NOTE TO USERS

Fault Recovery using Discrete Event Models

Anooshiravan Saboori

A Thesis

in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of Master of Applied Science at

Concordia University

Montréal, Québec, Canada

June 2005

Your file Votre référence
ISBN: 0-494-10250-0
Our file Notre référence
ISBN: 0-494-10250-0

ABSTRACT

Fault Recovery using Discrete Event Models

Anooshiravan Saboori

In this thesis, we study the synthesis of fault recovery procedures using discrete-event models. It is assumed that the faults are permanent and that a diagnosis system is available that detects and isolates the faults with a bounded delay. Thus, the combination of the plant and the diagnosis system, as the system to be controlled, will have three modes: normal, transient and recovery. Initially, the plant and thus the system to be controlled, are in the normal mode. Once a fault occurs in the plant, the system (to be controlled) enters the transient mode. After the fault is diagnosed by the diagnosis system, the system enters the recovery mode.

We study the design of a nonblocking supervisor that enforces the specifications of the system in all three modes. The solution is obtained by first transforming the problem to an equivalent Robust Nonblocking Supervisory Control Problem under Partial Observation (RNSC-PO) and then solving the robust control problem. In the robust control problem, the exact model of the plant is among a finite set of possible models. We show that under a "Blocking Invariance" assumption on the behaviors of the possible plant models, the solutions of RNSC-PO can be characterized in terms of certain controllable and observable languages. However the RSNC-PO problem resulting from Fault Recovery Problem (FRP), need not satisfy the blocking invariance assumption. To tackle this problem, it is shown that under some certain assumptions on the possible plant models, which holds for the RNSC-PO problem resulting from the FRP, the RNSC-PO problem can be replaced by an equivalent RNSC-PO for which the blocking invariance assumption holds and thus the solution of the

resulting RNSC-PO problem, or equivalently the original FRP, are obtained and parameterized in terms of certain controllable and observable languages. We discuss both single-failure scenario and simultaneous occurrence of multiple failures.

ACKNOWLEDGEMENTS

I am deeply indebted to my supervisors Dr. Shahin Hashtrudi Zad and Dr. Ferhat Khendek for their constant support and guidance throughout the work with this thesis.

I would like to thank my brothers, Arash and Ashkan for their never-ending support during my research.

I dedicate this thesis to my mother and father for their love, support, sacrifice and encouragement.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

Introduction

1.1 Fault Recovery in DES

As the complexity of control systems has increased, concerns for their safe and reliable operation in the presence of failures, especially in critical systems, have grown (see, e.g., [13],[14]). In order to ensure reliability in case of failure, the controller may take appropriate remedial actions such as control system reconfiguration. In this thesis we study the problem of the synthesis of fault recovery procedures using discrete-event models. A Discrete Event System (DES) is a discrete-state, event driven system, that is, its state evolution depends entirely on the occurrence of asynchronous events. This type of system includes a large class of systems such as manufacturing systems, database management, communication protocols, data communication networks, traffic systems and logistic systems. Fault recovery in discrete-event systems (DES) has been studied by researchers.

1.1.1 Background

In this subsection, we review some of the reported results on fault recovery using discrete event models.

1. **Transition–based fault models**: In this framework, faults are modeled as transitions

in the model of the plant. In [24], the system is modeled with an *extended I/O automaton* capturing timing information about the actions in the plant. Moreover the actions are classified into three classes: *normal, fault and recovery* and lower and upper bounds are associated with these actions, acting as deadlines for normal actions and duration of existence for temporary fault actions. Three different approaches are proposed for supervisor synthesis: *masking fault tolerant system* which can recover from any fault, *t-fault tolerant system* which can mask up to t faults during its life time and finally *gracefully degrading fault tolerant system* which resumes operation during recovery. However no algorithmic method for supervisor synthesis is given.

In comparison to our work in the thesis, [24] assumes transient failures however we assume permanent failures. Thus the set of problems addressed in this thesis is different from that studied in [24]. Moreover, unlike to our framework, marked behavior is not considered in [24] and also all faults are assumed to be observable.

2. **State based fault models**: In this type of approach the state set is divided into normal and error states and similar to classical control, certain notion of **stability** are introduced. The most common definitions are given in the following.

1. For a set of initial states, the system's state is guaranteed to enter a given set and stay there forever.

2. For a set of initial states, the system's state is guaranteed to visit a given set of states (E) infinitely often.

3. Stability in the sense of Lyapunov.

[20] takes definition 2 and with restricting E to "the states of system corresponding to normal operation", it restates the definition of a stable system as "*one which can recover from anomalies without catastrophic error propagation*". Faults are assumed to be unobservable. The proposed observer is *resilient* to errors in observation, i.e. inserted events, missed events

or mistaken events. In order to synthesize the supervisor, two notations of output stabilizability and strong output stabilizability are introduced, where the former is stronger in the sense that it could also tell us *when* the trajectory of events reaches the normal state set. They show that if the system is output stablizable then there exists a supervisor which is synthesized by synchronous product of an observer for the original system and an observer for the modified system in which transitions starting from normal state are removed. This framework, although having a meaningful physical interpretation for error recovery, in one view lacks a formal structure compared with classical control theory.

Compared to our framework, one important feature that is not considered in [20] is the nonblocking behavior of the resulting supervisor. Moreover [20] only finds a solution to the problem compared to our work which characterizes all the solutions to the problem.

[23] introduces the notion of Lyapunov stability in discrete event systems. The work is based on metric space analysis and Lyapunov functions. They propose sufficient and necessary conditions for the stability of invariant sets of DES in a metric space. An example of a manufacturing systems is also discussed in [23]. It also argues that the proposed method does not require the high computational complexity of typical DES methods, however the difficulty lies in specifying the Lyapunov function.

In comparison to our framework, [23] does not consider nonblocking as a requirement for supervisory control. Moreover, unlike our solution, the total set of solutions is not characterized in this work.

[5] proposes a definition of stability which enjoys the well-structured Lyapunov theory, in addition to the meaningful physical interpretation of [20] thanks to the introduction of resiliency parameter in the definition of Lyapunov stability. Connections are also established between this notion of stability and the property of fault tolerance. This framework suffers the same restrictions as the two previous cases in comparison to our framework.

3. **Symbolic Synthesis Approaches**: These approaches are developed in response to

the following known practical problems of developing supervisors in Ramadge and Wonham framework of supervisory control:

1. Computational complexity and state-space explosion.

2. Supervisor implementation.

[1] investigates these problems by accepting input-output interpretation of supervisory control theory. Moreover a synthesis fixpoint algorithm implementation using **binary decision diagrams** (**BDD**) is utilized to enable the design of supervisors of realistic size. The controller consists of two parts, a dynamic controller which implements the behaviorial specifications (explicit or task-related) and a static supervisor which filters the commands from the aforementioned controller according to equipment related specifications (implicit). These specifications are mainly related to *safety* and *liveness* of the system and that is the point which makes this work suitable for fault tolerant systems category. Moreover the nice feature of this work is the ability of the controller to adapt to new specifications by online synthesis of controller based on *BDD*. This framework is similar to our framework in the sense that the designer is given the freedom to control the system behavior after the occurrence of faults by assuming separate specifications for different modes of plant operation. A limiting assumption of [1] compared to our framework is that all events including the faults are observable. In [8], a framework for analyzing and verifying fault tolerant behavior of closed loop system (plant under supervision) is presented. Errors are modeled explicitly as separate entities (erroneous states) such that transient and permanent faults can be modeled uniformly. Furthermore, the framework can describe *fault-masking* and *t-fault tolerance*. The model is based on an automaton where transitions are labeled with expected events. Errors cause the events with similar "start state" to change behavior such that they lead to the error "end states" instead of the end state given by transition associated with the event. An approach for stating the fault tolerance requirements in CTL is given and the use of symbolic model checking to verify them is suggested.

The framework of [8] however, does not consider nonblocking. Also faults are assumed to be observable. On the other hand, their framework covers both transient and permanent failures whereas our framework only considers permanent failures.

4. **Explicit Synthesis Approaches**: This category is where our work falls into. This category covers a range of frameworks. [6] investigates fault tolerance in Ramadge and Wonham framework. Fault tolerance is defined as the ability of the system under supervision to reach a marked state despite the occurrence of faults and failures. A systematic way to classify faults and failures quantitatively is proposed.

Unlike our framework, all faults are assumed to be observable. They also do not parameterize the total set of solutions to their problem. It should also be noted that the solution we present in this thesis actually includes the solution to their problem as well.

[21] generalizes the results of [6] to the design of fault tolerant robust supervisors which guarantee fault–tolerant behavior represented as tolerable fault event sequences of a DES in spite of model uncertainty. As for robustness, it is assumed that the exact model of the plant is among a finite $\{G_i\}$, $i \in \mathcal{I}$, and the purpose of the controller is to guarantee that for each G_i, $i \in \mathcal{I}$ all event sequences reach marked states in spite of faults and failures. Multiple-failure scenarios are also considered.

Again in this framework, faults and failure are assumed to be observable. No other design specification is considered for faulty operation.

[19] investigates a modular approach to fault recovery in control systems. Failures are assumed to be permanent and unobservable. Single-failure scenario is considered. A diagnoser is assumed to be available which diagnoses the fault with bounded delay. The diagnosis is modeled as a finite-state generator. This model is an abstraction of the underlying diagnosis system and thus the framework does not depend on the technique used for diagnosis. The composition of plant and diagnoser will have three modes of operation: *normal, transient* and *recovery*. The design of supervisor consists of a normal-transient supervisor and multiple recovery supervisors each for recovery from a particular failure mode. Moreover nonblocking

and admissibility of the proposed supervisor are studied. Our work is actually based on [19]'s framework. We parameterize all of the solutions of the fault recovery problem, including that of [19] and state the necessary and sufficient conditions for the existence of the supervisor. We also generalize our results to simultaneous-failure scenarios.

5. **Planning and Reconfiguration Approach** In [29], *Plan* is a sequence of actions leading from the initial state to a goal state and since all actions are assumed to be controllable and deterministic there is no explicit way to model non-determinism caused by faults. The approach to this problem is either *conditional planning* or *replanning*. In conditional planning, the truth value of a set of state predicates is assumed to be unknown and they can become known during plan execution. Because of the high complexity of this approach, *replanning* and *reactive planning* are introduced. In replanning, the supervisor observes the execution of the plant and if the plan fails, a new plan is generated. A reactive planner is one which mixes execution and replanning. [29] considers a model-based approach to reactive self-configuring systems. Planner is equipped with a model based diagnoser. [4] proposes control reconfiguration for a sensor network which is based on a model-based fault isolation. The last two framework differs from our framework in the diagnoser module. In our case, we separate diagnoser module from supervisor module, by using an abstraction of the diagnoser module and assuming that a diagnoser is available which diagnoses the fault within bounded delay; however in [4] and [29] the supervisor and diagnoser are studied together. Moreover [29] considers probabilistic models for the generators as opposed to our deterministic models. In [10], a reconfigurable robot team is studied which can learn and modify its plans in response to one or more of the robots becoming faulty and thus being removed out of service. The system is ensured to remain within the bounds of prespecified behavioral constraints. This work can have applications in fault tolerance where the uncertainty in the system is due to the occurrence of faults. The interesting feature of this approach is that supervisor for a faulty mode is obtained by applying some modifications to the supervisor in use before

the fault. As a result, this method is very effective in responding to simultaneous failures.

Compared to our framework, in [10] all failures are assumed to be observable. Moreover the necessary and sufficient conditions of the existence of the solution is not investigated (unlike our framework). The faults are assumed to be in the form of a component of the system removed from the system; we do not have such assumptions in our work. Finally the proposed algorithm may not have a solution in which the supervisor has to be redesigned.

In another approach to control reconfiguration, [18] develops a framework for reconfiguration of a DES controller, which has a dynamic event observation set. A mega controller is designed that monitors the observation set of the DES controller and its state continuously. Upon a change in the observation set, the mega controller reconfigures the controller by a aggregation or disaggregation of the controller states. Sufficient and necessary conditions for the existence of the solution are provided.

Unlike our framework, [18] assumes the failures to be observable and the non-blocking property is not considered. Moreover the set of solutions of the problem is not provided.

It should also be noted that all of the methods proposed in planning and reconfiguration category, are on-line and thus differ from our framework and our design which is offline. One important issue for all reactive planning approaches is response latency which is an important implementation challenge.

As mentioned earlier, in this thesis, we study the problem of fault recovery in discrete-event systems. To obtain the set of solutions of this problem, we first transform the problem to an equivalent robust nonblocking supervisory control problem under partial observation and then solve the robust control problem. In the following section we briefly review recent research on the robust control of discrete event systems.

1.2 Robust Supervisory Control

Control of systems with modeling uncertainties is a complex problem. One of the approaches to deal with modeling uncertainties is adaptive control that invokes an identification procedure to resolve uncertainties and updates the control algorithm accordingly. The other approach is robust control in which the control system is designed to work for a family of plants. Robust and adaptive control of continuous systems have been studied by researchers in continuous systems as well as discrete event systems.

1.2.1 Background

There are different types of uncertainties in DES and hence different approaches in the literature to model uncertainty.

In one of these approaches[16], it is assumed that the exact model of the plant is not known, however it is among a finite set of possibilities, i.e. $\{G_i\}$ for $i \in \mathcal{I}$ for some finite index set $\mathcal{I}$. [16] formulates the Robust Supervisory Control–Partial Observation (RSC-PO) as follows.

Assume two nonempty and closed languages A and E are given such that

$$A \subseteq E \subseteq \bigcap_{i \in \mathcal{I}} L(G_i) \tag{1.1}$$

Synthesize a proper supervisor υ such that for all $G_i, i \in \mathcal{I}$:

$$A \subseteq L(\upsilon/G_i) \subseteq E \tag{1.2}$$

Here $L(G_i)$ and $L(\upsilon/G_i)$ refer to the closed behaviors of G_i and G_i under the supervision of υ. [16] finds sufficient and necessary conditions on the desired behavior of the closed loop

system, K, assumed to satisfy

$$K \subseteq \bigcap_{i \in \mathcal{I}} L_m(G_i), \tag{1.3}$$

such that there exists a supervisor v for which:

$$L_m(v_i/G) = K$$

Using this result [16] establishes sufficient and necessary conditions on A and E such that the RSC-PO is solvable.

In another approach to model uncertainty, the uncertainty is associated with internal and unobservable events occurring in systems and denoted by Δ-transitions. Lim [22] initiated this framework. He assigns a positive integer to some states to show that at most a $p - step$ state aggregation may occur by the internal and unobservable events. The Robust Nonblocking Supervisory Control–Partial Observation (RNSC-PO) is formulated in [22] as follows.

Given a specification K such that

$$K \subseteq P(L_m(G))$$

find a nonblocking supervisor such that

$$P(L(v/G)) = \overline{K}$$

where P is the projection with respect to the unobservable events in the nominal plant (excluding Δ).

Similar to [16], [22] finds sufficient and necessary conditions on K for the existence of a robust nonblocking supervisor, however [22] considers non deterministic generators as well.

In a framework closer to first approach, [9] formulates another robust supervisory control problem. This framework is based on infinite strings and infinite behavior of automata. All of the events are assumed to be observable and also no assumption is made on specifications (i.e.(1.1) is removed). Nonblocking is considered as a requirement and thus $L(v/G_i)$ in (1.2) is replaced with $L_m(v/G_i)$. The authors in [9] also generalize their formulation of robust supervisory control by adding the requirement that the solution should maximize robustness i.e. it should maximize the set of plants which, under supervision, satisfy the requirements of the RNSC problem. In other words, the following set should be maximized:

$$\varphi(A, E, v) := \{G \mid A \subseteq L_m(v/G) \subseteq E, L(v/G) = \overline{L_m(v/G)}\}$$

[9] shows that the above problem does not have a solution in general. However by restricting A and E in (1.1) to $A \cap L_\omega(G)$ and $E \cap L_\omega(G)$, [9] proves that $\varphi(A, E, v)$ has a maximizing element and proposes an algorithm for finding the element. [9] also investigates computational effort of the algorithm.

[28] generalizes the results of [9] to the unobservable case. [28] also removes the requirement of replacing A and E with $A \cap L_\omega(G)$ and $E \cap L_\omega(G)$ by assuming that A and E are closed. [28] also adds another part to the RNSC-PO (in addition to robustness in the sense of maximizing $\varphi(A, E, v)$) that should also be maximally permissive for each of the plants in $\varphi(A, E, v)$. [28] shows that the robustness can be maximized with partial observation, but permissiveness can not always be maximized. [28] also proves that under the assumption of observability of all controllable events, the obtained solution maximizes both robustness and permissiveness.

In [2] the authors choose the framework of [16] as the basis of their work. However instead of assuming upper and lower specifications (A, E), a separate specification is assumed

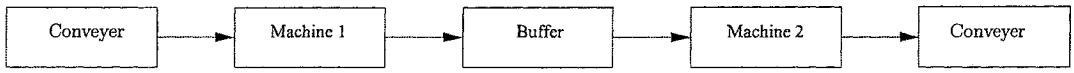


Figure 1.1: Example of a Fault Recovery Problem in a manufacturing system

to be given for each plant in the index set. i.e. (1.2) is changed to the following:

$$L_m(v/G_i) \subseteq E_i, \quad i \in \mathcal{I}$$

In this formulation, nonblocking of the resulting supervisor is also a requirement. All of the events are assumed to be observable. An algorithm to find a maximally permissive solution to RNSC is proposed and is shown to terminate in finite number of steps. Moreover the solution is proved to posses some adaptive behavior eventhough it is computed off-line.

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

1.3 Fault Recovery Problem: An Example

This thesis deals with Fault Recovery using Discrete Event Models. In following we provide an illustrative example to motivate our discussion.

Example: Consider two machines in a manufacturing system (Figure 1.1). Machine 1 accepts workpieces from the input conveyer and perform process P_1 on it. Upon termination of the process, the workpiece is deposited in the buffer. Machine two retrieves this work piece from the buffer and performs process P_2 on it. Note that these machines have different operations to perform on the workpieces meaning that P_1 is different from P_2. It is assumed that both machines are acceptable of performing P_1 and P_2 processes.

In the absence of failures (normal specification), it is required that the buffer neither underflow nor contain more than 10 workpieces. Any of the machines may break at any time and thus goes out of service. Let us assume that a diagnoser is available that diagnoses

any failure within a delay of 0 to 2 events. While one of the machines is broken and the fault has not been diagnosed, the system should be working under the same specification for the normal mode. That is the buffer should neither underflow nor contain more than 10 workpieces in it. We refer to this mode of operation as the transient mode. After the failure is diagnosed, if machine 1 has failed, machine 2 should process all workpieces in the buffer and then should switch to a new cycle. In this new cycle, P_1 must be performed on 5 workpieces and then machine 2 switches to P_2 and retrieves the 5 workpieces being deposited in the buffer and performs P_2 on them. This is the design specification for recovery from failure in machine 1. In case machine 2 fails, machine 1 should switch to P_2 and process all workpieces in the buffer. After that it must switch to the aforementioned $5P_1 + 5P_2$ recipe. (Specification for recovery from failure in machine 2).

□

In fault recovery problem, the objective is to design a supervisor that performs suitable control sequences and enure that the system meets its specifications in all normal, transient and recovery modes. Furthermore, the system should not block during its operation. Any supervisor that meets the aforementioned objectives is a solution to the problem. In this thesis, we manage to characterize the entire set of solutions of the above fault recovery problem. The contributions of this thesis are explained in more detail in the following section.

1.4 Thesis Contributions

The contributions of the thesis could be summarized as follows:

1. We show that the Fault Recovery Problem can be transferred to an equivalent Robust Nonblocking Supervisory Control problem under Partial Observation(RNSC-PO).

2. The solution to the Robust Nonblocking Supervisory Control Problem under partial

observation is obtained by extending the solution to the robust control with full observation existing in the literature. The solution is obtained assuming a "blocking invariance" condition on the behaviors of possible plant models. This property states that if a string creates blocking in one of the possible models, then in any other possible plant model, it either creates blocking or can not be generated at all. The property of blocking invariance originates in this thesis.

3. The set of Blocking Invariant language n-tuples is shown to posses a supremal element and an algorithm is proposed to obtain it. The convergence of the algorithm in the general case is still an open problem but under certain conditions (which holds in the problem derived from fault recovery problems), the algorithm is shown to converge to supremal blocking-invariant sublanguage n-tuple in one step.

4. In case of convergence of Algorithm A, it is shown that the original RNSC-PO can be converted to an equivalent RNSC-PO in which the marking languages of the plant models are replaced with the supremal blocking invariant sublanguages of the marked behaviors of the original plants. In other words, a sufficient condition is found under which the RNSC-PO problem can be converted to an equivalent problem in which the blocking-invariance condition holds.

5. The RNSC-PO resulting from Fault Recovery Problem in Single-Failure Scenario is shown to posses the aforementioned structural assumptions and thus the results of Robust Control Problem can be used to obtain the solutions to this problem. We show that the solutions of the RNSC-PO problem, and thus the fault recovery problem can be characterized in terms of certain controllable and observable languages.

1.5 Thesis Outline

In Chapter 2, we review the background material required in the solution of Fault Recovery Problem and Robust Nonblocking Supervisory Control Problem. In Chapter 3, we review the Robust Nonblocking Supervisory Control Problem under Partial Observation and find the solutions to this problem under. Chapter 4 discusses fault recovery problem and characterize its solution. An illustrative example is also provided. In Chapter 5, we provide a summary of the thesis contributions and discuss future directions for research.

Chapter 2

Background Overview

A Discrete Event System (DES) is a discrete-state, event driven system, that is, its state evolution depends entirely on the occurrence of asynchronous discrete events. This type of system includes a large class of systems such as manufacturing systems, database management, communication protocols, data communication networks, traffic systems and logistic systems.

Finite-state automaton represent a class of discrete-event systems. In this thesis, we study the design of fault recovery procedures for discrete-event systems modeled as finite-state automata. As we will see, this problem is a special case of robust supervisory control. In this chapter, we briefly review some of the results from the automata theory and the supervisory control of discrete-events systems required for our work in subsequent chapters.

2.1 Languages and Automata

2.1.1 Languages[12]

Let Σ be a finite set of distinct symbols called **alphabet**. Then define

$$\Sigma^+ := \{\alpha_1 \dots \alpha_k | \ k \geq 1, \ \alpha_i \in \Sigma\}$$

Defining ϵ as the sequence with no symbol, we write:

$$\Sigma^* = \Sigma^+ \cup \{\epsilon\}$$

Any element of Σ^* is a **string** over alphabet Σ. ϵ is the **empty string**. The concatenation of two strings s and t is the new string st consisting of the events in s followed by the events in t. A string t is called a prefix of s if:

$$\exists u \in \Sigma^* : s = tu.$$

Finally t is a suffix of s if :

$$\exists u \in \Sigma^* : s = ut$$

2.1.2 Operations on languages

1. Concatenation [12]: Let $L_1, L_2 \subseteq \Sigma^*$, then:

$$L_1 L_2 := \{s \in \Sigma^* : s = s_1 s_2, s_1 \in L_1, s_2 \in L_2\}$$

In other words, $L_1 L_2$ is the set of the strings resulted from the concatenation of the strings in L_1 with the strings in L_2.

2. Prefix–closure [12]: Let $L \subseteq \Sigma^*$, then:

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

Equivalently, the prefix closure of a language contains all the prefixes of all the strings in L.

3. Projection [3, 30]: Let $L_1 \subseteq \Sigma_1^*$ and $L_2 \subseteq \Sigma_2^*$ where we may have $\Sigma_1 \cap \Sigma_2 \neq \emptyset$. Let $\Sigma = \Sigma_1 \cup \Sigma_2$. We define natural projection of Σ^* onto Σ_i^*: $P_i : \Sigma^* \rightarrow \Sigma_i^*, (i = 1, 2)$ according

to

$$P_i(\epsilon) = \epsilon$$

$$P_i(\delta) = \begin{cases} \epsilon & \text{if } \delta \notin \Sigma_i \\ \delta & \text{if } \delta \in \Sigma_i \end{cases}$$

$$P_i(s\delta) = P_i(s)P_i(\delta), \quad s \in \Sigma^*, \delta \in \Sigma$$

Simply speaking P_i removes all the symbols in s which are not in Σ_i. Clearly P_i is catenative. i.e. $P(st) = P(s)P(t)$.

4. Complement [12]: Let $L \subseteq \Sigma^*$. Then:

$$L^{co} := \Sigma^* - L$$

2.1.3 Generators and Automata

Automata and generators are the visual means of representing DES. A deterministic generator G is a 5-tuple :

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

Here Σ is a finite alphabet of symbols referred to as **event set (label)**, Q is the **state set**, $\delta : Q \times \Sigma \rightarrow Q$ is the **partial transition function**, q_0 is the **initial state** and $Q_m \subseteq Q$ is the **marked states.** We refer to a generator as an automaton if its transition function is a total function over its domain.

The closed and marked languages of a generator are defined via:

$$L(G) := \{s \in \Sigma^* | \ \delta(q_0, s) \text{ is defined}\},$$

$$L_m(G) := \{s \in L(G) | \ \delta(q_0, s) \in Q_m\}.$$

$L(G)$ represents all the event sequences that can be generated by G, starting from the initial state q_0. Thus s is in $L(G)$ if and only δ is defined at (q_0, s). Clearly $L(G)$ is prefix-closed. Moreover if G is an automaton we have:

$$L(G) = \Sigma^*$$

$L_m(G)$ is a subset of $L(G)$ consisting of strings s for which $\delta(q_0, s) \in Q_m$.
Observe that we do not require Q to be finite. All the operations which will follow, actually work for infinite state generator. Moreover we can represent any language L by an automaton. The idea would be to represent the generator as a (possibly infinite) tree whose root is the initial state and nodes at level n are reached by either strings with length n or prefixes of length n of longer strings in language. In this case the state set would corresponds to the nodes of this tree and a state is marked if and only if the string which reaches that node from the root is a part of the language. We say G represents L and

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

It may not always be possible to represent a language by a *finite* state generator. A famous example of this type of language is $L = \{a^n b^n : n \geq 0\}$. We refer to languages that can be represented by a finite state generator as **regular languages**. The class of regular languages is very important since they require finite memory to be stored in computer and thus their manipulation in controller synthesis could be easier and more practical[3]. To have an alternative characterization of regular languages, we first need the definition **equivalence relation.**

Let X be a nonempty set and $E \subseteq X \times X$ a **binary relation**[30] (any subset of $E \times E$ is a

binary relation) on X. E is an **equivalence relation**[30] if

$$\forall x \in X : \; x \equiv_E x$$

$$\forall x, x' \in X : \; x \equiv_E x' \Rightarrow x' \equiv_E x$$

$$\forall x, x', x'' \in X : \; x \equiv_E x', x' \equiv_E x'' \Rightarrow x \equiv_E x''$$

We shall also write $x \equiv x'(modE)$ instead of $x \equiv_E x'$.

For $x \in X$, let $[x]$ denote the subset of elements x' that are equivalent to x:

$$[x] := \{x' \in X \mid x' \equiv_E x\}$$

The subset $[x]$ is called the **equivalence class**[30] of x with respect to the equivalence relation E. We also need the definition of **Nerode equivalence relation (L)**[31] to define regular languages formally. Consider a language $L \subseteq \Sigma^*$. For $s, t \in \Sigma^*$, we have $s \equiv_L t$ if

$$\{s' : s' \in \Sigma^* \; and \; ss' \in L\} = \{t' : t' \in \Sigma^* \; and \; tt' \in L\}$$

In other words, two strings are equivalent(mod L) if their continuations to form completed words of L are identical. We shall write

$$\| \; L \; \| := card(\Sigma^* / \equiv_L),$$

namely $\| \; L \; \|$ is the (possibly denumerable) number of equivalence classes of $\equiv_L$. Finally the language L is regular if $\| \; L \; \| < \infty$ [12]. $\| \; L \; \|$ corresponds to the number of states of the minimal-state generator representing L [31].

The following theorem summarizes the properties of regular languages.

Theorem 1. *[12] Let L_1 and L_2 be regular languages. Then the following languages are*

also regular:

$$\overline{L_1},\ \Sigma^* - L_1,\ L_1 \cup L_2,\ L_1 L_2,\ L_1 \cap L_2$$

Consider two generators with the same alphabet Σ: $G_1 = (Q_1, \Sigma, \delta_1, q_{01}, Q_{m1})$ and $G_2 = (Q_2, \Sigma, \delta_2, q_{02}, Q_{m2})$. We say that G_1 is a **subgenerator** of G_2, denoted by $G_1 \sqsubseteq G_2$, if $\delta_1(s, q_{01}) = \delta_2(s, q_{02})$ for all $s \in L(G_1)$. Note that marked states are not considered in this definition. Also $G_1 \sqsubseteq G_2$ implies $Q_1 \subseteq Q_2$, $q_{0,1} = q_{0,2}$ and $L(G_1) \subseteq L(G_2)$. However $L(G_1) \subseteq L(G_2)$ does not imply $G_1 \sqsubseteq G_2$ as languages with fewer elements may require more states in order to be generated or they may permit more aggregation of states.

2.1.4 Operations on Generators

1. **Accessible (Reachable) part**[3, 30]: Define

$$\begin{aligned} Ac(G) &:= (Q_{ac}, \Sigma, \delta_{ac}, q_0, Q_{m,ac}) \\ Q_{ac} &= \{q \in Q : \exists s \in \Sigma^*, \delta(q_0, s) = q\} \\ Q_{m,ac} &= Q_m \cap Q_{ac} \\ \delta_{ac} &= \delta \mid Q_{ac} \times \Sigma \to Q_{ac} \end{aligned}$$

The effect of this operation is to remove all the states in G that can not be reached by any of the strings in $L(G)$ starting from the initial state. Obviously we have:

$$L(Ac(G)) = L(G),\ L_m(Ac(G)) = L_m(G)$$

Thus we can assume all the generators are accessible.

2. **Coaccessible part**[3, 30]: Define

$$CoAc(G) := (Q_{coac}, \Sigma, \delta_{coac}, q_{0,coac}, Q_m)$$

$$Q_{coac} = \{q \in Q : \exists s \in \Sigma^*, \delta(q, s) \in q_m\}$$

$$q_{0,coac} = \begin{cases} q_0 & \text{if } q_0 \in Q_{coac} \\ \text{undefined} & \text{otherwise} \end{cases}$$

$$\delta_{coac} = \delta \mid Q_{coac} \times \Sigma \to Q_{coac}$$

This operation deletes all the sates from the state transition of G which are not coaccessible. A state is called coaccessible if there is a path from that state in the state transition diagram of G to a marked state. Obviously we have

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

3. **Trim operation**[3, 30]: Define

$$Trim(G) := CoAc(Ac(G)) = Ac(CoAc(G))$$

4. **Product (Meet)**[3, 30]: Define

$$G_1 \times G_2 := Ac(Q_1 \times Q_2, \Sigma_1 \times \Sigma_2, \delta, (q_{0,1}, q_{0,2}), (Q_{m,1}, Q_{m,2}))$$

$$\delta((q_1, q_2), \sigma) := \begin{cases} (\delta_1(q_1, \sigma), \delta_2(q_2, \sigma)) & \text{if } \delta_1(q_1, \sigma)! \text{ and } \delta_2(q_2, \sigma)! \\ \text{undefined} & \text{otherwise} \end{cases}$$

Thus at a state (q_1, q_2) of $G_1 \times G_2$, δ can occur, if and only if both G_1 and G_2 agree on the

occurrence of that event. It could be easily verified that

$$L(G_1 \times G_2) = L(G_1) \cap L(G_2)$$
$$L_m(G_1 \times G_2) = L_m(G_1) \cap L_m(G_2)$$

5. **Synchronous Product (Parallel Composition)**[3, 30]: Define

$$G_1 \parallel G_2 := Ac(Q_1 \times Q_2, \Sigma_1 \cup \Sigma_2, \delta, (q_{0,1}, q_{0,2}), Q_{m,1} \times Q_{m,2})$$

$$\delta((q_1, q_2), \sigma) := \begin{cases} (\delta_1(q_1, \sigma), \delta_2(q_2, \sigma)) & \text{if } \sigma \in \Sigma_1 \cap \Sigma_2 \\ (\delta_1(q_1, \sigma), q_2) & \text{if } \sigma \in (\Sigma_1 - \Sigma_2) \\ (q_1, \delta_2(q_2, \sigma)) & \text{if } \sigma \in (\Sigma_2 - \Sigma_1) \\ \text{undefined} & \text{otherwise} \end{cases}$$

In the parallel composition, a common event, that is, an event in $\Sigma_1 \cap \Sigma_2$, can only be executed if the two automata both execute it simultaneously. The other private events are not subject to such a constraint and can be executed whenever possible. It could be easily verified that

$$L(G_1 \parallel G_2) = P_1^{-1}(L(G_1)) \cap P_2^{-1}(L(G_2))$$
$$L_m(G_1 \parallel G_2) = P_1^{-1}(L_m(G_1)) \cap P_2^{-1}(L_m(G_2))$$

where P_i $(i = 1, 2)$ refers to natural projection w.r.t Σ_i.

2.2 Supervision of Discrete–Event Systems

2.2.1 Full Observation

In the Ramadge-Wonham (RW) approach to supervisory control, the plant to be controlled is modeled as a finite-state generator $G = (Q, \Sigma, \delta, q_0, Q_m)$. It is assumed that some of the events of the plant are not controllable such as failures and thus Σ can be partitioned into two sets, controllable events (Σ_c) and uncontrollable events (Σ_{uc}). Control is achieved by means of a supervisor which enables and disables controllable events. By applying supervisory control, we can vary the languages generated and marked by the plant and thus it is possible to limit the behavior of the plant within certain limit represented by another language, $E \subseteq L_m(G)$. E is called the *legal language.*

We can consider a supervisory control of G as a map

$$\upsilon : \Sigma^* \rightarrow \Gamma_\Sigma$$

where

$$\Gamma_\Sigma = \{\gamma \in Pwr(\Sigma) |\ \Sigma_{uc} \subseteq \gamma\}$$

For $s \in L(G)$, $\upsilon(s)$ denotes the set of events enabled by the supervisor. Note that υ should not disable uncontrollable events and thus $\Sigma_{uc} \subseteq \upsilon(s)$. The system under supervision is denoted by υ/G. The closed behavior of υ/G is the language $L(\upsilon/G)$ obtained inductively as follows:

1. $\epsilon \in L(\upsilon/G)$,
2. $(s, \sigma \in \Sigma^*) : s\sigma \in L(\upsilon/G) \Leftrightarrow \sigma \in \upsilon(s), s\sigma \in L(G)$.

The marked langauge of the plant under supervision is defined as

$$L_m(\upsilon/G) := L(\upsilon/G) \cap L_m(G).$$

v is said to be a **nonblocking supervisor**[25] if and only if $\overline{L_m(v/G)} = L(v/G)$.

Nonblocking Supervisory Control Problem (NSC)[25]: Given the plant $G = (Q, \Sigma, \delta, q_0, Q_m)$ with $\Sigma = \Sigma_c \cup \Sigma_{uc}$ and legal language $E \neq \emptyset$ and $E \subseteq L_m(G)$, find a nonblocking supervisor v such that:

1. $L_m(v/G) \subseteq E$
2. $L(v/G) = \overline{L_m(v/G)}$.

The NSC problem in general can have many solutions. Let

$$V := \{v |\ v \text{ solves the } NSC \text{ problem } \}.$$

A supervisor $v \in V$ is called **maximally permissive** if and only if

$$\forall v' \in V : L_m(v'/G) \subseteq L_m(v/G),\ L(v'/G) \subseteq L(v/G).$$

To characterize the solutions of the NSC we need the following definitions.

1. **Controllability**[25]: A language $K \subseteq \Sigma^*$ is controllable with respect to G if

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

In other words, K is controllable if and only if no string in $L(G)$ that is already a prefix of K when followed by an uncontrollable event in G, exits from the prefix closure of K.

2. $L_m(G)$-**closedness**[25]: $K \subseteq L_m(G)$ is $L_m(G)$-closed if:

$$K = \overline{K} \cap L_m(G)$$

The following theorem parameterizes all of the solutions to the NSC problem.

Theorem 2 ([25]). *Let G and E be defined as in NSC.*

1. *Suppose $K \subseteq E$, $K \neq \emptyset$ and K is*

(a) Controllable w.r.t G,

(b) $L_m(G)$ – closed.

Then a supervisor, $\widehat{v}$, exists that solves NSC with $L_m(\widehat{v}/G) = K$ and $L(\widehat{v}/G) = \overline{K}$.

2. *For any v which solves the NSC, $L_m(v/G)$ satisfies conditions (a)-(b) in part (1).*

Theorem 2 implies that if E satisfies conditions (a)-(b) in part 1 of the theorem, a nonblocking supervisor v for which

$$L_m(v/G) = E$$

exists and will be the maximally permissive solution to NSC. However we have to answer the following question: "What is the maximally permissive solution to NSC if E does not satisfy conditions (a)-(b) in Theorem 2?" The key to answer this question is in part (1) of the theorem which states that every subset of E which is controllable and $L_m(G)$-closed characterizes a solution to NSC. So the maximally permissive solution in this case would be the largest controllable and $L_m(G)$-closed subset of E, if any. To explore this issue further, we first review a few facts from lattice theory.

Poset (Partially ordered Set)[12, 30] : Let X be an arbitrary set and also let $\leq$ be a **binary relation**. The pair $(X, \leq)$ is called a **poset** if:

1. $\forall x \in X : x \leq x$
2. $\forall x, x', x'' \in X : x \leq x', x' \leq x'' \Rightarrow x \leq x''$
3. $\forall x, x' \in X : x \leq x', x' \leq x \Rightarrow x = x'$

If the relation $\leq$ could be understood from the context we could also say X is a poset.

Meet and Join Operations[12, 25]: An element $l \in X$ is the **meet** of $x, y \in X$ iff:

$$l \leq x \;\&\; l \leq y \;\&\; (\forall a \in X : a \leq x \;\&\; a \leq y \Rightarrow a \leq l)$$

Similarly an element $u \in X$ is the **join** of $x, y \in X$ iff:

$$x \leq u \;\&\; y \leq u \;\&\; (\forall b \in X : x \leq b \;\&\; y \leq b \Rightarrow u \leq b)$$

Lattice[30, 12]: A **lattice** is a poset L in which the meet and join operations of any two elements always exist. To generalize the notions of meet and join, consider a subset $S \subseteq L$. Define $l = \mathbf{inf(S)}$ as the largest lower bound of S, in other words,

$$[\forall y \in S : l \leq y] \quad \& \quad [(\forall z)(\forall y \in S : z \leq y) \Rightarrow z \leq l]$$

We can define $\mathbf{sup}(S)$ in a dual fashion. Note that if S is finite, these two elements always exist since $inf(S)$ and $sup(S)$ can be computed as the meet and join of the finite number of elements of S (and L) which are also guaranteed to exist since L is a lattice. However if S is not finite, it is not necessarily true that $inf(S)$ and $sup(S)$ exist. A lattice L is called **complete** if for any subset S of L, $inf(S)$ and $sup(S)$ exist as elements of L. Now if only $sup(S)$ exists we say L is a **complete upper semilattice**[1]. Let

$$\mathcal{C}_G(E) := \{K \subseteq E | \; K \text{ is controllable w.r.t } G\}$$

As a subset of the sublanguages of E, $\mathcal{C}_G(E)$ is a poset with respect to inclusion ($\subseteq$). The following proposition states that $\mathcal{C}_G(E)$ is a complete upper semilattice with the join operation of the lattice of sublanguages of E.

Proposition 1. *[25, 30] $\mathcal{C}_G(E)$ is nonempty and closed under arbitrary unions. In particular $\mathcal{C}_G(E)$ contains a supremal element defined as follows:*

$$sup\mathcal{C}_G(E) = \bigcup \{K \mid K \in \mathcal{C}_G(E)\}$$

[1]For an *upper semilattice*, the meet operation need not be defined.

To obtain the maximally permissive solution to NSC, we need another family of languages which is defined as follows:

$$\mathcal{R}_G(E) := \{K \subseteq E | \ K \text{ is } L_m(G)\text{-closed}\}$$

Proposition 2. *[15] $\mathcal{R}_G(E)$ is nonempty and closed under arbitrary unions. In particular $\mathcal{R}_G(E)$ contains a supremal element defined as follows:*

$$sup\mathcal{R}_G(E) = E - (L_m(G) - E)\Sigma^*$$

Proposition 3. *[25, 30] Let E be $L_m(G)$-closed. Then $sup\mathcal{C}_G(E)$ is also $L_m(G)$-closed.*

Theorem 2 states that every subset of E that is controllable and $L_m(G)$-closed is a solution to the NSC. Thus the following class of languages parameterizes all the solutions to NSC:

$$\mathcal{RC}_G(E) = \mathcal{R}_G(E) \cap \mathcal{C}_G(E)$$

Moreover $\mathcal{RC}_G(E)$ is nonempty and closed under arbitrary unions and thus contains a supremal element. Based on Proposition 3, $L_m(G)$-closedness is preserved under controllability operator, thus:

$$sup\mathcal{RC}_G(E) = sup\mathcal{C}_G(sup\mathcal{R}_G(E))$$

This characterizes the maximally permissive solution to NSC.

2.2.2 Partial Observation

Generally only a subset of the events generated by the plant (Σ_o) can be observed by supervisor. This may be due to limitations of sensors attached to the system or distributed nature of some systems where events at some locations are not observed in other locations.

Let $P : \Sigma^* \rightarrow \Sigma_o^*$ be the natural projection. In these situations, since the supervisor can not distinguish between two strings s_1 and s_2 with the same projection, that is $Ps_1 = Ps_2$, the same control action should be issued in response to each one. Note that the subset of unobservable events in general need not have any particular relation to the subset of controllable events. *NSC* could be naturally extended to include the unobservable events as follows:

Nonblocking Supervisory Control under Partial Observation Problem (NSC-PO)[17, 7]: Given $G = (Q, \Sigma, \delta, q_0, Q_m)$ with $\Sigma = \Sigma_c \cup \Sigma_{uc}, \Sigma = \Sigma_o \cup \Sigma_{uo}$ and legal langauge $E \subseteq L_m(G), E \neq \emptyset$, find a nonblocking supervisor control v such that:

1. $L_m(v/G) \subseteq E$
2. $L(v/G) = \overline{L_m(v/G)}$.

The concept of observability is crucial in finding the solution to NSC-PO.

Observability[17, 7]: A language $K \subseteq \Sigma^*$ is said to be $(L(G), P)$–observable if and only if

$$\forall s, s' \in \Sigma^*, \delta \in \Sigma : [s' \in \overline{K} \ \& \ s'\delta \in L(G) \ \& \ s\delta \in \overline{K} \ \& \ Ps = Ps'] \Rightarrow s'\delta \in \overline{K}$$

This property states that if two strings in the language have the same projection then a decision rule that apply to one can be used for the other.

The following theorem parameterizes all the solutions of the NSC-PO problem.

Theorem 3 ([17, 7]). *Let G and E be defined as in $NSC - PO$.*

1. *Suppose $K \subseteq E$, $K \neq \emptyset$ and K is*

   (a) Controllable w.r.t G,

   (b) $(L(G), P)$-observable

   (c) $L_m(G) - closed$.

   Then a supervisor, $\widehat{v}$, exists that solves $NSC - PO$ with $L_m(\widehat{v}/G) = K$ and $L(\widehat{v}/G) = \overline{K}$.

2. *For any v which solves the $NSC - PO$, $L_m(v/G)$ satisfies conditions $(a - c)$ in part (1).*

Now let us bring in the following class of languages:

$$\mathcal{O}_G(E) := \{K \subseteq E |\ K \text{ is } (L(G), P)\text{-observable}\}$$

Theorem 3 states that every subset of E that is controllable, observable and $L_m(G)$-closed expresses a solution to NSC-PO. Thus the following class of languages characterizes all of the solutions to NSC-PO:

$$\mathcal{ORC}_G(E) = \mathcal{O}_G(E) \cap \mathcal{R}_G(E) \cap \mathcal{C}_G(E)$$

However contrary to the full observation scenario, $\mathcal{O}(G)$ need not have a unique supremal element since it is not closed under arbitrary unions. Thus a maximally permissive solution may not exist. *Normality* is a property which is stronger than observability; however it is mathematically "well-behaved " and perhaps easier for analysis.

Normality:[17, 7] Also a language $K \subseteq L(G)$ is $(L(G), P)$-normal if and only if:

$$\overline{K} = L(G) \cap P^{-1}(P\overline{K})$$

In other words, K is $(L(G), P)$-normal if and only if it could be recovered from its projection along with a knowledge of the structure of $L(G)$.

One of the differences between normality and observability is that in the class of closed normal language we can say when the evolving string s exits from the language by watching its projection. This is not the case for observable language. Normality is a stricter property. In fact, a language which is normal is also observable but the converse statement is not necessarily true. Finally if the controllable events are all observable then every controllable, observable language is also normal.

Now let us consider the class of normal sublanguages of given language E.

$$\mathcal{N}_G(E) := \{K \subseteq E |\ K \text{ is } (L(G), P)\text{-normal}\}$$

Proposition 4. *[17, 30] $\mathcal{N}_G(E)$ is nonempty and closed under arbitrary unions. In particular $\mathcal{N}_G(E)$ contains a supremal element defined as follows:*

$$sup\mathcal{N}_G(E) = \bigcup\{[s] \cap L(G) \mid\ [s] \cap L(G) \subseteq E\}$$

where

$$[s] = \{s' : s' \in \Sigma^*, Ps' = Ps\}$$

Proposition 5. *[17, 30] Let E be $L_m(G)$-closed. Then $sup\mathcal{N}_G(E)$ is also $L_m(G)$-closed.*

From the definition of normal languages we have

$$\mathcal{N}_G(E) \subseteq \mathcal{O}_G(E)$$

Thus the following class of languages also characterizes solution to NSC-PO:

$$\mathcal{NRC}_G(E) = \mathcal{N}_G(E) \cap \mathcal{R}_G(E) \cap \mathcal{C}_G(E)$$

Observe that $\mathcal{NRC}_G(E)$ does not provide all of the solutions to NSC-PO.

$$sup\mathcal{NRC}_G(E) = sup\mathcal{NC}_G(sup\mathcal{R}_G(E)) = sup(\mathcal{N}_G(sup\mathcal{R}_G(E)) \cap \mathcal{C}_G(sup\mathcal{R}_G(E)))$$

gives the maximally permissive solution to NSC-PO among the solutions based on normal languages.

2.3 Robust Supervision of Discrete Event Systems

2.3.1 Full Observation

As discussed in Chapter 1, there are different framework for studying robust supervisory control. The framework of [2] is relevant to our work. In [2] the exact model of the plant (though not precisely known) is assumed to be among a finite set of plants. Each of these plants has its own set of alphabets (Σ_i). It is also assumed that all plants agree on the controllability of their common alphabets. In this setting a supervisor (solution for the robust supervisory control problem) is a map:

$$v : \Sigma^* \rightarrow \Gamma_\Sigma,$$

where

$$\Sigma = \bigcup \Sigma_i.$$

All events are assumed to be observable:

$$\Sigma_o = \bigcup \Sigma_{o_i}, \Sigma_o = \Sigma$$

A supervisor for G_i is a map: $v_i : \Sigma_i^* \rightarrow \Gamma_{\Sigma_i}$. In general: $\Sigma_i \neq \Sigma$. Having v, the supervisor v_i can be obtained as:

$$v_i(s) = v(s) \cap \Sigma_i, \quad s \in \Sigma_i^*$$

Robust Nonblocking Supervisory Control problem (RNSC)[2]: Given plants G_i with $L(G_i) \subseteq \Sigma_i^*$ and also legal languages E_i, where $E_i \subseteq L_m(G_i)$ for i in some finite index set $\mathcal{I} = \{1, \ldots, n\}$, synthesize a nonblocking supervisor

$$v : \Sigma^* \rightarrow \Gamma_\Sigma,$$

where $\Sigma = \bigcup \Sigma_i$, such that for all $i \in \mathcal{I}$:

(1) $L_m(v/G_i) \subseteq E_i$.

(2) $\overline{L_m(v/G_i)} = L(v/G_i)$.

The notation of maximally permissiveness is extended to *RNSC* as follows. Let

$$V := \{v |\ v \text{ solves the RNSC problem.}\}$$

A supervisor, $v \in V$ is called **maximally permissive** if and only if for any $v' \in V$ we have:

$$L_m(v'/G_i) \subseteq L_m(v/G_i) \quad (i \in \mathcal{I})$$
$$L(v'/G_i) \subseteq L(v/G_i) \quad (i \in \mathcal{I})$$

[2] discusses the solution of *RNSC*. In this thesis, we study fault recovery in DES. We show that the fault recovery problem under consideration can be transferred to an equivalent robust nonblocking supervisory control under partial observation. We solve the resulting robust control under partial observation. In our study we need the following definition and lemmas.

Definition 1. *[30] Two languages L and M are* ***nonconflicting*** *if and only if*

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

[2] introduces the following family of languages. Given languages E, $L_m(G_i) \subseteq \Sigma^*$ for i in some finite index set $\mathcal{I} = \{1, ..., n\}$, define

$$\mathcal{NF}(E) := \{K \subseteq E \mid (\forall i \in \mathcal{I}) :\ \overline{K \cap L_m(G_i)} = \overline{K} \cap \overline{L_m(G_i)}\}$$

Proposition 6. *[2] $\mathcal{NF}(E)$ is nonempty and closed under arbitrary unions. In particular*

$\mathcal{NF}(E)$ contains a supremal element defined as follows:

$$sup\mathcal{NF}(E) = \bigcup\{K \mid K \in \mathcal{NF}(E)\}$$

Lemma 1 ([9]). *Suppose G_1 and G_2 are DES over the alphabet Σ with*

$$L(G_1) \subseteq L(G_2).$$

If $\upsilon : \Sigma^* \to \Gamma_\Sigma$, then:

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

Lemma 2 ([2]). *Suppose G_1 and G_2 are DES over the alphabet Σ with*

$$L(G_1) \subseteq L(G_2), L_m(G_1) \subseteq L_m(G_2).$$

If $\upsilon : \Sigma^* \to \Gamma_\Sigma$, then

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

Chapter 3

Robust Nonblocking Supervisory Control Problem

As with many other areas of control theory, the notion of robust control has been introduced to discrete event systems in order to cope with situations in which a plant's dynamics are not precisely known. Among different frameworks, the most natural setting is the one in which the true model of the plant is assumed to be among a set of possibilities, none of which can be dismissed as unlikely or impossible. In Chapter 2, we discuss the most recent solution to RNSC in this setting by [2]. Here it is assumed that all events are assumed to be observable. In this chapter we extend results of [2] to control under partial observation where some events are unobservable. Later in Chapter 4, we will use these results to solve the fault recovery problem. Section 3.1 describes the problem formulation. In Section 3.2, the notion of Blocking-invariant languages are introduced which is later used for solving robust control problem. Section 3.3 shows that the class of Blocking-invariant languages posses a supremal element and proposes an algorithm to compute it. While the convergence of the proposed algorithm in the general case is an open problem, in Section 3.3.1, we investigate a case where the languages of the plants involved has a star structure and show that the proposed algorithm terminates in one iteration.

This result will enable us in Chapter 4 to find the solution to the fault recovery problem. Section 3.4 and 3.5 characterizes the solution to RNSC-PO in terms of certain controllable and observable languages.

3.1 Problem Formulation

In this section, we forumulate the problem of Robust Nonblocking Supervisory Control under partial observation, which can be regarded as an extension of the RNSC problem [2] mentioned in Chapter 2. We assume there are n different plants $G_1, \ldots, G_n$ each with its own event set with possibly unobservable events. We assume however, all plants agree on the controllability and observability of their common events. Therefore, the controllable and observable events of G_i are $\Sigma_{c,i} = \Sigma_c \cap \Sigma_i$ and $\Sigma_{o,i} = \Sigma_o \cap \Sigma_i$. For each plant G_i, a language describing the plant's legal marked behavior is also given.

Robust Nonblocking Supervisory Control under Partial Observation(RNSC-PO) Given plants G_i with $L(G_i) \subseteq \Sigma_i^*$, $\Sigma_{c_i} \subseteq \Sigma_i$ and $\Sigma_{o_i} \subseteq \Sigma_i$ and also legal languages E_i, where $E_i \subseteq L_m(G_i)$ for $i \in \mathcal{I} = \{1, \ldots, n\}$, synthesize a nonblocking supervisor

$$v : \Sigma^* \to \Gamma_\Sigma,$$

where

$$\Sigma = \bigcup \Sigma_i,\ \Sigma_o = \bigcup \Sigma_{o_i},\ \Sigma_o \subseteq \Sigma$$

such that:

(1) $L_m(v/G_i) \subseteq E_i, i \in \mathcal{I}$

(2) $\overline{L_m(v/G_i)} = L(v/G_i), i \in \mathcal{I}$

Note that in our setup, a supervisor for G_i is a map

$$v_i : \Sigma_i^* \to \Gamma_{\Sigma_i}$$

In general,

$$\Sigma_i \neq \Sigma$$

Having υ, the supervisor υ_i can obtained as

$$\forall s \in \Sigma_i^* : \ \upsilon_i(s) = \upsilon(s) \cap \Sigma_i$$

Therefore we do not distinguish between υ and υ_i when discussing the supervision of the plant G_i.

3.2 Blocking-Invariant Languages

In this section, we introduce a class of language n-tuples called **Blocking-Invariant Languages** which will be used later to solve the Robust Supervisory Control Problem.

Definition 2. *Assume $M_i, L_i \subseteq \Sigma_i^*$, $L_i \subseteq M_i = \overline{M_i}$ with $i \in \mathcal{I} = \{1, \ldots, n\}$. Then the n-tuple $(L_1, \ldots, L_n)$ is called **Blocking-Invariant (B.I.)** with respect to $(M_1, \ldots, M_n)$ if:*

$$\forall i, j \in \mathcal{I} : \ \overline{L_i} \cap M_j = \overline{L_j} \cap M_i \tag{3.1}$$

Remark 1. *In the robust control problem, L_i's are the marked behaviors and M_i's the closed behaviors of the plants involved. Thus:*

$$L_i = L_m(G_i), \ M_i = L(G_i)$$

Therefore condition 3.1 is

$$\overline{L_m(G_i)} \cap L(G_j) = \overline{L_m(G_j)} \cap L(G_i)$$

which is equivalent to

$$\overline{L_m(G_i)} \cap L(G_j) \subseteq \overline{L_m(G_j)}$$

or

$$\overline{L_m(G_i)} \cap (L(G_j) - \overline{L_m(G_j)}) = \emptyset$$

The above equation states that if a string s generated in G_j ($s \in L(G_j)$) creates blocking, then either it creates blocking in G_i or it can not be generated in G_i at all, thus the term "Blocking Invariance"

Lemma 3. *Blocking invariance (3.1) is equivalent to:*

$$\overline{L_i} \cap \overline{L_j} = \overline{L_i} \cap M_j, \quad \forall i, j \in \mathcal{I} \tag{3.2}$$

Proof. Assume blocking invariance.

$$\overline{L_i} \cap M_j = \overline{L_j} \cap M_i$$
$$\Rightarrow \overline{L_i} \cap M_j \cap \overline{L_j} = \overline{L_j} \cap M_i \cap \overline{L_j}$$
$$\Rightarrow \overline{L_i} \cap \overline{L_j} = \overline{L_j} \cap M_i \text{ (Since } L_j \subseteq M_j)$$

Conversely, assume (3.2) is true and thus:

$$M_j \cap \overline{L_i} = \overline{L_j} \cap \overline{L_i}$$

$$M_i \cap \overline{L_j} = \overline{L_j} \cap \overline{L_i}$$

Therefore:

$$M_j \cap \overline{L_i} = M_i \cap \overline{L_j}$$

□

The following Theorem provides an alternative description of B.I. languages and forms the basis of the computation of the supremal element of the class of **blocking-invariant** language n-tuple.

Theorem 4. *Let L_i's (for $i \in \mathcal{I}$) be as in Definition 2. Then $(L_1, \ldots, L_n)$ is Blocking-Invariant with respect to $(M_1, \ldots, M_n)$ iff:*

$$L_i = L_i - \bigcup_{j \in \mathcal{I}, j \neq i} (M_j - \overline{L_j})\Sigma^*, \quad \forall i \in \mathcal{I} \tag{3.3}$$

where $\Sigma = \bigcup_{i \in \mathcal{I}} \Sigma_i$.

Remark 2. *Note that (3.3) is equivalent to:*

$$L_i \cap \bigcup_{j \in \mathcal{I}, j \neq i} (M_j - \overline{L_j})\Sigma^* = \emptyset, \quad \forall i \in \mathcal{I}$$

Proof. (If) Assume for $i, j \in \mathcal{I}, i \neq j$

$$L_i = L_i - \bigcup_{j \in \mathcal{I}, j \neq i} (M_j - \overline{L_j})\Sigma^*$$

This implies

$$\overline{L_i} \cap (M_j - \overline{L_j}) = \emptyset$$

Since otherwise, assume $s \in \overline{L_i} \cap (M_j - \overline{L_j})$, Then

$$\exists s_0 \in \Sigma^* : ss_0 \in L_i, ss_0 \in (M_j - \overline{L_j})\Sigma^*$$

which contradicts the assumption. Thus we have

$$\overline{L_i} \cap M_j \subseteq \overline{L_j}$$

or

$$\overline{L_i} \cap M_j \subseteq \overline{L_j} \cap \overline{L_i}.$$

Moreover

$$\overline{L_j} \cap \overline{L_i} \subseteq \overline{L_i} \cap M_j$$

Thus

$$\overline{L_j} \cap \overline{L_i} = \overline{L_i} \cap M_j$$

Similarly

$$\overline{L_j} \cap \overline{L_i} = \overline{L_j} \cap M_i$$

(Only if) Assume for $i, j \in \mathcal{I}, i \neq j$

$$\overline{L_i} \cap \overline{L_j} = \overline{L_i} \cap M_j = \overline{L_j} \cap M_i$$

We have to show

$$\forall i \in \mathcal{I} : L_i = L_i - \bigcup_{j \in \mathcal{I}, j \neq i} (M_j - \overline{L_j})\Sigma^*$$

We have

$$L_i - \bigcup_{j \in \mathcal{I}, j \neq i} (M_j - \overline{L_j})\Sigma^* \subseteq L_i$$

Thus we only have to show

$$L_i \subseteq L_i - \bigcup_{j \in \mathcal{I}, j \neq i} (M_j - \overline{L_j})\Sigma^*$$

To do so we show

$$s \in L_i \Rightarrow s \notin (M_j - \overline{L_j})\Sigma^* \tag{3.4}$$

Assume

$$s \in L_i \ \& \ s \in (M_j - \overline{L_j})\Sigma^*$$

Therefore

$$\exists s_0 \in \Sigma^*, s_0 \in \overline{s} : s_0 \in (M_j - \overline{L_j}), s_0 \in \overline{L_i}$$
$$\Rightarrow s_0 \in \overline{L_i} \cap M_j, s_0 \notin \overline{M_i} \cap \overline{L_j}$$

which is not possible because of our assumption. Thus from (3.4) we can conclude

$$L_i = L_i - \bigcup_{j \in \mathcal{I}, j \neq i} (M_j - \overline{L_j})\Sigma^*, \quad \forall i \in \mathcal{I}$$

This completes the proof. □

3.3 Supremal Blocking-Invariant Sublanguages

In this section we will investigate the properties of blocking-invariant languages and show that this class of languages has a supremal element. However first we have to define the lattice of sub-language n-tuple. Recall that

$$L_i, M_i \subseteq \Sigma^*, L_i \subseteq M_i = \overline{M_i} \quad (i \in \mathcal{I})$$

Define:

$$(\Sigma^*)^n := \Sigma^* \times \ldots \times \Sigma^*$$

Also define (component-wise) intersection and union operators via:

$$(K_1^1, \ldots, K_n^1) \cap (K_1^2, \ldots, K_n^2) := (K_1^1 \cap K_1^2, \ldots, K_n^1 \cap K_n^2)$$

$$(K_1^1, \ldots, K_n^1) \cup (K_1^2, \ldots, K_n^2) := (K_1^1 \cup K_1^2, \ldots, K_n^1 \cup K_n^2)$$

Finally define the component-wise set inclusion, denoted by $\subseteq$, with:

$$(K_1^1, \ldots, K_n^1) \subseteq (K_1^2, \ldots, K_n^2) \Leftrightarrow K_1^1 \subseteq K_1^2, \ldots, K_n^1 \subseteq K_n^2$$

Now $(Pwr((\Sigma^*)^n), \subseteq)$ is a poset and under $\cup$ and $\cap$ (as join and meet operations) forms a complete lattice with $\perp = (\emptyset, \ldots, \emptyset)$ and $\top = (\Sigma^*, \ldots, \Sigma^*)$ (as the bottom and top elements)

Now consider the following class of sub language $n-tuple$:

$$\mathcal{L}_{BI}((L_1, \ldots, L_n)) = \{(K_1, \ldots, K_n) : (K_i \subseteq L_i,\ \forall i \in \mathcal{I})\ \&\ (\overline{K_i} \cap M_j = M_i \cap \overline{K_j}, \forall i, j \in \mathcal{I})\}$$

Theorem 5. *(i)* $(L_{BI1}, \ldots, L_{BIn}) := Sup\mathcal{L}_{BI}((L_1, \ldots, L_n))$ *is well-defined.*
(ii) $(L_{BI1}, \ldots, L_{BIn})$ *is the largest fix point of the operator* $\underline{\Omega} : (\Sigma^*)^n \to (\Sigma^*)^n$ *defined with*

$$\underline{\Omega}((K_1, \ldots, K_n)) := (\Omega_1(K_1), \ldots, \Omega_n(K_n))$$

$$\Omega_1(K_1) = K_1 - \bigcup_{j \in \mathcal{I},\, j \neq 1} (M_j - \overline{K_j})\Sigma^*,$$

$$\Omega_i(K_i) = (K_i - \bigcup_{j \in \mathcal{I},\, j > i} (M_j - \overline{K_j})\Sigma^*) - \bigcup_{j \in \mathcal{I},\, j < i} (M_j - \overline{\Omega_j(K_j)})\Sigma^*.$$

Proof. (i) Observe that $\mathcal{L}_{BI}((L_1, \ldots, L_n))$ is nonempty since $(\emptyset, \ldots, \emptyset) \in \mathcal{L}_{BI}((L_1, \ldots, L_n))$. Let $(K_{1,\alpha}, \ldots, K_{n,\alpha}) \in \mathcal{L}_{BI}((L_1, \ldots, L_n))$ for α in some index set $\mathcal{A}$, i.e.,

$$K_{i,\alpha} \subseteq L_i,\ , \forall i \in \mathcal{I} :\ \overline{K_{i,\alpha}} \cap M_j = \overline{K_{j,\alpha}} \cap M_i,\ i, j \in \mathcal{I}$$

Then $\bigcup_\alpha K_{i,\alpha} \subseteq L_i$. Also

$$\begin{aligned}(\overline{\bigcup_{\alpha\in\mathcal{A}} K_{i,\alpha}}) \cap M_j &= (\bigcup_{\alpha\in\mathcal{A}} \overline{K_{i,\alpha}}) \cap M_j \\ &= \bigcup_{\alpha\in\mathcal{A}} (\overline{K_{i,\alpha}} \cap M_j) \\ &= \bigcup_{\alpha\in\mathcal{A}} (\overline{K_{j,\alpha}} \cap M_i) \\ &= (\bigcup_{\alpha\in\mathcal{A}} \overline{K_{j,\alpha}}) \cap M_i \\ &= (\overline{\bigcup_{\alpha\in\mathcal{A}} K_{j,\alpha}}) \cap M_i\end{aligned}$$

Thus we have shown

$$(\bigcup_{\alpha\in\mathcal{A}} K_{1,\alpha}, \ldots, \bigcup_{\alpha\in\mathcal{A}} K_{n,\alpha}) \in \mathcal{L}_{BI}((L_1, \ldots, L_n))$$

which states that $\mathcal{L}_{BI}((L_1, \ldots, L_n))$ is closed under arbitrary unions. Therefore $(L_{BI1}, \ldots, L_{BIn}) := Sup\mathcal{L}_{BI}((L_1, \ldots, L_n))$ is well defined and exists and is given by

$$Sup\mathcal{L}_{BI}((L_1, \ldots, L_n)) = \bigcup\{(K_1, \ldots, K_n) \mid (K_1, \ldots, K_n) \in \mathcal{L}_{BI}((L_1, \ldots, L_n))\}$$

(ii) First we have to show L_{BIi} is a fix point of the operator Ω_i. According to Theorem 4 we have:

$$(L_{BI1}, \ldots, L_{BIn}) \in \mathcal{L}_{BI}((L_1, \ldots, L_n))$$

Therefore

$$\forall i \in \mathcal{I}: \; L_{BIi} = L_{BIi} - \bigcup_{j\in\mathcal{I}, j\neq i} (M_j - \overline{L_{BIj}})\Sigma^* \tag{3.5}$$

Moreover

$$\Omega_1(L_{BI1}) = L_{BI1} - \bigcup_{j\in\mathcal{I},j\neq 1} (M_j - \overline{L_{BIj}})\Sigma^*$$

$$= L_{BI1} \text{ (By (3.5))}$$

Now fix $i \leq n$. Assume

$$\forall j < i : \ \Omega_j(L_{BIj}) = L_{BIj}$$

We show:

$$\Omega_i(L_{BIi}) = L_{BIi}$$

We have

$$\Omega_i(L_{BIi}) = (L_{BIi} - \bigcup_{j\in\mathcal{I},j<i} (M_j - \overline{\Omega_j(L_{BIj})})\Sigma^*) - \bigcup_{j\in\mathcal{I},j>i} (M_j - \overline{L_{BIj}})\Sigma^*$$

$$= (L_{BIi} - \bigcup_{j\in\mathcal{I},j<i} (M_j - \overline{L_{BIj}})\Sigma^*) - \bigcup_{j\in\mathcal{I},j>i} (M_j - \overline{L_{BIj}})\Sigma^* \text{ (Using induction assumption)}$$

$$= L_{BIi} - \bigcup_{j\in\mathcal{I},j\neq i} (M_j - \overline{L_{BIj}})\Sigma^*$$

$$= L_{BIj} \text{ (Using 3.5)}$$

Thus we have shown $(L_{BI1}, \ldots, L_{BIn})$ is a fixed point of the operator $\underline{\Omega}$.

Next we show

$$(K_1, \ldots, K_n) = \underline{\Omega}(K_1, \ldots, K_n) \Rightarrow K_i \subseteq L_{BIi}, \quad i \in \mathcal{I}$$

To do so, it is sufficient to show

$$(K_1, \ldots, K_n) \in \mathcal{L}_{BI}((L_1, \ldots, L_n))$$

By definition of Ω_i we can conclude that:

$$K_i = \Omega_i(K_i)$$

which implies

$$\begin{aligned}
K_i &= (K_i - \bigcup_{j\in\mathcal{I},j<i} (M_j - \overline{\Omega_j(K_j)})\Sigma^*) - \bigcup_{j\in\mathcal{I},j>i} (M_j - \overline{K_j})\Sigma^* \\
&= (K_i - \bigcup_{j\in\mathcal{I},j<i} (M_j - \overline{K_j})\Sigma^*) - \bigcup_{j\in\mathcal{I},j>i} (M_j - \overline{K_j})\Sigma^* \ (K_j \text{ is the fixed point of the operator } \Omega_j) \\
&= K_i - \bigcup_{j\in\mathcal{I},j\neq i} (M_j - \overline{K_j})\Sigma^*
\end{aligned}$$

Thus by Theorem 4 we have:

$$\begin{aligned}
(K_1,\ldots,K_n) \in \mathcal{L}_{BI}((L_1,\ldots,L_n)) \Rightarrow (K_1,\ldots,K_n) &\subseteq Sup\mathcal{L}_{BI}((L_1,\ldots,L_n)) \\
&\subseteq (L_{BI1},\ldots,L_{BIn})
\end{aligned}$$

And this completes the proof. □

Up to this point, we have shown that the class of blocking-invariant languages has a supremal element but we have not yet proposed an algorithm to compute it. Moreover Theorem 5 and 4 suggests that this supremal element could be represented as the largest fixed point of operator Ω. This is the basis of the following algorithm for the computation of the L_{BIi} $(i \in \mathcal{I})$.

Algorithm A:

$$\begin{aligned}
(L_{1,0},\ldots,L_{n,0}) &= (L_1,\ldots,L_n) \\
(L_{1,q+1},\ldots,L_{n,q+1}) &= \underline{\Omega}((L_{1,q},\ldots,L_{n,q})), \quad q \geq 0 \\
M_{i,q} &= M_i, \quad ,\forall i \in \mathcal{I},\ q \geq 0
\end{aligned}$$

Observe that the above algorithm is an iterative algorithm. At iteration q a language n-tuple $(L_{1,q}, \ldots, L_{n,q})$ is generated. The following theorem proves that if Algorithm A converges (in a finite number of steps), it actually obtains $Sup\mathcal{L}_{BI}((L_1, \ldots, L_n))$.

Theorem 6. *Consider Algorithm A.*

(i) $L_{i,\infty} := \lim_{n\to\infty} L_{i,n}$ *exists.*

(ii)$L_{BI_i} \subseteq L_{i,\infty}$, $i \in \mathcal{I}$.

(iii) If Algorithm A converges in a finite number of steps, say q^*, *then*

$$(L_{1,q^*}, \ldots, L_{n,q^*}) = Sup\mathcal{L}_{BI}((L_1, \ldots, L_n)) = (L_{BI1}, \ldots, L_{BIn})$$

Proof. (i) $\{L_{i,n}\}$ is a non-increasing sequence with a lower bound of $\emptyset$. Thus $L_{i,\infty} := \lim_{n\to\infty} L_{i,n}$ exists and $L_{i,\infty} = \bigcap_{n=0}^{\infty} L_{i,n}$.

(ii) We show this by induction. Obviously,

$$L_{BIi} \subseteq L_i = L_{i,0}$$

Now assume $L_{BIi} \subseteq L_{i,j}$. Since $\underline{\Omega}$ is a monotone operator:

$$L_{BIi} = \Omega_i(L_{BIi}) \subseteq \Omega_i(L_{i,j}) = L_{i,j+1}$$

Thus $L_{BIi} \subseteq L_{i,j}$ (for $j \geq 0$) which implies $L_{BIi} \subseteq \bigcap_{j=0}^{\infty} L_{i,j} = L_{i,\infty}$

(iii) Suppose that Algorithm A converges in a finite number of steps, say q^*, then:

$$L_{i,q^*} = L_{i,q^*+1}, \quad (i \in \mathcal{I})$$

Thus it will converge to a fix point of $\underline{\Omega}$, which by Theorem 5-*ii* is the supremal element. □

Example: Consider the plants G_1, G_2 and G_3 in Fig. 3.1. The marked behaviors of these

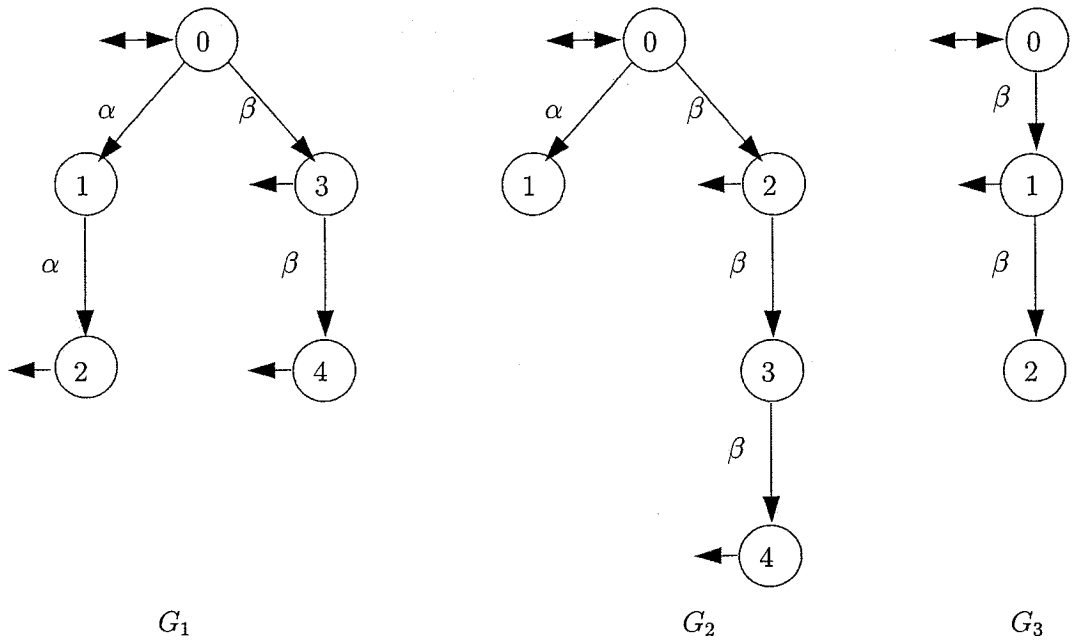


Figure 3.1: G_1, G_2 and G_3

plants do not form a blocking invariant tuple w.r.t their closed behaviors. In this example we apply Algorithm A to the marked behaviors of G_1, G_2, G_3 to find $Sup\mathcal{BI}(L_m(G_1), L_m(G_2), L_m(G_3))$. Following the notation of Algorithm A, we have

$$L_i = L_m(G_i)\ , (i = 1, 2, 3).$$

Let $G_{i,j}$ be finite state generators with

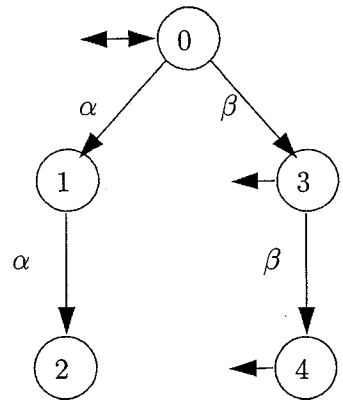
$$L(G_{i,j}) = M_i = L(G_i)$$

and

$$L_m(G_{i,j}) = L_{i,j}$$

where M_i and $L_{i,j}$ are as defined in Algorithm A.

Iteration 1: The action of the operator Ω_1 is to compare $G_{1,0} = G_1$ with each of the rest of



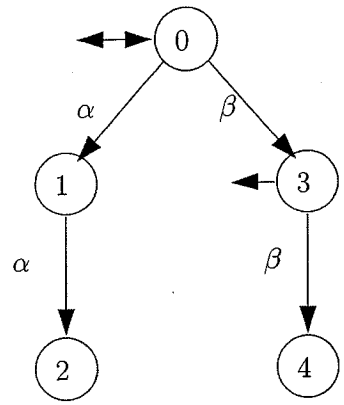
$G_{1,\frac{1}{2}}$

Figure 3.2: $G_{1,\frac{1}{2}}$

possible plant models ($G_{2,0} = G_2$ and $G_{3,0} = G_3$) and remove the strings in the marked behavior that violate blocking invariance. This will be done in 2 steps. At step 1, Ω_1 compares $G_{1,0}$ with $G_{2,0}$. α creates blocking in $G_{2,0}$ but does not cause blocking in $G_{1,0}$. Thus state 2 is unmarked to remove this string and its continuations from $L_m(G_{1,0})$. Now α creates blocking in both of them. Figure 3.2 depicts the resulting plant $G_{1,\frac{1}{2}}$. It should be noted that

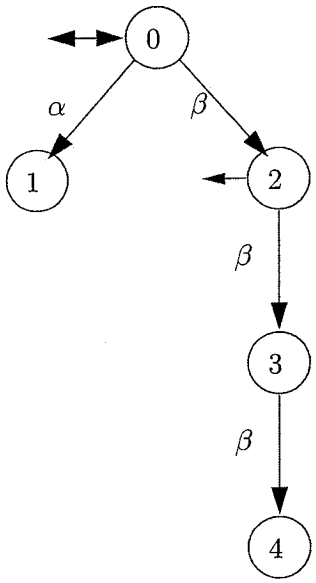
$$L(G_{1,\frac{1}{2}}) = L(G_{1,0}) = L(G_1)$$

At step 2, Ω_1 compares $G_{1,\frac{1}{2}}$ with $G_{3,0}$. $\beta\beta$ creates blocking in $G_{3,0}$ but does not result in blocking in $G_{1,\frac{1}{2}}$. Thus the marking of state 4 is removed in $G_{1,\frac{1}{2}}$ to remove this violating string from the marking of $G_{1,\frac{1}{2}}$. Figure 3.3 depicts the resulting plant, $G_{1,1}$. Next Ω_2 should continue the same procedure for $G_{2,0}$ with respect to $G_{1,1}$ and $G_{3,0}$. This is again done in two steps. At the first level, $G_{2,0}$ is compared with $G_{1,1}$. $\beta\beta$ creates blocking in $G_{1,1}$ but does not generate blocking in $G_{2,0}$. Thus the marking of state 4 in $G_{2,0}$ is changed. In this way, $G_{2,\frac{1}{2}}$ is generated. (Figure 3.4). Next $G_{2,\frac{1}{2}}$ is compared with $G_{3,0}$. It can be verified that there is no string in $G_{2,\frac{1}{2}}$ violating blocking invariance. Finally Ω_3 will follow



$G_{1,1}$

Figure 3.3: $G_{1,1}$



$G_{2,\frac{1}{2}}$

Figure 3.4: $G_{2,\frac{1}{2}}$ and $G_{2,1}$

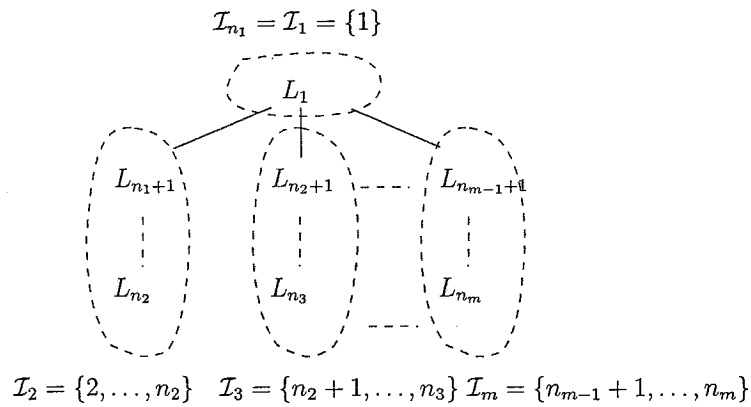


Figure 3.5: Special case of Algorithm A

the same procedure for $G_{3,0}$ and it is not difficult to verify that

$$G_{3,1} = G_{3,0}$$

At this point iteration 1 is finished.

Iteration 2: It could be easily verified that the operator Ω actually does not change the marked behaviors of any of $G_{1,1}, G_{2,1}$ or $G_{3,1}$. Thus the algorithm terminates in this iteration.□

While the convergence of the proposed algorithm in general case is still an open problem, we can establish convergence in a special case where the languages involved have a star structure as defined in the following subsection. The languages involved in the *Fault Recovery problem* studied in Chapter 4, have this configuration and thus the results of the next subsection can be applied to Fault Recovery Problem.

3.3.1 Special case of Algorithm A

In this section we show that in cases where the languages L_i and M_i posses a "star structure", Algorithm A converges in one iteration. The results of this section are applicable to *Fault Recovery Problem* discussed in next chapter. Figure 3.5, demonstrates

the special case which will be considered in the following theorem.

Theorem 7. *Assume the languages L_i (and M_i) can be partitioned into m subsets with index sets $\mathcal{I}_l = \{n_{l-1}+1, \ldots, n_l\}$ with $1 \leq l \leq m$, (Figure 3.5), such that:*

(i) $\mathcal{I}_1 = \{1\}$

(ii) $L_1 \subseteq L_i$, $M_1 \subseteq M_i$, $i \in \mathcal{I}$

(iii) $L_i \subseteq L_{i+1}$, $M_i \subseteq M_{i+1}$, $n_{l-1}+1 \leq i < i+1 \leq n_l$, $2 \leq l \leq m$

(iv) $L_i \cap L_j = L_1$, $M_i \cap M_j = M_1$,

$\forall i, j : n_{l-1}+1 \leq i \leq n_l$, $n_{t-1}+1 \leq j \leq n_t, l, t \in \{2, \ldots, m\}$.

Then we have

$$L_{BI1} = L_1$$

$$L_{BIi} = L_i - (M_1 - \overline{L_1})\Sigma^*, \quad i = n_1+1, n_2+1, n_3+1, \ldots, n_{m-1}+1$$

$$L_{BIi} = L_i - (M_{i-1} - \overline{L_{i-1}})\Sigma^*, \quad n_{l-1}+1 \leq i-1 < i \leq n_l, \ 2 \leq l \leq m$$

Therefore, Algorithm A converges in one step.

Lemma 4. *Assume languages L_1, L_2, M_1 and M_2, satisfy*

$$L_1 \subseteq L_2, \ M_1 \subseteq M_2$$

Then we have:

$$L_1 \cap (M_2 - \overline{L_2})\Sigma^* = \emptyset$$

Proof.

$$s \in (M_2 - \overline{L_2})\Sigma^* \Rightarrow \exists s_1 : s = s_1 s_2, s_1 \in (M_2 - \overline{L_2})$$
$$\Rightarrow s_1 \notin \overline{L_2}$$
$$\Rightarrow s_1 \notin \overline{L_1} \ (since L_1 \subseteq L_2)$$
$$\Rightarrow s \notin L_1$$

Thus

$$L_1 \cap (M_2 - \overline{L_2})\Sigma^* = \emptyset$$

□

Lemma 5. *Let $L_{i,q}, M_i$ and $L_{j,q}, M_j$ for $n_{l-1} + 1 \leq j < i \leq n_l$, $2 \leq l \leq m$ and $q \geq 1$ be as in Algorithm A. Then:*

$$(M_j - \overline{L_{j,q}})\Sigma^* \subseteq (M_i - \overline{L_{i,q}})\Sigma^*$$

Proof.

$$L_{i,q} = (L_{i,q-1} - (M_j - \overline{L_{j,q}})\Sigma^*) - \{\text{other terms}\}$$

Now we have

$$s \in (M_j - \overline{L_{j,q}})\Sigma^* \Rightarrow \exists s_1, s_2 : s = s_1 s_2, s_1 \in (M_j - \overline{L_{j,q}})$$
$$\Rightarrow s_1 \in M_i \text{ (since } M_j \subseteq M_i)$$

Moreover

$$s_1 \notin \overline{L_{i,q}}$$

Since otherwise:

$$\begin{aligned} s_1 \in \overline{L_{i,q}} &\Rightarrow \exists s_3, s_1 s_3 \in L_{i,q} \\ &\Rightarrow s_1 s_3 \notin (M_j - \overline{L_{j,q}})\Sigma^* \text{ (By definition of } L_{i,q}) \\ &\Rightarrow s_1 \notin (M_j - \overline{L_{j,q}}) \end{aligned}$$

which is not the case. Thus we have shown

$$s \in (M_j - \overline{L_{j,q}})\Sigma^*.$$

Thus

$$\begin{aligned} s_1 \notin \overline{L_{i,q}}, s_1 \in M_i &\Rightarrow s_1 \in M_i - \overline{L_{i,q}} \\ &\Rightarrow s \in (M_i - \overline{L_{i,q}})\Sigma^* \end{aligned}$$

□

Lemma 6. *Let $L_{i,q}, M_i$ for $q \geq 1$ be as in Algorithm A. Then for $i \in \mathcal{I}$:*

$$(M_1 - \overline{L_{1,q}})\Sigma^* \subseteq (M_i - \overline{L_{i,q}})\Sigma^*$$

Proof. Similar to the proof of Lemma 5. □

Lemma 7. *Let $L_{i,q}, M_i$ and $L_{j,q}, M_j$ be as in Algorithm A. Then:*

(i) $L_{i,q} \cap L_{j,q} = L_1, \quad n_{l-1} + 1 \leq i \leq n_l,\ n_{t-1} + 1 \leq j \leq n_t, l, t \in \{2, \ldots, m\}, q \geq 0$

(ii) $L_{j,q} \subseteq L_{i,q}, \quad n_{l-1} + 1 \leq j \leq i \leq n_l,\ 2 \leq l \leq mq \geq 0$

Proof. (i) We first show by induction that:

$$L_1 \subseteq L_{i,q}, i \in \mathcal{I}, q \geq 0$$

For $q = 0$, by assumption *(ii)* of Theorem 7, we have:

$$L_1 \subseteq L_{i,0}, i \in \mathcal{I}$$

Now assume for all $i \in \mathcal{I}$ and with $n \geq 1$

$$L_1 \subseteq L_{i,n-1} \tag{3.6}$$

We show

$$L_1 \subseteq L_{i,n} \tag{3.7}$$

To show (3.7), we use induction on i. For $i = 1$, we have:

$$L_{1,n} = L_{1,n-1} - \bigcup_{j>1,\, j\in\mathcal{I}} (M_j - \overline{L_{j,n-1}})\Sigma^*$$

By Lemma 4 and (3.6):

$$(M_j - \overline{L_{j,n-1}})\Sigma^* \cap L_1 = \emptyset \,, j \in \mathcal{I}$$

Thus

$$\bigcup_{j\in\mathcal{I},\, j>1} (M_j - \overline{L_{j,n-1}})\Sigma^* \cap L_1 = \emptyset$$

Therefore,

$$L_1 \subseteq L_{1,n}$$

and the induction statement holds for $i = 1$. Now fix l and assume

$$\forall i, 1 \leq i \leq l-1 : L_1 \subseteq L_{i,n} \tag{3.8}$$

Now we have:

$$L_{l,n} = (L_{l,n-1} - \bigcup_{j<l, j\in\mathcal{I}} (M_j - \overline{L_{j,n}})\Sigma^*) - \bigcup_{j>l, j\in\mathcal{I}} (M_j - \overline{L_{j,n-1}})\Sigma^*)$$

From Lemma 4 and assumption 3.6 we have:

$$\bigcup_{j>l, j\in\mathcal{I}} (M_j - \overline{L_{j,n-1}})\Sigma^* \cap L_1 = \emptyset$$

Also by Lemma 4 and assumption 3.8 we have:

$$\bigcup_{j<l, j\in\mathcal{I}} (M_j - \overline{L_{j,n}})\Sigma^* \cap L_1 = \emptyset$$

Thus

$$L_1 \subseteq L_{l,n}$$

This completes both inductions. Thus we have shown

$$L_1 \subseteq L_{i,q}, i \in \mathcal{I}, q \geq 0 \tag{3.9}$$

Now assuming

$$n_{l-1} + 1 \leq i \leq n_l,\ n_{t-1} + 1 \leq j \leq n_t, l, t \in \{2, \ldots, m\}, q \geq 0$$

(3.9) implies:

$$L_1 \subseteq L_{i,q} \cap L_{i,q}$$

Moreover by assumption *(ii)* of Theorem 7 and the fact that $\underline{\Omega}$ is a contractive map (i.e. $\underline{\Omega}(K_1, \ldots, K_n) \subseteq (K_1, \ldots, K_n)$), we have:

$$L_{i,q} \cap L_{i,q} \subseteq L_i \cap L_j = L_1$$

Thus

$$L_{i,q} \cap L_{i,q} = L_1$$

(ii) We show by induction that for $n_{l-1} + 1 \leq i < i + 1 \leq n_l,\ 2 \leq l \leq m,\ q \geq 0$:

$$L_{i,q} \subseteq L_{i+1,q}$$

The above inequality holds for $q = 0$ by assumption (iii) of Theorem 7. Now assume

$$L_{i,q-1} \subseteq L_{i+1,q-1}$$

We show

$$L_{i,q} \subseteq L_{i+1,q}$$

For this, consider:

$$\begin{aligned} L_{i,q} &= (L_{i,q-1} - \bigcup_{k<i,k\in\mathcal{I}} (M_k - \overline{L_{k,q}})\Sigma^*) - \bigcup_{k>i,k\in\mathcal{I}} (M_k - \overline{L_{k,q-1}})\Sigma^* \\ &= (L_{i,q-1} - (M_{i+1} - \overline{L_{i+1,q-1}})\Sigma^*) - \{\text{other terms}\} \end{aligned}$$

Moreover:

$$\begin{aligned} L_{i+1,q} &= (L_{i+1,q-1} - \bigcup_{k<i+1,k\in\mathcal{I}} (M_k - \overline{L_{k,q}})\Sigma^*) - \bigcup_{k>i+1,k\in\mathcal{I}} (M_k - \overline{L_{k,q-1}})\Sigma^* \\ &= (L_{i+1,q-1} - (M_i - \overline{L_{i,q}})\Sigma^*) - \{\text{other terms}\} \end{aligned}$$

Note that {other terms} in both expressions are the same. Now we have:

$$\begin{aligned} s \in L_{i,q} &\Rightarrow s \in L_{i,q-1} \\ &\Rightarrow s \in L_{i+1,q-1} \text{ (By the assumption of induction)} \\ &\Rightarrow s \notin (M_{i+1} - \overline{L_{i+1,q-1}})\Sigma^* \end{aligned}$$

Moreover:

$$s \in L_{i,q} \Rightarrow s \notin \{\text{other terms}\}$$

Thus we show

$$s \in L_{i,q} \Rightarrow s \in L_{i+1,q-1} \;\&\; s \notin (M_{i+1} - \overline{L_{i+1,q-1}})\Sigma^* \;\&\; s \notin \{\text{other terms}\}$$

Therefore by definition of $L_{i+1,q}$ we have:

$$s \in L_{i,q} \Rightarrow s \in L_{i+1,q}$$

Thus we show

$$L_{i,q} \subseteq L_{i+1,q}, \quad n_{l-1} + 1 \leq i < i+1 \leq n_l,\ 2 \leq l \leq m,\ q \geq 0$$

which implies

$$L_{j,q} \subseteq L_{i,q}, \quad n_{l-1} + 1 \leq j \leq i \leq n_l,\ 2 \leq l \leq m, q \geq 0$$

□

Lemma 8. *Let L_i and M_i's be as in Algorithm A. Then*

(i) for $i \in \{n_{l-1}+1, \ldots, n_l\} = I_l$ and $q \geq 0$, we have:

1. $L_{i,q} - \bigcup_{j<i,\, j \notin I_l} (M_j - \overline{L_{j,q+1}})\Sigma^* = L_{i,q} - (M_1 - \overline{L_1})\Sigma^*$

2. $L_{i,q} - \bigcup_{j>i,\, j\notin I_l} (M_j - \overline{L_{j,q}})\Sigma^* = L_{i,q} - (M_1 - \overline{L_1})\Sigma^*$

(ii) for $i, j \in \{n_{l-1}+1, \ldots, n_l\} = I_l$, $2 \leq l \leq m$, *we have:*

1. $L_{i,q} - \bigcup_{j>i,\, j\in I_l} (M_j - \overline{L_{j,q}})\Sigma^* = L_{i,q}$

2. $(L_{i,q} - \bigcup_{j<i,\, j\in I_l} (M_j - \overline{L_{j,q+1}})\Sigma^*) - \bigcup_{j>i,\, j\in I_l} (M_j - \overline{L_{j,q}})\Sigma^* = L_{i,q} - \bigcup_{j<i,\, j\in I_l} (M_j - \overline{L_{j,q+1}})\Sigma^*$

3. $L_{i,q} - \bigcup_{j<i,\, j\in I_l} (M_j - \overline{L_{j,q+1}})\Sigma^* = L_{i,q} - (M_{i-1} - \overline{L_{i-1,q+1}})\Sigma^*$

Proof. (i)(1) By Lemma 6 for $q \geq 0$, we have:

$$(M_1 - \overline{L_{1,q}})\Sigma^* \subseteq (M_j - \overline{L}_{j,q})\Sigma^* \subseteq (M_j - \overline{L_{j,q+1}})\Sigma^*, \quad j \in \mathcal{I}$$

Moreover

$$L_{1,q+1} \subseteq L_1 \Rightarrow (M_1 - \overline{L_1})\Sigma^* \subseteq (M_1 - \overline{L_{1,q+1}})\Sigma^* \Rightarrow$$

$$(M_1 - \overline{L_1})\Sigma^* \subseteq (M_j - \overline{L_{j,q+1}})\Sigma^*$$

Thus

$$(M_1 - \overline{L_1})\Sigma^* \subseteq \bigcup_{j<i,\, j\notin I_l} (M_j - \overline{L_{j,q+1}})\Sigma^*$$

Therefore

$$L_{i,q} - \bigcup_{j<i,\, j\notin I_l} (M_j - \overline{L_{j,q+1}})\Sigma^* \subseteq L_{i,q} - (M_1 - \overline{L_1})\Sigma^*$$

Next we show

$$L_{i,q} - (M_1 - \overline{L_1})\Sigma^* \subseteq L_{i,q} - \bigcup_{j<i,\, j\notin I_l} (M_j - \overline{L_{j,q+1}})\Sigma^*$$

which is the same as showing:

$$s \in (M_j - \overline{L_{j,q+1}})\Sigma^* (j < i, j \in I_l),\ s \in L_{i,q} \Rightarrow s \in (M_1 - \overline{L_1})\Sigma^*$$

We have

$$s \in (M_j - \overline{L_{j,q+1}})\Sigma^*,\ s \in L_{i,q} \Rightarrow \exists s_1, s_2 : s = s_1 s_2, s_1 \in M_j, s_1 \in M_i, s_1 \in \overline{L_{i,q}}, s_1 \notin \overline{L_{j,q+1}}$$

$$\Rightarrow s_1 \in M_i \cap M_j$$

$$\Rightarrow s_1 \in M_1 \text{ (By assumption (iv) in Theorem 7)} \tag{3.10}$$

Moreover from Lemma 7-i for $n_{l-1}+1 \leq i \leq n_l,\ n_{t-1}+1 \leq j \leq n_t, l, t \in \{2, \ldots, m\}$, we have:

$$\overline{L_1} \subseteq \overline{L_{i,q+1}} \cap \overline{L_{j,q+1}}$$

Also :

$$\overline{L_{i,q+1}} \subseteq \overline{L_{i,q}}$$

Thus:

$$\overline{L_1} \subseteq \overline{L_{i,q}} \cap \overline{L_{j,q+1}}$$

Therefore:

$$s_1 \notin \overline{L_{j,q+1}}, s_1 \in \overline{L_{i,q}} \Rightarrow s_1 \notin \overline{L_1} \tag{3.11}$$

Thus from (3.10) and (3.11) it follows that $s_1 \in M_1 - \overline{L_1}$ and $s \in (M_1 - \overline{L_1})\Sigma^*$. And this completes the proof of *(i)(1)*.

(i)(2) Similar to *(i)(1)*.

(ii)(1) By Lemma 7-ii for $n_{l-1}+1 \leq i \leq j \leq n_l,\ 2 \leq l \leq m, q \geq 0$, we have:

$$L_{i,q} \subseteq L_{j,q}$$

Thus by Lemma 4:

$$L_{i,q} \cap (M_j - \overline{L_{j,q}})\Sigma^* = \emptyset \Rightarrow$$

$$L_{i,q} \cap \bigcup_{j>i, j\in I_l} (M_j - \overline{L_{j,q}})\Sigma^* = \emptyset$$

Therefore:

$$L_{i,q} - \bigcup_{j>i, j\in I_l} (M_j - \overline{L_{j,q}})\Sigma^* = L_{i,q}$$

(ii)(2) We have

$$(L_{i,q} - \bigcup_{j<i, j\in I_l} (M_j - \overline{L_{j,q+1}})\Sigma^* \subseteq L_{i,q}$$

Thus

$$(L_{i,q} - \bigcup_{j<i, j\in I_l} (M_j - \overline{L_{j,q+1}})\Sigma^*) \cap \bigcup_{j>i, j\in I_l} (M_j - \overline{L_{j,q}})\Sigma^* = \emptyset$$

Therefore

$$(L_{i,q} - \bigcup_{j<i, j\in I_l} (M_j - \overline{L_{j,q+1}})\Sigma^*) - \bigcup_{j>i, j\in I_l} (M_j - \overline{L_{j,q}})\Sigma^* = L_{i,q} - \bigcup_{j<i, j\in I_l} (M_j - \overline{L_{j,q+1}})\Sigma^*$$

(ii)(3) Follows from Lemma 5.

This completes the proof of the Lemma.

□

Proof of Theorem 7. We first show:

$$L_{i,1} = L_i - (M_{i-1} - \overline{L_{i-1}})\Sigma^*, \quad n_{l-1} + 1 \leq i - 1 < i \leq n_l,\ 2 \leq l \leq m \tag{3.12}$$

$$L_{i,1} = L_i - (M_1 - \overline{L_1})\Sigma^*, \quad i = n_1 + 1, n_2 + 1, n_3 + 1, \ldots, n_{m-1} + 1 \tag{3.13}$$

and then:

$$L_{i,2} = L_{i,1},\ i \in \mathcal{I}. \tag{3.14}$$

To show (3.12):

$$L_{i,1} = (L_{i,0} - \bigcup_{j\in\mathcal{I}, j<i} (M_j - \overline{L_{j,1}})\Sigma^*) - \bigcup_{j\in\mathcal{I}, j>i} (M_j - \overline{L_{j,0}})\Sigma^*$$

Now there are two possibilities here:

1) $i-1 \geq n_{l-1}+1$: Then by Lemma 8-*(i)(1)* and*(i)(2)*:

$$L_{i,1} =$$

$$((L_{i,0} - \bigcup_{j<i, j\in I_l} (M_j - \overline{L_{j,1}})\Sigma^*) - \bigcup_{j>i, j\in I_l} (M_j - \overline{L_{j,0}})\Sigma^*) - (M_1 - \overline{L_1})\Sigma^*, n_{l-1}+1 \leq j \leq n_l,\ 2 \leq l \leq m$$

$$= (L_{i,0} - \bigcup_{j<i}(M_j - \overline{L_{j,1}}\Sigma^*) - (M_1 - \overline{L_1})\Sigma^* \text{ (By Lemma 8-}\textit{(ii)(2)}\text{)}$$

$$= (L_i - (M_{i-1} - \overline{L_{i-1,1}})\Sigma^*) - (M_1 - \overline{L_1})\Sigma^* \text{ (By Lemma 8-}\textit{(ii)(3)}\text{)}$$

$$= L_i - (M_{i-1} - \overline{L_{i-1,1}})\Sigma^* \text{ (By Lemma 6)}$$

2) $i-1 = n_{l-1}$: Again by Lemma 8-*(i)(1)* and*(i)(2)* :

$$L_{i,1} = ((L_i - \bigcup_{j>i, j\in I_l} (M_j - \overline{L_{j,0}})\Sigma^*)) - (M_1 - \overline{L_1})\Sigma^*,\ n_{l-1}+1 \leq j \leq n_l,\ 2 \leq l \leq m$$

$$= L_i - (M_1 - \overline{L_1})\Sigma^* \text{ (By Lemma 8-}\textit{(ii)(1)}\text{)}$$

Thus we have shown (3.12) and (3.13). To show (3.14) we use induction on i. It follows from Lemma 8-*(i)* and Lemma 4 that

$$L_{1,2} = L_{1,1} = L_1$$

Now assume :

$$L_{k-1,2} = L_{k-1,1}$$

we show

$$L_{k,2} = L_{k,1}$$

There are two scenarios:

1) $n_{l-1}+1 \leq k-1 < k \leq n_l,\ 2 \leq l \leq m$:

$$L_{k,2} = (L_{k,1} - \bigcup_{j\in\mathcal{I},\, j>k} (M_j - \overline{L_{j,1}})\Sigma^*)) - \bigcup_{j\in\mathcal{I},\, j<k} (M_j - \overline{L_{j,2}})\Sigma^*$$

By lemma8-*(i)(1)* and*(i)(2)* for $n_{l-1}+1 \leq j \leq n_l$:

$$\begin{aligned}
&= ((L_{k,1} - \bigcup_{j>k, j\in I_l} (M_j - \overline{L_{j,1}})\Sigma^*) - \bigcup_{j<k, j\in I_l} (M_j - \overline{L_{j,2}})\Sigma^*) - (M_1 - \overline{L_1})\Sigma^*) \\
&= (L_{k,1} - \bigcup_{j<k, j\in I_l} (M_j - \overline{L_{j,2}})\Sigma^*) - (M_1 - \overline{L_1})\Sigma^* \text{ (By Lemma8-}\textit{(ii)(2)}\text{)} \\
&= (L_{k,1} - (M_{k-1} - \overline{L_{k-1,2}})\Sigma^*) - (M_1 - \overline{L_1})\Sigma^* \text{ (By Lemma8- part}\textit{(ii)(3)}\text{)} \\
&= (L_{k,1} - (M_{k-1} - \overline{L_{k-1,1}})\Sigma^*) - (M_1 - \overline{L_1})\Sigma^* \text{ (By the induction assumption)} \\
&= L_{k,1} - (M_{k-1} - \overline{L_{k-1,1}})\Sigma^* \text{ (By Lemma 6)} \\
&= L_{k,1} \text{ (By (3.12))}
\end{aligned}$$

2) $k = n_{l-1}+1,\ 2 \leq l \leq m$

$$L_{k,2} = (L_{k,1} - \bigcup_{j\in\mathcal{I},\, j>k} (M_j - \overline{L_{j,1}})\Sigma^*) - \bigcup_{j\in\mathcal{I},\, j<k} (M_j - \overline{L_{j,2}})\Sigma^*$$

By Lemma 8-*(i)(1)* and*(i)(2)* for j in the branch of the star which starts from k:

$$\begin{aligned}
&= (L_{k,1} - \bigcup_{j>k, j\in I_l} (M_j - \overline{L_{j,2}})\Sigma^*) - (M_1 - \overline{L_1})\Sigma^* \\
&= L_{k,1} - (M_1 - \overline{L_1})\Sigma^* \text{ (By Lemma 8-}\textit{(ii)(1)}\text{)} \\
&= L_{k,1} \text{ (By (3.14))}
\end{aligned}$$

Thus in both scenarios the induction is proved and thus the following is true for $i \in \mathcal{I}$:

$$L_{i,2} = L_{i,1}$$

Therefore the algorithm converges in one step and $(L_{1,1}, \ldots, L_{n_m,1})$ is the fixed point of the operator $\underline{\Omega}$ and by Theorem 5-ii we have

$$L_{BIi} = L_i, \quad i \in \mathcal{I}$$

□

3.4 Application of Blocking-Invariant Languages to RNSC-PO

In this section, we discuss the solution of RNSC-PO. As we will see later in Section 3.5, our solution assume that the marked behaviors of the plants in RNSC-PO are blocking-invariant. The following theorem states that if Algorithm A converges (in a finite number of steps), the RNSC-PO ca be transferred into an equivalent problem in which the marked behaviors of the plants are blocking-invariant.

Theorem 8. *Consider the RNSC-PO problem. Assume that Algorithm A converges in a finite number of steps. Then a supervisor υ solves RNSC-PO if and only if:*

1. $L_m(\upsilon/\widetilde{G}_i) \subseteq \widetilde{E}_i$, *for* $i \in \mathcal{I}$,
2. $\overline{L_m(\upsilon/\widetilde{G}_i)} = L(\upsilon/\widetilde{G}_i)$, *for* $i \in \mathcal{I}$.

Here the plants $\widetilde{G}_i$ are finite-state generators with:

$$(L_m(\widetilde{G}_1), \ldots, L_m(\widetilde{G}_n)) = SupBI(L_m(G_1), \ldots, L_m(G_n))$$

$$L(\widetilde{G}_i) = L(G_i),$$

Blocking-invariance is defined w.r.t $(L(G_1), \ldots, L(G_n))$, *and*

$$E_i' = Sup\mathcal{R}_{G_i}(E_i), \quad i \in \mathcal{I},$$

$$\widetilde{E}_i = E_i' \cap L_m(\widetilde{G}_i), \ i \in \mathcal{I}.$$

Lemma 9. *Assume* $G_{i,q}$ *to be as in Algorithm A. Also assume* υ/G_i *is nonblocking for all* $i \in \mathcal{I}$. *Then for* $i \in \mathcal{I}$ *and* $q \geq 0$*:*

$$L_m(\upsilon/G_{i,q}) = L_m(\upsilon/G_i) \tag{3.15}$$

and $\upsilon/G_{i,q}$ *is nonblocking.*

Proof. We show this by induction. For $q = 0$ we have:

$$\forall i \in \mathcal{I}: \ L_m(G_{i,0}) = L_m(G_i) \Rightarrow L_m(\upsilon/G_{i,0}) = L_m(\upsilon/G_i)$$

Now assume (3.15) is valid for $q = n$. We show it is also valid for $q = n+1$. In other words we want to show

$$\forall i \in \mathcal{I}: \ L_m(\upsilon/G_{i,n+1}) = L_m(\upsilon/G_i) \tag{3.16}$$

To show (3.16) we use induction but this time on i. For $i = 1$ we have

$$\begin{aligned} L_m(\upsilon/G_{1,n+1}) &= L_m(\upsilon/G_{1,n}) \cap L_m(G_{1,n+1}) \text{ (By Lemma 2)} \\ &= L_m(\upsilon/G_{1,n}) \cap (L(G_1) - \bigcup_{i\in\mathcal{I},\, i\neq 1} (L(G_i) - \overline{L_m(G_{i,n})})\Sigma^*) \\ &= L_m(\upsilon/G_{1,n}) - \bigcup_{i\in\mathcal{I},\, i\neq 1} (L(G_i) - \overline{L_m(G_{i,n})})\Sigma^* \end{aligned}$$

Now we show

$$L_m(\upsilon/G_{1,n}) \cap (L(G_i) - \overline{L_m(G_{i,n})})\Sigma^* = \emptyset, \quad , i \in \mathcal{I} - \{1\}$$

If there exists s for which:

$$s \in L_m(\upsilon/G_{1,n}) \;\&\; s \in (L(G_i) - \overline{L_m(G_{i,n})})\Sigma^* \Rightarrow$$

$$\exists s_1, s_2 : s = s_1 s_2, s_1 \in L(G_i), s_1 \notin \overline{L_m(G_{i,n})}, s_1 \in L_m(\upsilon/G_{1,n}) \Rightarrow$$

$$s_1 \in L(\upsilon/G_{1,n}) = L(\upsilon/G_1), s_1 \in L(G_i), s_1 \notin \overline{L_m(\upsilon/G_i)}$$

Moreover we have:

$$L(\upsilon/G_1) \cap L(G_i) \subseteq L(\upsilon/G_i)$$

since

$$\begin{aligned} L(\upsilon/meet(G_1, G_i)) &\subseteq L(\upsilon/G_i) \\ L(\upsilon/meet(G_1, G_i)) &= L(\upsilon/G_1) \cap L(G_1) \cap L(G_i) \\ &= L(\upsilon/G_1) \cap L(G_i) \end{aligned}$$

Thus:

$$s_1 \in L(\upsilon/G_i),$$

which is not possible since υ is a nonblocking supervisor for G_i and we assume $s_1 \in L(G_i) - \overline{L_m(G_i)}$. Hence:

$$L_m(\upsilon/G_{1,n+1}) = L_m(\upsilon/G_{1,n}) = L_m(\upsilon/G_1).$$

Now we have to show if

$$1 \leq i \leq l-1: \; L_m(v/G_{i,n+1}) = L_m(v/G_i) \tag{3.17}$$

then

$$L_m(v/G_{l,n+1}) = L_m(v/G_l).$$

Since we are dealing with two inductions on q and i, we restate the assumptions to avoid ambiguity:

$$L_m(v/G_{i,n}) = L_m(v/G_i) \quad i \in \mathcal{I} \tag{3.18}$$

$$L_m(v/G_{i,n+1}) = L_m(v/G_i) \quad \forall i \leq l-1, i \in \mathcal{I} \tag{3.19}$$

To finalize the induction on i:

$$\begin{aligned} L_m(v/G_{l,n+1}) &= L_m(v/G_{l,n}) \cap L_m(G_{l,n+1}) \\ &= (L_m(v/G_{l,n}) \cap (L_m(G_{l,n}) - \bigcup_{i<l}(L(G_i) - \overline{L_m(G_{i,n+1})})\Sigma^*) - \bigcup_{i>l}(L(G_i) - \overline{L_m(G_{i,n})})\Sigma^*) \\ &= (L_m(v/G_{l,n}) - \bigcup_{i<l}(L(G_i) - \overline{L_m(G_{i,n+1})})\Sigma^*) - \bigcup_{i>l}(L(G_i) - \overline{L_m(G_{i,n})})\Sigma^* \end{aligned}$$

Now we show:

(i) $L_m(v/G_{l,n}) \cap (L(G_i) - \overline{L_m(G_{i,n+1})})\Sigma^* = \emptyset \,, i \in \mathcal{I}, i < l$

(ii) $L_m(v/G_{l,n}) \cap (L(G_i) - \overline{L_m(G_{i,n})})\Sigma^* = \emptyset \,, i \in \mathcal{I}, i > l$

To prove *(i)*, note that:

$$s \in L_m(v/G_{l,n}) \;\&\; s \in (L(G_i) - \overline{L_m(G_{i,n+1})})\Sigma^* \Rightarrow$$

$$\exists s_1, s_2 : s = s_1 s_2, s_1 \in \overline{L_m(v/G_{l,n})}, s_1 \in L(G_i) - \overline{L_m(G_{i,n+1})} \Rightarrow$$

$$s_1 \in L(v/G_{l,n}) \;\&\; s_1 \in L(G_i)$$

Moreover since $L(G_{l,j}) = L(G_l)$, $j \geq 0$:

$$s_1 \in L(v/G_l) \;\&\; s_1 \in L(G_i)$$

Following the same discussion as in the case i=1, we can conclude:

$$s_1 \in L(v/G_i)$$

and since $L(G_{i,n+1}) = L(G_i)$:

$$s_1 \in L(v/G_{i,n+1}).$$

Thus we find a string, s_1, such that:

$$s_1 \in L(v/G_{i,n+1}) \;\&\; s_1 \notin \overline{L_m(G_{i,n+1})}$$

This implies

$$s_1 \notin \overline{L_m(v/G_{i,n+1})}$$

By (3.19):

$$s_1 \notin \overline{L_m(v/G_i)}$$

However this is not possible since $s_1 \in L(v/G_i)$ and v is assumed to be nonblocking for G_i.

Thus:

$$L_m(v/G_{l,n}) \cap (L(G_i) - \overline{L_m(G_{i,n+1})})\Sigma^* = \emptyset \;, i \in \mathcal{I}, i < l$$

Similar to *(i)*, *(ii)* can be shown and hence we have shown

$$L_m(v/G_{l,n+1}) = L_m(v/G_{l,n}) = L_m(v/G_l).$$

This completes the induction on i and q and therefore we have:

$$L_m(\upsilon/G_{i,q}) = L_m(\upsilon/G_i), \quad q \geq 0.$$

Thus:

$$\overline{L_m(\upsilon/G_{i,q})} = \overline{L_m(\upsilon/G_i)} = L(\upsilon/G_i) = L(\upsilon/G_{i,k})$$

and $\upsilon/G_{i,k}$ is nonblocking.

Proof of Theorem 8(If). We assume that there exists a supervisor, υ, which solves the RNSC-PO. Now by Lemma 9 for $k > 0$ we have:

$$L_m(\upsilon/G_{i,k}) = L_m(\upsilon/\widetilde{G}_i).$$

Replacing $k = q^*$ (assuming the algorithm stops in q^* steps):

$$L_m(\upsilon/\widetilde{G}_i) := L_m(\upsilon/G_{i,q^*}) = L_m(\upsilon/\widetilde{G}_i).$$

Moreover $L_m(\upsilon/\widetilde{G}_i)$ must be $L_m(\widetilde{G}_i)$-closed; thus:

$$L_m(\upsilon/\widetilde{G}_i) \subseteq E_i'.$$

Therefore:

$$L_m(\upsilon/\widetilde{G}_i) \subseteq E_i' \cap L_m(\upsilon/\widetilde{G}_i) \subseteq \widetilde{E}_i.$$

Also by Lemma 9, $\upsilon/\widetilde{G}_{i,k}$ is nonblocking for all $i \in \mathcal{I}$, $k \geq 0$ especially $k = q^*$. This completes the proof of the (If) part.

(Only If.) We have to show if there exists a supervisor υ for which $L_m(\upsilon/\widetilde{G}_i) \subseteq \widetilde{E}_i$ and $\overline{L_m(\upsilon/\widetilde{G}_i)} = L(\upsilon/\widetilde{G}_i)$, for $i \in \mathcal{I}$, then υ solves the RNSC-PO. For this, we first show:

$L_m(v/G_i) \subseteq E_i$, which is straightforward according to the following:

$$\begin{aligned}
L_m(v/G_i) &= L(v/G_i) \cap L_m(G_i) \\
&\subseteq \overline{\widetilde{E}_i} \cap L_m(G_i) \\
&\subseteq \overline{\widetilde{E}_i} \cap L_m(\widetilde{G}_i) \\
&\subseteq \overline{E'_i \cap L_m(\widetilde{G}_i)} \cap L_m(\widetilde{G}_i) \\
&\subseteq \overline{E'_i} \cap L_m(\widetilde{G}_i). \\
&= E'_i \quad (\ E'_i \text{ is } L_m(\widetilde{G}_i) - closed) \\
&\subseteq E_i.
\end{aligned}$$

Thus:

$$L_m(v/G_i) \subseteq E_i.$$

Next we show

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

which could be shown as follows:

$$\begin{aligned}
\overline{L_m(v/\widetilde{G}_i)} &\subseteq \overline{L_m(v/G_i)} \\
&\subseteq L(v/G_i) \\
&= L(v/\widetilde{G}_i) \\
&= \overline{L_m(v/\widetilde{G}_i)} \quad \text{(By assumption)}
\end{aligned}$$

Thus:

$$\overline{L_m(v/G_i)} = L(v/G_i).$$

And this completes the proof of Theorem 8.

□

3.5 Solution to RNSC-PO

In the following theorem, we investigate the solution to the RNSC-PO. First a new DES (G) is defined via the union of the behavior of plants G_i. Afterwards a new design specification (E) is introduced which consists of the common stings of the union of each E_i and the complement of $L_m(G_i)$ w.r.t $L_m(G)$. It should be noted that the supervisor v only monitors the observable events and thus :

$$\forall s, s' : \ Ps = Ps' \Rightarrow v(s) = v(s')$$

where $P : \Sigma^* \rightarrow \Sigma_o^*$ is the natural projection.

Theorem 9 (Solution to RNSC-PO). *Let G be a DES defined via:*

$$L_m(G) = \bigcup_{i \in \mathcal{I}} L_m(G_i),$$
$$L(G) = \bigcup_{i \in \mathcal{I}} L(G_i).$$

Moreover define E to be :

$$E = \bigcap_{i \in \mathcal{I}} (E_i \cup (\Sigma^* - L_m(G_i))) \cap L_m(G)$$

Also assume $(L_m(G_1), \ldots, L_m(G_n))$ is blocking-invariant w.r.t $(L(G_1), \ldots, L(G_n))$. Then :

1. *(i) Suppose $K \subseteq E$, $K \neq \emptyset$ and K is*

   (a) Controllable w.r.t G,

   (b) $(L(G), P)$-observable.

   (c) $L_m(G)$ – closed,

   (d) nonconflicting w.r.t $L_m(G_i)$, $\forall i \in \mathcal{I}$.

Then a supervisor, $\widehat{\upsilon}$, exists that solves the RNSC-PO with $L_m(\widehat{\upsilon}/G) = K$ and $L(\widehat{\upsilon}/G) = \overline{K}$.

(ii) For any υ which solves the RNSC-PO, $L_m(\upsilon/G)$ satisfies conditions $(a-d)$ in part(1)(i).

2. *If a supervisor, υ, solves the RNSC-PO then $L_m(\upsilon/G) \subseteq E$.*

3. *If $E \neq \emptyset$ and E satisfies conditions (a-d) in part (1)(i), then (i)a supervisor υ^* exists that solves RNSC-PO with $L_m(\upsilon^*/G) = E$ and $L(\upsilon^*/G) = \overline{E}$. (ii) υ is a maximally permissive solution to RNSC-PO if and only if $L_m(\upsilon/G) = E$ and $L(\upsilon/G) = \overline{E}$.*

Lemma 10. *For any $M \subseteq L_m(G_i)$ we have:*

$$(\forall i, j \in \mathcal{I}): \quad \overline{M} \cap L(G_j) = \overline{M} \cap \overline{L_m(G_j)}.$$

Proof. $(L_m(G_1), \ldots, L_m(G_n))$ is *B.I.* w.r.t $(L(G_1), \ldots, L(G_n))$. Thus by Lemma 3 we have:

$$(\forall i, j \in \mathcal{I}): \quad \overline{L_m(G_i)} \cap L(G_j) = \overline{L_m(G_i)} \cap \overline{L_m(G_j)}$$
$$\overline{L_m(G_i)} \cap L(G_j) \cap \overline{M} = \overline{L_m(G_i)} \cap \overline{L_m(G_j)} \cap \overline{M}$$
$$L(G_j) \cap \overline{M} = \overline{L_m(G_j)} \cap \overline{M} \text{ (since } M \subseteq L_m(G_i))$$

□

Lemma 11. *With $E, L_m(G_i)$ and $L(G_i)$, for $i \in \mathcal{I}$, as in Theorem 9, then for any $K \subseteq E$,*

$$\overline{K} \cap \overline{L_m(G_i)} = \overline{K} \cap L(G_i). \tag{3.20}$$

Proof. Without lost of generality we assume $\mathcal{I} = \{1, 2, 3\}$. It is straightforward that

$$\begin{aligned} E =& (E_1 \cap E_2 \cap E_3) \cup (E_1 \cap E_2 \cap L_m^{co}(G_3)) \cup (E_1 \cap E_3 \cap L_m^{co}(G_2)) \cup (E_2 \cap E_3 \cap L_m^{co}(G_1)) \\ & \cup (E_1 \cap L_m^{co}(G_2) \cap L_m^{co}(G_3)) \cup (E_2 \cap L_m^{co}(G_1) \cap L_m^{co}(G_3)) \cup (E_3 \cap L_m^{co}(G_1) \cap L_m^{co}(G_2)). \end{aligned}$$

and since $E_1 \subseteq L(G_1)$:

$$\begin{aligned} \overline{E} \cap L(G_1) = & (\overline{E_1 \cap E_2 \cap E_3}) \cup (\overline{E_1 \cap E_2 \cap L_m^{co}(G_3)}) \cup (\overline{E_1 \cap E_3 \cap L_m^{co}(G_2)}) \\ & \cup (\overline{E_1 \cap L_m^{co}(G_2) \cap L_m^{co}(G_3)}) \cup (\overline{E_2 \cap E_3 \cap L_m^{co}(G_1)} \cap L(G_1)) \\ & \cup (\overline{E_2 \cap L_m^{co}(G_1) \cap L_m^{co}(G_3)} \cap L(G_1)) \cup (\overline{E_3 \cap L_m^{co}(G_1) \cap L_m^{co}(G_2)} \cap L(G_1)) \end{aligned}$$

Moreover by Lemma 10:

$$\begin{aligned} \overline{L_m(G_2)} \cap L(G_1) = \overline{L_m(G_1)} \cap \overline{L_m(G_2)} \Rightarrow \\ \overline{E_2 \cap E_3 \cap L_m^{co}(G_1)} \cap L(G_1) = \overline{E_2 \cap E_3 \cap L_m^{co}(G_1)} \cap \overline{L_m(G_1)} \end{aligned}$$

Here the last equation follows from $E_2 \cap E_3 \cap L_m^{co}(G_1) \subseteq E_2 \subseteq \overline{L_m(G_2)}$.

The same justification could be done for other terms of $\overline{E} \cap L(G_1)$ which contain $L(G_1)$, so:

$$\overline{E} \cap L(G_1) = \overline{E} \cap \overline{L_m(G_1)}$$

Similarly for $i \in \{2, 3\}$,

$$\overline{E} \cap L(G_i) = \overline{E} \cap \overline{L_m(G_i)}.$$

Now for any $K \subseteq E$ we can say

$$\begin{aligned}\overline{K} \cap L(G_1) &\subseteq \overline{E} \cap L(G_1) \\ &= \overline{E} \cap \overline{L_m(G_1)} \\ &\subseteq \overline{L_m(G_1)} \Rightarrow \\ \overline{K} \cap L(G_1) &\subseteq \overline{K} \cap \overline{L_m(G_1)}\end{aligned}$$

Moreover,

$$\overline{K} \cap \overline{L_m(G_1)} \subseteq \overline{K} \cap L(G_1).$$

Hence,

$$\overline{K} \cap L(G_1) = \overline{K} \cap \overline{L_m(G_1)}.$$

Similarly for $i \in \{2, 3\}$,

$$\overline{K} \cap L(G_i) = \overline{K} \cap \overline{L_m(G_i)}.$$

□

Lemma 12. *Assume υ is a supervisor which solves the* RNSC-PO *with $K_i = L_m(\upsilon/G_i)$, and $K = L_m(\upsilon/G)$. Then:*

$$K = \bigcup_{i \in \mathcal{I}} K_i.$$

Proof.

$$\begin{aligned} K &= L(\upsilon/G) \cap L_m(G) \\ &= L(\upsilon/G) \cap (\bigcup_{i\in\mathcal{I}} L_m(G_i)) \\ &= \bigcup_{i\in\mathcal{I}} (L(\upsilon/G) \cap L_m(G_i)) \\ &= \bigcup_{i\in\mathcal{I}} L_m(\upsilon/G_i) \\ &= \bigcup_{i\in\mathcal{I}} K_i \end{aligned}$$

□

Lemma 13. *Let G be defined as in Theorem 9. Also suppose υ solves the* RNSC-PO *and let*

$$K_i = L_m(\upsilon/G_i),$$
$$K = L_m(\upsilon/G).$$

Then for $i, j \in \mathcal{I}, i \neq j$:

$$\overline{K_j}\Sigma_{uc}^* \cap L(G_i) \subseteq \overline{K_i}\Sigma_{uc}^* \cap L(G_i) \tag{3.21}$$

$$\overline{K_j} \cap L_m(G_i) \subseteq \overline{K_i} \cap L_m(G_i) \tag{3.22}$$

Proof. We will show the results for the case when we have two plants G_1 and G_2. Generalization to n plants would be straightforward. Hence assume,

$$K_1 = L_m(\upsilon/G_1),$$
$$K_2 = L_m(\upsilon/G_2),$$
$$K = L_m(\upsilon/G).$$

Thus:

$$\begin{aligned}
\overline{K_1} = L(v/G_1) &= L(v/G) \cap L(G_1) \Rightarrow \\
\overline{K_1} \cap L(G_2) &= L(v/G) \cap L(G_1) \cap L(G_2) \\
&= L(v/G_2) \cap L(G_1) \\
&\subseteq L(v/G_2) \\
&= \overline{K_2} \\
&= \overline{K_2} \cap L(G_2) \Rightarrow
\end{aligned}$$

$$\overline{K_1}\Sigma_{uc}^* \cap L(G_2) \subseteq \overline{K_2}\Sigma_{uc}^* \cap L(G_2)$$

Similarly

$$\overline{K_2}\Sigma_{uc}^* \cap L(G_1) \subseteq \overline{K_1}\Sigma_{uc}^* \cap L(G_1)$$

Next,

$$\overline{K_1} \cap L(G_2) \subseteq \overline{K_2} \cap L(G_2)$$

Hence by taking the intersection of both sides with $L_m(G_2)$.

$$\overline{K_1} \cap L_m(G_2) \subseteq \overline{K_2} \cap L_m(G_2)$$

Similarly,

$$\overline{K_2} \cap L_m(G_1) \subseteq \overline{K_1} \cap L_m(G_1)$$

And this completes the proof of Lemma. □

Proof of Theorem 9. (1)(i) If K satisfies conditions (a-c) in part i in Theorem 9 then by Theorem 3, there exists a supervisor $\widehat{v}$ for which $L_m(\widehat{v}/G) = K$ and $L(\widehat{v}/G) = \overline{K}$. We have to show $\widehat{v}$ solves the *RNSC-PO*.

Using Lemma 2 :

$$\begin{aligned} L_m(\widehat{v}/G_i) &= L_m(\widehat{v}/G) \cap L_m(G_i) \\ &= K \cap L_m(G_i) \\ &\subseteq E \cap L_m(G_i) \\ &\subseteq E_i \end{aligned}$$

As for non–blocking,

$$\begin{aligned} \overline{L_m(\widehat{v}/G_i)} &= \overline{K \cap L_m(G_i)} \text{ (By Lemma 2)} \\ &= \overline{K} \cap \overline{L_m(G_i)} \text{ (K is nonconflicting w.r.t } L_m(G_i)) \\ &= \overline{K} \cap L(G_i) \text{ (By Lemma 11)} \\ &= L(\widehat{v}/G) \cap L(G_i) \\ &= L(\widehat{v}/G_i) \text{ (By Lemma 1)} \end{aligned}$$

Hence $\widehat{v}$ solves the *RNSC-PO* and this completes the proof for part (i).

(1)(ii) In this part we assume we have two plants $\mathcal{I} = \{1, 2\}$, extension to the general case is straightforward. Suppose,

$$\begin{aligned} K_1 &= L_m(v/G_1), \\ K_2 &= L_m(v/G_2), \\ K &= L_m(v/G). \end{aligned}$$

Thus using Lemma 12

$$K = K_1 \cup K_2.$$

We also have for i=1,2 :

1. K_i is controllable w.r.t G_i.

2. K_i is $(L(G_i), P_i)$-observable.

3. K_i is $L_m(G_i)$−closed.

and we want to show the same properties hold for K w.r.t G and also K is nonconflicting w.r.t $L_m(G_i)$ for $i = 1, 2$. We start by controllability of K w.r.t G:

$$\begin{aligned}\overline{K}\Sigma_{uc}^* \cap L(G) &= (\overline{K_1 \cup K_2})\Sigma_{uc}^* \cap (L(G_1) \cup L(G_2)) \\ &= (\overline{K_1}\Sigma_{uc}^* \cap L(G_1)) \cup (\overline{K_2}\Sigma_{uc}^* \cap L(G_2)) \cup (\overline{K_1}\Sigma_{uc}^* \cap L(G_2)) \cup (\overline{K_2}\Sigma_{uc}^* \cap L(G_1))\end{aligned}$$

Using (3.21):

$$\begin{aligned}\overline{K}\Sigma_{uc}^* \cap L(G) &= (\overline{K_1}\Sigma_{uc}^* \cap L(G_1)) \cup (\overline{K_2}\Sigma_{uc}^* \cap L(G_2)) \\ &\subseteq K_1 \cup K_2 = K.\end{aligned}$$

Next observability of K is investigated. Suppose v solves the *RNSC-PO* then v only monitors the observable events. Thus

$$\forall i, j \in \mathcal{I}: \; s \in L(G_i) \;\&\; s' \in L(G_j) \;\&\; Ps = Ps \Rightarrow v(s) = v(s')$$

Using this fact we show K is observable w.r.t to G. We have to show

$$\forall s, s', \sigma: \; Ps = Ps' \;\&\; s' \in \overline{K} \;\&\; s'\sigma \in L(G) \;\&\; s\sigma \in \overline{K} \Rightarrow s'\sigma \in \overline{K}$$

Four situations are possible for $i, j \in \{1, 2\}, i \neq j$.
1) $s'\sigma \in L(G_i) \;\&\; s\sigma \in \overline{K_i} \;\&\; s' \in \overline{K_i} \Rightarrow s'\sigma \in \overline{K_i}$. Now since K_i is observable w.r.t $L(G_i))$ we have:

$$s'\sigma \in \overline{K}$$

2) $s'\sigma \in L(G_i)$ & $s\sigma \in \overline{K_j} \subseteq L(G_j)$ & $s' \in \overline{K_i} \Rightarrow s' \in L(G_i)$ & $s \in L(G_j)$ & $Ps = Ps'$. This corresponds to a situation in which two strings with the same projection are generated in different plants. As we discussed earlier, we have:

$$v(s) = v(s') \;\&\; \sigma \in v(s) \Rightarrow \sigma \in v(s') \Rightarrow s'\sigma \in \overline{K}$$

3) $s'\sigma \in L(G_i)$ & $s\sigma \in \overline{K_i}$ & $s' \in \overline{K_j} \Rightarrow s \in L(G_i)$ & $\sigma \in v(s) = v(s') \Rightarrow$

$$s'\sigma \in \overline{K_j} \subseteq \overline{K} \Rightarrow s'\sigma \in \overline{K}$$

4) $s'\sigma \in L(G_i)$ & $s\sigma \in \overline{K_j}$ & $s' \in \overline{K_j} \Rightarrow s' \in L(G_i)$ & $s \in L(G_j)$ & $\sigma \in v(s) = v(s') \Rightarrow$

$$s'\sigma \in \overline{K_j} \subseteq \overline{K} \Rightarrow s'\sigma \in \overline{K}$$

As for $L_m(G)$-closedness,

$$\begin{aligned}\overline{K} \cap L_m(G) &= (\overline{K_1} \cup \overline{K_2}) \cap (L_m(G_1) \cup L_m(G_2)) \\ &= (\overline{K_1} \cap L_m(G_1)) \cup (\overline{K_2} \cap L_m(G_2)) \cup (\overline{K_1} \cap L_m(G_2)) \cup (\overline{K_2} \cap L_m(G_1))\end{aligned}$$

By (3.22):

$$\begin{aligned}\overline{K} \cap L_m(G) &= (\overline{K_1} \cap L_m(G_1)) \cup (\overline{K_2} \cap L_m(G_2)) \\ &= K_1 \cup K_2 \\ &= K\end{aligned}$$

Finally we have to show K is nonconflicting w.r.t $L_m(G_1)$ and $L_m(G_2)$.

$$\begin{aligned}\overline{K \cap L_m(G_1)} &= \overline{L_m(\upsilon/G_1)} \text{ (By Lemma 1)} \\ &= L(\upsilon/G_1) \quad (\upsilon \text{ solves the } \textit{RNSC-PO}) \\ &= L(\upsilon/G) \cap L(G_1) \\ &= \overline{K} \cap L(G_1)\end{aligned}$$

Moreover,

$$\overline{K \cap L_m(G_1)} \subseteq \overline{K} \cap \overline{L_m(G_1)} \subseteq \overline{K} \cap L(G_1) \Rightarrow$$
$$\overline{K \cap L_m(G_1)} = \overline{K} \cap \overline{L_m(G_1)}.$$

Similarly

$$\overline{K \cap L_m(G_2)} = \overline{K} \cap \overline{L_m(G_2)}.$$

And this completes the proof of part 1 of Theorem 9.

(2). If υ solves the RSNC-PO then

$$L_m(\upsilon/G_i) \subseteq E_i \text{ (By Theorem 8)} \Rightarrow$$
$$L_m(\upsilon/G) \cap L_m(G_i) \subseteq E_i \Rightarrow$$
$$L_m(\upsilon/G) \subseteq E_i \cup (\Sigma^* - L_m(G_i))$$

Moreover,

$$L_m(\upsilon/G) \subseteq L_m(G)$$

Hence,

$$L_m(\upsilon/G) \subseteq L_m(G) \cap (\bigcap_{i \in \mathcal{I}} E_i \cup (\Sigma^* - L_m(G_i)) = E$$

(3)(i) If E satisfies conditions (a-c) in part (1)(i) then by Theorem 3, there exists a supervisor v^* for which $L_m(v^*/G) = E$ and $L(v^*/G) = \overline{E}$. We have to show v^* solves the *RNSC-PO*.

Using result of Lemma 2 :

$$\begin{aligned} L_m(v^*/G_i) &= L_m(v^*/G) \cap L_m(G_i) \\ &= E \cap L_m(G_i) \\ &= \bigcap_{j \in \mathcal{I}} (E_j \cup (\Sigma^* - L_m(G_j))) \cap L_m(G_i) \\ &\subseteq (E_i \cup (\Sigma^* - L_m(G_i)) \cap L_m(G_i) \\ &= E_i \end{aligned}$$

As for non–blocking,

$$\begin{aligned} \overline{L_m(v^*/G_i)} &= \overline{E \cap L_m(G_i)} \text{ (By Lemma 2)} \\ &= \overline{E} \cap \overline{L_m(G_i)} \text{ (E is nonconflicting w.r.t } L_m(G_i)) \\ &= \overline{E} \cap L(G_i) \text{ (By Lemma 11)} \\ &= L(v^*/G) \cap L(G_i) \\ &= L(v^*/G_i) \text{ (By Lemma 1)} \end{aligned}$$

Hence v^* solves the *RNSC-PO* and this completes the proof for part i.

(3)(ii)(If.) Suppose $L_m(v/G) = E$ then similar to part i, v solves the *RNSC-PO*. Moreover,

according to part 2, for any other supervisor, v' solving the *RNSC-PO*, we have

$$\begin{aligned} L_m(v'/G) &\subseteq E \\ &= L_m(v/G) \Rightarrow \\ L_m(v'/G) \cap L_m(G_i) &\subseteq L_m(v/G) \cap L_m(G_i) \Rightarrow \\ L_m(v'/G_i) &\subseteq L_m(v/G_i) \end{aligned}$$

Hence v is a maximally permissive solution to *RNSC-PO*.

(Only if.) Suppose, v, is a maximally permissive solution to *RNSC-PO*. Also assume $L_m(v^*/G) = E$ with v^* as in part i. Then,

$$\begin{aligned} L_m(v^*/G_i) &\subseteq L_m(v/G_i) \Rightarrow \\ L_m(v^*/G) \cap L_m(G_i) &\subseteq L_m(v/G) \cap L_m(G_i) \Rightarrow \\ E \cap L_m(G_i) &\subseteq L_m(v/G) \cap L_m(G_i) \end{aligned}$$

Hence,

$$\begin{aligned} E \cap (\bigcup_{i \in \mathcal{I}} L_m(G_i)) &\subseteq L_m(v/G) \cap (\bigcup_{i \in \mathcal{I}} L_m(G_i)) \Rightarrow \\ E \cap L_m(G) &\subseteq L_m(v/G) \cap L_m(G) \Rightarrow \\ E &\subseteq L_m(v/G) \end{aligned}$$

Moreover, v is a solution to *RNSC-PO*:

$$L_m(v/G) \subseteq E \text{ (By part 2)}$$

Hence

$$L_m(v/G) = E.$$

And this completes the proof of Theorem 9. □

Remark 3. *Note that in the solution obtained for RNSC-PO, we assume that* $(L_m(G_1), \ldots, L_m(G_n))$ *is B.I w.r.t* $(L(G_1), \ldots, L(G_n)$. *If this assumption does not hold, we may attempt to use Algorithm A to find plant's* $\widetilde{G}_1, \ldots, \widetilde{G}_n$ *satisfying the blocking-invariant assumption. Specially, if Algorithm A converges in a finite number of steps (for example in the case described in Theorem 7), then by Theorem 8, the original robust control problem can be transferred to an equivalent robust control problem satisfying the blocking-invariant assumption. In next chapter, we will use our results on robust control under partial observation to find the solution to the Fault Recovery Problem.*

Chapter 4

Fault Recovery

As the complexity of control systems has increased, concerns for their safe operation and reliability, especially in the presence of failures, have grown. In order to ensure reliability in cases of failure, the controller (supervisor) may take appropriate remedial actions such as control system reconfiguration. In this chapter, we study the problem of the synthesis of fault recovery procedures using discrete–event models.

Fault recovery in DES has been studied by researchers (e.g., [27, 4, 19, 10]). [19] studies fault recovery in DES assuming permanent faults. It is assumed that a diagnosis system is available which detects and isolates faults with a bounded delay. The diagnosis system is modeled as a finite–state automaton that generates a "detection event," within a bounded number of events after a fault occurs. The technique used in the design of the diagnosis system could be either based on continuous–variable models such as differential equations ([13]) or DES–based (e.g., [11]). In other words, the finite–state automaton model is an abstraction of the underlying diagnosis system. One of the advantages of this approach is that the diagnosis and control problems are almost separated, allowing simpler solutions for both problems.

The system to be controlled which consists of the plant and the diagnosis system will have three modes of operation: (i) *normal* when the plant is functioning properly, (ii) *transient*

when a fault has occurred but still not diagnosed by the diagnosis system, and (iii) *recovery*, when the fault is diagnosed (and appropriate actions may have to be taken for recovery). Based on the supervisory control theory of Ramadge–Wonham (RW) [26], [19] proposes a modular switching supervisory control scheme to enforce design specifications in the three modes. The issues of supervisor existence and optimality are not addressed in [19].

In this chapter, we re-examine the problem posed in [19] and manage to characterize the solutions of the problem. For this, we first transfer the problem into an equivalent *Robust Nonblocking Supervisory Control under Partial Observation* and then solve the resulting robust control problem. We obtain the solution to our robust control problem using the results of the previous chapter. We show that the solutions of the fault recovery problem can be characterized in terms of certain controllable and observable languages. Thus we obtain necessary and sufficient conditions for the existence of the supervisor.

Section 4.1 deals with *single-failure scenarios*. In Section 4.1.1 modeling of the plant and diagnoser is described. In Section 4.1.2 the statement of the *fault recovery problem* is given. In Section 4.1.3, we obtain the solutions to the fault recovery problem. Section 4.2 generalizes the results developed in Section 4.1 to *simultaneous-failure* scenarios. Finally in Section 4.3 an illustrative example is provided.

4.1 Single-Failure Scenario

4.1.1 Plant and Diagnoser

We model the plant as a finite state automaton $H = (Q, \Sigma_H, \delta, q_0, Q_m)$. This model describes the behavior of the plant in normal and faulty situations. We assume there are n failure modes $F_1, \ldots, F_n$. Furthermore, we assume:

(1) All failure modes are **permanent**, which means that when a failure occurs the system remains in the faulty condition indefinitely.

(2) **Single–failure scenario**, which means that at most one failure mode may occur at

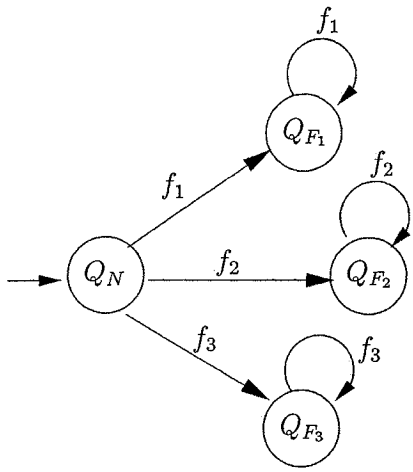


Figure 4.1: Plant with three permanent failure modes (Single-failure scenario)

any time. This assumption will be removed in Section 4.2.

As a result of these assumptions, the plant can be in one of $n+1$ conditions: N (Normal), $F_1, \ldots, F_n$.

Based on these assumptions, we can partition the state set of the plant into normal and faulty states according to: $Q = Q_N \cup Q_{F_1} \cup \ldots \cup Q_{F_n}$. Let $\Sigma_f = \{f_1, \ldots, f_n\}$ be the set of failure events. As a result of a failure event f_i, the system enters failure mode F_i. Failure events are assumed to be unobservable and uncontrollable. We partition the alphabet, Σ_H according to $\Sigma_H = \Sigma_p \cup \Sigma_f$ where as mentioned before, Σ_f is the set of failure events and Σ_p contains the remaining of plant events. Fig. 4.1 shows an example in which the plant has three different permanent failure modes.

We assume that a diagnosis system is available which detects and isolates failures with bounded delay (for example, 3 to 5 events). If a failure f_i is detected and isolated, the diagnoser reports the failure by generating a "diagnosis event" d_i. The set of diagnosis events is denoted by Σ_d. Diagnosis events are assumed observable, but uncontrollable.

We use an automaton $D = (Y, \Sigma_D, \eta, y_0, Y_m)$ to model the diagnosis system. Here $\Sigma_D = \Sigma_H \cup \Sigma_d$. We use the example in Fig. 4.2 to explain the structure of the diagnosis system model. In this example, the plant is assumed to have two permanent failure modes.

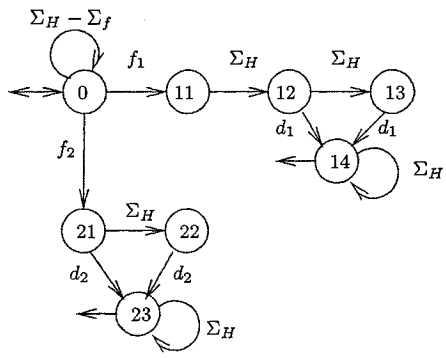


Figure 4.2: Example of Diagnoser Delay Model

The minimum and maximum diagnosis delays for F_1 are assumed to be 1 and 2 events and for F_2, 0 and 1 event. Initially when (by assumption) the plant is in the normal condition and the diagnosis system is in state 0. If f_1 occurs, the system enters the state 11. States 11, 12 and 13 correspond to transient mode when f_1 has not been diagnosed yet. Once f_1 is diagnosed, d_1 is generated and the diagnosis system enters state 14 and recovery may begin. Similarly, states 21 and 22 correspond to the transient mode following the occurrence of f_2. Once f_2 is diagnosed, the diagnosis system enters state 23 and the recovery mode begins. The model used for the diagnosis system is an abstraction of the underlying diagnosis system.The diagnosis system could have been designed based on any technique, as long as the bounds for diagnosis delay are available. In general, these bounds are functions of fault type, plant dynamics and diagnosis technique. One of the advantages of using an abstraction is that it results in the separation of the diagnosis problem and the control problem, thus simplifying the design process. Of course, the time bounds for diagnosis delay have to be computed and since we use an abstraction of the diagnosis system, the approach may be to some extent conservative. For example, in a certain cycle of operation, the diagnosis delays may be smaller than the upper bound used for the design of supervisory control; this may lead to a more conservative control algorithm.
Note that here we assume that after a failure occurs, the plant under supervision generates the minimum number of events necessary for diagnosis (i.e., the minimum diagnosis delay).

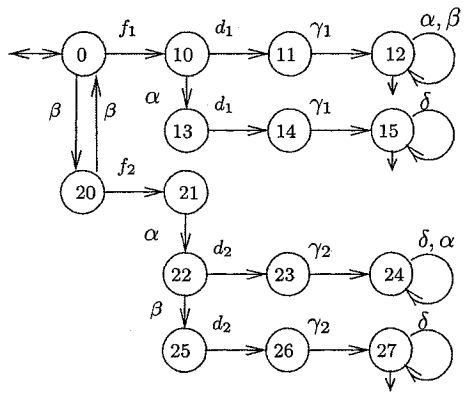


Figure 4.3: Example of G

This assumption holds (trivially) if the minimum diagnosis delay is zero. Another case in which the above assumption is true is where the system is **live** in the sense that it does not have a state at which no events are enabled.[1]

The system to be controlled, G, is the synchronous product of H and D ($G = H \| D$). G has three modes of operation: *(i) Normal (N)* when all components of the system are working properly *(ii) Transient (T)* when a failure occurs in the system but the diagnosis system has not diagnosed it yet and*(iii) Recovery (R)* when the failure is detected and isolated and the corresponding diagnosis event is generated. Accordingly we can partition the states of G into three different blocks corresponding to different modes of operation: $Q^G = Q_N^G \cup Q_T^G \cup Q_R^G$. Moreover Q_T^G and Q_R^G can be further partitioned according to failure modes. Finally let G_N be the subgenerator of G corresponding to states Q_N^G. This generator describes the behavior of the system in normal mode. Also let G_{NT} and G_{NTR} be the subgenerators of G corresponding to $Q_N^G \cup Q_T^G$ and $Q_N^G \cup Q_T^G \cup Q_R^G$ respectively. Also we define sub–generators G_{NT_i} and $G_{NT_iR_i}$ to correspond to $Q_N^G \cup Q_{T_i}^G$ and $Q_N^G \cup Q_{T_i}^G \cup Q_{R_i}^G$ respectively.

Example 1. *Consider a plant with two permanent failures f_1 and f_2. Let the minimum and maximum diagnosis delays be 0 and 1 events for f_1, and 1 and 2 events for f_2. Fig. 4.3*

[1]Timed DES are examples of this case since in a timed DES, the clock keeps generating tick events.

shows the generator $G = H \| D$, *where* $\Sigma_p = \{\alpha, \beta, \delta, \gamma_1, \gamma_2\}, \Sigma_f = \{f_1, f_2\}, \Sigma_d = \{d_1, d_2\}$. *Also,* $Q_N^G = \{0, 20\}$, $Q_{T_1}^G = \{10, 13\}$, $Q_{T_2}^G = \{21, 22, 25\}$, $Q_T^G = Q_{T_1}^G \cup Q_{T_2}^G$, $Q_{R_1}^G = \{11, 12, 14, 15\}$ $Q_{R_2}^G = \{23, 24, 26, 27\}$, *and* $Q_R^G = Q_{R_1}^G \cup Q_{R_2}^G$.

4.1.2 The Fault Recovery Problem

Let E_N ($\subseteq L_m(G_N)$), E_{NT_i}($\subseteq L(G_{NT_i})$) and $E_{NT_iR_i}$($\subseteq L_m(G_{NT_iR_i})$) denote legal languages for the modes $N, T_1, \ldots, T_n$ and $R_1, \ldots, R_n$. The normal specification does not include fault and diagnosis events ($E_N \subseteq \Sigma_p^*$). For each of the transient specifications we have $E_{NT_i} \subseteq (\Sigma_p \cup \{f_i\})^*$ and finally, $E_{NT_iR_i} \subseteq (\Sigma_p \cup \{f_i\} \cup \{d_i\})^*$.

Fault Recovery Problem-Single Failure (*FRP-SF*) Given the plant $G = H \| D$ and legal languages E_N, E_{NT_i} and $E_{NT_iR_i}$ for $i \in \mathcal{I} = \{1, ..., n\}$, synthesize a supervisor $\upsilon : \Sigma^* \rightarrow \Gamma_\Sigma$ such that:

(1) $L_m(\upsilon/G_N) \subseteq E_N$

(2) $L(\upsilon/G_{NT_i}) \subseteq E_{NT_i}$

(3) $L_m(\upsilon/G_{NT_iR_i}) \subseteq E_{NT_iR_i}$

(4) $\overline{L_m(\upsilon/G_N)} = L(\upsilon/G_N)$

(5) $\overline{L_m(\upsilon/G_{NT_iR_i})} = L(\upsilon/G_{NT_iR_i})$.

Note that we require the plant under supervision to be nonblocking in the normal mode and (in case of failure) in the recovery mode. Nonblocking property during the transient mode (for G_{NT_i}) is not required. The reason is that by assumption the failures are diagnosed with a bounded delay and G enters recovery mode (from transient mode) after the execution of a bounded number of events. Also note that E_{NT_i} is assumed to be prefix–closed.

As we will see in next section, *FRP-SF* can be transformed to an equivalent *RNSC-PO*.

4.1.3 Supervisor Synthesis

In this section we use the tools we developed in previous chapter to find the solution to *FRP-SF*. Specifically, we transform the *FRP-SF* to an equivalent *RNSC-PO* problem. Note that *FRP-SF* resembles *RNSC-PO*. The difference is that in *FRP-SF*, nonblocking property is not required in the transient mode and the legal behavior limits the closed behavior (not the marked behavior).

Theorem 10 (*FRP-SF* to *RNSC-PO*). *A supervisor, v, solves the* FRP-SF *if and only if:*

1. $L_m(v/G_N) \subseteq E_N$
2. $L_m(v/\widehat{G}_{NT_i}) \subseteq \widehat{E}_{NT_i}$
3. $L_m(v/\widehat{G}_{NT_iR_i}) \subseteq \widehat{E}_{NT_iR_i}$
4. $\overline{L_m(v/G_N)} = L(v/G_N)$
5. $\overline{L_m(v/\widehat{G}_{NT_i})} = L(v/\widehat{G}_{NT_i})$
6. $\overline{L_m(v/\widehat{G}_{NT_iR_i})} = L(v/\widehat{G}_{NT_iR_i})$.

Here,

$$
\begin{aligned}
L(\widehat{G}_{NT_i}) &= L(G_{NT_i}),\\
L_m(\widehat{G}_{NT_i}) &= L_m(G_N) \cup (\overline{L_m(G_N)} f_i \Sigma^* \cap L(G_{NT_i})),\\
L(\widehat{G}_{NT_iR_i}) &= L(G_{NT_iR_i}),\\
L_m(\widehat{G}_{NT_iR_i}) &= L_m(\widehat{G}_{NT_i}) \cup L_m(G_{NT_iR_i}),\\
\widehat{E}_{NT_i} &= E_{NT_i} \cap L_m(\widehat{G}_{NT_i}),\\
\widehat{E}_{NT_iR_i} &= E'_{NT_iR_i} \cup (\overline{E'_{NT_iR_i}} \cap L_m(\widehat{G}_{NT_i})),\\
E'_{NT_iR_i} &= Sup\mathcal{R}_{G_{NT_iR_i}}(E_{NT_iR_i}).
\end{aligned}
$$

Lemma 14. *Assume $\widehat{E}_{NT_iR_i}$ and $L_m(\widehat{G}_{NT_iR_i})$ are defined as in Theorem 10. Then:*

$$\overline{\widehat{E}_{NT_iR_i}} \cap L_m(\widehat{G}_{NT_iR_i}) = \widehat{E}_{NT_iR_i}.$$

Proof.

$$\begin{aligned}
\overline{\widehat{E}_{NT_iR_i}} \cap L_m(\widehat{G}_{NT_iR_i}) &= (\overline{E'_{NT_iR_i}} \cup \overline{E'_{NT_iR_i} \cap L_m(\widehat{G}_{NT_i})}) \cap (L_m(\widehat{G}_{NT_i}) \cup L_m(G_{NT_iR_i})) \\
&= \overline{E'_{NT_iR_i}} \cap (L_m(\widehat{G}_{NT_i}) \cup L_m(G_{NT_iR_i})) \quad (\text{ Since } \overline{E'_{NT_iR_i} \cap L_m(\widehat{G}_{NT_i})} \subseteq \overline{E'_{NT_iR_i}}) \\
&= (\overline{E'_{NT_iR_i}} \cap L_m(\widehat{G}_{NT_i})) \cup (\overline{E'_{NT_iR_i}} \cap L_m(G_{NT_iR_i})) \\
&= (\overline{E'_{NT_iR_i}} \cap L_m(\widehat{G}_{NT_i})) \cup E'_{NT_iR_i} \quad (\text{ Since } E'_{NT_iR_i} \text{ is } L_m(\widehat{G}_{NT_iR_i})\text{-closed}) \\
&= \widehat{E}_{NT_iR_i}.
\end{aligned}$$

□

Lemma 15. *Let $G_{NT_iR_i}$ and $\widehat{G}_{NT_iR_i}$ be defined as in Theorem 10. Moreover assume υ satisfies conditions (1-6) in Theorem 10. Then:*

$$\overline{L_m(\upsilon/\widehat{G}_{NT_iR_i})} = \overline{L_m(\upsilon/G_{NT_iR_i})}.$$

Proof. We have to show

$$\overline{L_m(\upsilon/\widehat{G}_{NT_iR_i})} \subseteq \overline{L_m(\upsilon/G_{NT_iR_i})}. \tag{4.1}$$

$$\overline{L_m(\upsilon/G_{NT_iR_i})} \subseteq \overline{L_m(\upsilon/\widehat{G}_{NT_iR_i})}. \tag{4.2}$$

(4.2) follows from $L_m(G_{NT_iR_i}) \subseteq L_m(\widehat{G}_{NT_iR_i})$. To show (4.1), we need to show:

$$\forall s \in \overline{L_m(\upsilon/\widehat{G}_{NT_iR_i})} : s \in \overline{L_m(\upsilon/G_{NT_iR_i})}$$

For $s \in \overline{L_m(\upsilon/\widehat{G}_{NT_iR_i})}$ there exists

$$s_0 \in \Sigma^* : ss_0 \in L_m(\upsilon/\widehat{G}_{NT_iR_i}).$$

Due to our problem formulation, ss_0 could have one of the following forms:

1) $ss_0 \in \Sigma_p^*$

2) $ss_0 \in \Sigma_p^* f_i(\Sigma_p \cup \{f_i\})^*$

3) $ss_0 \in \Sigma_p^* f_i(\Sigma_p \cup \{f_i\})^* d_i(\Sigma_p \cup \{f_i\})^*$.

However if ss_0 is in the first or third group, then:

$$ss_0 \in L_m(\widehat{G}_{NT_iR_i}) \Leftrightarrow ss_0 \in L_m(G_{NT_iR_i})$$

Since

$$L_m(\upsilon/G_{NT_iR_i}) = L(\upsilon/G_{NT_iR_i}) \cap L_m(G_{NT_iR_i}),$$

$$L_m(\upsilon/\widehat{G}_{NT_iR_i}) = L(\upsilon/G_{NT_iR_i}) \cap L_m(\widehat{G}_{NT_iR_i}),$$

We have

$$ss_0 \in L_m(\upsilon/\widehat{G}_{NT_iR_i}) \Leftrightarrow ss_0 \in L_m(\upsilon/G_{NT_iR_i})$$

As a result, to show (4.1) we have to show

$$ss_0 \in \Sigma_p^* f_i(\Sigma_p \cup \{f_i\})^*, ss_0 \in L_m(\upsilon/\widehat{G}_{NT_iR_i}) \Rightarrow s \in \overline{L_m(\upsilon/G_{NT_iR_i})}$$

Since $ss_0 \in L_m(\upsilon/\widehat{G}_{NT_iR_i})$ we conclude that

$$ss_0 \in L(\upsilon/\widehat{G}_{NT_iR_i}) \Rightarrow ss_0 \in L(\upsilon/G_{NT_iR_i})$$

By assumption, in case of failure, the diagnosis event will occur with bounded delay, say

D_{max}. Therefore:

$$\exists s_d \in (\Sigma_p \cup \{f_i\})^* d_i : ss_0 s_d \in L(\upsilon/G_{NT_iR_i}) = L(\upsilon/\widehat{G}_{NT_iR_i}) = \overline{L_m(\upsilon/\widehat{G}_{NT_iR_i})}$$

Based on our previous discussion on the strings in group (3), we have

$$ss_0 s_d \in \overline{L_m(\upsilon/G_{NT_iR_i})}$$

which implies

$$s \in \overline{L_m(\upsilon/G_{NT_iR_i})}$$

And this completes the proof of Lemma 15. □

Proof of Theorem 10. We show

$$[L_m(\upsilon/\widehat{G}_{NT_i}) \subseteq \widehat{E}_{NT_i} \;\&\; \overline{L_m(\upsilon/\widehat{G}_{NT_i})} = L(\upsilon/\widehat{G}_{NT_i})] \Leftrightarrow L(\upsilon/G_{NT_i}) \subseteq E_{NT_i}$$

$$L_m(\upsilon/\widehat{G}_{NT_iR_i}) \subseteq \widehat{E}_{NT_iR_i} \Leftrightarrow L_m(\upsilon/G_{NT_iR_i}) \subseteq E_{NT_iR_i}$$

$$\overline{L_m(\upsilon/\widehat{G}_{NT_iR_i})} = L(\upsilon/\widehat{G}_{NT_iR_i}) \Leftrightarrow \overline{L_m(\upsilon/G_{NT_iR_i})} = L(\upsilon/G_{NT_iR_i})$$

(If) We have

$$\begin{aligned}
L(G_{NT_i}) &= L(\widehat{G}_{NT_i}) \Rightarrow \\
L(\upsilon/G_{NT_i}) &= L(\upsilon/\widehat{G}_{NT_i}) \\
&\subseteq \overline{\widehat{E}_{NT_i}} \\
&\subseteq \overline{E_{NT_i} \cap L_m(\widehat{G}_{NT_i})} \\
&\subseteq \overline{E_{NT_i}} \\
&\subseteq E_{NT_i} \text{ (Since } E_{NT_i} \text{ is assumed to be prefix closed)}
\end{aligned}$$

Moreover

$$\begin{aligned}
L_m(\upsilon/G_{NT_iR_i}) &= L_m(\upsilon/\widehat{G}_{NT_iR_i}) \cap L_m(G_{NT_iR_i}) \\
&\subseteq \widehat{E}_{NT_iR_i} \cap L_m(G_{NT_iR_i}) \\
&= (E'_{NT_iR_i} \cup (\overline{E'_{NT_iR_i}} \cap L_m(\widehat{G}_{NT_i}))) \cap L_m(G_{NT_iR_i}) \\
&= (E'_{NT_iR_i} \cap L_m(G_{NT_iR_i})) \cup (\overline{E'_{NT_iR_i}} \cap L_m(G_{NT_iR_i}) \cap L_m(\widehat{G}_{NT_i})) \\
&\subseteq (E'_{NT_iR_i} \cap L_m(G_{NT_iR_i})) \cup (E'_{NT_iR_i} \cap L_m(\widehat{G}_{NT_i})) \ (E'_{NT_iR_i} \text{ is } L_m(G_{NT_iR_i}) - \textit{closed}) \\
&\subseteq E'_{NT_iR_i} \\
&\subseteq E_{NT_iR_i}.
\end{aligned}$$

Next we show

$$\overline{L_m(\upsilon/G_{NT_iR_i})} = L(\upsilon/G_{NT_iR_i})$$

By Lemma 15:

$$\begin{aligned}
\overline{L_m(\upsilon/G_{NT_iR_i})} &= \overline{L_m(\upsilon/\widehat{G}_{NT_iR_i})} \\
&= L(\upsilon/\widehat{G}_{NT_iR_i}) \\
&= L(\upsilon/G_{NT_iR_i})
\end{aligned}$$

And this completes the proof for the (If) part.

(Only If) In this part we assume a supervisor υ exists which solves the *FRP-SF* problem. Thus:

$$\begin{aligned}
L_m(\upsilon/\widehat{G}_{NT_i}) &= L(\upsilon/\widehat{G}_{NT_i}) \cap L_m(\widehat{G}_{NT_i}) \\
&= L(\upsilon/G_{NT_i}) \cap L_m(\widehat{G}_{NT_i}) \\
&\subseteq E_{NT_i} \cap L_m(\widehat{G}_{NT_i}) \\
&= \widehat{E}_{NT_i}
\end{aligned}$$

Moreover:

$$\begin{aligned}
L_m(\upsilon/\widehat{G}_{NT_iR_i}) &= L(\upsilon/G_{NT_iR_i}) \cap L_m(\widehat{G}_{NT_iR_i}) \\
&= (L(\upsilon/G_{NT_iR_i}) \cap L_m(G_{NT_iR_i})) \cup (L(\upsilon/G_{NT_iR_i}) \cap L_m(\widehat{G}_{NT_i})) \\
&= L_m(\upsilon/G_{NT_iR_i}) \cup (L(\upsilon/G_{NT_iR_i}) \cap L_m(\widehat{G}_{NT_i})) \\
&\subseteq E'_{NT_iR_i} \cup (\overline{E'_{NT_iR_i}} \cap L_m(\widehat{G}_{NT_i})) \\
&= \widehat{E}_{NT_iR_i}
\end{aligned}$$

As for non blocking:

$$\begin{aligned}
\overline{L_m(\upsilon/\widehat{G}_{NT_i})} &= \overline{L(\upsilon/\widehat{G}_{NT_i}) \cap L_m(\widehat{G}_{NT_i})} \\
&= \overline{(L(\upsilon/\widehat{G}_{NT_i}) \cap L_m(G_N))} \cup \overline{(L(\upsilon/\widehat{G}_{NT_i}) \cap (\overline{L_m(G_N)} f_i \Sigma^* \cap L(G_{NT_i})))}
\end{aligned}$$

υ is a nonblocking supervisor for G_N, thus:

$$\begin{aligned}
\overline{L_m(\upsilon/\widehat{G}_{NT_i})} &= (L(\upsilon/\widehat{G}_{NT_i}) \cap L(G_N)) \cup \overline{(L(\upsilon/\widehat{G}_{NT_i}) \cap (\overline{L_m(G_N)} f_i \Sigma^* \cap L(G_{NT_i})))} \\
&= (L(\upsilon/\widehat{G}_{NT_i}) \cap L(G_N)) \cup \overline{(L(\upsilon/\widehat{G}_{NT_i}) \cap \overline{L_m(G_N)} f \Sigma^*)} \text{ (Since } L(\upsilon/\widehat{G}_{NT_i}) \subseteq L(G_{NT_i}))
\end{aligned}$$

Observe that:

$$L(\upsilon/\widehat{G}_{NT_i}) \cap L(G_N) = \{s \in L(\upsilon/\widehat{G}_{NT_i}) \mid \text{ s does not contain failure event } \text{“}f_i\text{”}\}.$$

Also since υ/G_N is nonblocking

$$L(\upsilon/\widehat{G}_{NT_i}) \cap L(G_N) \subseteq \overline{L_m(G_N)}$$

Thus

$$L(\upsilon/\widehat{G}_{NT_i}) \cap \overline{L_m(G_N)} f_i \Sigma^* = \{s \in L(\upsilon/\widehat{G}_{NT_i}) \mid \text{s contains at least one failure event "}f_i\text{"}\}$$

Which results in:

$$\overline{L_m(\upsilon/\widehat{G}_{NT_i})} = L(\upsilon/\widehat{G}_{NT_i})$$

Also:

$$\overline{L_m(\upsilon/G_{NT_iR_i})} = L(\upsilon/G_{NT_iR_i}) = L(\upsilon/\widehat{G}_{NT_iR_i}).$$

Moreover:

$$\overline{L_m(\upsilon/G_{NT_iR_i})} \subseteq \overline{L_m(\upsilon/\widehat{G}_{NT_iR_i})} \subseteq L(\upsilon/\widehat{G}_{NT_iR_i}).$$

Hence:

$$\overline{L_m(\upsilon/\widehat{G}_{NT_iR_i})} = L(\upsilon/\widehat{G}_{NT_iR_i})$$

And this completes the proof of Theorem 10. □

Theorem 10 transfers the *FRP-SF* to an *RNSC-PO*. Hence we can find a solution for this problem using Theorem 9. The result is stated in the following theorem.

Theorem 11 (Solution of *FRP-SF*). *Let G be defined via:*

$$L_m(G) = \bigcup_{i \in \mathcal{I}} L_m(\widetilde{G}_{NT_iR_i}),$$

$$L(G) = \bigcup_{i \in \mathcal{I}} L(\widetilde{G}_{NT_iR_i}) \quad .$$

Also let E be defined via:

$$E = \left(\left[E_N \cup L_m^{co}(G_N)\right] \cap \left[\bigcap_{i \in \mathcal{I}} \left(\widetilde{E}_{NT_i} \cup L_m^{co}(\widetilde{G}_{NT_i})\right)\right] \cap \left[\bigcap_{i \in \mathcal{I}} \left(\widetilde{E}_{NT_iR_i} \cup L_m^{co}(\widetilde{G}_{NT_iR_i})\right)\right]\right) \cap L_m(G)$$

where

$$
\begin{aligned}
L(\widetilde{G}_{NT_i}) &= L(\widehat{G}_{NT_i}),\\
L_m(\widetilde{G}_{NT_i}) &= L_m(\widehat{G}_{NT_i}),\\
L(\widetilde{G}_{NT_iR_i}) &= L(G_{NT_iR_i}),\\
L_m(\widetilde{G}_{NT_iR_i}) &= L_m(\widetilde{G}_{NT_i}) \cup (\overline{L_m(\widetilde{G}_{NT_i})}d_i\Sigma^* \cap L_m(G_{NT_iR_i})),\\
\widetilde{E}_{NT_i} &= \widehat{E}_{NT_i} = E_{NT_i} \cap L_m(\widetilde{G}_{NT_i}),\\
\widetilde{E}_{NT_iR_i} &= (E'_{NT_iR_i} \cap L_m(\widetilde{G}_{NT_iR_i})) \cup (\overline{E'_{NT_iR_i}} \cap L_m(\widetilde{G}_{NT_i})),\\
E'_{NT_iR_i} &= Sup\mathcal{R}_{G_{NT_iR_i}}(E_{NT_iR_i}).
\end{aligned}
$$

Then :

1. *(i) Suppose $K \subseteq E$, $K \neq \emptyset$ and K is:*

   (a) Controllable w.r.t G

   (b) $(L(G), P)$-observable

   (c) $L_m(G)$ – closed

   (d) Nonconflicting w.r.t $L_m(G_N)$ and all $L_m(\widetilde{G}_{NT_iR_i})$, $L_m(\widetilde{G}_{NT_i})$ $(i \in \mathcal{I})$.

   Then a supervisor, $\widehat{v}$, exists that solves FRP-SF *with $L_m(\widehat{v}/G) = K$ and $L(\widehat{v}/G) = \overline{K}$.*

   (ii) For any v which solves FRP-SF*, $L_m(v/G)$ satisfies conditions (a-d) in part (1)(i).*

2. *If a supervisor, v, solves* FRP-SF *then $L_m(v/G) \subseteq E$.*

3. *If $E \neq \emptyset$ and also E satisfies conditions (a-d) in part (1)(i), then (i) a supervisor, v^*, exists that solves* FRP-SF *with $L_m(v^*/G) = E$ and $L(v^*/G) = \overline{E}$ (ii) v is a maximally permissive solution to the* FRP-SF *if and only if $L_m(v/G) = E$ and $L(v/G) = \overline{E}$*

Proof. We show this theorem assuming single-failure mode ($\Sigma_f = \{f_1\}$). Generalization to n failures would be straightforward. For simplicity we drop index 1 from the formulas. Theorem 9 assumes B.I., so first we use Algorithm A to construct generators G_N, $\widetilde{G}_{NT}$ and $\widetilde{G}_{NTR}$ such that:

$$(L_m(G_N), L_m(\widetilde{G}_{NT}), L_m(\widetilde{G}_{NTR})) = Sup\mathcal{BI}(L_m(G_N), L_m(\widehat{G}_{NT}), L_m(\widehat{G}_{NTR}))$$

To calculate $Sup\mathcal{BI}$, let:

$$G_1 = G_N,\ G_2 = \widehat{G}_{NT},\ G_3 = \widehat{G}_{NTR}$$
$$\mathcal{I} = \{1,2,3\},\ \mathcal{I}_1 = \{1\}, \mathcal{I}_2 = \{2,3\}$$
$$L_1 = L_m(G_N), L_2 = L_m(\widehat{G}_{NT}), L_3 = L_m(\widehat{G}_{NTR})$$
$$M_1 = L(G_N), M_2 = L(\widehat{G}_{NT}), M_3 = L(\widehat{G}_{NTR})$$

Moreover it is obvious that:

$$L_1 = L_m(G_N) \subseteq L_m(\widehat{G}_{NT}) = L_2,$$
$$L_2 = L_m(\widehat{G}_{NT}) \subseteq L_m(\widehat{G}_{NTR}) = L_3,$$
$$M_1 = L(G_N) \subseteq L(\widehat{G}_{NT}) = M_2,$$
$$M_2 = L(\widehat{G}_{NT}) \subseteq L(\widehat{G}_{NTR}) = M_3$$

In other words L_i's and M_i's satisfy the conditions in Theorem 7. Thus Algorithm A converges in one step and the expressions given for $L_m(\widetilde{G}_{NT})$ and $L_m(\widetilde{G}_{NTR})$ in the statement of Theorem 11 are obtained from that theorem. Note that by Remark 3, since Algorithm A converges, the RNSC-PO resulted from Theorem 8 with

$$(L_m(G_N), L_m(\widetilde{G}_{NT}), L_m(\widetilde{G}_{NTR})) = Sup\mathcal{ND}(L_m(G_N), L_m(\widehat{G}_{NT}), L_m(\widehat{G}_{NTR}))$$

is equivalent to the original one with $(L_m(G_N), L_m(\widehat{G}_{NT}), L_m(\widehat{G}_{NTR}))$. Moreover in the new RNSC-PO, the assumption of B.I is satisfied and thus Theorem 9 could be utilized. To apply Theorem 8 we must calculate $\widetilde{E}_{NT}$ and $\widetilde{E}_{NTR}$. The formula for $\widetilde{E}_{NT}$ is obvious. For $\widetilde{E}_{NTR}$ we have by Theorem 8:

$$\widetilde{E}_{NTR} = Sup\mathcal{R}_{\widehat{G}_{NTR}}(\widehat{E}_{NTR}) \cap L_m(\widetilde{G}_{NTR})$$

Moreover by Lemma 14:

$$\widetilde{E}_{NTR} = \widehat{E}_{NTR} \cap L_m(\widetilde{G}_{NTR})$$

After substituting for $\widehat{E}_{NTR}$:

$$\widetilde{E}_{NTR} = (E'_{NTR} \cap L_m(\widetilde{G}_{NTR})) \cup (\overline{E_{NTR}} \cap L_m(\widetilde{G}_{NT})).$$

Now Theorem 11 follows from Theorem 9 with

$$E_1 = E_N,\ E_2 = \widetilde{E}_{NT},\ E_3 = \widetilde{E}_{NTR}$$
$$G_1 = G_N,\ G_2 = \widetilde{G}_{NT},\ G_3 = \widetilde{G}_{NTR}.$$

□

4.2 Simultaneous-Failure Scenario

4.2.1 Plant and Diagnoser

In this section we generalize the results of Section 4.1 to the case of simultaneous-failure scenarios. Similar to single-failure scenario, the system to be controlled has three modes of operation: *(i) Normal (N)* when all components of the system are working properly, *(ii) Transient(T)* when there is at least one failure which has occurred but *not* diagnosed yet

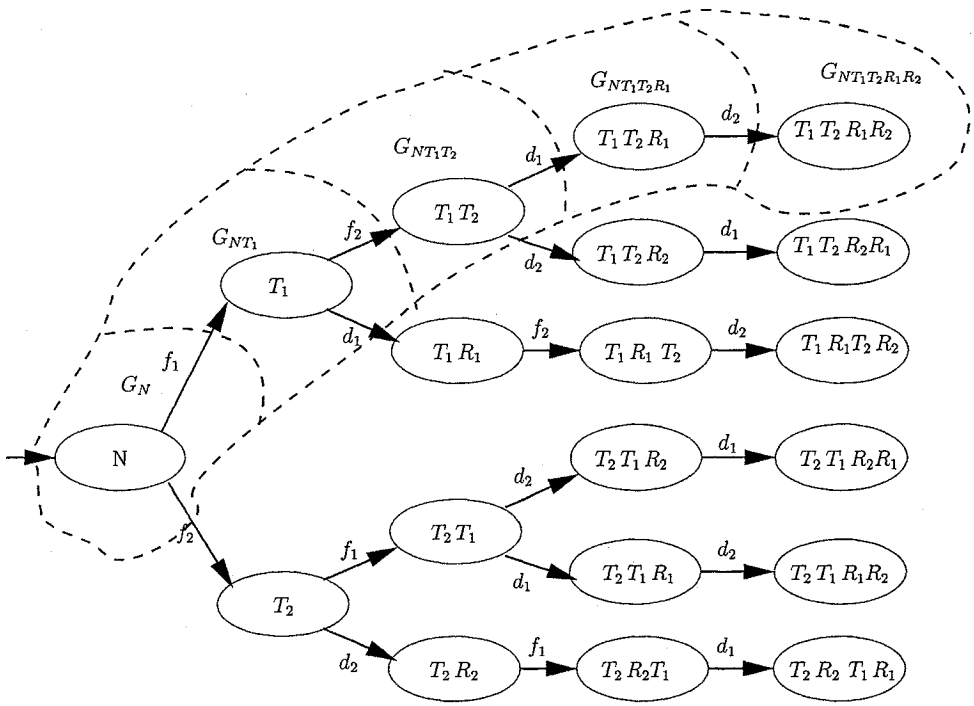


Figure 4.4: Multiple Failure Scenario

and *(iii) Recovery* when all failures occurred have been diagnosed by the diagnoser. Accordingly we can partition the state space of the system to be controlled into three different blocks: $Q^G = Q_N^G \cup Q_T^G \cup Q_R^G$. Moreover Q_T^G and Q_R^G can be further partitioned according to the failure and diagnosis event(s) occurred in the system. For example, $Q_{T_1T_2}^G$ are those faulty states reached after first f_1 and then f_2 has occurred and $Q_{T_2T_1}^G$ includes those faulty states in which the occurrence of f_2 precedes the occurrence of f_1.

Corresponding to this partitioning of states, subgenerators of G could be defined e.g. $G_{NT_1T_2}$ corresponding to $Q_N^G \cup Q_{T_1T_2}^G$ and $G_{NT_2T_1}$ corresponding to $Q_N^G \cup Q_{T_2T_1}^G$. Fig 4.4 illustrates these subgenerators assuming a plant with failure events f_1 and f_2.

For design specification, we assume for each of these subgenerators a design specification in the form of legal marked language is present. Furthermore, the system under supervision is required to be nonblocking in recovery and normal modes.

4.2.2 The Fault Recovery Problem

Here for brevity, we assume the system has two failure events f_1, f_2 (Figure 4.4). Extension to more failures is straightforward. However it could be noted that the number of subgenerators in the general case would increase exponentially in the number of the failure events and the problem may become very complex. In practice, though simultaneous occurrence of say more than 3 failures is rare specially when failures are independent.

Fault Recovery Problem- Two Simultaneous Failure(FRP-TSF) Given the plant $G = H \| D$ with two failures f_1 and f_2 ($\Sigma_f = \{f_1, f_2\}$), and legal languages E_N, E_{NT_i}, $E_{NT_iT_j}$, $E_{NT_iR_i}$, $E_{NT_iR_iT_j}$, $E_{NT_iT_jR_i}$, $E_{NT_iT_jR_j}$, $E_{NT_iT_jR_iR_j}$, $E_{NT_iT_jR_jR_i}$, $E_{NT_iR_iT_jR_j}$ for $i, j = 1, 2$ and $i \neq j$, synthesize a supervisor $\upsilon : \Sigma^* \rightarrow \Gamma_\Sigma$ such that (for $i, j \in \{1, 2\}$ and $i \neq j$)

1. $L_m(\upsilon/G_N) \subseteq E_N$
2. $L(\upsilon/G_{NT_i}) \subseteq E_{NT_i}$,
3. $L(\upsilon/G_{NT_iT_j}) \subseteq E_{NT_iT_j}$,
4. $L_m(\upsilon/G_{NT_iR_i}) \subseteq E_{NT_iR_i}$,
5. $L(\upsilon/G_{NT_iR_iT_j}) \subseteq E_{NT_iR_iT_j}$,
6. $L(\upsilon/G_{NT_iT_jR_i}) \subseteq E_{NT_iT_jR_i}$,
7. $L(\upsilon/G_{NT_iT_jR_j}) \subseteq E_{NT_iT_jR_j}$,
8. $L_m(\upsilon/G_{NT_iT_jR_iR_j}) \subseteq E_{NT_iT_jR_iR_j}$,
9. $L_m(\upsilon/G_{NT_iT_jR_jR_i}) \subseteq E_{NT_iT_jR_jR_i}$,
10. $L_m(\upsilon/G_{NT_iR_iT_jR_j}) \subseteq E_{NT_iR_iT_jR_j}$
11. $L(\upsilon/G_N) = \overline{L_m(\upsilon/G_N)}$

12. $L(\upsilon/G_{NT_iR_i}) = \overline{L_m(\upsilon/G_{NT_iR_i})}$

13. $L(\upsilon/G_{NT_iT_jR_iR_j}) = \overline{L_m(\upsilon/G_{NT_iT_jR_iR_j})}$

14. $L(\upsilon/G_{NT_iT_jR_jR_i}) = \overline{L_m(\upsilon/G_{NT_iT_jR_jR_i})}$

15. $L(\upsilon/G_{NT_iR_iT_jR_j}) = \overline{L_m(\upsilon/G_{NT_iR_iT_jR_j})}$

4.2.3 Supervisor Synthesis

We will follow the same approach as in Section 4.1 and transform the *FRP-TSF* to a *RNSC-PO* whose solutions are characterized by Theorem 9.

Theorem 12 (*FRP-TSF* into *RNSC-PO*). *A supervisor, υ, solves the FRP-TSF if and only if:*

1. $L_m(\upsilon/G_N) \subseteq E_N$

2. $L_m(\upsilon/\widehat{G}_{NT_i}) \subseteq \widehat{E}_{NT_i}$,

3. $L_m(\upsilon/\widehat{G}_{NT_iT_j}) \subseteq \widehat{E}_{NT_iT_j}$,

4. $L_m(\upsilon/\widehat{G}_{NT_iR_i}) \subseteq \widehat{E}_{NT_iR_i}$,

5. $L_m(\upsilon/\widehat{G}_{NT_iR_iT_j}) \subseteq \widehat{E}_{NT_iR_iT_j}$,

6. $L_m(\upsilon/\widehat{G}_{NT_iT_jR_i}) \subseteq \widehat{E}_{NT_iT_jR_i}$,

7. $L_m(\upsilon/\widehat{G}_{NT_iT_jR_j}) \subseteq \widehat{E}_{NT_iT_jR_j}$,

8. $L_m(\upsilon/\widehat{G}_{NT_iT_jR_iR_j}) \subseteq \widehat{E}_{NT_iT_jR_iR_j}$,

9. $L_m(\upsilon/\widehat{G}_{NT_iT_jR_jR_i}) \subseteq \widehat{E}_{NT_iT_jR_jR_i}$,

10. $L_m(\upsilon/\widehat{G}_{NT_iR_iT_jR_j}) \subseteq \widehat{E}_{NT_iR_iT_jR_j}$

11. $L(\upsilon/G_N) = \overline{L_m(\upsilon/G_N)}$

12. $L(\upsilon/\widehat{G}_{NT_iR_i}) = \overline{L_m(\upsilon/\widehat{G}_{NT_iR_i})}$

13. $L(\upsilon/\widehat{G}_{NT_iT_jR_iR_j}) = \overline{L_m(\upsilon/\widehat{G}_{NT_iT_jR_iR_j})}$

14. $L(\upsilon/\widehat{G}_{NT_iT_jR_jR_i}) = \overline{L_m(\upsilon/\widehat{G}_{NT_iT_jR_jR_i})}$

15. $L(\upsilon/\widehat{G}_{NT_iR_iT_jR_j}) = \overline{L_m(\upsilon/\widehat{G}_{NT_iR_iT_jR_j})}$

Here,

$$L(\widehat{G}_{NT_i}) = L(G_{NT_i}),$$

$$L(\widehat{G}_{NT_iR_i}) = L(G_{NT_iR_i}),$$

$$L(\widehat{G}_{NT_iR_iT_j}) = L(G_{NT_iR_iT_j})$$

$$L(\widehat{G}_{NT_iR_iT_jR_j}) = L(G_{NT_iR_iT_jR_j})$$

$$L(\widehat{G}_{NT_iT_j}) = L(G_{NT_iT_j})$$

$$L(\widehat{G}_{NT_iT_jR_i}) = L(G_{NT_iT_jR_i})$$

$$L(\widehat{G}_{NT_iT_jR_j}) = L(G_{NT_iT_jR_j})$$

$$L(\widehat{G}_{NT_iT_jR_iR_j}) = L(G_{NT_iT_jR_iR_j})$$

$$L(\widehat{G}_{NT_iT_jR_jR_i}) = L(G_{NT_iT_jR_jR_i})$$

$$L_m(\widehat{G}_{NT_i}) = L_m(G_N) \cup (\overline{L_m(G_N)} f_i \Sigma^* \cap L(\widehat{G}_{NT_i})),$$

$$L_m(\widehat{G}_{NT_iT_j}) = L_m(\widehat{G}_{NT_i}) \cup (\overline{L_m(\widehat{G}_{NT_i})} f_j \Sigma^* \cap L(\widehat{G}_{NT_iT_j}))$$

$$L_m(\widehat{G}_{NT_iT_jR_i}) = L_m(\widehat{G}_{NT_iT_j}) \cup L_m(G_{NT_iT_jR_i}))$$

$$L_m(\widehat{G}_{NT_iT_jR_iR_j}) = L_m(\widehat{G}_{NT_iT_jR_i}) \cup L_m(G_{NT_iT_jR_iR_j}))$$

$$L_m(\widehat{G}_{NT_iT_jR_j}) = L_m(\widehat{G}_{NT_iT_j}) \cup L_m(G_{NT_iT_jR_j}))$$

$$L_m(\widehat{G}_{NT_iT_jR_jR_i}) = L_m(\widehat{G}_{NT_iT_jR_j}) \cup L_m(G_{NT_iT_jR_jR_i}))$$

$$
\begin{aligned}
L_m(\widehat{G}_{NT_iR_i}) &= L_m(\widehat{G}_{NT_i}) \cup L_m(G_{NT_iR_i}),\\
L_m(\widehat{G}_{NT_iR_iT_j}) &= L_m(\widehat{G}_{NT_iR_i}) \cup (\overline{L_m(\widehat{G}_{NT_iR_i})} f_j \Sigma^* \cap L(\widehat{G}_{NT_iR_iT_j}))\\
L_m(\widehat{G}_{NT_iR_iT_jR_j}) &= L_m(\widehat{G}_{NT_iR_iT_j}) \cup L_m(G_{NT_iR_iT_jR_j})\\
\widehat{E}_{NT_i} &= E_{NT_i} \cap L_m(\widehat{G}_{NT_i}),\\
\widehat{E}_{NT_iT_j} &= E_{NT_iT_j} \cap L_m(\widehat{G}_{NT_iT_j}),\\
\widehat{E}_{NT_iR_i} &:= E'_{NT_iR_i} \cup (\overline{E'_{NT_iR_i}} \cap L_m(\widehat{G}_{NT_i})),\\
E'_{NT_iR_i} &= Sup\mathcal{R}_{G_{NT_iR_i}}(E_{NT_iR_i})\\
\widehat{E}_{NT_iT_jR_i} &:= E'_{NT_iT_jR_i} \cup (\overline{E'_{NT_iT_jR_i}} \cap L_m(\widehat{G}_{NT_iT_j})),\\
E'_{NT_iT_jR_i} &= Sup\mathcal{R}_{G_{NT_iT_jR_i}}(E_{NT_iT_jR_i})\\
\widehat{E}_{NT_iT_jR_j} &:= E'_{NT_iT_jR_j} \cup (\overline{E'_{NT_iT_jR_j}} \cap L_m(\widehat{G}_{NT_iT_j})),\\
E'_{NT_iT_jR_j} &= Sup\mathcal{R}_{G_{NT_iT_jR_j}}(E_{NT_iT_jR_j})\\
\widehat{E}_{NT_iT_jR_iR_j} &:= E'_{NT_iT_jR_iR_j} \cup (\overline{E'_{NT_iT_jR_iR_j}} \cap L_m(\widehat{G}_{NT_iT_jR_i})),\\
E'_{NT_iT_jR_iR_j} &= Sup\mathcal{R}_{G_{NT_iT_jR_iR_j}}(E_{NT_iT_jR_iR_j})\\
\widehat{E}_{NT_iT_jR_jR_i} &:= E'_{NT_iT_jR_jR_i} \cup (\overline{E'_{NT_iT_jR_jR_i}} \cap L_m(\widehat{G}_{NT_iT_jR_j})),\\
E'_{NT_iT_jR_jR_i} &= Sup\mathcal{R}_{G_{NT_iT_jR_jR_i}}(E_{NT_iT_jR_jR_i})\\
\widehat{E}_{NT_iR_iT_j} &:= E_{NT_iR_iT_j} \cap L_m(\widehat{G}_{NT_iR_iT_j}),\\
\widehat{E}_{NT_iR_iT_jR_j} &:= E'_{NT_iR_iT_jR_j} \cup (\overline{E'_{NT_iR_iT_jR_j}} \cap L_m(\widehat{G}_{NT_iR_iT_jR_j})),\\
E'_{NT_iR_iT_jR_j} &= Sup\mathcal{R}_{G_{NT_iR_iT_jR_j}}(E_{NT_iR_iT_jR_j}).
\end{aligned}
$$

Proof. The proof is done by repeated application of Theorem 10 to appropriate defined

plants as follows. As part of Theorem 10 it was shown that

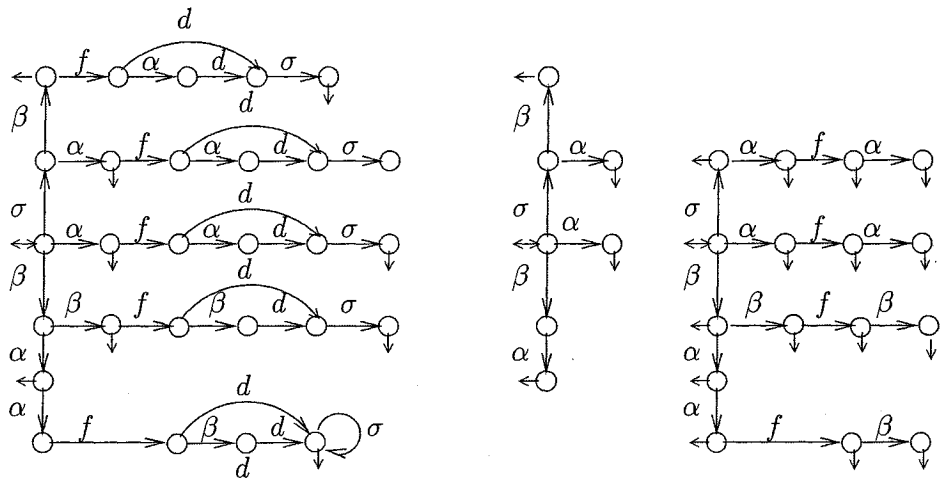
$$\begin{cases} L_m(\upsilon/G_N) \subseteq E_N \\ \overline{L_m(\upsilon/G_N)} = L(\upsilon/G_N) \\ L(\upsilon/G_{NT}) \subseteq E_{NT} \end{cases}$$

is equivalent to

$$\begin{cases} L_m(\upsilon/G_N) \subseteq E_N \\ \overline{L_m(\upsilon/G_N)} = L(\upsilon/G_N) \\ L_m(\upsilon/\widehat{G}_{NT}) \subseteq \widehat{E}_{NT} \\ \overline{L_m(\upsilon/\widehat{G}_{NT})} = L(\upsilon/\widehat{G}_{NT}) \end{cases} \tag{4.3}$$

Expressions for $\widehat{G}_{NT_i}$ and $\widehat{E}_{NT_i}$ in Theorem 12 follow from this equivalence applied to the pari $\{(G_N, E_N), (G_{NT_i}, E_{NT_i})\}$. Next if we consider $\widehat{G}_{NT_i}$ as the normal plant and $G_{NT_iT_j}$ as the transient plant for an FRP problem and applying the transformation in (4.3) to $\{(\widehat{G}_{NT_i}, \widehat{E}_{NT_i}), (G_{NT_iT_j}, E_{NT_iT_j})\}$ we obtain equivalent RNSC problem with $\{(\widehat{G}_{NT_i}, \widehat{E}_{NT_i}), (\widehat{G}_{NT_iT_j}, \widehat{E}_{NT_iT_j})\}$. Following this procedure by assuming $\widehat{G}_{NT_iT_j}$ as normal plant and $G_{NT_iT_jR_i}$ (resp. $G_{NT_iT_jR_j}$) as transient plant, expressions for $\widehat{G}_{NT_iT_jR_i}$ and $\widehat{E}_{NT_iT_jR_i}$ (resp. $\widehat{G}_{NT_iT_jR_j}$ and $\widehat{E}_{NT_iT_jR_j}$) will follow. Finally by Theorem 10, FRP with the pairs
$\{(\widehat{G}_{NT_iT_j}, \widehat{E}_{NT_iT_j}), (\widehat{G}_{NT_iT_jR_i}, \widehat{E}_{NT_iT_jR_i}), (G_{NT_iT_jR_iR_j}, E_{NT_iT_jR_iR_j}))\}$ is equivalent to RNSC-PO for $\{(\widehat{G}_{NT_iT_j}, \widehat{E}_{NT_iT_j}), (\widehat{G}_{NT_iT_jR_i}, \widehat{E}_{NT_iT_jR_i}), (\widehat{G}_{NT_iT_jR_iR_j}, \widehat{E}_{NT_iT_jR_iR_j}))\}$. This technique could be applied to $\{(G_N, E_N), (\widehat{G}_{NT_i}, \widehat{E}_{NT_i}), (G_{NT_iR_i}, E_{NT_iR_i})\}$ to find the expressions for $\widehat{G}_{NT_iR_i}$ and $\widehat{E}_{NT_iRT_i}$. By continuing this method, all other expressions in the statement of Theorem 12 can be shown. □

Up to this point, FRP-TSF is transferred to a RNSC-PO. However, to use Theorem 9, B.I. assumption should be satisfied. Repeated application of Theorem 8 (Section 3.4) to an



$G_{NTR}, E_{NTR} = L_m(G_{NTR})$ E_N E_{NT}

Figure 4.5: G_{NTR}, E_N, E_{NT} and E_{NTR} in Example with one failure mode

equivalent RNSC-PO problem in which the blocking-invariance condition holds. First we apply Theorem 8 to the plants G_N,$\widehat{G}_{NT_1}$ and $\widehat{G}_{NT_2}$ to obtain an equivalent problem involving G_N, $\widetilde{G}_{NT_1}$ and $\widetilde{G}_{NT_2}$. Note that G_N, $\widetilde{G}_{NT_1}$ and $\widetilde{G}_{NT_2}$ form the "star-structure" and thus the convergence of Algorithm A is guaranteed by Theorem 7. Secondly, note that in the transformation G_N does not change (since in Theorem 7, L_1 and M_1 do not change). Next we apply Theorem 7 to $\widetilde{G}_{NT_1}, \widehat{G}_{NT_1T_2}, \widehat{G}_{NT_1R_1}, \widehat{G}_{NT_1R_1T_2}$ and $\widehat{G}_{NT_1R_2T_2R_2}$ to obtain an equivalent problem with $\widetilde{G}_{NT_1}, \widetilde{G}_{NT_1T_2}, \widetilde{G}_{NT_1R_1}, \widetilde{G}_{NT_1R_1T_2}$ and $\widetilde{G}_{NT_1R_2T_2R_2}$. Once again note that $\widetilde{G}_{NT_1}, \widehat{G}_{NT_1T_2}, \widehat{G}_{NT_1R_1}, \widehat{G}_{NT_1R_1T_2}$ and $\widehat{G}_{NT_1R_2T_2R_2}$ form the required star structure. The procedure is repeated until we arrive at an equivalent RNSC-PO problem satisfying the blocking invariance condition.

4.3 Example

Fig. 4.5 shows the generator for G_{NTR}. Single failure mode is assumed ($\Sigma_f = \{f\}$). Diagnoser diagnoses the fault with a delay of 0 to 1 event. $\Sigma_p = \{\alpha, \beta, \sigma\}$, $\Sigma_f = \{f\}$,

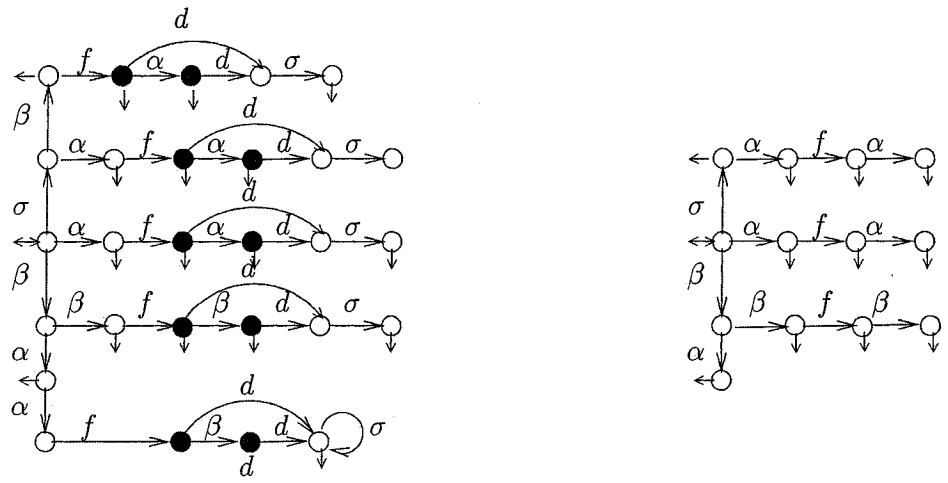


Figure 4.6: $\widehat{G}_{NTR}, \widehat{E}_{NTR}$ and $\widehat{E}_{NT}$ in the procedure of supervisor synthesis

$\Sigma_{uc} = \{f, d, \alpha\}$ and $\Sigma_o = \Sigma - \{f\}$. E_{NTR} is assumed to be the same as G_{NTR} meaning that there is no restriction on the behavior of G_{NTR} in recovery mode (Figure 4.5). Normal and transient legal behaviors, E_N and E_{NT}, are also depicted in Fig. 4.5. Following the procedure for supervisor synthesis, we first have to transfer the problem into an RNSC-PO problem. Figure 4.6 depicts the resulting plants. In this example, $\widehat{G}_{NTR}$ is actually obtained by changing the marking of the faulty states (Black circles in Figure 4.6). Here the rule to change the marking of black states is as follows: if the string that reaches the black state from initial state does not generate blocking in normal mode mark the black state. As a result, the black states in the lowest branch are not marked. E_{NT} and E_{NTR} are altered similarly. The resulting robust control problem does not satisfy blocking invariance assumption. For example, $\beta\alpha\alpha \in \overline{L_m(G_{NTR})} \cap L(G_N)$ but $\beta\alpha\alpha \notin \overline{L_m(G_N)}$. Thus we have to apply Algorithm A and by Theorem 7 and 11 the algorithm is guaranteed to converge in first iteration. Figure 4.7 depicts the resulting plant and specifications. As can be seen, the only thing changed here is the marking of the state shown as $\otimes$. This state is marked in recovery but the string that reaches this state from initial state creates

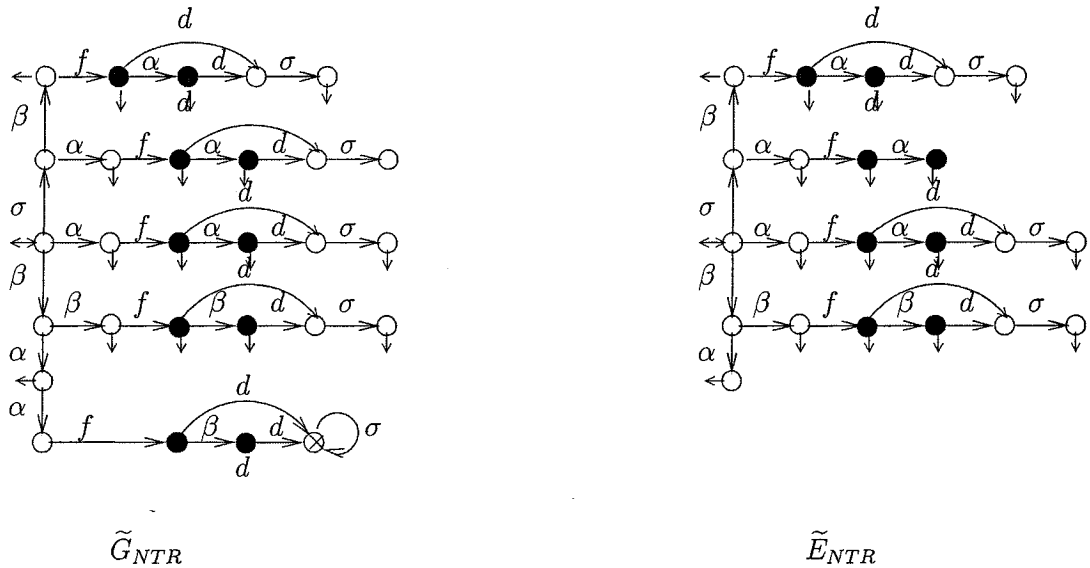


Figure 4.7: $\widetilde{G}_{NTR}$ and $\widetilde{E}_{NTR}$ resulted from Algorithm A

blocking in the normal mode; thus this state should be unmarked. Now, we construct the "super plant" G and specification E as in Theorem 11. It could be easily verified that $G = \widetilde{G}_{NTR}$. A generator representing E is depicted in Figure 4.8.

In this example, since all controllable events are observable, the notion of observability and normality coincide. Thus E has a supremal sub-language satisfying conditions (a-d) in Theorem 11. The supervisor v in Figure 4.8 is designed based on this sub-language. The action of supervisor on the plant models contain useful insights to this framework. In normal mode, $\beta\alpha\alpha$ must be removed from the marked behavior. Consequently the lowest branch starting from this sequence is removed in E. However α is an uncontrollable event and thus the supervisor has to disable β from the initial state. Since $\sigma\beta$ is not allowed in the transient mode and thus the β (after σ) leading to the upper branch of G is disabled in the supervisor. $\sigma\alpha$ is allowed in normal mode and does not generate blocking. However, the continuation of this string generates blocking in recovery mode. Thus the supervisor disables σ in normal mode in initial state.

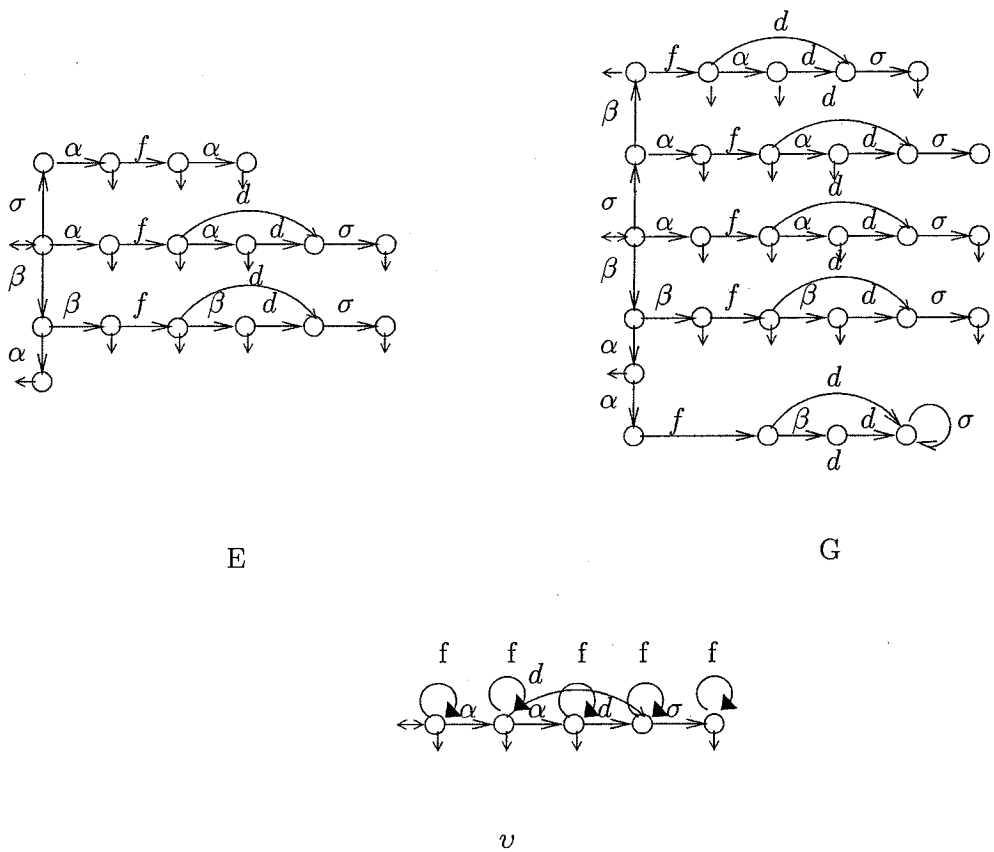


Figure 4.8: Super-plant G and specification E and the resulting supervisor υ

Chapter 5

Conclusion

5.1 Summary

In this thesis, we study the synthesis of fault recovery procedures using discrete-event models. It is assumed that the faults are permanent and that a diagnosis system is available that detects and isolates the faults with a bounded delay. We use a finite-state automaton to model the diagnosis delay. The system to be controlled which consists of the plant and the diagnosis system will have three modes of operation: (i) *normal* when the plant is functioning properly, (ii) *transient* when a fault has occurred but still not diagnosed by the diagnosis system, and (iii) *recovery*, when the fault is diagnosed (and appropriate actions may have to be taken for recovery). In this way the design of the supervisor and diagnosis systems are almost separated, resulting in a simpler design. We study the design of a nonblocking supervisor that enforces the specifications of the system in all three modes. To obtain the solutions, we first transform the problem to an equivalent Robust Nonblocking Supervisory Control problem under Partial Observation and then obtain the solutions to the equivalent robust control problem.

To obtain the solution to Robust Nonblocking Supervisory Control under Partial Observation, we first extend the solutions to Nonblocking Supervisory Control Problem

under Full Observation to the case with partial observation. Our solution is obtained assuming a blocking-invariance property which states that if a string creates blocking in one of the possible plant models (in the robust control problem), then in any other possible plant models, it either causes blocking or is not generated at all. This property is introduced for the first time in this thesis. The set of Blocking Invariant languages is shown to posses a supremal element and an algorithm is proposed to obtain it. It is shown that if the behaviors of the plant models have a certain "star" structure, the algorithm converges to the supremal blocking invariant sublanguages n-tuple. Moreover convergence occurs in one step. We also show that in case of convergence of the algorithm, the original Robust Nonblocking Supervisory Control under Partial Observation could be replaced with an equivalent Robust Nonblocking Supervisory Control under Partial Observation for which the marking language of the plants are the supremal blocking invariant sublanguages of the marking language of the original plants, The closed languages remain the same. In other words, we find a sufficient condition under which the RNSC-PO problem can be converted to an equivalent problem in which the blocking-invariance condition holds. Next we use these results to obtain the solution of the Robust Nonblocking Supervisory Control under Partial Observation problem equivalent to the Fault Recovery Problem. It is shown that the equivalent Robust Nonblocking Supervisory Control under Partial Observation need not satisfy the blocking invariance assumption. However we show that in this robust control problem the plant models posses the aforementioned star structure and thus the problem could be transferred to an equivalent Robust Nonblocking Supervisory Control under Partial Observation which satisfies the blocking invariance property. In this way, we manage to characterize all of the solutions to the robust control problem and thus the Fault Recovery Problem in the form of certain controllable and observable languages. We show that our results hold for single-failure scenario as well as simultaneous failures.

5.2 Future Work

The following two directions may be considered for future research:

1. Fault Recovery Problem

   (a) Computational issues: Computational procedures for the solution of the fault recovery problem in the case of regular languages should be developed. Obviously this is of high practical importance.

   (b) Timed DES: Timing information could be crucial in many recovery problems and thus the extension of our works to timed DES should be an important topic for future research.

   (c) Robust Fault Recovery Problem: In the current framework for Fault Recovery Problem, we assume that the model of the plant in any of its operational mode is exactly known. The framework could be generalized by assuming that the exact model of the plant is not known. To solve the Fault Recovery Problem in this framework, a more general framework have to be examined for Robust Nonblocking Supervisory Control under Partial Observation by requiring the supervisor not only to maximize permissiveness, but also to maximize robustness. In other words, the set of plants which satisfies the specification under the supervision of the optimal supervisor, should be maximized.

2. Robust Nonblocking Supervisory Control under Partial Observation.

   (a) Convergence of Algorithm A: The sufficient condition for the convergence of Algorithm A may be relaxed. This will extend the application of the results to all Robust Nonblocking Supervisory Control problems.

Bibliography

[1] S. Balemi, G. J. Hoffmann, P. Gyugyi, H. Wong-Toi, and G. F. Franklin. Supervisory control of a rapid thermal multiprocessor. *IEEE Trans. on Automat. Cont.*, 38(7):1040–1059, July 1993.

[2] S. E. Bourdon, M. Lawford, and W.M. Wonham. Robust nonblocking supervisory control of discrete–event systems. In *Proc. of the American Control Conference*, volume 1, pages 730–735, Anchorage AK., May 2002.

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

[4] Y. Liang Chen and G. Provan. Model–based control fault–tolerant control reconfiguration for general network topologies. *IEEE MICRO*, 21(5):64–76, September 2001.

[5] K. H. Cho and J. T. Lim. A study on stability analysis of discrete event dynamic systems. *IEICE Trans. Inf. & Syst.*, E80-D(12):1149–1154, December 1997.

[6] K. H. Cho and J. T. Lim. Synthesis of fault-tolerant supervisor for automated manufacturing systems: a case study for photolithographic process. *IEEE Trans. Robot. Automat.*, 14(2):348–351, April 1998.

[7] R. Cieslak, C. Desclaux, A.S. Fawaz, and P. Varaiya. Supervisoy control of discrete-event processes with partial observation. *IEEE. Trans. Automat. Cont.*, 33(3):249–260, March 1988.

[8] M. Dal Cin. Verifying fault tolerant behavior of state machines. In *Proceeding of the International Conference on Computer-Aided Verification*, pages 97–99, 1997.

[9] J. E.R. Cury and B. H. Krogh. Robustness of supervisors for discrete–event systems. *IEEE Trans. on Automat. Contr.*, 2(44):376–379, 1999.

[10] D. Gordon-Spears and K. Kiriakidis. Reconfigurable robot teams: Modeling and supervisory control. *IEEE Trans. Contr. Syst. Technology*, 12(5):763–769, 2004.

[11] S. HashtrudiZad, R.H. Kwong, and W.M. Wonham. Fault diagnosis in discrete-event systems: Framework and model reduction. *IEEE Trans. Automat. Contr.*, 48(7):1199–1212, July 2003.

[12] J.E. Hopcraft and J.D. Ullman. *Introduction to Automata Theorey, Languages and Computation.* Addison-Welsley, 1979.

[13] R. Isermann. Supervision, fault–detection and fault–diagnosis methods – an introduction. *Control Eng. Practice*, 5(5):639–652, 1997.

[14] R. M. Jensen. Discrete event system controller synthesis and fault tolerant control, a survey of recent advances. Technical Report TR-2003-40, The IT University of Copenhagen, December 2003.

[15] R. Kumar and V. Garg. *Modeling and Control of Logical Discrete Event Systems.* Kluwer Academic Publisher, 1995.

[16] F. Lin. Robust and adaptive supervisory control of discrete event systems. *IEEE Trans. on Auto. Cont.*, 38(12):1848–1852, Decemeber 1993.

[17] F. Lin and W.M. Wonham. On observability of discrete-event systems. *Inform. Sci.*, 44:173–198, 1988.

[18] J. Liu and H. Darabi. Control reconfiguration of discrete event systems controllers with partial observation. *IEEE. Trans. Syst. Man Cybern. B*, 34(6):2262–2272, December 2004.

[19] M. Mossaei and Shahin Hashtrudi Zad. Fault recovery in control systems: A modular discrete–event approach. In *Proc. 2004 Internat. Conf. Electrical and Electronics Engineering (CINVESTAV / IEEE)*, pages 445–450, Acapulco, Mexico., Sep 2004.

[20] C. M. Ozveren and A. S. Willsky. Output stabilizability of discrete event dynamic systems. *IEEE Trans. on Auto. Cont.*, 36(8):925–935, August 1991.

[21] S. J. Park and J. T. Lim. Fault-tolerance robust supervisor for discrete event systems with model uncertainty and its application to a workcell. *IEEE Trans. on Robot. Auto.*, 15(2):386–391, April 1999.

[22] S. J. Park and J. T. Lim. Robust and nonblocking supervisor for discrete-event systems with model uncertainty under partial observation. *IEEE Trans. on Auto. Cont.*, 45(12):2393–2396, December 2000.

[23] K. M. Passino, A. N. Michel, and P. J. Antsaklis. Lyapunov stability of a class of discrete event systems. *IEEE Trans. on Auto. Cont.*, 39(2):269–279, Feburary 1994.

[24] T. S. Perraju, S. P. Rana, and S. P. Sarkar. Specifying fault tolerance in mission critical systems. In *Proc. IEEE High Assurance Systems Engineering Workshop*, pages 24–31, 1996.

[25] P. J. Ramadge and W. M. Wonham. Supervisory control of a class of discrete event processes. *SIAM Journal on Control and Optimization*, 25:206–230, 1987.

[26] P.J. Ramadge and W.M Wonham. Supervisory control of a class of discrete event processes. *SIAM Journal on Contr. and Optimazation*, 25(1):206–230, Jan. 1987.

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

[28] S. Takai. Maximizing robustness of supervisors for partially observed discrete event systems. *Automatica*, 40(3):531–535, March 2004.

[29] Brian C. Williams, Michel D. Ingham, Seung H. Chung, and Paul H. Elliott. Model-based programming of intelligen embedded sysems and robotic space explorers. In *Proc. of IEEE-96*, volume 91, pages 212–237, January 2003.

[30] W. M. Wonham. *Supervisory Control of Discrete–Event Systems.* System Contro Group, Edward S. Rogers Sr. Dept. of Elec. and Comp. Eng., University of Toronto, 2004. Available at http://www.control.utoronto.ca/DES.

[31] W. M. Wonham and P. J. Ramadge. On the supremal controllable sublanguage of a given language. *SIAM Journal on Control and Optimization*, 25(3):637–659, May 1987.