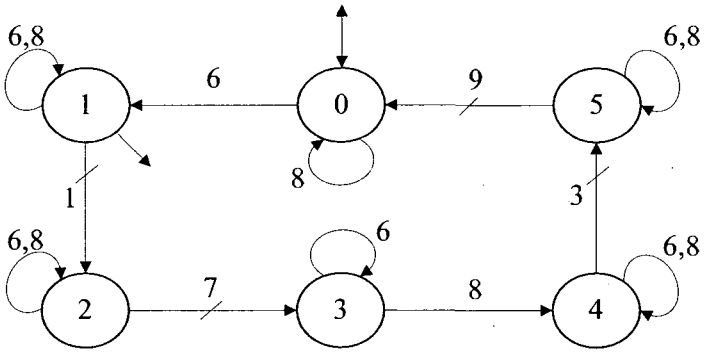

Fig. 5.10: Supervisor: V_0^*

The recovery mode supervisor $V_{0'}^*$ is shown in Fig. 5.11.

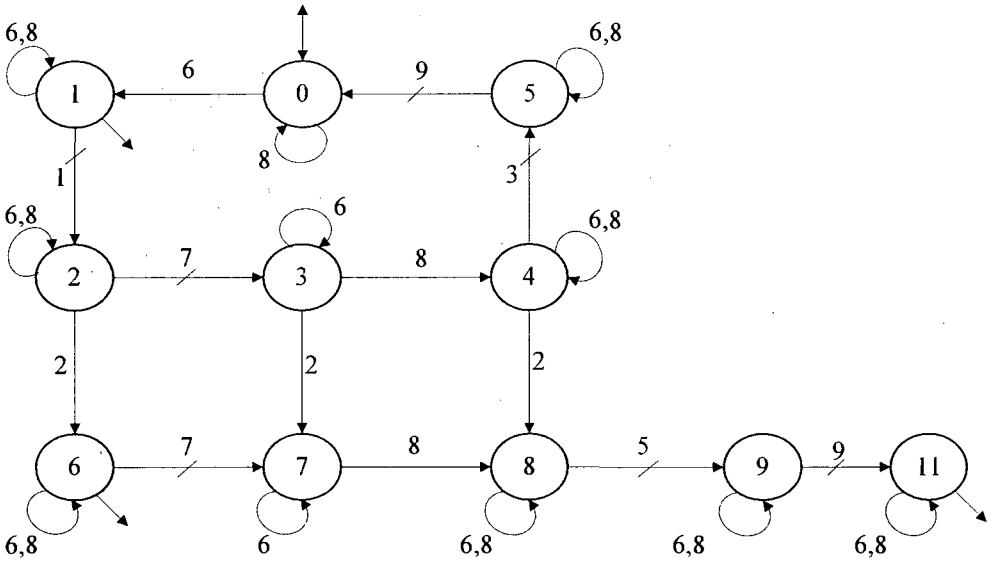

Fig. 5.11: Supervisor: $V_{0'}^*$

Next, we find V_1^* which is shown in Fig. 5.12.

It can be observed that repeat occurrences of the failure event or return-to-normal event do not add any new states to the plant model. Hence, the iteration of computing

supervisors (as per algorithm in Chapter 4) can be stopped here. Hence $V_\infty^* = V_1^*$ and convergence is achieved.

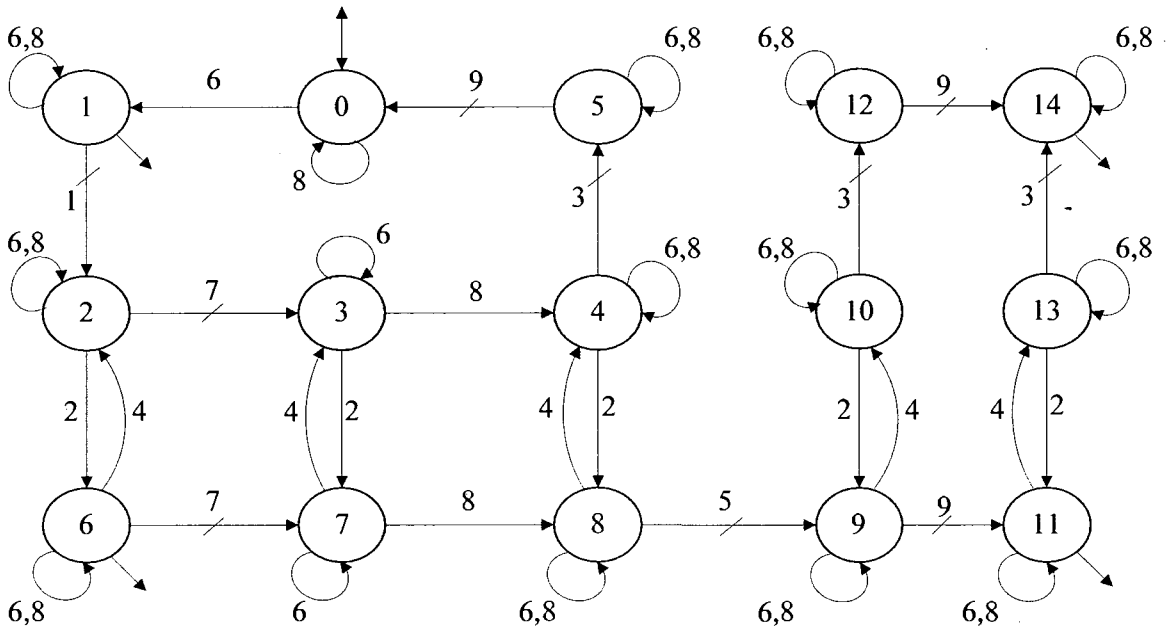

Fig. 5.12: Supervisor V_1^*

The actions of the supervisor V_∞^* can be summarized as follows. During the normal operation, it only uses the regular valve (with the appropriate sequence). In case of failure, if by the time stop command is issued, the failure persists, it will shutdown the engine indefinitely following the proper sequence. In the next section, we present an extended version of SPS. The Extended Propulsion System (EPS) also contains a controllable return-to-normal event and more plant dynamics involved.

5.3 Extended Propulsion System (EPS)

In this example, we consider the extended version of simplified spacecraft propulsion system (SPS). This extended example represents a more realistic model of spacecraft propulsion systems. EPS is a bi-propellant system. The system uses Liquid Oxygen (LOX) along with the main propellant. This example also represents a system wherein return-to-normal event is an uncontrollable event (failure-correct).

5.3.1 Setup of EPS

The Extended Propulsion System (EPS) contains the following components:

1) Pilot: To initiate start and stop commands to start and stop the individual operations involving individual engines. There are two sets of pilot commands in the system for the two engines.
2) Valves: Regular valves to control the flow of the propellants to the engines. There are five valves used in EPS. One of them is used as a backup valve which is initially open. All the other valves are initially closed.
3) Thrust Engines: Two thrust engines to produce required thrust are used in EPS. These two engines are capable of producing thrust in opposite directions. Initially the engines are in off position.

The following diagram shows the setup of the components of EPS:

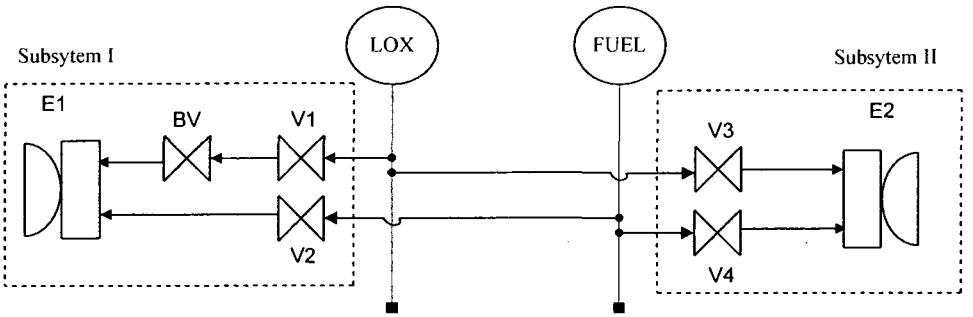

Fig. 5.13: Setup for EPS

The setup diagram does not show the pilot. Also the fuel tanks are not part of DES modeling in the solution presented. Valves, V1 and V2 are used to control the propellant flow to engine, E1. Backup valve, BV is used as a backup for V1. Valves, V3 and V4 control the propellant flow to engine, E2. Both the engines are setup in such a way that they produce thrust in the opposite directions. Both engines are used to accelerate or decelerate. If E1 is used to accelerate then E2 can be used to decelerate in the same direction and vice versa. Both engines are not used simultaneously but used alternatively to cancel excessive thrust in the directions the engines are setup to produce thrust.

The normal operation of the plant is similar to that of SPS. Each engine is operated in similar way. Upon receiving the start command from the pilot, the valves are opened leading to that particular engine. Subsequently, that particular engine is turned on to start producing thrust. At the discretion of the pilot, depending on need for decelerating the first engine that is producing the thrust is stopped and the second engine is started.

A failure event stuck-open is assumed for V1 in open position. In this situation, engine E1 gets continuous supply of LOX even after the stop command is issued by the pilot. This situation creates potential depletion of LOX. Also when it is time to run E2, there will not be enough pressure in LOX supply due to depletion. The return-to-normal event is defined for V1 to come out of the failure mode back to normal mode and to be closed subsequently if necessary. In the case that the return-to-normal event does not happen, the backup valve is used to stop the flow of the LOX. Once the valve V1 returns to normal mode and subsequently closed, the backup valve (BV) can be returned to its normal open position. The major difference in SPS and EPS is that in SPS, once the pyro-valve is closed, the plant can not be reused; whereas in EPS, the system is reusable. The next section presents all the DES models for all the components in EPS.

5.3.2 DES Models

In modeling of these components, we use the same procedures used in SPS. The following table describes all the events used in this example:

Event	Tag	Description
1	V1_open	Valve1 opens
2	V1_fail	Valve1 stuck open (failure event)
3	V1_close	Valve1 closes
4	V1_return	V1 Return-to-normal event (r)
5	BV_close	Backup valve closes
6	Start (P1)	Start command from the pilot for engine 1
7	BV_oepn	Backup valve opens
8	Stop (P1)	Stop command from the pilot for engine 1
9	V2_open	Valve2 opens
10	Start (P2)	Start command from the pilot for engine 2
11	V2_close	Valve2 closes
12	Stop (P2)	Stop command from the pilot for engine 2
13	V3_open	Valve3 opens
15	V3_close	Valve3 closes

17	V4_open	Valve4 opens
19	V4_close	Valve4 closes
21	E1_on	Engine1 turns on
23	E1_off	Engine1 turns off
25	E2_on	Engine2 turns on
27	E2_off	Engine2 turns off

Table 5.2: Event list for EPS

The following diagram shows the model for the pilot:

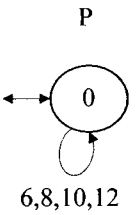

Fig. 5.14: Model for pilot (EPS)

The pilot may issue start and stop commands to both engine subsystems. The models for all the valves are shown in the following diagram:

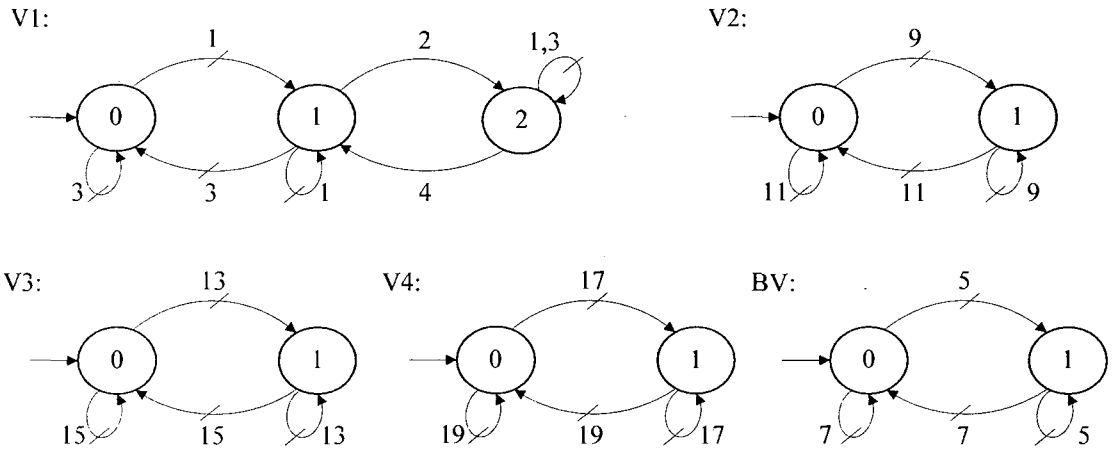

Fig. 5.15: Models for all valves (EPS)

All the valves share similar states either open or close except for V1, which is prone to failure and has an additional state: fail.

The following schematic shows the models for both engines: E1 and E2:

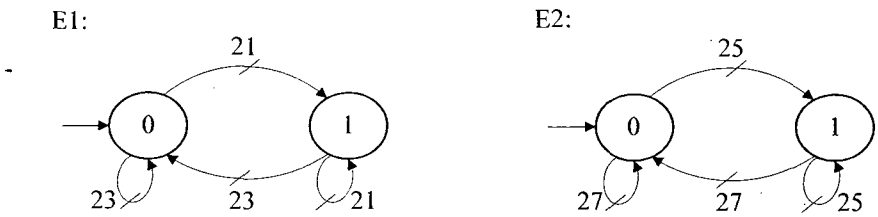

Fig. 5.16: Engine models (EPS)

5.3.3 Plant Models

We use TTCT [15] to compute the plant models. Plant in normal mode comes out to have 64 states and 1024 event transitions. Due its large size, the plant model is not shown here. Similarly, the plant in its full mode comes out to have 192 states and 3456 event transitions. It is observed that those states are marked in which the EPS engines do not generate thrust. The plant in its full mode is not shown here.

5.3.4 State Specifications

The specifications are presented in simple linguistic sequences. The specifications deal with the steps to be followed in operating EPS. The following list outlines the specifications for normal mode of operation:

(i) Each subsystem is operated only after receiving a start command from the pilot.

(ii) The valves that allow LOX have to be opened first before opening the valves that allow the fuel.

(iii) Before starting an engine, the corresponding valves must be open.

(iv) Before shutting down the engine, the corresponding valves must be closed.

(v) The following combination of valves can not be in open position simultaneously:

a. V1 and V3; and V2 and V4 (These combinations will cause depletion of the propellants).

(vi) Both engines must not be on simultaneously.

The following DES models present the required state models to implement the above normal mode specifications. SPEC1 and 2 model (i), (ii), (iii), and (iv). SPEC3 and 4 model (v) and SPEC5 models (vi).

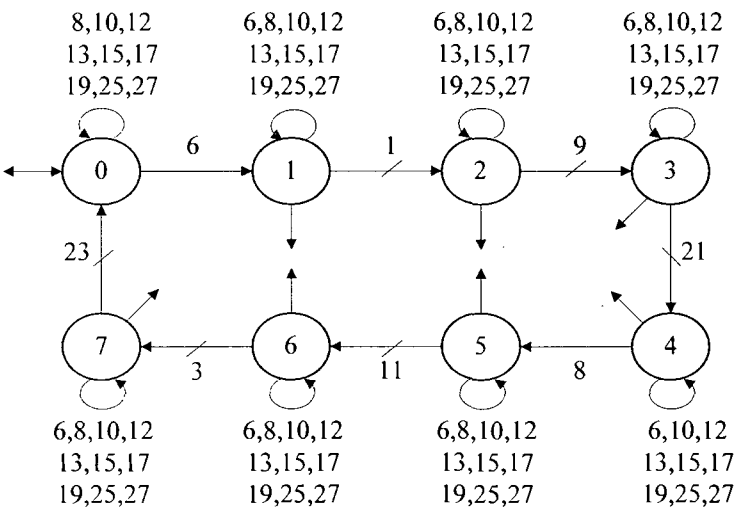

Fig. 5.17: SPEC1 (Normal mode) (EPS)

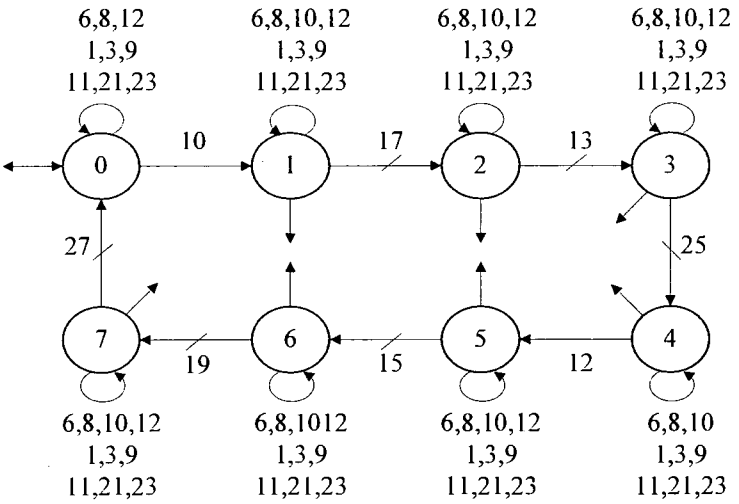

Fig 5.18: SPEC2 (Normal mode) (EPS)

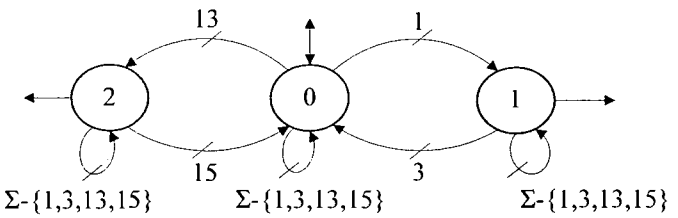

Fig. 5.19: SPEC3 (Normal mode) (EPS)

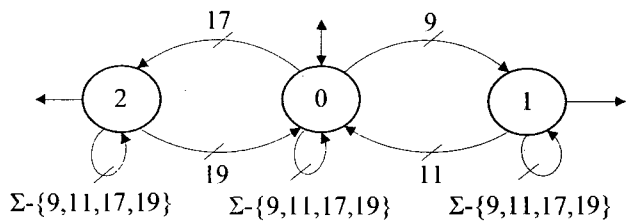

Fig. 5.20: SPEC4 (Normal mode) (EPS)

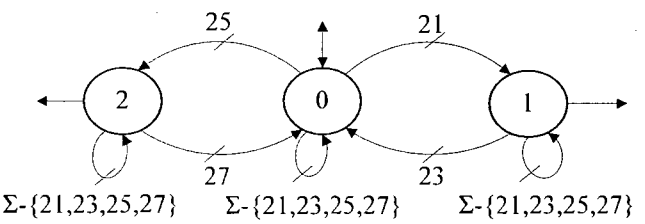

Fig. 5.21: SPEC5 (Normal mode) (EPS)

The recovery specifications are related to V1. V1 is prone to failure. A return-to-normal event is also defined for V1. In the case, where stop command has been issued by pilot but the return-to-normal event has not happened, the backup valve (BV) is used to close the propellant flow as required.

The following list outlines the additional specifications required for recovery mode of EPS. Note that all normal mode specifications still apply to recovery mode specifications.

i) When V1 fails, the backup valve is used to stop the fuel flow when it is time to close the valve after receiving a stop command.

ii) The backup valve can be reopened once the return-to-normal event happens and the V1 is closed.

The following diagram shows the modified specification to control E1 considering the failure and return-to-normal events for V1:

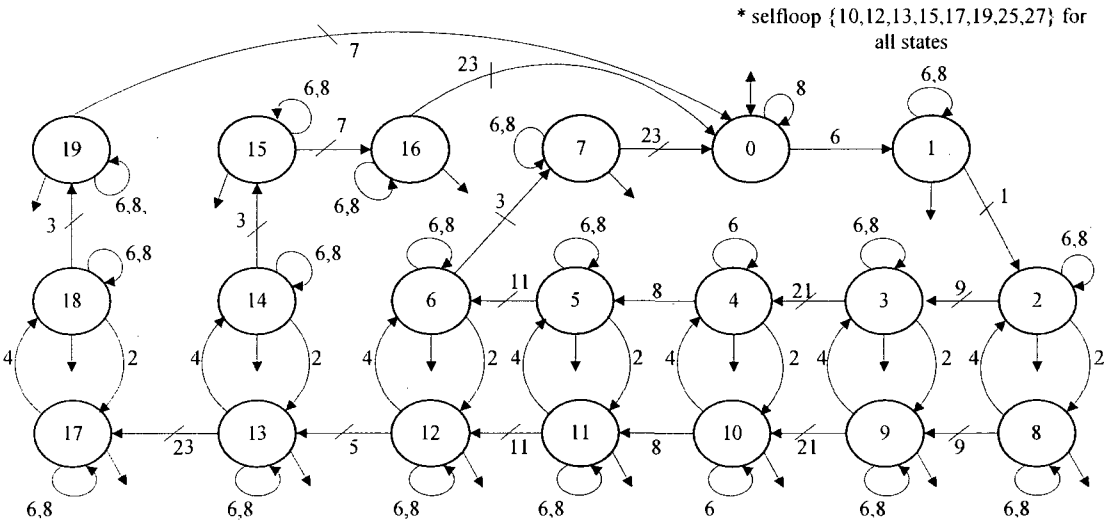

Fig. 5.22: Recovery specifications (EPS)

A suitable supervisor is computed to control the plant under consideration. Our solution particularly tackles a situation which is normally not tackled by standard RW-based solutions. For example, let us consider the plant in recovery mode. State 0 is the initial state of the plant where all the components are at their initial states. State 8 represents the state where all components are at their initial states except V1. V1 is in its failed position at state 8. The plant is not producing any thrust at this state. The following is a partial extract from the plant which includes the recovery mode:

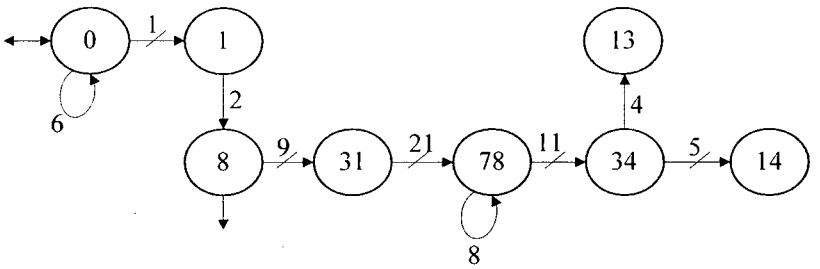

Fig. 5.23: Extract from G (Plant in recovery mode: EPS)

The state that has to be considered is state 34. At this state, the subsystem has received stop command and the fuel valve, V2 is closed. Now is the time to close the valve, V1 which has already failed. At this stage, we expect the return-to-normal event to occur. However, it is not guaranteed that this event will occur. Also state 34 is not marked and the plant can not stay at this state indefinitely where the engine is not off. At this state, standard (non-robust) solutions depend upon the return-to-normal event and expect to solve the problem. However, for practical reasons, a solution can not depend on this event and hence our solution points to the necessity of an alternative way, in this case, a backup valve to solve the problem.

In the two examples considered so far, the return-to-normal events are failure-correct events and controllable. The next example Automatic Resource Allocator (ARA) contains a recovery-action event as return-to-normal event and it is uncontrollable.

5.4 Automatic Resource Allocator (ARA)

In this example, we consider an automatic resource allocator, which supervises the allocation of given resources. In this example, the return-to-normal event (r) is a controllable event as opposed to SPS in the previous section. For simplicity, we consider two users, two resources in the plant. This plant can represent two computers as two users and two parallel processors as two resources.

5.4.1 Setup of ARA

The following schematic shows the setup for ARA.

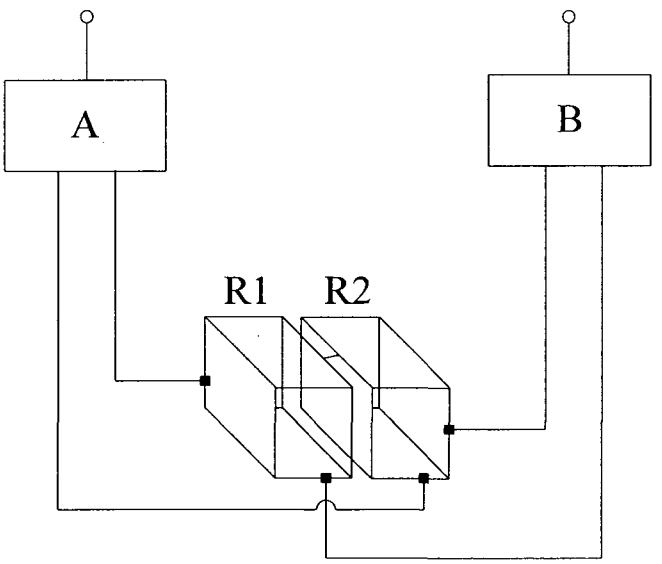

Fig. 5.24: Setup for ARA

The ARA consists of two users, user A and user B, and two resources, R1 and R2. We assume that there is a task allocator which allocates tasks to user A and user B. The resources are allocated as per the availability. ARA does not store any data to save the queuing of the tasks. Both users A and B need both resources to finish any given task.

The normal mode operation of the plant is as follows: Depending upon task assignment, either user A or user B may acquire either resource. Then the same user should acquire the second resource to be able to finish the task. Completion of a task by a particular user will release both resources thus enabling another task to be initiated by either user. The different states of a user are i) Idle, ii) wait, and iii) busy. At the wait state, the user has acquired one resource and ready to acquire the second resource to finish the task. The busy state represents where the user is busy finishing the task before releasing both resources. Each resource can be in two different states i) Available; and ii) Unavailable.

The problem the plant may face in normal mode operation is of 'mutual exclusion.' In this scenario, each user acquires one resource and waits to acquire the second resource. Since the second resource is already acquired by the other user, this scenario can continue indefinitely. The plant observes deadlock in this scenario.

The plant enters into failure (recovery) mode, when user A observes a failure. The failure event (f) is defined for user A, as user A malfunctions after acquiring either one resource or both resources. This event makes the user A unable to finish the task at hand.

At this state, we define a controllable return-to-normal event (r) on user A. This return-to-normal event is a reset action on user A, which resets user A to its initial state. This reset action also makes user A to release the resource(s) at hand. We assume that the rest of the components work without any failures. Since this return-to-normal event is defined as a controllable event, we assume that the event will eventually occur and bring user A back to normal operation thus bringing the plant back to normal operation.

However, as we see in Section 5.4.3, the plant in recovery mode faces blocking at various states. We also see in that section that standard RW-based supervisory control solutions would depend upon the failure event to occur to come out of blocking.

The next section presents the DES models for the users and the resources.

5.4.2 DES Models

The following table describes all the events used in this example:

Event	Tag	Description
1	A_R1	User A acquires R1
2	A_fail	User A malfunctions
3	A_R2	User A acquires R2
5	B_R1	User B acquires R1
7	B_R2	User B acquires R2
9	A_task_complete	User A completes the task
11	B_task_complete	User B completes the task
15	reset	User A gets reset (recovery-action event)

Table 5.3: Event list for ARA

The following diagram shows user A in its full mode:

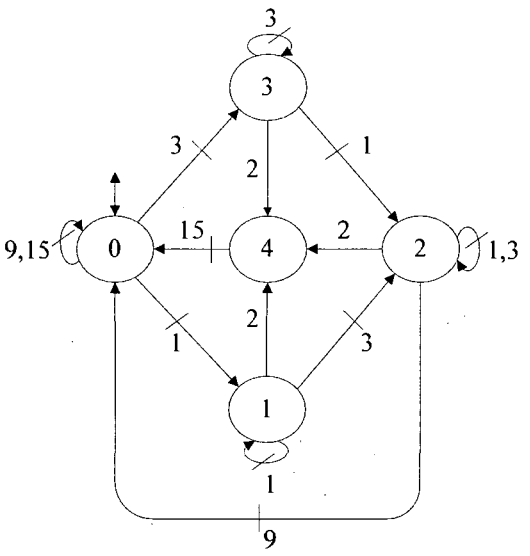

Fig. 5.25: User A (ARA)

Upon receiving a task assignment, user A acquires either R1 or R2. Then it acquires the second resource and completes the task. Once the task is completed, it goes to its initial state, simultaneously releasing both resources. As defined, user A is prone to failure (state 4) and might malfunction at any state after acquiring one resource. The return-to-normal event is a reset and brings user A from malfunctioning state to normal initial state.

The following diagram shows the DES model for user B. User B works just like user A, except that by assumption, (for simplicity), it does not malfunction at any time.

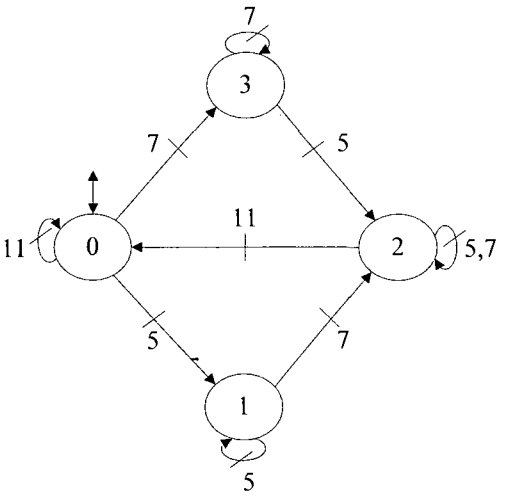

Fig. 5.26: User B (ARA)

The next two diagrams show resources: R1 and R2.

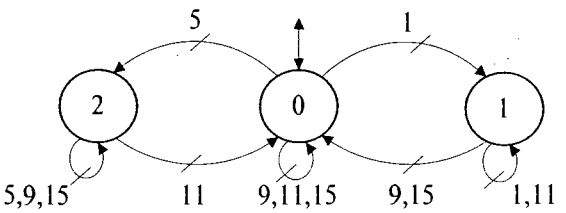

Fig. 5.27: R1 (ARA)

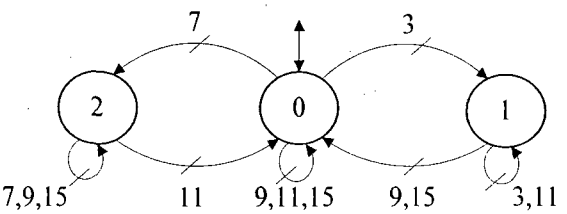

Fig. 5.28: R2 (ARA)

Note that the return-to-normal (reset) event (15) makes both resources available only if they are already acquired by user A.

5.4.3 Plant Models

We use TTCT [15] to compute the ARA in its full mode including the failure event and return-to-normal events. The following diagram shows the plant model:

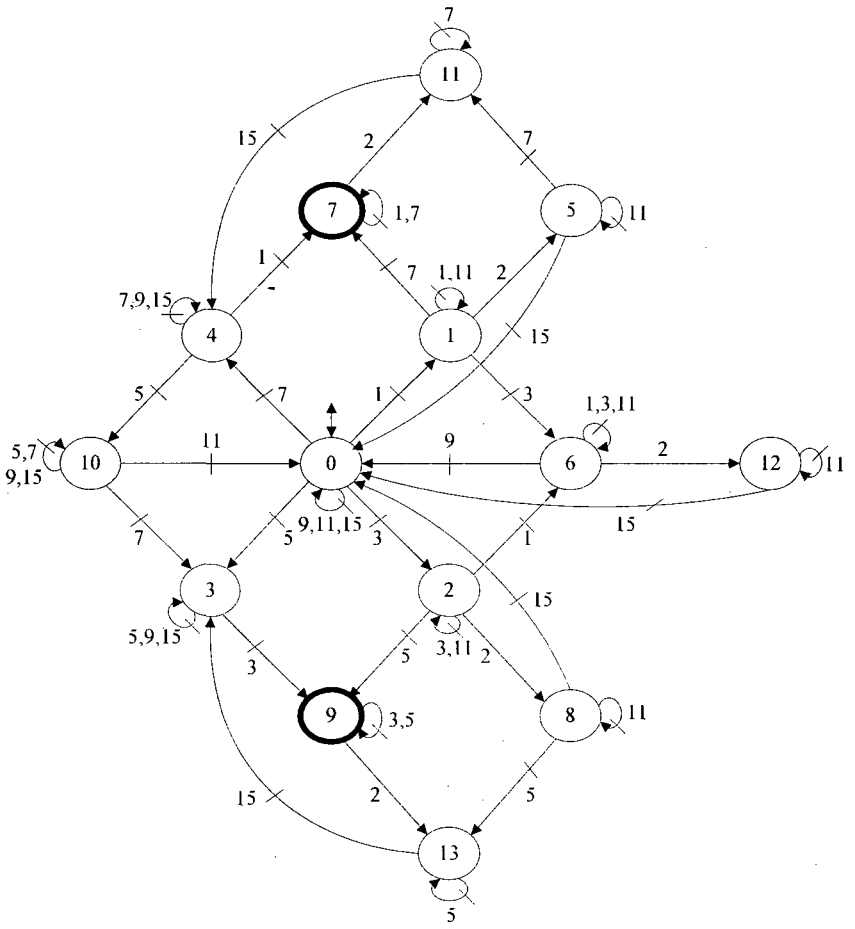

Fig. 5.29: Complete model of ARA (G)

In this model, it can be seen that at states 7 and 9, the plant observes dead blocking unless the failure event (2) occurs. In state 7, User A has R1 and user B has R2, hence deadlock, unless as a result of failure in User A, it relinquishes R1 and then User B can acquire R1. A similar situation occurs in state 9 as well with User A holding R2 and User B holding R1. It is not advised to depend on the failure event to come out of blocking. Standard RW-based supervisory control solutions assume that there is a path to come out of blocking. However, this path starts with a failure event. Our solution

considers this possibility along with possible blocking scenarios when the plant returns to normal mode from recovery mode. However, in this case, we do not foresee any blocking issues in recovery mode since we assume that the controllable return-to-normal event would eventually occur.

For the purposes of understanding the dynamics of ARA, we present the plant model in its normal mode in the following diagram. Note that some states have been renamed to match the plant model.

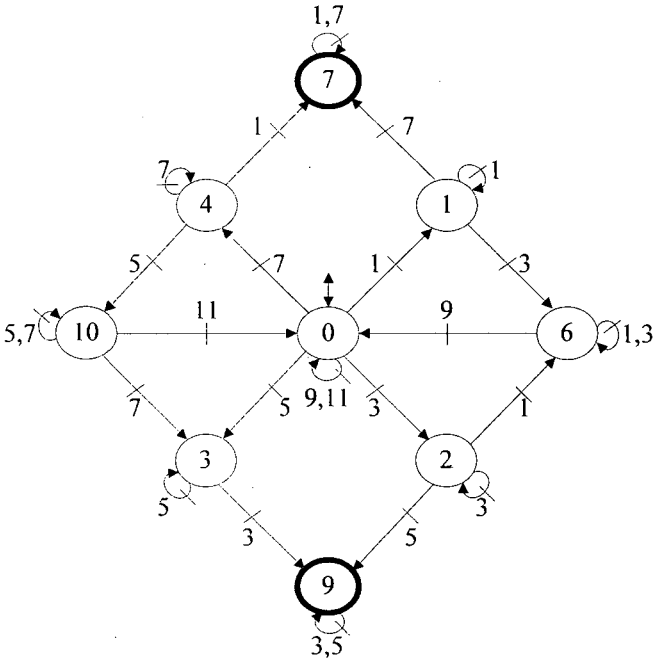

Fig. 5.30: Normal mode Plant (ARA) (G_N)

We obtain the normal mode plant from the full mode plant by removing the failure event and return-to-normal events, and the corresponding states.

In this problem, the job of the supervisor is to ensure nonblocking in normal mode (both initially and after any return-to-normal event). Therefore all states are considered legal: $E_N = Q_N$, $E_R = Q_R$.

5.4.4 Supervisors

Following our algorithm presented in Chapter 4, we compute the sequence of supervisors $V_0^*, V_1^*, \ldots$ for ARA. The final robust supervisor (V_∞^*) is found as the limit of the sequence. Note that in the faulty (recovery) mode no nonblocking condition is enforced since the return-to-normal event is controllable. Also all faulty (recovery) states are legal ($E_R = Q_R$). Therefore, supervisors $V_{0'}^*, V_{1'}^*, \ldots$ need not be constructed.

The following diagram shows supervisor V_0^*:

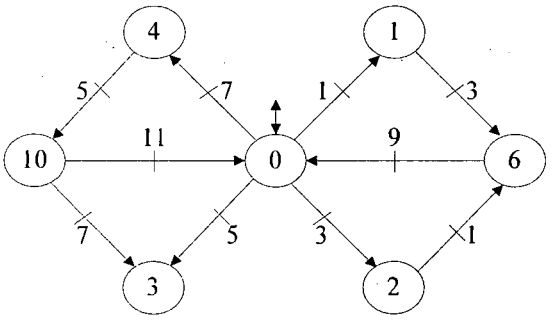

Fig. 5.31: Normal mode Supervisor: V_0^*

And the following diagram shows the supervisor V_1^*:

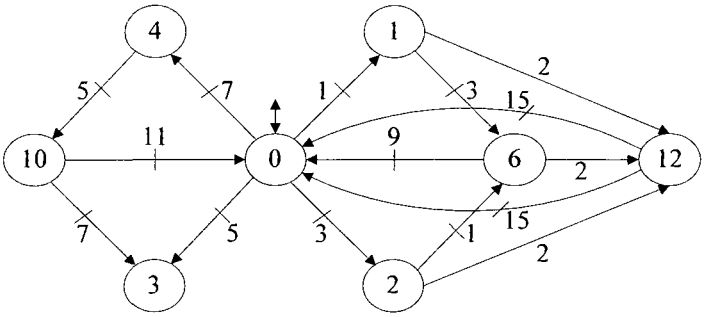

Fig. 5.32: Recovery mode Supervisor: V_1^*

It can be easily verified that the condition for convergence is satisfied and $V_1^* = V_\infty^*$. The automatic resource allocator (ARA) under supervision is shown in the following diagram:

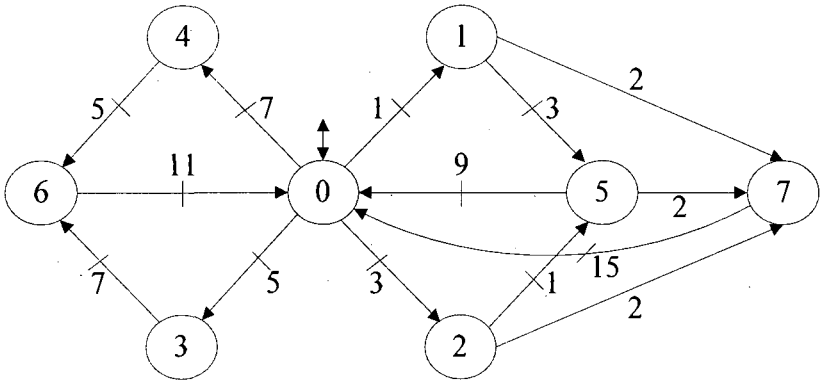

Fig. 5.33: ARA under supervision

It can be seen that the plant under supervision follows 'mutual exclusion' scenario.

5.5 Conclusion

In this chapter, three application examples are given. The first two examples that deal with propulsion systems have controllable return-to-normal events and the third example ARA has an uncontrollable return-to-normal events. It has been shown how the

algorithm presented in Chapter 4 provides the required supervisor, taking into account the plant dynamics and the nature of failure and return-to-normal events.

Chapter 6

Conclusions and Future Research

6.1 Conclusions

In this thesis, fault recovery is studied using discrete event models. In particular, fault recovery when the plant is capable of returning to normal mode of operation is studied. A return-to-normal event is defined as an event that can bring the plant from various faulty modes to normal mode of operation and is defined in the faulty mode. Return-to-normal can be either because the faults are intermittent or because the plant has the capacity to repair or reset. A design algorithm is presented to compute a robust nonblocking supervisor under above said failure conditions.

We assume that the plant can be modeled as DES automaton and it contains finite number of states. We also assume that all events including the failure events are observable. The failure events are uncontrollable and return-to-normal events can be

either controllable or uncontrollable. The failures are assumed to be detected and isolated without any delay.

The algorithm is developed using state-based approach. In our problem setup, we assumed that the specifications are given in form of a set of safe states. The specification for each failure mode is also the same and it does not depend upon the number of times the failure event previously occurred. Then the pre-defined safe states remain unaffected. This results in an easier and more transparent design for the control policies in terms of safe and unsafe states compared to legal and illegal event strings. We consider two cases of supervisory control problem, one with controllable return-to-normal (recovery-action) events and the second one with uncontrollable return-to-normal (recovery-correction) events. We have showen the resulting problems are instances of Robust Nonblocking Supervisory Control of countably infinite plants. We provided a procedure for computing maximally permissive supervisors for the fault recovery problem.

Three illustrative examples were provided. Two of the examples belong to the spacecraft propulsion systems. DES theory lends itself perfectly to be used in the high level supervisory control of spacecraft subsystems such as propulsion systems. Both Simplified Propulsion System (SPS) and Extended Propulsion System (EPS) examples depict propulsion systems used in spacecraft. These two examples contain uncontrollable return-to-normal events. The Automatic Resource Allocator (ARA) represents computer queuing systems and contains controllable return-to-normal events. The resulting supervisors were also provided as DES models.

The problem considered in this thesis is all about plant dynamics and how these dynamics affect overall results expected from specific plant operations. As shown in the examples, all standard (non-robust) RW-based supervisory control methods fail to specifically address the nonblocking properties of plants under supervision after certain failures occur. The solutions provided by standard methods tend to have blocking issues when the plant moves to normal mode and start the cycle of operation again. Providing a robust solution that considers all the dynamics of the plant is very important in the face of ever growing systems and their mission criticality. Generally, DES theory is well suited to provide high level control in complex systems. However, DES theory developed with the view of internal event dynamics best provides a complete solution to achieve desired supervisory control.

6.2 Future Research

The current research can be continued in the following areas:

- In this thesis, the supervisory control problem for fault recovery is solved using untimed discrete event models. This recovery framework can be extended to timed discrete event systems (TDES).
- We assumed that all events are observable. In future work, either the failure events or return-to-normal events can be considered as unobservable events. This would induce transient modes into the problem. Then the computed supervisor must enforce safety requirements in transient modes too.

- In this thesis, only 'single failure scenarios' are considered through the design algorithm and examples provided. However, in general, failures can occur simultaneously in DES. The framework presented here can be extended to work for 'multiple failure scenarios.' However, this would make the computations very complex. A systematic way has to be found to deal with this problem. To that end, it would be useful to develop a software program to implement the design and verification so that multiple failure scenarios can be dealt with.
- Only one kind of return-to-normal events is considered at a time in solving the examples, either controllable or uncontrollable. However, in general, a plant might contain a mix of controllable and uncontrollable return-to-normal events related to different kinds of failure scenarios. Solving the robust nonblocking supervisory problems for these cases would be very challenging and computationally exhaustive. Further investigation is needed to implement the design algorithm for these cases.

Bibliography

[1] S.E. Bourdon, M. Lawford, and W.M. Wonham, 'Robust nonblocking supervisory control of discrete-event systems.' *IEEE. Trans. On Automatic Control*, vol 50, Pgs: 2015 – 2021, 2005.

[2] S.E. Bourdon, M. Lawford, and W.M. Wonham, 'Robust nonblocking supervisory control of discrete-event systems.' *Proc. of the American Control Conference*, vol 1, pgs: 730 – 735, May 2002.

[3] Charles D. Brown, 'Spacecraft Propulsion.' *Text book*, AIAA, Edition 1996.

[4] R. Cieslak, C. Desclaux, A. S. Fawaz, and P. Varaiya. 'Supervisory control of discrete-event processes with partial observation.' *IEEE Trans. On Automatic Control*, 33:249-260, 1988.

[5] Song Foh Chew, and Mark A. Lawley, 'Robust supervisory control for production systems with multiple resource failures.' *Proc. of the 2005 IEEE Int. Conf. on Automation Science and Engineering*, pages: 375 – 380, 2005.

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

[7] Dieter K. Huzel, and David H.H, 'Modern Engineering for Design of Liquid-Propellant Rocket Engines.' *Text book*, AIAA, edition 1992.

[8] F. Lin and W. M. Wonham, 'On observability of discrete event systems.' *InformationSciences*, 44:173-198, 1988.

[9] Rudolf X. Meyer, 'Elements of Space Technology.' *Text book*, Academic Press, 1st edition 1999.

[10] M. Moosaei, and Dr. S. Hashtrudi Zad, 'Fault recovery in control systems: A modular discrete-event approach.' *1st Int. Conf. on Electrical and Electronics Engineering.* Pgs: 445 – 450, 2004.

[11] P.J. Ramadge and W. M. Wonham, 'Supervisory control of a class of discreet event processes,' *SIAM Journal of Control and Optimization*, vol. 25, no. I, pp. 206-230, Jan, 1987.

[12] A. Saboori, and S. Hashtrudi Zad, 'Fault recovery in discrete event systems.' *Int. conf. on Computational Intelligence Methods and Applications*, 2005. 6 Pages.

[13] A. Saboori and S. Hashtrudi Zad, 'Robust nonblocking supervisory control of discrete-event systems under partial observation.' *Systems & Control Letters*, vol. 55, no. 10, pp. 839-848, Oct 2006.

[14] C. Wang, and Hashtrudi Zad, S, 'Fault recovery in discrete-event systems using observer-based supervisors.' *Int. Conf. INDICON*, 2005, Pgs: 442 – 445.

[15] TTCT Software. [Online] Available: http://www.control.toronto.edu/people/profs/wonham/wonham.html

[16] Q. Wen, R. Kumar, J. Huang, and H. Liu, 'Weakly fault-tolerant supervisory control of discrete event systems.' *Proc. of the 2007 American Control Conference.*

[17] B.C. Williams, M.D. Ingham, S.H. Chung, and P.H. Elliott, 'Model-based programming of intelligent embedded systems and robotic space explorers.' *Proc. of the IEEE,* vol. 91: 212 – 237, 2003.

[18] W. M. Wonham. Supervisory control of discrete-event systems. [Online] Available: http://www.control.utoronto.ca/DES.

[19] Hashtrudi Zad, S, and M. Moosaei, 'Fault recovery in control systems: A modular discrete event system approach.' *1st Int. Conf. on Electrical and Electronics Engineering 2004 (ICEEE)*, Pgs: 445 – 450.