# Robust State-Based Supervisory Control of Hierarchical Discrete-Event Systems

Nazanin Hashemi Attar

A Thesis
In the Department
of
Electrical and Computer Engineering

Presented in Partial Fulfilment of the Requirements
For the Degree of
Doctor of Philosophy (Electrical and Computer Engineering) at
Concordia University
Montréal, Québec, Canada

April 2021

**CONCORDIA UNIVERSITY**

**SCHOOL OF GRADUATE STUDIES**

This is to certify that the thesis prepared

By: Nazanin Hashemi Attar

Entitled: Robust State-based Supervisory Control of Hierarchical Discrete-Event Systems

and submitted in partial fulfillment of the requirements for the degree of

Doctor Of Philosophy (Electrical and Computer Engineering)

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

______________________________ Chair
Dr. Liangzhu Wang

______________________________ External Examiner
Dr. Feng Lin

______________________________ External to Program
Dr. Wen-Fang Xie

______________________________ Examiner
Dr. Kash Khorasani

______________________________ Examiner
Dr. Mustafa Mehmet Ali

______________________________ Thesis Supervisor
Dr. Shahin Hashtrudi Zad

Approved by ______________________________
Dr. Wei-Ping Zhu, Graduate Program Director

April 8, 2021 ______________________________
Dr. Mourad Debbabi, Dean
Gina Cody School of Engineering and Computer Science

# Abstract

**Robust State-Based Supervisory Control of Hierarchical Discrete-Event Systems**

Nazanin Hashemi Attar, Ph.D.

Concordia University, 2021

Model uncertainty due to unknown dynamics or changes (such as faults) must be addressed in supervisory control design. Robust supervisory control, one of the approaches to handle model uncertainty, provides a solution (i.e., supervisor) that simultaneously satisfies the design objectives of all possible known plant models. Complexity has always been a challenging issue in the supervisory control of discrete-event systems, and different methods have been proposed to mitigate it. The proposed methods aim to handle complexity either through a structured solution (e.g. decentralized supervision) or by taking advantage of computationally efficient structured models for plants (e.g., hierarchical models). One of the proposed hierarchical plant model formalisms is State-Tree-Structure (STS), which has been successfully used in supervisor design for systems containing up to $10^{20}$ states.

In this thesis, a robust supervisory control framework is developed for systems modeled by STS. First, a robust nonblocking supervisory control problem is formulated in which the plant model belongs to a finite set of automata models and design specifications are expressed in terms of state sets. A state-based approach to supervisor design is more convenient for implementation using symbolic calculation tools such as Binary Decision Diagrams (BDDs). In order to ensure that the set of solutions for robust control problem can be obtained from State Feedback Control (SFBC) laws and hence suitable for symbolic calculations, it is assumed, without loss of generality, that the plant models satisfy a mutual refinement assumption. In this thesis, a set of necessary and sufficient conditions is derived for the solvability of the robust control problem, and a procedure for finding the maximally permissive solution is obtained.

Next, the robust state-based supervisory framework is extended to systems modeled by STS. A sufficient condition is provided under which the mutual refinement property can be verified without converting the hierarchical model of STS to a flat automaton model. As an illustrative example, the developed approach was successfully used to design a robust supervisor for a Flexible Manufacturing System (FMS) with a state set of order $10^8$.

## Acknowledgements

Foremost, I would like to express my sincere gratitude to my research supervisor Dr. Shahin Hashtrudi-Zad for giving me the opportunity to learn and providing me with invaluable guidance throughout this research. Without his priceless advice and constant support, this dissertation would not have been possible.

I would like to thank my thesis committee members Dr. Feng Lin, Dr. Wen-Fang Xie, Dr. Khashayar Khorasani, and Dr. Mustafa K. Mehmet Ali. I appreciate their time and their valuable comments.

I am very thankful to my friends for their support and friendship. In particular, Amir, Irina, Bahareh, Navid, Mahsa, Hossein, Golsa, Mahmoud, Adriana, Mitra, and Fahimeh for their continuous love and support.

I would like to express my eternal appreciation and love to my siblings Alireza, Amirhossein and Asana. I profoundly thank my parents for their love and support.

# Contents

# List of Tables

# List of Figures

# Acronyms

**AI** Artificial Intelligence

**AIP** Atelier Interetablissement de Productique

**BDD** Binary Decision Diagram

**DES** Discrete-Event Systems

**FMS** Flexible Manufacturing System

**HFSM** Hierarchical Finite-State Machine

**HISC** Hierarchical Interface-Based Supervisory Control

**MR** Mutually Refined

**MRI** Magnetic Resonance Imaging

**NCA** Nearest Common Ancestor

**OBDD** Ordered Binary Decision Diagrams

**PID** Proportional–Integral–Derivative

**RNSCP-STS** Robust Nonblocking Supervisory Control Problem for State-Tree-Structure

**RNSSCP** Robust Nonblocking State-based Supervisory Control Problem

**SCT** Supervisory Control Theory

**SFBC** State Feedback Control

**ST** State-Tree

**STS** State-Tree-Structure

**STSM** State-Tree-Structure with conditional-preemption matrices

# Chapter 1

# Introduction

Control systems are typically hierarchical. At the lowest level, control loops are designed based on the continuous-variable models of the plant (such as differential equations). Examples of these controllers include Proportional–Integral–Derivative (PID) and lead/lag controllers. At the middle layer, supervisory control monitors the plant and issues sequencing commands. For example, it enables or disables lower-level control loops and controls the system's startup and shutdown sequences. These control sequences can be analyzed and designed using Discrete-Event Systems (DES) models. Finally, at the highest level, scheduling and planning are done for the system over longer time horizons.

In this thesis, our focus is on the supervisory control of DES (from now on for brevity, supervisory control). Given a DES plant with a set of design specifications, the supervisory control problem is to design a control law to alter the plant's behavior such that the plant under supervision meets the given specifications. These specifications usually address safety properties and the generation of desirable sequences. Furthermore, in most cases, meeting the aforementioned specifications is not enough, and the system under supervision is expected to be nonblocking (i.e. be free of deadlocks and livelocks).

Faults are inevitable during the operation of a system. One approach to handle faults is to use robust supervisory control methods and design the system to be fault-tolerant.

One of the biggest problems in supervisory control of real-world complex systems is the so-called state explosion. Different methods have been proposed to tackle this problem, and we will review some of them later in this chapter. Besides state explosion, other challenges in designing supervisory control include measurement uncertainty (i.e. uncertainty in determining the moment an event in a plant occurs) and model uncertainty (either due to limitations in the designer's knowledge of plant's behavior or unexpected changes due to, say, faults). In this thesis, we will specifically focus on model uncertainty. Researchers have extensively studied robust supervisory control for handling this type of uncertainty. A robust supervisory control method that can handle both model uncertainty and the complexity of real-world systems would be advantageous.

In this chapter, first, we briefly review supervisory control using DES models. Then, we discuss some of the related works in robust and state-based supervisory control, followed by hierarchical discrete-event systems and symbolic supervisory control. Finally, we provide an overview of the research contributions discussed in this thesis.

## 1.1 Supervisory Control Using Discrete-Event Models

Different methods are used to model, analyze and control systems. Depending on the information needed to achieve the control objectives, one can choose to model a plant as a continuous-time, discrete-time or discrete-event system. A detailed model is not necessary to design supervisory control sequences, and DES models usually suffice. Ramadge and Wonham first introduced a formal systematic approach to supervisory control using DES models in [53]. In the so-called RW supervisory control theory, it is assumed that design safety requirements are specified as a set of safe event sequences and that the supervisor can disable and prevent the occurrence of a subset of plant events called the controllable events. The role of the supervisor (Figure 1.1) is to monitor the events generated in the plant and, based on the given requirements, disable some of the controllable events [74]. The supervisor should also ensure that the system under supervision is nonblocking (i.e., it is free of deadlocks and livelocks) [74]. In Figure 1.1, the system information that the supervisor receives is the sequence of events that the plant generates.

The original formulation of supervisory control problem in [53] and [72] was language-based in the sense that the safety requirements were expressed as a design specification language, i.e., a set of event sequences. An alternative way is to express the safety requirements in terms of subsets of safe states (rather than languages). This led to a state-based framework for supervisory control developed in [73], [2], [24] and [35] using predicate calculus. Extensions of state-based approach include State-Tree-Structure STS [68] and Hierarchical Finite-State Machine (HFSM) [5]. The linguistic and state-based approaches are equivalent: every problem in one setup can be transformed and solved using the other approach. The choice of approach is a matter of convenience. For symbolic calculations, a state-based approach is more convenient.

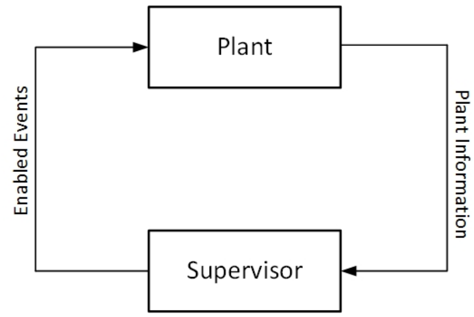



Figure 1.1: The block diagram of a supervisory control system.

## 1.2 Literature Review

The focus of this thesis will be robust supervisory control, especially in plants modeled by hierarchical STS models. This section will review some papers on the topics of robust supervisory control, state-based supervisory control, supervisory control of hierarchical discrete-event systems, and symbolic supervisory control.

### 1.2.1 Supervisory Control

Supervisory control for DES systems was first proposed in [53]. A supervisor's goal is to make sure that the plant achieves its desired behavior and does not get blocked. However, the supervisor can only prevent the controllable events from happening, and it does not have any control over the uncontrollable ones. The application of Supervisory Control Theory (SCT) to real-world complex DES plants poses serious challenges. As a result, different supervisory control methods have been proposed to deal with different situations. In this section, we briefly review the related works that have been done in the following areas: state-based supervisory control, hierarchical discrete-event systems and symbolic supervisory control.

#### 1.2.1.1 State-Based Supervisory Control

State-based formulation of the supervisory control problem was introduced in [52] for automaton plants and further developed in [35]. Other state-based approaches based on Petri Nets and Vector DES have been proposed (See, e.g, [27] and [36]). The introduction of hierarchical structures in supervisory control [5], [68] has led to more recent research activities in state-based supervisory control. In Section 1.2.1.3, we will review the use of hierarchical structures in DES control. [51] has introduced a state-based supervisory control for timed DES. Meanwhile, [69] examined the state-based control of DES under partial observation.

Recently, [17] solved the state-based supervisory control for systems that have restrictions on the controller implementation.

#### 1.2.1.2 Robust Supervisory Control

Similar to continuous-time systems, modeling uncertainties also exist in DES models and the related supervisory control problems. The prominent cases of uncertainty considered in this research are those in which the plant dynamics are known (at the design stage) to belong to a finite set of models. This set of uncertainty cases is encountered, for example, in fault accommodation and recovery problems. There are two main approaches to tackle this type of problem: 1. adaptive supervisory control; 2. robust supervisory control. In adaptive supervisory control (see, e.g., [37]), the supervisor tries to overcome the effects of plant model uncertainties by updating itself accordingly. In robust supervisory control, however, the supervisor is synthesized only once at the beginning in such a way that it meets the design specifications for each of the possible finite set of models (i.e. a standard solution for multiple supervisory control problems). Different methods have been proposed to solve various robust supervisory control problems. The majority of them have adopted the linguistics approach based on the closed/marked specification languages. In the following, we will briefly review some of the highlighted works in the robust supervisory control.

In [37], it is assumed that the DES model of the plant is not unique and belongs to a finite set of models. Moreover, the union and the intersection of marked (resp. closed) languages of all the possible models form an upper and a lower bound for the plant's marked (resp. closed) behavior and the *common* desired behavior (design specification) is a subset of the lower bound of marked behavior. A solution for the robust supervisory control problem that marks the desired behavior exists if and only if the given desired behavior is controllable and observable with respect to the upper bound of the closed behavior. The results of [37] are extended in [4] where it is assumed that each possible model in the finite set of models has its own specification. Full event observation is, however, assumed in [4]. *Nonconflicting* property of the marked behaviors of plant models is presented [4] which as shown in [57] serves as a necessary condition for the solution. Meanwhile, [57] continues the direction of [4] and extends its results for systems with partial observable events. Furthermore, it replaces the nonconflicting condition with a stronger condition called $\mathbf{G}_i$-nonblocking to obtain a set of necessary and sufficient conditions for robust control under partial observation. In recent years, studies of robust control applications have been undertaken. For example, [76] uses the results of [4] and [57] to synthesize a robust supervisor for the fault recovery of a spacecraft propulsion system.

The results of [37], [4], [57] and [76] fall into the category of indirect approaches to robust supervisory control problem. In the indirect approach, the solution is characterized by a sub-language of the union of all marked behaviors (from which each plant model's behaviour under supervision can be derived). On the other hand, in the direct approach proposed by [12], the controlled behavior of each plant model is considered individually. Along the same line, [60] considers partially observed timed-DESs and designs a supervisor such that all the possible models have legal behavior under its supervision. Meanwhile, [49] solved the robust nonblocking supervisory control problem for nondeterministic DESs. This work was extended to decentralized supervisory control problem in [50]. All the robust supervisory control approaches that we have discussed until now are computed off-line; however, there are some approaches such as [3] and [11] that have proposed algorithms to compute the supervisor on-line (by extending the lookahead policy results of [13] and [25].

Another approach to robust control is proposed in [14] in which a robust nonblocking supervisor for plant models describing infinite behavior is studied. Here, instead of using a finite set of models, [14] uses a nominal plant model and assumes that the desired specification has a lower and an upper bound. Then a supervisor for the nominal plant is found that maximizes the set of closed-loop models that satisfy the lower and the upper bound specifications. This work on maximally permissive supervisors was extended by [61] by removing the restrictions on upper bound specifications in the case of closed languages, and later in [62] to systems with partially observed events.

Besides the problems that concern model uncertainty, some other problems can also be solved as robust supervisory control problems. For example, the supervisory control problem with multiple tasks (multiple sets of marked states) introduced by [15] was solved as a robust supervisory control problem by [12]. [76] also used robust supervisory control to solve a fault recovery problem. The robustness issue has also been studied in the fault diagnosis [63][7][65][77] and fault prognosis [64][75] problems. For example, [64] assumes that the current automaton model of the system belongs to a set of possible automaton models and designs a robust prognosis scheme for such system.

#### 1.2.1.3 Hierarchical Discrete Event Systems

For a system with structure, a structured model is more understandable than a flat (unstructured) model with individual states connected through events. Moreover, the structure in a model may be used to reduce the computational complexity associated with control and observation problems. For these reasons, Statecharts are proposed in [26] as a modeling formalism for systems with a hierarchical structure. However, [26] mainly focuses on the system's visual representation and fails to give a mathematical definition. Based on the definition given in [26], [68] develops a modeling structure named State-Tree-Structure (STS) to model the state space and dynamics of systems. STS have vertical and horizontal modularity; vertical modularity comes from placing states of the system in ordered layers, and horizontal modularity is the effect of using modules called holons [68]. Another version of statechart is Hierarchical Finite-State Machine (HFSM) used in [5]. In [5], shared events are not allowed among horizontal modules. To overcome this problem, [40] introduced a new definition of STS that is discussed in details in Chapter 2. Hierarchical Interface-Based Supervisory Control (HISC) is another hierarchical method that is proposed in [33] and [38]; HISC is a language-based approach in which the system has only two levels of hierarchy and $n \geq 1$ modules. Therefore, we can only expand the models in one direction rather than two. From this point of view, HISC is similar to modular discrete-event systems [29].

Predicates can be used to represent states in both flat models and STS. However, predicates of a system modeled with STS are noticeably simpler than those in the equivalent flat model. Therefore, synthesizing a supervisory controller for STS requires less time and space. [40] and [39] propose an algorithm to synthesize the optimal (maximally permissive[1]) supervisory controller for systems modeled by STS. In addition to benefiting from the advantages of a structured model of the plant, [40] and [39] use Binary Decision Diagrams (BDDs) for the calculation of the supervisor. The proposed method is tested on two benchmark problems, one with $10^8$ and the other with $10^{24}$ states. For both systems, the supervisor was calculated within a short period of time. Later, [9] expands the results of [39] to modular supervisory control. Recently, [71] studied the problem of real-time scheduling for systems with STS models.

The problem of fault diagnosis has also been explored in hierarchical DES. [47] studies the diagnosis of Hierarchical Finite-State Machine (HFSM) and proposes a semi-modular approach. There the concept of holons from [68] is replaced with D-holons. D-holons' definition is the same as holons except that their boundary events should be observable. Later, [46] develops a recursive multi-level algorithm to design a diagnosis system for hierarchical DES. [48] takes a different approach to diagnosis and introduces the concept of L1-diagnosability. If a system is L1-diagnosable, then a fault event can be detected from the first (top) level of the hierarchy. For those systems that are not L1-diagnosable, [48] develops an algorithm that transfers the fault information from lower levels until a fault event can be detected with certainty. [55] uses the results of [47] and expands it to STS; however, no diagnosability verification algorithm is considered.

#### 1.2.1.4 Symbolic Supervisory Control

BDD was first introduced by [34] and [1] to symbolically represent boolean functions. Later, [6] expanded their results to Ordered Binary Decision Diagrams (OBDD) and simplified the computational procedure of BDD. In supervisory control problem, BDDs have been used to synthesize the supervisor symbolically [2] and in most cases they are associated with hierarchical structures and state-based supervisory control

[1] A maximally permissive supervisor keeps the set of disabled events minimal, resulting in the largest reachable state set for the system under supervision.

[40][56][42][43][9][8][41][40][67]. As it has been previously mentioned in Section 1.2.1.1 and 1.2.1.3, predicates have been used to represent state space of a system and BDDs are the best tools that can simplify the calculations of predicates. Usually, the supervisory control problems that use BDDs in their synthesis are called symbolic supervisory control. The computational complexity of symbolic supervisory control is polynomial in the number of BDD nodes. In the worst case scenario, this computational complexity is polynomial in the number of states (exponential in the number of components). The following paragraph reviews some of the related works.

BDDs have been applied in the diagnosis of DES; [59] and [58] used BDDs to reduce the memory space required for performing the computations and storing the diagnoser. BDDs have also been used in supervisory control of extended finite automata [45][67][18][19] and timed extended finite automata[43]. [44] used a simple Artificial Intelligence (AI) search method in the synthesize of a symbolic supervisory control problem for deterministic finite automata. BDDs have also been used for some of the hierarchical structures that were mentioned in Section 1.2.1.3. For example, [32] proposed the symbolic version of HISC. Moreover, symbolic computation of STS (Section 1.2.1.3) was proposed by [40][41][8][21][28][67]. [21] has specifically synthesized a symbolic supervisor for a sub-system of a Magnetic Resonance Imaging (MRI) scanner. [16] also synthesized a symbolic supervisor for an autonomous aerial refueling system modeled by STS.

The synthesis of supervisory control for STS under partial observation is introduced in [23] and [22]. The symbolic calculation of STS is used in [54] for synthesizing a fault-tolerant supervisory control. [30] has specifically synthesized a supervisor for advanced driver assistance systems. Recently, [70] studied the concept of *priority* (e.g., the priority of events occurrences) in systems with STS models. They developed a new framework called State-Tree-Structure with conditional-preemption matrices (STSM), and utilized BDD to synthesize a nonblocking supervisory control for STSM.

## 1.3 Research Objectives and Methodology

In this section, we explain our objectives and the methodologies that we have used to reach those objectives.

### 1.3.1 Objectives

State-explosion is one of the main barriers to using supervisory control theory for large-scale industrial systems. As the size of state space grows, the computational complexity and the required memory to perform supervisory control calculations also grow rapidly. In fact, the complexity is exponential in the number of the system's components. Over the years, different methods have been proposed to deal with this issue. All the proposed methods fall into two major categories: 1. solutions in the form of structured supervisor and 2. solutions based on a structured plant model.

Addressing model uncertainty and varying dynamics has also been an issue in supervisory control of DES. This problem becomes even more challenging in complex industrial systems. To our knowledge, no robust supervisory control method has been proposed to deal with model uncertainty in industrial-size examples so far.

In this thesis, our objective is to develop a robust supervisory control method that can deal with large-scale DES.

### 1.3.2 Methodology

We formulate a robust supervisory control method for plants modeled with STS to deal with model uncertainty in large-scale industrial systems. In order to do so, first, we define a novel robust state-based supervisory control for automata and then expand our method to STS. This method synthesizes the supervisor offline and can use BDD to accelerate the calculations. Moreover, it makes sure that the systems under supervision stay nonblocking.

Model uncertainty in DES can be dealt with by robust or adaptive supervisory control methods. Fuzzy DES are also used to represent model uncertainty; however, the supervisory control of fuzzy DES does not represent any computational advantages over the existing robust or adaptive supervisory control approaches. The stochastic approaches in DES used to deal with model uncertainty are beyond this thesis's scope.

Using BDD in supervisory control calculations has shown promising results in combating the state-explosion problem in DES. The method in [66] is shown to handle a transfer line model with up to $10^{210}$ states and 100 cells. [43] tested their method on some industrial benchmarks with up to $10^{17}$ states. [39] also simulated a model of Atelier Interetablissement de Productique (AIP) that has $10^{24}$ states; the calculations took less than 20 seconds for a personal computer with 1GHz Athlon CPU and 256-MB RAM.

For industrial systems, STS is a more suitable modeling formalism as adding or removing the components can be quickly done by adding or removing new branches to the existing STS model. Moreover, the model is more comprehensible to the users.

## 1.4 Contributions

The thesis formulates a novel robust state-based nonblocking supervisory control problem for DES modeled by automata. To our knowledge, no robust state-based supervisory control method has been defined previously. Next, the necessary and sufficient conditions are obtained for the existence of a solution for the problem. An algorithm is also introduced to calculate the maximally permissive solution within a finite number of iterations.

We extend the robust state-based supervisory control problem and examine the robust nonblocking supervisory control problem for systems modeled by STS. A set of necessary and sufficient conditions for the existence of solution are derived, and an algorithm is also explained that calculates the maximally permissive solution within a finite number of iterations. This STS-based robust supervisory control method is suitable for large scale and industrial-size systems.

The solutions of a conventional supervisory control problem[2](in which the plant model is known) can always be represented via SFBC. In the robust supervisory control problem, to make sure that we can characterize the solutions via SFBC laws, we assume that the automata satisfy the Mutually Refined (MR) condition. The MR property in automata does not impose any restrictions on the formulation of the robust supervisory control problem. We derive conditions to verify the MR property in STS without building the flat (unstructured) model.

To illustrate our results, we formulate the robust nonblocking supervisory control problem for a Flexible Manufacturing System (FMS) with a state set of order $10^8$. We synthesized the solution (supervisor) using our algorithm and a BDD-based program. The solution was synthesized in less than 0.5 seconds on a personal computer.

## 1.5 Organization

The rest of the thesis is organized as follows. In Chapter 2, we review some preliminaries. In Chapter 3, the robust nonblocking state-based supervisory control method is formulated and solved. Chapter 4 extends the results of Chapter 3 to robust supervisory control of systems with STS models. Chapter 5 summarizes the thesis and discusses directions for future research.

[2]In this thesis, the original supervisory control problem introduced by [53], in which the exact plant model is known, is referred to as the conventional supervisory control problem.

# Chapter 2

# Background

In this chapter, we briefly review some of the required preliminaries. First, in Section 2.1, automata, languages, and predicates are introduced. Then, in Section 2.2, the robust supervisory control problem, introduced in [4] and [57], is discussed. In Section 2.3, state-based supervisory control is reviewed. In Sections 2.4 and 2.5, State-Tree-Structure (STS) models and supervisory control of STS are discussed. Finally, in Section 2.6, some relevant results on the Binary Decision Diagram (BDD) are presented.

## 2.1 Automata, Languages, and Predicates

Let us assume $\Sigma$ is a finite nonempty set of symbols each representing an *event* in a DES. $\Sigma$ is called an *alphabet*. The events that belong to $\Sigma$ can be used to form a sequence of events called a *string*. The set of all finite strings over an alphabet $\Sigma$ is shown by $\Sigma^+$,

$$\Sigma^+ = \{\sigma_1 \ldots \sigma_k \mid k \geq 1,\ \sigma_i \in \Sigma\}. \tag{2.1}$$

An *empty string* is shown by $\epsilon$. The set of all finite strings over an alphabet $\Sigma$ is shown by $\Sigma^*$,

$$\Sigma^* = \Sigma^+ \cup \{\epsilon\}. \tag{2.2}$$

Any subset of $\Sigma^*$ is called a *language*. Let $L$ be a language over $\Sigma$. For $s \in L$, $L/s = \{t \in \Sigma^* \mid st \in L\}$ is called the *post-language* of $L$ after $s$. Language $L$ is *live* if

$$\forall s \in L,\ \exists t \in \Sigma^*, \text{ and } t \neq \epsilon \text{ such that } st \in L. \tag{2.3}$$

For $s \in \sigma^*$, $t \in \Sigma^*$ is the *prefix* of $s$ if $s = tu$ for some $u \in \Sigma^*$. The *prefix-closure* of a language $L \subseteq \Sigma^*$ is denoted as $\overline{L}$,

$$\overline{L} = \{s \in \Sigma^* \mid \exists t \in \Sigma^* \text{ such that } st \in L\}. \tag{2.4}$$

The language $L$ is called prefix-closed (or closed) if $L = \overline{L}$.

The dynamics of continuous-variable systems are time-driven, while those of DES are event-driven. In a time-driven system, state changes are represented against the passage of time. However, in an event-driven system, occurrences of events cause the system to go from one state to another. There are different approaches for representing DES. Here, we only discuss DES that are modeled by finite-state deterministic automata. A deterministic automaton is a five-tuple

$$\mathbf{G} = (Q, \Sigma, \delta, q_0, Q_m), \tag{2.5}$$

where $Q$ is the *state set*, $\Sigma$ is the finite set of events, $\delta : Q \times \Sigma \rightarrow Q$ is the *partial transition function*, $q_0$ is the *initial state* and $Q_m \subseteq Q$ is the set of *marked states*.

At a state $q \in Q$, $\delta(q, \sigma)!$ means a transition $\sigma$ is possible. The set of strings generated by the automaton $\mathbf{G}$ is called the *closed language*: $L(\mathbf{G}) = \{s \in \Sigma^* \mid \delta(q_0, s)!\}$. The *marked language* is the set of strings generated by $\mathbf{G}$ that end in a marked state: $L_m(\mathbf{G}) = \{s \in L(\mathbf{G}) \mid \delta(q_0, s)! \; \& \; \delta(q_0, s) \in Q_m\}$.

Consider $\mathbf{G}_1 = (Q_1, \Sigma_1, \delta_1, q_{01}, Q_{m1})$ and $\mathbf{G}_2 = (Q_2, \Sigma_2, \delta_2, q_{02}, Q_{m2})$. $\mathbf{G}_1$ is a *sub-automaton* of $\mathbf{G}_2$ ($\mathbf{G}_1 \subseteq \mathbf{G}_2$) if

- $Q_1 \subseteq Q_2$,
- $Q_{m1} \subseteq Q_{m2}$,
- $q_{01} = q_{02}$, and
- $\forall s \in L(\mathbf{G}_1) \left(\delta_1(q_{01}, s) = \delta_2(q_{02}, s)\right)$.

For the rest of this section, consider a DES $\mathbf{G} = (Q, \Sigma, \delta, q_0, Q_m)$. The *reachable* sub-automaton of $\mathbf{G}$ is denoted by $\text{reach}(\mathbf{G}) = (Q_r, \Sigma_r, \delta_r, q_{0r}, Q_{mr})$.

A system can have multiple components (sub-systems) modeled by automata. To model the interconnections of these components, the *product* and the *synchronous product* (parallel composition) operations are defined over automata.

**Definition 2.1.** *([74]) Consider two automata* $\mathbf{G}_1 = (Q_1, \Sigma_1, \delta_1, q_{01}, Q_{m1})$ *and* $\mathbf{G}_2 = (Q_2, \Sigma_2, \delta_2, q_{02}, Q_{m2})$*. The*

***product*** *of* $\boldsymbol{G}_1$ *and* $\boldsymbol{G}_2$ *is defined as follows:*

$$reach(\boldsymbol{G}_1 \times \boldsymbol{G}_2) = \left(Q_1 \times Q_2, \Sigma_1 \cup \Sigma_2, \delta, (q_{01}, q_{02}), Q_{m1} \times Q_{m2}\right) \tag{2.6}$$

*where for* $q_1 \in Q_1$ *and* $q_2 \in Q_2$,

$$\delta\big((q_1, q_2), \sigma\big) = \begin{cases} (\delta_1(q_1, \sigma), \delta_2(q_2, \sigma)), & \text{if } \delta_1(q_1, \sigma)! \text{ and } \delta_2(q_2, \sigma)! \\ \text{undefined}, & \text{otherwise} \end{cases}. \tag{2.7}$$

**Definition 2.2.** *([74]) Consider two automata* $\boldsymbol{G}_1 = (Q_1, \Sigma_1, \delta_1, q_{01}, Q_{m1})$ *and* $\boldsymbol{G}_2 = (Q_2, \Sigma_2, \delta_2, q_{02}, Q_{m2})$*. The* ***synchronous product*** *of* $\boldsymbol{G}_1$ *and* $\boldsymbol{G}_2$ *is defined as follows:*

$$reach(\boldsymbol{G}_1 \| \boldsymbol{G}_2) = \left(Q_1 \times Q_2, \Sigma_1 \cup \Sigma_2, \delta, (q_{01}, q_{02}), Q_{m1} \times Q_{m2}\right) \tag{2.8}$$

*where for* $q_1 \in Q_1$ *and* $q_2 \in Q_2$,

$$\delta\big((q_1, q_2), \sigma\big) = \begin{cases} (\delta_1(q_1, \sigma), \delta_2(q_2, \sigma)), & \text{if } \delta_1(q_1, \sigma)! \text{ and } \delta_2(q_2, \sigma)! \\ (\delta_1(q_1, \sigma), q_2), & \text{if } \sigma \in \Sigma_1 - \Sigma_2 \text{ and } \delta_1(q_1, \sigma)! \\ (q_1, \delta_2(q_2, \sigma)), & \text{if } \sigma \in \Sigma_2 - \Sigma_1 \text{ and } \delta_2(q_2, \sigma)! \\ \text{undefined}, & \text{otherwise} \end{cases}. \tag{2.9}$$

The synchronous product of automata can also be constructed using the product operation. To do so, first $\mathbf{G}'_1$ (by adding self-loops of $\Sigma_2 - \Sigma_1$ to $\mathbf{G}_1$) and $\mathbf{G}'_2$ (by adding self-loops of $\Sigma_1 - \Sigma_2$ to $\mathbf{G}_2$) are constructed. The synchronous product of $\mathbf{G}_1$ and $\mathbf{G}_2$ is calculated as follows:

$$\mathbf{G}_1 \| \mathbf{G}_2 = \mathbf{G}'_1 \times \mathbf{G}'_2 \tag{2.10}$$

A *predicate* $P$ is a function $P : Q \rightarrow \{0, 1\}$ that maps each of the states in $Q$ to 0 or 1. Here, 0 and 1 represent *false* and *true* respectively. We say a state $q$ satisfies a predicate $P$ if and only if $P(q) = 1$ (or *true*) and write $q \models P$. Predicates are always identified with a state subset; for example, we say $P_1$ is identified with $Q_1 = \{q \in Q \mid P_1(q) = 1\}$. The set of all predicates that can be defined over $Q$ is shown by $\text{Pred}(Q)$. We can also form the *conjunction* and *disjunction* of two predicates denoted by $\wedge$ and $\vee$ corresponding to the intersection and union set operations. Moreover, the partial order " $\leq$ " is defined over $\text{Pred}(Q)$ as $P_1 \leq P_2$ if and only if $P_1 \wedge P_2 = P_1$. For predicate $P$, $\neg P$ denotes the *negation* of $P$.

The *reachability predicate* $R(\mathbf{G}, P)$ is defined as follows [74]:

- If $q_0 \not\models P$, then $R(\mathbf{G}, P) = \mathit{false}$, otherwise if $q_0 \models P$, then $q_0 \models R(\mathbf{G}, P)$.

- $q \models R(\mathbf{G}, P)$, $\sigma \in \Sigma$, $\delta(q, \sigma)!$ & $\delta(q, \sigma) \models P \Rightarrow \delta(q, \sigma) \models R(\mathbf{G}, P)$.

- No other state satisfies $R(\mathbf{G}, P)$.

$R(\mathbf{G}, P)$ holds for all states in $\mathbf{G}$ that can be reached from $q_0$ via some states that satisfy $P$. $R(.,.)$ is a *monotonically increasing* function.

**Lemma 2.1.** *([74]) (Monotonically increasing) Suppose $P_1$ and $P_2$ are two predicates such that $P_1 \leq P_2$. Then $R(\mathbf{G}, P_1) \leq R(\mathbf{G}, P_2)$.*

The *coreachability predicate* $CR(\mathbf{G}, P)$ [39][40] is defined below.

- If $\nexists q \in Q$ such that $q \in Q_m$ and $q \models P$, then $CR(\mathbf{G}, P) = \mathit{false}$, otherwise $(\forall q \in Q)\ q \in Q_m$ & $q \models P \Rightarrow q \models CR(\mathbf{G}, P)$.

- $q \models CR(\mathbf{G}, P)$, $\sigma \in \Sigma$, $\delta(q', \sigma)!$, $\delta(q', \sigma) = q$ & $q' \models P \Rightarrow q' \models CR(\mathbf{G}, P)$.

- No other state $q$ satisfies $CR(\mathbf{G}, P)$.

$CR(\mathbf{G}, P)$ holds for all states in $\mathbf{G}$ that can reach at least one of the marked states in $Q_m$ via states that satisfy $P$. Similar to $R(.,.)$, $CR(.,.)$ is a monotonically increasing function.

## 2.2 Robust Supervisory Control

As it has been previously mentioned in Section 1.2.1.2, there are various formulations for language-based robust supervisory control problem. We have chosen the one proposed in [4] and [57].

Assume that a plant's model is not known with certainty. However, it can be one of the $N$ models in $\mathcal{G} = \{\mathbf{G}_1, \ldots, \mathbf{G}_N\}$. Here, all $\mathbf{G}_i$ for $i \in I = \{1, \ldots, N\}$ are finite state automata with $\mathbf{G}_i = (Q_i, \Sigma_i, \delta_i, q_0, Q_{mi})$. Let $\Sigma_{ci}$ and $\Sigma_{uci}$ denote the controllable and uncontrollable event sets of $\mathbf{G}_i$ $(i \in I)$. We assume the controllability state of an event does not change from one model to another:

$$\text{if } \sigma \in \Sigma_i \cap \Sigma_j\ (i, j \in I), \text{ then either } \sigma \in \Sigma_{ci} \cap \Sigma_{cj} \text{ or } \sigma \in \Sigma_{uci} \cap \Sigma_{ucj}. \tag{2.11}$$

Suppose language $K_i$ is the design specification for $\mathbf{G}_i$. The *legal marked behavior* is $E_i = K_i \cap L_m(\mathbf{G}_i)$. The set of robust nonblocking supervisory controls for this plant is defined below.

$$V = \{v : \Sigma^* \to \Gamma_\Sigma \mid L_m(v/\mathbf{G}_i) \subseteq E_i \;\&\; \overline{L_m(v/\mathbf{G}_i)} = L(v/\mathbf{G}_i)\} \tag{2.12}$$

where $\Gamma_\Sigma = \{\gamma \in P(\Sigma) \mid \Sigma_u \subseteq \gamma\}$ is the set of all *control patterns* on $\Sigma = \bigcap_{i\in I} \Sigma_i$. Moreover, $L(v/\mathbf{G}_i)$ and $L_m(v/\mathbf{G}_i)$ are the closed and marked languages of $\mathbf{G}_i$ under supervision. Here, $v/\mathbf{G}_i$ denotes *system under supervision*.

The language-based robust nonblocking supervisory control problem is defined below.

**Problem 2.1.** *Consider the set of automata $\mathcal{G} = \{\boldsymbol{G}_1, \ldots, \boldsymbol{G}_N\}$ and the legal languages $E_i \in L_m(\boldsymbol{G}_i)$ $(i \in I)$. Find a supervisor $v : \Sigma^* \to \Gamma_\Sigma$, with $\Sigma = \bigcap_{i\in I} \Sigma_i$ such that*

1. $L_m(v/\boldsymbol{G}_i) \subseteq E_i$,
2. $\overline{L_m(v/\boldsymbol{G}_i)} = L(v/\boldsymbol{G}_i)$.

The theorem below gives the necessary and sufficient conditions for the existence of a solution for Problem 2.1.

**Theorem 2.1.** *([57]) Suppose $\boldsymbol{G}$ is an automaton with $L(\boldsymbol{G}) = \bigcup_{i\in I} L(\boldsymbol{G}_i)$, $L_m(\boldsymbol{G}) = \bigcup_{i\in I} L_m(\boldsymbol{G}_i)$, $\Sigma = \bigcup_{i\in I} \Sigma_i$ and $\Sigma_u = \bigcup_{i\in I} \Sigma_{iu}$. We define $E$ as,*

$$E = \bigcap_{i\in I} (E_i \cup (\Sigma^* - L_m(\boldsymbol{G}_i)) \cap L_m(\boldsymbol{G}). \tag{2.13}$$

*For any nonempty sublanguage $K \subseteq E$ that satisfies the following conditions:*

1. *Controllable with respect to $\boldsymbol{G}$ $\left(\text{i.e., } \overline{K}\Sigma_{uc} \cap L(\boldsymbol{G}) \subseteq \overline{K}\right)$,*
2. *$\boldsymbol{G}_i$-nonblocking $\left(\text{i.e., } \overline{K \cap L_m(\boldsymbol{G}_i)} = \overline{K} \cap L(\boldsymbol{G}_i)\right)$, and*
3. *$L_m(\boldsymbol{G})$-closed $\left(\text{ i.e., } K = \overline{K} \cap L_m(\boldsymbol{G}) \right)$,*

*there exists a solution $v \in V$ to the robust nonblocking control problem such that $L_m(v/\boldsymbol{G}) = K$ and $L(v/\boldsymbol{G}) = \overline{K}$. And conversely, if $v$ solves the robust nonblocking supervisory control problem, then $K = L_m(v/\boldsymbol{G})$ meets conditions (1) to (3).*

## 2.3 State-Based Supervisory Control

A state-based formulation of supervisory control is introduced in [52] for automaton plants and further developed in [35].

It is assumed that the set of events $\Sigma$ can be divided into the set of *controllable events* $\Sigma_c$ and the set of *uncontrollable events* $\Sigma_{uc}$. In the state-based supervisory control approach, the safety requirements are expressed in terms of a set of *safe states*. Let a predicate $P$ represent the set of safe states of automaton $\mathbf{G}$. Feedback can be used to limit the behavior of $\mathbf{G}$ to safe states.

A State Feedback Control (SFBC) $f$ is a function $f : Q \to \Gamma$ where $\Gamma = \{\Sigma' \subseteq \Sigma \mid \Sigma_{uc} \subseteq \Sigma'\}$. For $q \in Q$, $f(q)$ is the set of events that are allowed by $f$ to be enabled at state $q$. Furthermore, for $\sigma \in \Sigma$, the function $f_\sigma : Q \to \{0,1\}$ is defined as $f_\sigma(q) = 1$ if and only if $\sigma \in f(q)$. If $f_\sigma(q) = 1$, $f_\sigma$ asserts that at a state $q \in Q$, the event $\sigma$ should be enabled and it should be disabled if $f_\sigma(q) = 0$. The automaton $\mathbf{G}$ under the supervision of SFBC $f$ is shown by $\mathbf{G}^f = (Q^f, \Sigma^f, \delta^f, q_0, Q_m^f)$. It is clear that $\mathbf{G}^f \subseteq \mathbf{G}$.

An SFBC $f : Q \to \{0,1\}$ is *balanced* if

$$(\forall q, q' \in Q)\ (\forall \sigma \in \Sigma)\ q, q' \models R(\mathbf{G}^f, true) \text{ such that } \delta(q,\sigma)!\ \&\ \delta(q,\sigma) = q' \Rightarrow f_\sigma(q) = 1. \tag{2.14}$$

A predicate $P$ is called *controllable* if and only if

$$P \le R(\mathbf{G}, P)\ \&\ (\forall \sigma \in \Sigma_{uc})\ P \le M_\sigma(P), \tag{2.15}$$

where $M_\sigma : \text{Pred}(Q) \to \text{Pred}(Q)$ is defined as below.

$$M_\sigma(P)(q) = \begin{cases} 1 & \text{if either } \delta(q,\sigma)!\ \&\ \delta(q,\sigma) \models P, \text{ or } \neg(\delta(q,\sigma)!), \\ 0 & \text{otherwise } (i.e.,\ \delta(q,\sigma)! \text{ and } \delta(q,\sigma) \not\models P) \end{cases}. \tag{2.16}$$

The theorem below gives the necessary and sufficient conditions for the existence of a SFBC under the specifications defined by a predicate $P$.

**Theorem 2.2.** *([74]) Assume $P \in \text{Pred}(Q)$, $P \neq false$, and $q_0 \models P$. Then there exists a SFBC $f$ for $\mathbf{G}$ such that $R(\mathbf{G}^f, true) = P$ if and only if $P$ is controllable.*

Denote the set of all controllable sub-predicates of $P$ by

$$\text{CP}(P) = \{K \in \text{Pred}(Q) \mid K \le P\ \&\ K \text{ is controllable with respect to } \mathbf{G}\}. \tag{2.17}$$

$\mathrm{CP}(P)$ is nonempty and has a supremal element [74].

For a predicate $P$, the sub-predicate $\langle P \rangle \leq P$ is defined as follows:

$$\text{For a state } q,\ q \models \langle P \rangle \text{ whenever } \forall w \in \Sigma_{uc}^*,\ \delta(q,w)! \Rightarrow \delta(q,w) \models P. \tag{2.18}$$

$\langle P \rangle \leq P$ holds for those states in $Q_P$ from which uncontrollable event sequences never leave $P$.

**Lemma 2.2.** *([74]) The supremal element of* $\mathrm{CP}(P)$ *is* $\sup\mathrm{CP}(P) = R(\mathbf{G}, \langle P \rangle)$.

A predicate $P$ defined over the state set of $\mathbf{G}$ is called *coreachable* if $P \leq CR(\mathbf{G}, P)$ and *reachable* if $P \leq R(\mathbf{G}, P)$.

**Definition 2.3.** *([39]) A predicate* $P \in \mathrm{Pred}(Q)$ *is called nonblocking with respect to* $\mathbf{G}$ *if and only if* $R(\mathbf{G}, P) \leq CR(\mathbf{G}, P)$.

The above definition states that starting from $q_0$, any path consisting of states that satisfy $P$ can be extended inside $P$ to some marked state. In the following definition, the nonblocking SFBC is defined.

**Definition 2.4.** *([74]) An SFBC $f$ is called a nonblocking SFBC for* $\mathbf{G}$ *if* $R(\mathbf{G}^f, true) \leq CR(\mathbf{G}^f, true)$. *In other words, for a nonblocking SFBC,* $R(\mathbf{G}^f, true)$ *is coreachable.*

The theorem below gives the necessary and sufficient conditions for the existence of a nonblocking SFBC.

**Theorem 2.3.** *([74]) Assume* $P \in \mathrm{Pred}(Q)$, $P \neq false$ *and* $q_0 \models P$. *Then there exists a nonblocking SFBC $f$ for* $\mathbf{G}$ *such that* $R(\mathbf{G}^f, true) = P$ *if and only if $P$ is controllable and nonblocking.*

Note that with $P$ and $f$ as in Theorem 2.3, we have

$$P = R(\mathbf{G}^f, \mathrm{true}) = R(\mathbf{G}, P) \leq CR(\mathbf{G}, P) \leq P = R(\mathbf{G}^f, \mathrm{true}). \tag{2.19}$$

Thus,

$$R(\mathbf{G}, P) = CR(\mathbf{G}, P) = P = R(\mathbf{G}^f, \mathrm{true}) \tag{2.20}$$

and

$$R(\mathbf{G}^f, \mathrm{true}) \leq CR(\mathbf{G}^f, \mathrm{true}). \tag{2.21}$$

Suppose $P \in \mathrm{Pred}(Q)$ is not controllable and/or nonblocking with respect to $\mathbf{G}$. Let $\mathrm{CN_bP}(P)$ denote the set of all controllable and nonblocking sub-predicates of $P$:

$$\mathrm{CN_bP}(P) = \{K \in \mathrm{Pred}(Q) \mid K \leq P \ \&\ K \text{ controllable and nonblocking with respect to } \mathbf{G}\}. \tag{2.22}$$

$CN_bP(P)$ is nonempty and closed under arbitrary disjunctions and has a supremal element $supCN_bP(P)$ [74]. A nonblocking SFBC $f$ that is balanced and results in $R(\mathbf{G}^f, \text{true}) = P$ is the maximally permissive solution of supervisory control problem.

## 2.4 State-Tree-Structure

[68] introduces the State-Tree Structure (STS) based on the definition of statecharts in [26]. STS has both the vertical and the horizontal modularity. In the STS, the State-Tree (ST) and the holons form the horizontal (the hierarchical structure of state space) and the vertical modularity respectively. The weakness of STS proposed by [68] appears when the model has an AND root state. For these cases, the AND root state is converted to some OR super-state by using the synchronous product operation which can transform the model from structured to flat. Furthermore, in [68], the STS dynamics are not defined formally and fully.

[20] introduces Hierarchical Finite State Machine (HFSM) based on the definition of statecharts. The root state for HFSM has to be an OR state. Moreover, HFSM does not allow shared events between the AND components which adds more restrictions to the modeling of plant. Based on the results of [26] and [68], [40] proposes a new STS modeling formalism that has solved the problems mentioned above. In this formalism, it is not an obligation to have an alternating layers of AND and OR super-states. We use the method proposed in [40] in this research.

This section is a brief mathematical description of STS defined in [40]. This review is mainly to introduce the notations. For details and examples, the reader is referred to [40]. In order to define STS, first the definition of State-Tree(ST) has to be introduced.

**Definition 2.5.** ***State-Tree (ST)*** *is a 4-tuple* $(X, x_0, \mathcal{T}, \varepsilon)$*, where*

1. *$X$ is an structured state set.*
2. *$x_0 \in X$ is the root state.*
3. *$\mathcal{T} : X \longrightarrow \{AND, OR, simple\}$ is the type function.*
4. *$\varepsilon : X \longrightarrow 2^X$ is the expansion function.*

where the expansion function is defined below:

$$(x \in X)\ \varepsilon(x) = \begin{cases} Y, & \text{(for some } \emptyset \subseteq Y \subseteq X \text{ such that } x \notin Y) \text{ if } \mathcal{T}(x) \in \{\text{AND}, \text{OR}\} \\ \emptyset, & \text{if } \mathcal{T}(x) = \text{simple} \end{cases} . \tag{2.23}$$

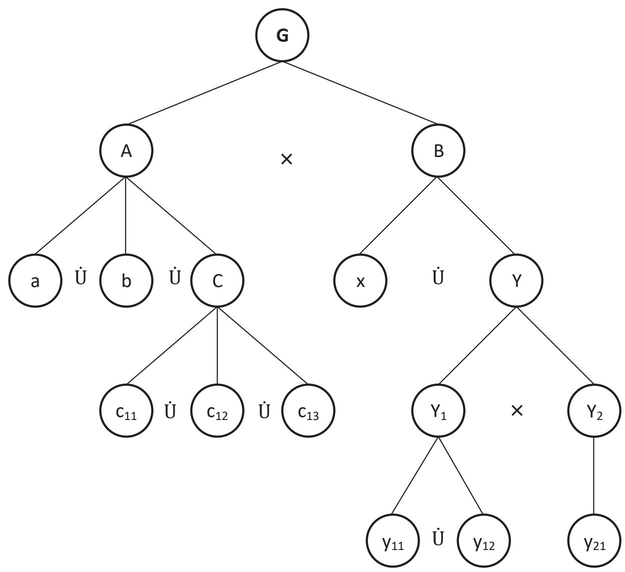



Figure 2.1: An Example of ST of a plant called **G**.

The *reflexive and transitive closure of* $\varepsilon(x)$ is shown by $\varepsilon^*(x) : X \rightarrow 2^X$. This definition can be extended to subsets of $x$:

$$\varepsilon^*(Y) = \bigcup_{x \in Y} \varepsilon^*(x), \ \forall Y \subseteq X. \tag{2.24}$$

Based on (2.24), $X = \varepsilon^*(x_0)$. Furthermore, $\varepsilon^+(x) = \varepsilon^*(x) - \{x\}$ includes all the descendants of $x \in X$. A state $x$ is called a *super-state* if $\varepsilon^+(x) \neq \emptyset$.

**Example 2.1.** *Consider the ST illustrated in Figure 2.1. In this example,* $\mathbf{G}$ *is the root state as well as an AND superstate* $\mathbf{G} = A \times B$ *(*$\varepsilon(G) = \{A, B\}$ *and* $\mathcal{T}(\mathbf{G}) = AND$*) where* $A = a \dot{\cup} b \dot{\cup} C$ *and* $B = x \dot{\cup} Y$ *are OR superstates. The expansion function for* $A$ *is* $\varepsilon(A) = \{a, b, C\}$*, and* $\varepsilon^*(A) = \{A, a, b, c_{11}, c_{12}, c_{13}\}$.

The restriction of $\mathcal{T}$ to $X' \subseteq X$ is shown by $\mathcal{T}_{X'} : X' \rightarrow$ {OR, AND, simple} and is defined below:

$$\forall x \in X', \ \mathcal{T}_{X'}(x) = \mathcal{T}(x). \tag{2.25}$$

Any 4-tuple ST $= (X, x_0, \mathcal{T}, \varepsilon)$ is a ST if it satisfies the following conditions.

1. (terminal case) $X = \{x_0\}$ or

2. (recursive tree) $(\forall y \in \varepsilon(x_0))$ $ST^y = (\varepsilon^*(y), y, \mathcal{T}_{\varepsilon^*(y)}, \varepsilon_{\varepsilon^*(y)})$ is a ST where $(\forall y, \ y' \in \varepsilon(x_0))$ $(y \neq y' \Rightarrow \varepsilon^*(y) \cap \varepsilon^*(y') = \emptyset)$ and $\dot{\bigcup}_{y \in \varepsilon(x_0)} \varepsilon^*(y) = \varepsilon^+(x_0)$.

A ST is called *well-formed* if no AND component is a simple state or in other words

$$(\forall x, y \in X) \text{ if } \mathcal{T}(x) = \text{AND} \ \& \ y \in \varepsilon(x) \implies \mathcal{T}(y) \in \{\text{OR, AND}\}. \tag{2.26}$$

For the rest of this thesis, we assume that all the STs are well-formed. If $y \in \varepsilon^*(x)$, we write $x \leq y$ and it indicates that either $x$ is an ancestor of $y$ ($x < y$) or $x = y$. For a ST $ST = (X, x_0, T, \varepsilon)$, ($\leq$) is a partial order on state space $X$ and $X$ is a poset.

The definitions of *Nearest Common Ancestor (NCA)* in ST is given below.

**Definition 2.6.** *Assume that $ST = (X, x_0, T, \varepsilon)$ is a ST and $x, y, z \in X$. $z$ is the* ***Nearest Common Ancestor (NCA)*** *of $x$ and $y$ if*

- *$z < x$ and $z < y$;*
- *$(\forall a \in \varepsilon^+(z))$ $a \not< x$ or $a \not< y$.*

In a ST, any two different states $a, b \in X$ are related through the following three options.

1. a is an ancestor of b ($a < b$) or vice versa ($b < a$).
2. a and b are parallel ($a \mid b$) : the system can be at both states a and b at the same time (NCA of $a$ and $b$ is an AND state).
3. a and b are exclusive ($a \oplus b$) : the system cannot be at both states a and b at the same time (NCA of $a$ and $b$ is an OR state).

The notion of *sub-ST* is defined below.

**Definition 2.7.** *Let $ST = (X, x_0, \mathcal{T}, \varepsilon)$ and $Y \subseteq X$. Suppose $ST' = (Y, x_0, \mathcal{T}', \varepsilon')$ is well-formed, ST' is a* ***sub-ST*** *of ST with $\mathcal{T}' : Y \to \{AND, OR, simple\}$ and $\varepsilon' : Y \to 2^Y$ if for any $y \in Y$,*

$$\mathcal{T}'(y) = \mathcal{T}(y) \tag{2.27}$$

$$\varepsilon'(y) = \begin{cases} \varepsilon(y), & \text{if } \mathcal{T}'(y) = AND \\ Z, & \big(\text{for some } \emptyset \subset Z \subseteq \varepsilon(y)\big) \text{ if } \mathcal{T}'(y) = OR \\ \emptyset, & \text{if } \mathcal{T}'(y) = simple \end{cases} . \tag{2.28}$$

The set of all sub-STs of ST is **ST**(ST) = {ST' | ST' is a sub-ST of ST}.

**Definition 2.8.** *Let $ST = (X, x_0, \mathcal{T}, \varepsilon)$ and $ST_1, ST_2 \in \boldsymbol{ST}(ST)$. Then*

$$ST_1 \le ST_2 \text{ if and only if } ST_1 \in \boldsymbol{ST}(ST_2). \tag{2.29}$$

For a ST, $(\le)$ is a partial order on **ST**(ST). Theorem 2.4 and 2.5 define the *conjunction* and *disjunction* of sub-STs.

**Theorem 2.4.** *Let $ST = (X, x_0, \mathcal{T}, \varepsilon)$ and $ST_1, ST_2 \in \boldsymbol{ST}(ST)$. In the poset $(\boldsymbol{ST}(ST), \le)$, the **conjunction** of two sub-STs $ST_1 \wedge ST_2$ is defined below.*

1. *If either $ST_1$ or $ST_2$ is empty, $ST_1 \wedge ST_2 = \emptyset$,*

2. *If $ST_1$ and $ST_2$ are not empty, then let $ST_1 = (X_1, x_0, \mathcal{T}_1, \varepsilon_1)$, $ST_2 = (X_2, x_0, \mathcal{T}_2, \varepsilon_2)$, and $ST_3 = ST_1 \wedge ST_2$. Here $ST_3 = (X_3, x_0, \mathcal{T}_3, \varepsilon_3)$ is defined by recursion. We only need to define $\varepsilon_3$ and $\mathcal{T}_3$ since $X_3 = \varepsilon_3^*(x_0)$ and $\mathcal{T}_3 = \mathcal{T}_{X_3}$ is the restriction of $\mathcal{T}$ to $X_3$. For $\varepsilon_3$, all the possible cases are*

   (a) *(terminal case): $\mathcal{T}(x_0) = simple$. Then $\varepsilon_3(x_0) = \emptyset$.*

   (b) *(recursive case 1): $\mathcal{T}(x_0) = OR$. Then $y \in \varepsilon_3(x_0)$ if and only if*

      i. *$y \in \varepsilon_1(x_0) \cup \varepsilon_2(x_0)$ and*

      ii. *$\varepsilon_3(y) = \emptyset \implies \mathcal{T}_3(y) = simple$.*

      *$ST_3 = \emptyset$ if and only if $\varepsilon_3(x_0) = \emptyset$.*

   (c) *(recursive case 2): $\mathcal{T}(x_0) = AND$. Then*

$$\varepsilon_3(x_0) = \begin{cases} \varepsilon_1(x_0) \cup \varepsilon_2(x_0) = \varepsilon_(x_0), & \text{if } (\forall y \in \varepsilon(x_0))\varepsilon_3(y) \neq \emptyset \\ \emptyset, & \text{otherwise} \end{cases}.$$

      *$ST_3 = \emptyset$ if and only if $\varepsilon_3(x_0) = \emptyset$.*

**Theorem 2.5.** *Let $ST = (X, x_0, \mathcal{T}, \varepsilon)$ and $ST_1, ST_2 \in \boldsymbol{ST}(ST)$. In poset $(\boldsymbol{ST}(ST), \le)$, the **disjunction** of two sub-STs $ST_1 \vee ST_2$ always exists and is defined below.*

1. *If $ST_1$ is empty, $ST_1 \vee ST_2 = ST_2$.*

2. *If $ST_2$ is empty, $ST_1 \vee ST_2 = ST_1$.*

3. *Assume that $ST_1$ and $ST_2$ are nonempty. Let $ST_1 = (X_1, x_0, \mathcal{T}_1, \varepsilon_1)$, $ST_2 = (X_2, x_0, \mathcal{T}_2, \varepsilon_2)$, and $ST_3 = ST_1 \vee ST_2$. Then $ST_3 = (X_3, x_0, \mathcal{T}_3, \varepsilon_3)$ is defined below.*

(a) $\forall x \in X_1 \cup X_2$,

$$\varepsilon_3(x) = \begin{cases} \varepsilon_1(x), & if\ x \in X_1 - X_2 \\ \varepsilon_2(x), & if\ x \in X_2 - X_1 \\ \varepsilon_1(x) \cup \varepsilon_2(x), & if\ x \in X_1 \cup X_2 \end{cases}.$$

(b) $X_3 = \varepsilon_3^*(x_0)$.

(c) $\mathcal{T}_3 = \mathcal{T}_{X_3}$, *the restriction of* $\mathcal{T}$ *to* $X_3$.

**Definition 2.9.** *Let* $ST = (X, x_0, \mathcal{T}, \epsilon)$. *The size of* $ST_1 \in \boldsymbol{ST}(ST)$ *is defined below.*

$$count(ST_1) = \begin{cases} \prod_{\forall y \in \varepsilon(x_0)} count(ST^y), & if\ \mathcal{T}(x_0) = AND \\ \sum_{\forall y \in \varepsilon(x_0)} count(ST^y), & if\ \mathcal{T}(x_0) = OR \\ 1, & if\ \mathcal{T}(x_0) = simple \end{cases}. \tag{2.30}$$

The notion of *basic-sub-ST* is defined below.

**Definition 2.10.** *Assume* $ST = (X, x_0, \mathcal{T}, \varepsilon)$ *be a ST.* $b \in \boldsymbol{ST}(ST)$ *is a* ***basic-sub-ST*** *of ST if* $count(b) = 1$. *The set of basic-sub-STs' of ST is* $B(ST) = \{b \mid count(b) = 1\}$.

A basic-sub-ST corresponds to the state of the flat automaton obtained from the hierarchical model described by STS. For the rest of this thesis, the term basic-ST is used in place of basic-sub-ST.

The notion of *holon* is used to describe the local (component) behavior of the plant.

**Definition 2.11.** ***Holon*** *is a 5-tuple* $(X, \Sigma, \delta, X_0, X_m)$ *where*

1. $X = X_E \dot{\cup} X_I$, $X_I$ *is the internal state set and* $X_E$ *is the external one. Note that* $X_E \cap X_I = \emptyset$.

2. $\Sigma = \Sigma_B \dot{\cup} \Sigma_I$, $\Sigma_I$ *is the internal event set and* $\Sigma_B$ *is the external one. Note that* $\Sigma_E \cap \Sigma_I = \emptyset$. *We can also partition* $\Sigma$ *into controllable and uncontrollable events,* $\Sigma = \Sigma_c \dot{\cup} \Sigma_{uc}$.

3. $\delta : X \times \Sigma \to X$ *is a partial function that defines transition structure. A transition from state x with event* $\sigma$ *is defined as:*

$$\delta(x,\sigma) = \begin{cases} \delta_I(x,\sigma), & if\ x \in X_I\ \&\ \sigma \in \Sigma_I \\ \delta_{BI}(x,\sigma), & if\ x \in X_E\ \&\ \sigma \in \Sigma_B \\ \delta_{BO}(x,\sigma), & if\ x \in X_I\ \&\ \sigma \in \Sigma_B \end{cases} \tag{2.31}$$

*where* $\delta_I(x,\sigma) : X_I \times \Sigma_I \to X_I$, $\delta_{BI}(x,\sigma) : X_E \times \Sigma_B \to X_I$, *and* $\delta_{BO}(x,\sigma) : X_I \times \Sigma_B \to X_E$ *are internal, incoming boundary, and outgoing boundary transition structures respectively.*

4. $X_0 \subseteq X_I$ *is the initial state set.*

5. $X_m \subseteq X_I$ *is the set of marked (terminal) states.*

STs and holons define the state space and local behavior of the plant respectively. STS deploys these two concepts to form a DES model that has modular and top-down structure.

**Definition 2.12.** ***STS*** *is a 6-tuple,* $(ST, H, \Sigma, \delta, ST_0, ST_m)$*, where*

1. $ST = (X, x_0, \mathcal{T}, \varepsilon)$ *is a ST.*

2. $H = \{H^a \mid \mathcal{T}(a) = OR \;\&\; H^a = (X^a, \Sigma^a, \delta^a, X_0^a, X_m^a)\}$ *defines the set of holons that are matched to each OR super-state in ST.*

3. $\Sigma = \bigcup_{\forall H^a \in H} \Sigma_I^a$ *is the set of all events that appears in H.*

4. $\Delta : \boldsymbol{ST}(ST) \times \Sigma \to \boldsymbol{ST}(ST)$ *is the transition function.*

5. $ST_0 \in \boldsymbol{ST}(ST)$ *is the initial ST.*

6. $ST_m \subseteq \boldsymbol{ST}(ST)$ *is the marked ST.*

Consider the Example 2.1. The STS model of Figure 2.1 is illustrated in Figure 2.2. In Figure 2.2, the plant **G** has two AND super-states (components) $A$ and $B$. The AND super-states are separated by dashed lines and the OR super-states (holons) are framed within a solid line. The states $C$, $Y$, and $c_{12}$ are the examples of OR states in Figure 2.2.

An STS is called deterministic, if all of its holons are deterministic.

**Definition 2.13.** *Consider* $H^x = (X^x, \Sigma^x, \delta^x, X_0^x, X_m^x)$ *matched to state x. The function* $\bar{\delta}_I^x : B(ST^x) \times \Sigma_I^x \to B(ST^x)$ *denotes the transition function defined in terms of (child) sub-STs.*

The *largest eligible* ST of **G** is formulated in the following definition.

**Definition 2.14.** *Let* $\boldsymbol{G} = (ST, H, \Sigma, \delta, ST_0, ST_m)$*,* $ST = (X, x_0, \mathcal{T}, \varepsilon)$*, and* $\sigma \in \Sigma$*. The* ***largest eligible ST*** *of* ***G****,* $Elig_{\boldsymbol{G}}(\sigma) : \Sigma \to \boldsymbol{ST}(ST)$*, is the largest sub-ST where* $\sigma$ *is allowed to happen. Define* $D_\sigma = \{x \mid \sigma \in \Sigma_I^x\}$ *to be the set of all OR super-states that have a holon assigned to them and* $\sigma$ *belongs to their internal event set. Then* $a \in Elig_{\boldsymbol{G}}(\sigma)$ *if and only if*

1. $(\forall x \in D_\sigma)\; a|x$*, or,*

2. $(\forall x \in D_\sigma)\; a \le x$*, or,*

3. $(\exists x \in D_\sigma,\; T \in B(ST^x))\; a \in T \;\&\; \bar{\delta}_I^x(T, \sigma)!$*.*

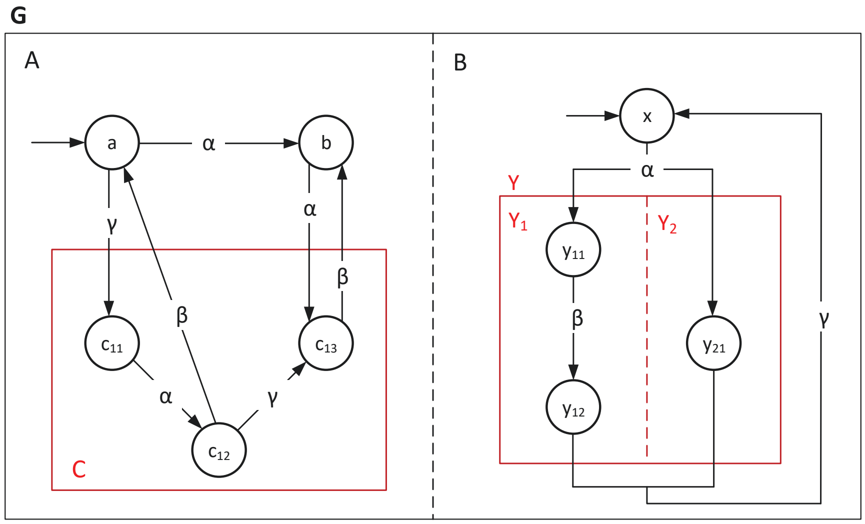



Figure 2.2: The STS model of Figure 2.1.

**Definition 2.15.** *Consider* $\mathbf{G} = (ST, H, \Sigma, \delta, ST_0, ST_m)$, $ST = (X, x_0, \mathcal{T}, \varepsilon)$, *and* $\sigma \in \Sigma$. *Define* $D_\sigma = \{x \mid \sigma \in \Sigma_I^x\}$. *The function* $replace\_source_{\mathbf{G},\sigma} : \mathbf{ST}(Elig_{\mathbf{G}}(\sigma)) \rightarrow \mathbf{ST}(ST)$ *maps a sub-ST of* $Elig_{\mathbf{G}}(\sigma)$ *to another sub-ST of ST. Let* $ST_2$ *denote* $replace\_source_{\mathbf{G},\sigma}(ST_1)$.

1. $(\forall x \in D_\sigma)\ a \in ST_1 \ \&\ a|x$, *or*

2. $(\forall x \in D_\sigma)\ a \in ST_1 \ \&\ a \leq x$, *or*

3. $(\exists x \in D_\sigma,\ b \in B(ST_1^x),\ b' \in ST^x)\ a \in b' \ \&\ b' = \bar{\delta}_I^x(b, \sigma)$

The *transition* function $\Delta(\text{ST})$ is formulated in the following definition.

**Definition 2.16.** *Let* $\mathbf{G} = (ST, H, \Sigma, \delta, ST_0, ST_m)$ *and* $ST = (X, x_0, \mathcal{T}, \varepsilon)$. *Assume that* $ST_1 \in \mathbf{ST}(ST)$ *and* $\sigma \in \Sigma$. *For* $ST_1$ *and* $\sigma$, *the* ***transition*** *function* $\Delta : \mathbf{ST}(ST) \times \Sigma \rightarrow \mathbf{ST}(ST)$ *is defined below.*

$$\Delta(ST_1, \sigma) = replace\_source_{\mathbf{G},\sigma}(ST_1 \wedge Elig_{\mathbf{G}}(\sigma)) \tag{2.32}$$

**Definition 2.17.** *If the transition function* $\Delta(.,.)$ *is in correspondence with the state transitions of the flat automaton, then* $\Delta(.,.)$ *is said to be* ***sound***.

**Lemma 2.3.** *For* $\Delta(.,.)$ *to be sound,* $replace\_source_{\mathbf{G},\sigma}(.)$ *and* $Elig_{\mathbf{G}}(.)$ *(for all* $\sigma \in \Sigma$*) have to be sound.*

**Lemma 2.4.** *For* $replace\_source_{\mathbf{G},\sigma}(.)$ *and* $Elig_{\mathbf{G}}(.)$ *to be sound, a set of sufficient conditions are:*

- *(soundness of* $replace\_source_{\mathbf{G},\sigma}(.)$*) every incoming boundary transition of the holon matched to an AND component must have a unique event label.*

- *(soundness of* $\mathrm{Elig}_{\mathbf{G}}(.)$*) every outgoing boundary transition of the holon matched to an AND component must have a unique event label.*

If the two conditions above do not met in a STS, then one can simply re-label the events to make the transition function $\Delta(.,.)$ be sound.

For example, in the STS $\mathbf{G}$ shown in figure 2.2, holon $Y$ is matched to the AND components $Y_1$ and $Y_2$. Holon $Y$ has just one incoming boundary transition with the event label $\alpha$ and one outgoing boundary transitions with the event label $\gamma$. Therefore, in $\mathbf{G}$, $replace\_source_{\mathbf{G},\sigma}(.)$ and $\mathrm{Elig}_{\mathbf{G}}(.)$ are sound. Thus, $\Delta(.,.)$ is not also sound. For the rest of this thesis, we assume that we are dealing with STSs that have a sound transition function $\Delta(.,.)$.

In the STS defined by [40], unlike the other hierarchical methods, holons can have shared events. However, they have to be assigned to the OR states that have the same AND super-state as their nearest common ancestor. The structure of STS has been briefly discussed in this section. More details can be found in [40] and [68].

In the STS framework, a *predicate* $P : B(\mathrm{ST}) \rightarrow \{0,1\}$ is a function that maps basic-STs to 0 (false) or 1 (true). A basic-ST $b$ is said to satisfy a predicate $P$ if $P(b) = 1$ and it is shown by $b \models P$. The set of all predicates defined on ST is Pred(ST). Here, $B_P = \{b \in B(\mathrm{ST}) | P(b) = 1\}$ is the set of basic-STs that satisfy $P$.

If one can find a $\mathrm{ST}_1 \in \mathbf{ST}(\mathrm{ST})$ such that $B_P = B(\mathrm{ST}_1)$, then it is said that $\mathrm{ST}_1$ identifies $P$.

In that sense, $\mathrm{ST}_0$ and $\mathrm{ST}_m$ identify $P_0$ and $P_m$ respectively, where $B_{P_0} = B(\mathrm{ST}_0)$ and $B_{P_m} = B(\mathrm{ST}_m)$. Let $P_0$ and $P_m$ denote the predicates corresponding to $\mathrm{ST}_0$ and $\mathrm{ST}_m$. For the rest of this section, we assume that $P \in \mathrm{Pred}(\mathrm{ST})$ and $\mathbf{G} = (\mathrm{ST}, H, \Sigma, \delta, P_0, P_m)$.

The *reachability predicate* $R(\mathbf{G},P)$ $\big(R(\mathbf{G},.) : \mathrm{Pred}(\mathrm{ST}) \rightarrow \mathrm{Pred}(\mathrm{ST})\big)$ represents the set of basic-ST that can be reached from $\mathrm{ST}_0$ via some basic-ST satisfying $P$; $R(\mathbf{G},P)$ is defined below.

- If $P \wedge P_0 =$ false, then $R(\mathbf{G},P) = \mathit{false}$, otherwise $\forall b \models P \wedge P_0$, $b \models R(\mathbf{G},P)$.
- $b \models R(\mathbf{G},P)$, $\sigma \in \Sigma$, $\Delta(b,\sigma) \neq \emptyset$ & $\Delta(b,\sigma) \models P \Rightarrow \Delta(b,\sigma) \models R(\mathbf{G},P)$.
- No other basic-ST satisfies $R(\mathbf{G},P)$.

The *coreachability predicate* $CR(\mathbf{G},P)$ $\big(CR(\mathbf{G},.) : \mathrm{Pred}(\mathrm{ST}) \rightarrow \mathrm{Pred}(\mathrm{ST})\big)$ represents the set of basic-ST that can reach at least one of the marked basic-ST in $\mathrm{ST}_m$ via some basic-ST satisfying $P$; $CR(\mathbf{G},P)$ is defined below.

- If $P \wedge P_m =$ false, then $CR(\mathbf{G},P) = \mathit{false}$, otherwise $(\forall b \in B(\mathrm{ST}))\ b \models P \wedge P_m \;\Rightarrow\; b \models CR(\mathbf{G},P)$.

- $b \models CR(\mathbf{G}, P)$, $\sigma \in \Sigma$, $\Delta(b', \sigma)!$, $\Delta(b', \sigma) = b$ & $b' \models P \Rightarrow b' \models CR(\mathbf{G}, P)$

- No other basic-ST satisfies $CR(\mathbf{G}, P)$.

## 2.5 Nonblocking Supervisory Control of State-Tree-Structure

In this thesis, we examine the robust state-based supervisory control of systems modeled by STS. In this section, we review some of the definitions, theorems, and lemmas related to the nonblocking supervisory control of STS by using the SFBC. Here we present some of previously discussed definitions in terms of ST.

Suppose that $P$ represents a sub-ST of ST. Let $b \in \mathbf{ST}(\text{ST})$ and $\sigma \in \Sigma$. The *weakest liberal preconditions* $M_\sigma : \text{Pred}(\text{ST}) \to \text{Pred}(\text{ST})$ is defined below.

$$b \models M_\sigma(P) \text{ if and only if } \Delta(b, \sigma) \models P. \tag{2.33}$$

For $b \in B(\text{ST})$, $M_\sigma(P)(b)$ is defined below.

$$M_\sigma(P)(b) = \begin{cases} 1, & \text{if either } \Delta(b,\sigma) \neq \emptyset \text{ \& } \Delta(b,\sigma) \models P, \text{ or } \Delta(b,\sigma) = \emptyset, \\ 0, & \text{otherwise} \end{cases}. \tag{2.34}$$

**Definition 2.18.** *$P$ is called **controllable** with respect to $\mathbf{G}$ if*

$$P \leq R(\mathbf{G}, P) \text{ \& } (\forall \sigma \in \Sigma_{uc})\ P \leq M_\sigma(P). \tag{2.35}$$

For automata, [74] defines a predicate transformer $\langle . \rangle$. We expand the definition to STS. For a predicate $P$, $\langle P \rangle \leq P$ is defined such that $b \models \langle P \rangle$ if

$$\forall w \in \Sigma_{uc}^*,\ \Delta(b, w) \neq \emptyset \Rightarrow \Delta(b, w) \models P. \tag{2.36}$$

**Definition 2.19.** *For a STS $\mathbf{G}$, $f : B(ST) \to \Pi$ represents a SFBC, where $\Pi = \{\Sigma' \subseteq \Sigma \mid \Sigma_{uc} \subseteq \Sigma'\}$. If $\sigma \in f(b)$, then $\sigma$ is enabled at $b$ in $\mathbf{G}$.*

For an event $\sigma \in \Sigma$, $f_\sigma : B(\text{ST}) \to \{0, 1\}$ is defined below.

$$f_\sigma(b) = 1 \text{ if and only if } \sigma \in f(b). \tag{2.37}$$

If $f$ is *nonblocking*, then the inequality below should be true.

$$R(\mathbf{G}^f, \text{true}) \leq CR(\mathbf{G}^f, \text{true}). \tag{2.38}$$

A predicate $P$ is called nonblocking for $\mathbf{G}$ if

$$R(\mathbf{G}, P) \leq CR(\mathbf{G}, P). \tag{2.39}$$

## 2.6 Binary Decision Diagram

Binary Decision Diagrams (BDDs) were proposed in [1] to represent Boolean functions. Later, [6] suggested using an ordering for the functions' input variables to accelerate the calculations further. Finding an efficient ordering of variables is one of the challenges of Ordered Binary Decision Diagram (OBDD). In the context of supervisory control, the structured state set of STS may offer suitable orderings. The results of some papers such as [40] and [45] strongly suggest that using BDD can hugely affect the calculations, especially in large and complex systems.

[1] and [6] developed a graphical presentation of Boolean functions using the Shannon's expansion. Assume that we have a set of variables $X = \{x_1, \dots, x_i, \dots, x_n\}$ and a Boolean function $f : 2^X \to \{0, 1\}$. Shannon's expansion can be written as:

$$f(x_1, \dots, x_n) = (x_i \wedge f|_{x_i=1}) \vee (\neg x_i \wedge f|_{x_i=0}), \ \forall x_i \in X, \tag{2.40}$$

where $\neg x_i$ is the negation of $x_i$, $f|_{x_i=1} = f(x_1, \dots, x_n)|_{x_i=1}$ and $f|_{x_i=0} = f(x_1, \dots, x_n)|_{x_i=0}$.

In BDD, there are two types of nodes: 1. decision nodes and 2. terminal nodes. Terminal nodes can be either 0 or 1.

**Example 2.2.** *Let us have $X = \{x_1, x_2\}$ and $f(x_1, x_2) = x_1 \vee x_2$. The expansion is:*

$$f(x_1, x_2) = \big(x_1 \wedge f(1, x_2)\big) \vee \big(\neg x_1 \wedge f(0, x_2)\big) = (x_1 \wedge 1) \vee (\neg x_1 \wedge x_2). \tag{2.41}$$

*Figure 2.3 shows the BDD graph of (2.41), where circles and squares represent decision and terminal nodes. The dashed line (solid line) shows that the variable's logical value is 0 (1).*

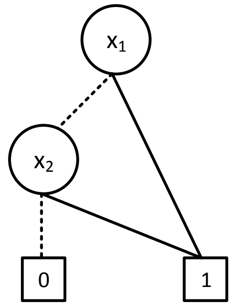



Figure 2.3: The BDD graph of $f = (x_1 \vee x_2)$.

## 2.7 Summary

In this chapter, we have reviewed some of the preliminaries used throughout this thesis. A brief review of DES, STS, and the supervisory control have been covered.

# Chapter 3

# Robust Nonblocking State-based Supervisory Control

In this chapter, the robust nonblocking supervisory control problem of DES is studied. In this framework, the plant model is unknown, but it is assumed to belong to a finite set of models. The safety requirements are expressed in terms of a set of safe states for each model. A set of necessary and sufficient conditions is obtained for the existence of a solution, and an algorithm is developed to calculate the supremal solution within a finite number of iterations. The resulting supervisor will be maximally permissive.

The rest of this chapter is organized as follows. In Section 3.1, the problem is formulated, and the MR property of automata is explained. Section 3.2 discusses some implications of MR property in automata and Section 3.3 defines the solution to the problem presented in Section 3.1. A simple illustrative example is provided in Section 3.5 and finally, the summary is given in Section 3.6.

## 3.1 Problem Formulation

In this section, we define our problem. Let us consider a DES plant and assume that due to some existing model uncertainty, the actual model of the plant belongs to a finite set of models $\mathcal{G} = \{\mathbf{G}_1, \ldots, \mathbf{G}_N\}$, where $\mathbf{G}_i = (Q_i, \Sigma_i, \delta_i, q_0, Q_{mi})$ for $i \in I = \{1, ..., N\}$. For each model, the design specification (safe states) is defined by a predicate $P_i$.

We assume that $\mathbf{G}_i$'s are reachable ($i \in I$). Moreover, we assume that for any pair of $\mathbf{G}_i, \mathbf{G}_j \in \mathcal{G}$, $\mathbf{G}_i$ and $\mathbf{G}_j$ are *Mutually Refined* (*MR*) ($i, j \in I$).

**Definition 3.1.** *([11]) Let $\mathbf{G}_1 = (Q_1, \Sigma_1, \delta_1, q_{01}, Q_{m1})$ and $\mathbf{G}_2 = (Q_2, \Sigma_2, \delta_2, q_{02}, Q_{m2})$ be two automata. $\mathbf{G}_1$ and $\mathbf{G}_2$ are Mutually Refined (**MR**) if*

1. $\forall s \in L(\mathbf{G}_1) \cap L(\mathbf{G}_2)$, $\delta_1(q_{01}, s) = \delta_2(q_{02}, s)$.
2. $\forall s \in L(\mathbf{G}_1) - L(\mathbf{G}_2)$ & $t \in L(\mathbf{G}_2)$, $\delta_1(q_{01}, s) \neq \delta_2(q_{02}, t)$.
3. $\forall s \in L(\mathbf{G}_2) - L(\mathbf{G}_1)$ & $t \in L(\mathbf{G}_1)$, $\delta_1(q_{01}, t) \neq \delta_2(q_{02}, s)$.

The first condition states that in two models $\mathbf{G}_1$ and $\mathbf{G}_2$, the corresponding states have the same label. Conditions (2) and (3) ensure that the solutions of robust control can be characterized by state feedback control [11]. We will elaborate on this issue and its importance in robust state-based supervisory control after the robust control problem is formally presented as Problem 3.1.

Note that in the non-trivial cases where $L(\mathbf{G}_1) \neq \emptyset$ and $L(\mathbf{G}_2) \neq \emptyset$, it follows from (1) in Definition 3.1 with $s = \epsilon$ that $q_{01} = q_{02}$.

The MR property defined in Definition 3.1 can easily be extended for more than two automata.

**Definition 3.2.** *Consider a finite set of automata $\mathcal{G} = \{\mathbf{G}_1, \ldots, \mathbf{G}_N\}$. We call these $N$ models MR if they are MR pairwise (i.e., for any $\mathbf{G}_i$ and $\mathbf{G}_j$ $(i, j \in I)$, $\mathbf{G}_i$ and $\mathbf{G}_j$ are MR).*

As we will see in Section 3.5, in fault recovery problems, $\mathbf{G}_i$'s are MR. If $\mathbf{G}_i$'s are not MR, there exists a procedure explained in Appendix 3.A that can be used to convert $\mathbf{G}_i$'s to MR automata.

In our problem, each $\mathbf{G}_i$ has its own event set $\Sigma_i$ and the controllability (or uncontrollability) of events does not change from one automaton to another. We want to find a SFBC $f$ for $\mathbf{G}$ such that $\mathbf{G}_i$ under the supervision of $f$, shown as $\mathbf{G}_i^f$, satisfies the specifications $P_i$ $(i \in I)$. Furthermore, we want to make sure that for any $i \in I$, $\mathbf{G}_i^f$ is nonblocking.

In the problem studied in this thesis, it is useful to consider predicates whose domain is a superset of the state set of an automaton. Consider an automaton $\mathbf{G}_1 = (Q_1, \Sigma_1, \delta_1, q_{01}, Q_{m1})$ and a state set $Q \supseteq Q_1$. Suppose $P$ is a predicate $P : Q \rightarrow \{0, 1\}$. We define the reachability and the coreachability predicates, namely $R(\mathbf{G}_1, P)$ and $\mathrm{CR}(\mathbf{G}_1, P)$ exactly as it was done in Section 2.1 and for brevity, we do not repeat them here. We observe that $R(\mathbf{G}_1, P)$ still corresponds to states that can be reached from $q_{01}$ using transitions in $\mathbf{G}_1$ via states that satisfy $P$ (of course, if $q_{01} \not\models P$, then $R(\mathbf{G}_1, P) = \text{false}$). Thus, $R(\mathbf{G}_1, P) \leq P_{Q_1}$, where $P_{Q_1}$ is the predicate identified by $Q_1$.

Furthermore, $\mathrm{CR}(\mathbf{G}_1, P)$ is satisfied, exactly on those states that can reach a state in $Q_{m1}$ using transitions in $\mathbf{G}_1$ via states satisfying $P$. Thus, $\mathrm{CR}(\mathbf{G}_1, P) \leq P_{Q_1}$.

$R(.,.)$ and $\mathrm{CR}(.,.)$ are still monotonically increasing functions. The following is an extension of Definition 2.3.

**Definition 3.3.** *Consider an automaton $\boldsymbol{G}$ with state set $Q_1$ and marked states $Q_{m1}$. Let $Q_1 \subseteq Q$ and $P \in Pred(Q)$. Predicate $P$ is nonblocking with respect to $\boldsymbol{G}_1$ or $\boldsymbol{G}_1$-nonblocking if*

$$R(\boldsymbol{G}_1, P) \leq \mathrm{CR}(\boldsymbol{G}_1, P). \tag{3.1}$$

Finally, for predicate $P \in \mathrm{Pred}(Q)$, denote the *restriction of* $P$ to $Q_1 \subseteq Q$ as $P|_{Q_1}$ and define it as $P|_{Q_1} : Q_1 \to \{0, 1\}$,

$$\forall\, q \in Q_1 \; q \models P|_{Q_1} \;\Leftrightarrow\; q \models P. \tag{3.2}$$

**Problem 3.1.** *(Robust Nonblocking State-based Supervisory Control Problem (RNSSCP)): Consider $N$ Mutually Refined (MR) models named $\boldsymbol{G}_i = (Q_i, \Sigma_i, \delta_i, q_0, Q_{mi})$ ($i \in I = \{1, \ldots, N\}$). There is a consistency in controllability/uncontrollability of events in automata. For each model $\boldsymbol{G}_i$, a safety predicate $P_i \in \mathrm{Pred}(\cup_{j \in I} Q_j)$ is assumed with $q_0 \models P_i$. Find a State Feedback Control (SFBC) $f : \bigcup_{i \in I} Q_i \to \Gamma$ such that*

1. *$R(\boldsymbol{G}_i^f, true) \leq P_i$ (safety property)*
2. *$R(\boldsymbol{G}_i^f, true) \leq CR(\boldsymbol{G}_i^f, true)$ (nonblocking property)*

In a conventional (in which the plant model is known) state-based supervisory control problem, the set of solutions can be characterized by sub-predicates of safety predicate. Each solution can be realized using a state feedback law. Therefore, in solving the control problem, one may only consider SFBC laws.

In a robust state-based control problem with $N$ models $\mathbf{G}_1, \ldots, \mathbf{G}_N$, each solution can be characterized by a set of $N$ suitable sub-predicates, one for each model. In this case, a solution can not always be realized with a state feedback law in general.

**Example 3.1.** *Let $\boldsymbol{G}_1$ and $\boldsymbol{G}_2$ in Figure 3.1 be the automata in a robust control problem. In $\boldsymbol{G}_1$ and $\boldsymbol{G}_2$, state 1 reaches to state 2 via two different events. Based on Definition 3.1, $\boldsymbol{G}_1$ and $\boldsymbol{G}_2$ are not MR. Suppose all events are controllable and safe in $\boldsymbol{G}_1$, but state 4 is unsafe for $\boldsymbol{G}_2$. Therefore, if state 2 is reached using string aa (in $\boldsymbol{G}_1$), a can remain enabled, and if state 2 is reached via string ab (in $\boldsymbol{G}_2$), then a should be disabled. Thus, one solution of robust control results in the removal of state 4 from reachable states. We observe that the control decision at state 2 depends not just on the current state but on the sequence of events that led to the current state. Therefore, a SFBC cannot remove state 4 without removing state 3 from reachable states.*

In this thesis, we want to set up the robust control problem in such a way that all solutions can be realized using state feedback. The mutual refinement condition ensures that there is enough information about the

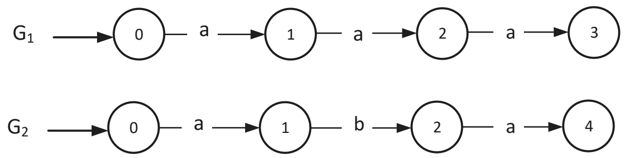



Figure 3.1: Example 3.1: The automata $\mathbf{G}_1$ and $\mathbf{G}_2$.

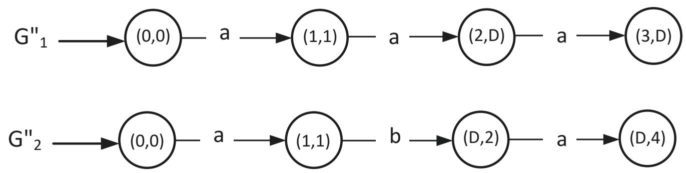



Figure 3.2: The result of applying Procedure 3.1 to automata $\mathbf{G}_1$ and $\mathbf{G}_2$ in Example 3.1.

dynamics of the model $\mathbf{G}_i$ in the state labels so that control decisions can be made based on the current state only. A procedure from [11] is provided in Appendix 3.A that refines the transition structures of $\mathbf{G}_i$'s to satisfy the MR property. We apply the Procedure 3.1 to $\mathbf{G}_1$ and $\mathbf{G}_2$ in Example 3.1. The MR automata $\mathbf{G}''_1$ and $\mathbf{G}''_2$ are shown in Figure 3.2. Now the previous state 2 is related to $(2,D)$ and $(D,2)$. In this case, a state feedback law to remove only state 4 (at $(D,2)$) can be used that keeps state $(3,D)$ reachable.

## 3.2 Implications of Mutually Refinement Property

This section discusses some implications of MR property in automata and introduces new definitions. The results of this section are used in Section 3.3 to explain the solution of RNSSCP. First, we merge all $N$ models defined in Section 3.1 to form a "union" automaton called $\mathbf{G}$.

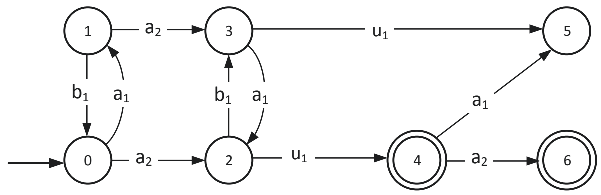
**Definition 3.4.** *Consider a finite set of MR models $\mathcal{G} = \{\mathbf{G}_1, \ldots, \mathbf{G}_N\}$, where $\mathbf{G}_i = (Q_i, \Sigma_i, \delta_i, q_0, Q_{mi})$ $(i \in I)$. Let $\mathbf{G}$ be an automaton such that $\mathbf{G} = (Q, \Sigma, \delta, q_0, Q_m)$, where $Q = \bigcup_{i\in I} Q_i$, $\Sigma = \bigcup_{i\in I} \Sigma_i$, $Q_m = \bigcup_{i\in I} Q_{mi}$ and $\delta : Q \to Q$ is defined below.*

- *For $q, q' \in Q$ and $\sigma \in \Sigma$, if for some $i \in I$, $\delta_i(q,\sigma) = q'$, then $\delta(q,\sigma)!$ and $\delta(q,\sigma) = q'$.*

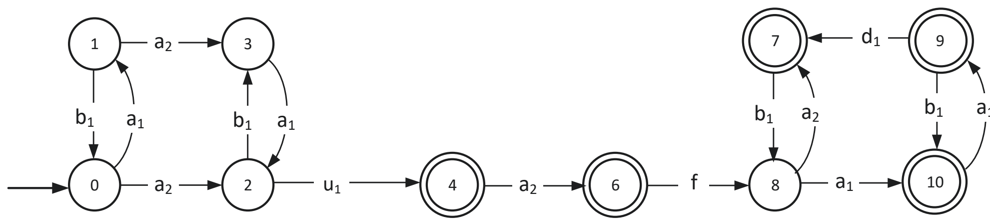
**Remark 3.1.** *Note that it follows from the MR property that if for $i, j \in I$, $\delta_i(q,\sigma) = q'$ and $\delta_j(q,\sigma)!$, then $\delta_i(q,\sigma) = \delta_j(q,\sigma) = q'$. Thus, in $\mathbf{G}$, transition $\delta(q,\sigma)$ has a unique target state and $\mathbf{G}$ is deterministic.*

It can easily be observed that $\mathbf{G}_i \subseteq \mathbf{G}$ $(i \in I)$.

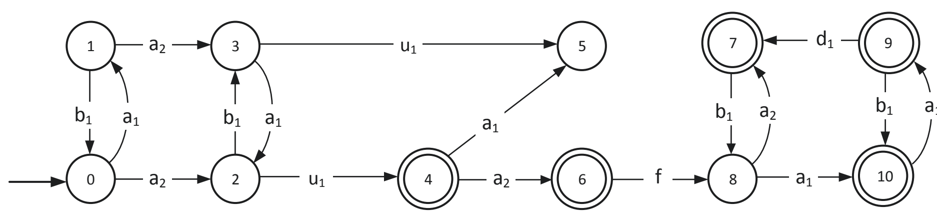
**Example 3.2.** *Assume that a plant has two possible MR models $\mathbf{G}_1$ and $\mathbf{G}_2$ shown in Figure 3.3a and 3.3b. In this example, the set of controllable events is $\Sigma_c = \{a_1, a_2, b_1\}$ and the set of uncontrollable events is $\Sigma_{uc} = \{\mathrm{f}, u_1, d_1\}$. The union automaton is shown in Figure 3.3c.*




(a) $\mathbf{G}_1$




(b) $\mathbf{G}_2$




(c) $\mathbf{G}$

Figure 3.3: Example 3.2: The two possible models of a plant and the union model $\mathbf{G}$.

**Lemma 3.1.** *Consider the set of MR models $\mathcal{G}$ and the automaton $\mathbf{G}$ in Definition 3.4. For any $q$, $q' \in Q$ and $s \in \Sigma^*$ such that $\delta(q,s) = q'$, there exists $i \in I$ such that $s \in \Sigma_i^*$ and $\delta_i(q,s) = \delta(q,s) = q'$.*

*Proof.*
For $s \in \epsilon$ the lemma is trivially true. Suppose $s \neq \epsilon$ and for some $n \geq 1$, $s = \sigma_0 \ldots \sigma_{n-1}$. Also there exists $q_1, \ldots, q_n \in Q$ with $\delta(q_l, \sigma_l) = q_{l+1}$ $(1 \leq l \leq n-1)$, $\delta(q, \sigma_0) = q_1$, and $q_n = q'$. Based on Definition 3.4, for each $l$, there exists $i_l \in I$ such that in $\mathbf{G}_{i_l}$, $\delta_{i_l}(q_l, \sigma_l)!$ and $\delta_{i_l}(q_l, \sigma_l) = \delta(q_l, \sigma_l) = q_{l+1}$. We prove that $\delta_{i_{n-1}}(q,s)!$ and $\delta_{i_{n-1}}(q,s) = \delta(q,s) = q'$.

Consider transitions $\delta_{i_0}(q, \sigma_0) = \delta(q, \sigma_0) = q_1$ and $\delta_{i_1}(q_1, \sigma_1) = \delta(q_1, \sigma_1) = q_2$. We claim that $\delta_{i_1}(q, \sigma_0)!$ and $\delta_{i_1}(q, \sigma_0) = \delta(q, \sigma_0) = q_1$. Assume that $\delta_{i_1}(q, \sigma_0)$ is not defined. Since $\mathbf{G}_{i_0}$ and $\mathbf{G}_{i_1}$ are reachable, there exists $s'_0 \in \Sigma_{i_0}^*$ and $s'_1 \in \Sigma_{i_1}^*$ such that $\delta_{i_0}(q_0, s'_0\sigma_0) = q_1$ and $\delta_{i_1}(q_0, s'_1) = q_1$ and $s'_0\sigma_0 \neq s'_1$. But by assumption, $\mathbf{G}_{i_0}$ and $\mathbf{G}_{i_1}$ are MR, and it follows from condition (2) in Definition 3.1 $\delta_{i_0}(q_0, s'_0\sigma_0) \neq \delta_{i_1}(q_0, s'_1)$, which is a contradiction. Therefore, $\delta_{i_1}(q, \sigma_0)!$. Now assume that $\delta_{i_1}(q, \sigma_0)!$ and $\delta_{i_1}(q, \sigma_0) \neq \delta(q, \sigma_0) = q_1$. Therefore, $\delta_{i_1}(q, \sigma_0) \neq \delta_{i_0}(q, \sigma_0)$. By assumption, $\mathbf{G}_{i_0}$ and $\mathbf{G}_{i_1}$ are MR, and it follows from condition (1) in Definition

3.1 that $\delta_{i_1}(q,\sigma_0) = \delta_{i_0}(q,\sigma_0)$, which is a contradiction. Therefore, $\delta_{i_1}(q,\sigma_0) = \delta(q,\sigma_0) = q_1$. So far, we have proved that $\delta_{i_1}(q,\sigma_0\sigma_1) = \delta(q,\sigma_0\sigma_1)$.

Now assume that $\delta_{i_l}(q,\sigma_0\ldots\sigma_l) = \delta(q,\sigma_0\ldots\sigma_l) = q_{l+1}$. We have to prove that $\delta_{i_{l+1}}(q,\sigma_0\ldots\sigma_{l+1}) = \delta(q,\sigma_0\ldots\sigma_{l+1})$ $= q_{l+2}$. We know that $\delta_{i_{l+1}}(q_{l+1},\sigma_{l+1}) = q_{l+2}$; therefore, we need to prove that $\delta_{i_{l+1}}(q,\sigma_0\ldots\sigma_1)!$ and $\delta_{i_{l+1}}(q,\sigma_0\ldots\sigma_l) = \delta(q,\sigma_0\ldots\sigma_l) = q_{l+1}$. Suppose that $\delta_{i_{l+1}}(q,\sigma_0\ldots\sigma_l)$ is not defined. Since $\mathbf{G}_{i_l}$ and $\mathbf{G}_{i_{l+1}}$ are reachable, there exists $s'_{i_l}$ and $s'_{i_{l+1}}$ such that $\delta_{i_l}(q_0,s'_{i_l}\sigma_0\ldots\sigma_l) = \delta_{i_{l+1}}(q_0,s'_{i_{l+1}}) = q_{l+1}$ and $s'_{i_l}\sigma_0\ldots\sigma_l \neq s'_{i_{l+1}}$. But this would contradict the assumption that $\mathbf{G}_{i_l}$ and $\mathbf{G}_{i_{l+1}}$ are MR. Now suppose that $\delta_{i_{l+1}}(q,\sigma_0\ldots\sigma_l)!$, but $\delta_{i_{l+1}}(q,\sigma_0\ldots\sigma_l) \neq \delta(q,\sigma_0\ldots\sigma_l) = q_{l+1}$. We know that $\delta_{i_l}(q,\sigma_0\ldots\sigma_l) = \delta(q,\sigma_0\ldots\sigma_l) = q_{l+1}$. Therefore, $\delta_{i_{l+1}}(q,\sigma_0\ldots\sigma_l) \neq \delta_{i_l}(q,\sigma_0\ldots\sigma_l)$. But this would also contradicts the assumption that $\mathbf{G}_{i_l}$ and $\mathbf{G}_{i_{l+1}}$ are MR.

Finally by induction, we can conclude that $\delta_{i_{n-1}}(q,s)!$ and $\delta_{i_{n-1}}(q,s) = \delta(q,s)$. □

Lemma 3.1 states that any sequence of events in the union model $\mathbf{G}$ belongs to at least one of the automaton in $\mathcal{G}$. In other words, by merging MR models, new sequences of events will not be generated. In Example 3.2, the sequence of $a_1a_2a_1u_1$, in $\mathbf{G}$, exists in both $\mathbf{G}_1$ and $\mathbf{G}_2$, and the sequence of $a_1a_2a_1u_1a_1$ only belongs to $\mathbf{G}_1$.

In the following lemma, we use Lemma 3.1 to prove that $\mathbf{G}$ and $\mathbf{G}_i$ are MR.

**Lemma 3.2.** *Consider the set of MR models $\mathcal{G}$ and automaton $\mathbf{G}$ defined in Definition 3.4. For any $i \in I$, $\mathbf{G}$ and $\mathbf{G}_i$ are MR.*

*Proof.*
We prove that $\mathbf{G}$ and $\mathbf{G}_i$ meet the three conditions mentioned in Definition 3.1.

1. Suppose $s \in L(\mathbf{G}) \cap L(\mathbf{G}_i)$. Thus, $\delta_i(q_0,s)!$ and $\delta(q_0,s)!$. It follows from Lemma 3.1 that $\delta_j(q_0,s) = \delta(q_0,s)$ for some $j \in I$. Since $\mathbf{G}_i$ and $\mathbf{G}_j$ are MR, $\delta_i(q_0,s) = \delta_j(q_0,s) = \delta(q_0,s)$.
2. Assume that $s \in L(\mathbf{G}) - L(\mathbf{G}_i)$ and $s = \sigma_0\sigma_1\ldots\sigma_{n-1}$ $(n \geq 1)$, where $\delta(q_l,\sigma_l) = q_{l+1}$ for $l \in \{0,\ldots,n-1\}$. Also let $t \in L(\mathbf{G}_i) \subseteq L(\mathbf{G})$. Based on Lemma 3.1, $\exists j \in I$ such that $\delta_j(q_0,s) = \delta(q_0,s)$ and $i \neq j$ since $s \notin L(\mathbf{G}_i)$. Since $\mathbf{G}_i$ and $\mathbf{G}_j$ are MR, then $\delta_i(q_0,t) \neq \delta_j(q_0,s)$ and $\delta_i(q_0,t) \neq \delta(q_0,s)$.
3. Condition (3) is trivially true since $L(\mathbf{G}_i) - L(\mathbf{G}) = \emptyset$.

□

In Lemmas 3.3 and 3.4, we prove that the MR property is preserved under the supervision of a SFBC $f : Q \to \Gamma$.

**Lemma 3.3.** *Consider the set of MR models $\mathcal{G}$ and automaton $\mathbf{G}$ defined in Definition 3.4. Let $f : Q \to \Gamma$ be an SFBC defined for $\mathbf{G}$. For any $i, j \in I$, if $s \in L(\mathbf{G}_i) \cap L(\mathbf{G}_j)$ and $s \notin L(\mathbf{G}_i^f)$, then $s \notin L(\mathbf{G}_j^f)$.*

*Proof.* Since $\mathbf{G}_i$ and $\mathbf{G}_j$ are MR, the sequence of states traversed in $\mathbf{G}_i$ and $\mathbf{G}_j$ using sequence $s$ are identical. Therefore, if the SFBC removes $s$ from $L(\mathbf{G}_i^f)$, it will do the same in $L(\mathbf{G}_j^f)$. □

**Lemma 3.4.** *Consider the set of MR models $\mathcal{G}$ and automaton $\mathbf{G}$ defined in Definition 3.4. Let $f : Q \to \Gamma$ be an SFBC defined for $\mathbf{G}$. For any $i, j \in I$, $\mathbf{G}_i^f$ and $\mathbf{G}_j^f$ are MR.*

*Proof.*
We need to prove that for $\mathbf{G}_i^f$ and $\mathbf{G}_j^f$, the three conditions in Definition 3.1 are met.

1. Suppose $s \in L(\mathbf{G}_i^f) \cap L(\mathbf{G}_j^f)$ and $\delta_i^f(q_{i0}, s) \neq \delta_j^f(q_{j0}, s)$. Therefore, there exists $q \in Q_i$ and $q' \in Q_j$ with $q \neq q'$ such that $\delta_i^f(q_{i0}, s) = q$ and $\delta_j^f(q_{j0}, s) = q'$. Since $\mathbf{G}_i^f \subseteq \mathbf{G}_i$ and $\mathbf{G}_j^f \subseteq \mathbf{G}_j$, we can conclude that $\delta_i(q_{i0}, s) = q$ and $\delta_j(q_{j0}, s) = q'$. We have assumed that $\mathbf{G}_i$ and $\mathbf{G}_j$ are MR. Therefore, we must have $\delta_i(q_{i0}, s) = \delta_j(q_{j0}, s)$, which is not possible. Thus, our assumption ($\delta_i^f(q_{i0}, s) \neq \delta_j^f(q_{j0}, s)$) is not true and we can conclude that $\forall s \in L(\mathbf{G}_i^f) \cap L(\mathbf{G}_j^f)$, $\delta_i^f(q_{i0}, s) = \delta_j^f(q_{j0}, s)$.

2. Suppose for some $s \in L(\mathbf{G}_i^f) - L(\mathbf{G}_j^f)$ and $t \in L(\mathbf{G}_j^f)$, we have $\delta_i^f(q_{i0}, s) = \delta_j^f(q_{j0}, t)$. Since $\mathbf{G}_i^f \subseteq \mathbf{G}_i$ and $\mathbf{G}_j^f \subseteq \mathbf{G}_j$, we have $\delta_i(q_{i0}, s)!$, $\delta_j(q_{j0}, t)!$, and $\delta_i(q_{i0}, s) = \delta_j(q_{j0}, t)$.

   Observe that $s \in L(\mathbf{G}_i^f)$ and $s \notin L(\mathbf{G}_j^f)$. Since $s \in L(\mathbf{G}_i^f)$, $s \in L(\mathbf{G}_i)$. Based on 3.3, we should have $s \notin L(\mathbf{G}_j)$, otherwise $s \in L(\mathbf{G}_i) \cap L(\mathbf{G}_j)$ and $s \notin L(\mathbf{G}_j^f)$; therefore, we would have $s \notin L(\mathbf{G}_i^f)$, which contradicts the assumption.

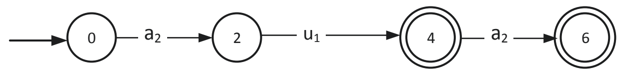
   Thus, for $s \in L(\mathbf{G}_i^f) - L(\mathbf{G}_j^f)$ and $t \in L(\mathbf{G}_j)$, we have $\delta_i(q_{i0}, s) = \delta_j(q_{j0}, t)$. But this is not possible, since $\mathbf{G}_i$ and $\mathbf{G}_j$ are MR. Therefore, condition (2) must be true.

3. Proof of condition (3) in Definition 3.1 is similar to the proof of condition (2).

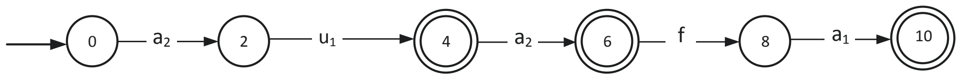
□

In Example 3.2, assume a SFBC $f$ defined for $\mathbf{G}$ (Figure 3.3c) such that for the reachable states under the supervision of $f$, we have $f(0) = \{\epsilon, a_2\}$, $f(2) = \{\epsilon, u_1\}$, $f(4) = \{\epsilon, a_2\}$, $f(6) = \{\epsilon, \text{f}\}$, $f(8) = \{\epsilon, a_1\}$, and $f(10) = \{\epsilon\}$. Figure 3.4 illustrates the three automata of Figure 3.3 under the supervision of $f$. As it can be seen, $\mathbf{G}^f$, $\mathbf{G}_1^f$ and $\mathbf{G}_2^f$ are MR.

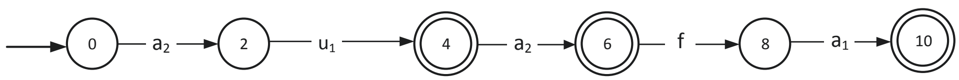
Now we define the relationship between the reachability and the coreachability predicates of $\mathbf{G}$ and $\mathbf{G}_i$s.




(a) $\mathbf{G}_1^f$




(b) $\mathbf{G}_2^f$




(c) $\mathbf{G}^f$

Figure 3.4: The automata models in Figure 3.3 under the supervision of a SFBC $f : Q \to \Gamma$.

**Lemma 3.5.** *Let $\mathbf{G}$ be the automaton defined in Definition 3.4. Then we have*

$$R(\mathbf{G}, true) = \bigvee_{i \in I} R(\mathbf{G}_i,\ true), \tag{3.3}$$

$$\mathrm{CR}(\mathbf{G}, true) = \bigvee_{i \in I} \mathrm{CR}(\mathbf{G}_i, true). \tag{3.4}$$

*Proof.*

1. We prove that, (i) $\bigvee_{i\in I} R(\mathbf{G}_i, true) \le R(\mathbf{G}, true)$ and (ii) $R(\mathbf{G}, true) \le \bigvee_{i\in I} R(\mathbf{G}_i, true)$.

   i. Assume $q \models \bigvee_{i\in I} R(\mathbf{G}_i, true)$, then $\exists j \in I$ such that $q \models R(\mathbf{G}_j, true)$. State $q$ is reachable in $\mathbf{G}_j$; thus, $\exists s \in \Sigma_j^*$ such that $\delta_j(q_0, s) = q$. Based on Definition 3.4, $s \in \Sigma^*$ and $\delta(q_0, s) = q$. Therefore, $q$ is also reachable in $\mathbf{G}$ and $q \models R(\mathbf{G}, true)$. We have proven that $\bigvee_{i\in I} R(\mathbf{G}_i, true) \le R(\mathbf{G}, \mathrm{true})$.

   ii. Assume $q \models R(\mathbf{G}, true)$. Therefore, $\exists t \in \Sigma^*$ such that in $\mathbf{G}$, $\delta(q_0, t) = q$. Based on Lemma 3.1, $\exists j \in I$ such that $\delta_j(q_0, t) = \delta(q_0, t) = q$. Thus, $q \models R(\mathbf{G}_j, \mathrm{true})$ and $q \models \bigvee_{i\in I} R(\mathbf{G}_i, true)$.
   Thus, we proved that $R(\mathbf{G},\ true) \le \bigvee_{i\in I} R(\mathbf{G}_i, true)$.

2. We prove that (i) $\bigvee_{i\in I} CR(\mathbf{G}_i,\ true) \le CR(\mathbf{G},\ true)$ and (ii) $CR(\mathbf{G}, true) \le \bigvee_{i\in I} CR(\mathbf{G}_i, true)$.

   i. The proof will be similar to section (i) in part 1 above.

   ii. Assume $q \models CR(\mathbf{G}, true)$, then $\exists q_m \in Q_m$ and $t = \sigma_0 \ldots \sigma_{n-1} \in \Sigma^*$ $(n \ge 1)$ such that $\delta(q, t) = q_m$. Similar to the proof of section (ii) in part 1, it can be shown that $\exists j \in I$ such that $\delta_j(q, t) = q_m$. Therefore, $q \models CR(\mathbf{G}_j, true)$ and $q \models \bigvee_{i\in I} CR(\mathbf{G}_i, true)$.

Thus, we proved that $CR(\mathbf{G}, true) \leq \bigvee_{i \in I} CR(\mathbf{G}_i, true)$.

□

**Remark 3.2.** *Using Lemma 3.4, we can easily show that the results of Lemmas 3.1 and 3.5 also hold for the automata under the supervision of a SFBC $f : Q \to \Gamma$. In particular,*

$$R(\mathbf{G}^f, true) = \bigvee_{i \in I} R(\mathbf{G}_i^f,\ true), \tag{3.5}$$

$$\mathrm{CR}(\mathbf{G}^f, true) = \bigvee_{i \in I} \mathrm{CR}(\mathbf{G}_i^f,\ true). \tag{3.6}$$

## 3.3 Solution: Necessary and Sufficient Conditions

In this section, we obtain the set of solutions of Robust Nonblocking State-based Supervisory Control Problem (RNSSCP). Theorem 3.1 is our main result. It presents a set of necessary and sufficient conditions for having a solution for RNSSCP.

**Theorem 3.1.** *Let $\mathbf{G}$ be the finite state automaton introduced in Definition 3.4. Define the predicate $P$ as*

$$P = \left[ \bigwedge_{j \in I} \left( P_j \vee \left[ R(\mathbf{G}, true) \wedge \neg R(\mathbf{G}_j, true) \right] \right) \right] \wedge R(\mathbf{G}, true). \tag{3.7}$$

1. *If there exists a predicate $K \leq P$ with $K \neq false$ such that*

   i. *$K$ is controllable with respect to $\mathbf{G}$,*

   ii. *$K$ is $\mathbf{G}_i$-nonblocking for all $i \in I$,*

   *then Robust Nonblocking State-based Supervisory Control Problem (RNSSCP) has a solution $f$ and $R(\mathbf{G}^f, true) = K$.*

2. *Conversely, if $f$ is a solution of RNSSCP, then $K$ defined as $K = R(\mathbf{G}^f, true)$ is controllable with respect to $\mathbf{G}$, $\mathbf{G}_i$-nonblocking for all $i \in I$ and $K \leq P$.*

To prove Theorem 3.1, we need the results in Lemmas 3.6 to 3.9. In Lemmas 3.6, 3.7, and 3.8, we have an automaton $\mathbf{G}_1$, which is a sub-automaton of another automaton $\mathbf{G}_2$. All predicates are defined over $Q_1 \cup Q_2 = Q_2$.

**Lemma 3.6.** *Consider two automata $\mathbf{G}_1$ and $\mathbf{G}_2$. Assume $\mathbf{G}_1$ is a sub-automaton of $\mathbf{G}_2$. Then $R(\mathbf{G}_1, true) \leq R(\mathbf{G}_2, true)$ and $CR(\mathbf{G}_1, true) \leq CR(\mathbf{G}_2, true)$.*

*Proof.*

1. Let $q \models R(\mathbf{G}_1, true)$. If $q = q_0$ (initial state), then obviously $q \models R(\mathbf{G}_2, \text{true})$. Suppose $q \neq q_0$. Therefore, $q \in Q_1$, $\exists q_1, \ldots, q_{n-1} \in Q_1$, and $\sigma_0 \ldots \sigma_{n-1} \in \Sigma_1^*$ $(n \geq 1)$ such that $\delta_1(q_l, \sigma_l) = q_{l+1}$ $(0 \leq l \leq n-2)$, $\delta_1(q_{n-1}, \sigma_{n-1}) = q$, and $q_l \models R(\mathbf{G}_1, true)$ for $l \in \{0, \ldots, n-1\}$. Based on the definition of sub-automaton in Section 2.1, since $\mathbf{G}_1 \subseteq \mathbf{G}_2$, then $q_0, \ldots, q_{n-1}, q \in Q_2$, $\sigma_0 \ldots \sigma_{n-1} \in \Sigma_2^*$, $\delta_2(q_l, \sigma_l) = q_{l+1}$ $(0 \leq l \leq n-2)$, and $\delta_2(q_{n-1}, \sigma_{n-1}) = q$. Therefore, $q_l \models R(\mathbf{G}_2, true)$ $(0 \leq l \leq n-1)$ and $q \models R(\mathbf{G}_2, \text{true})$. We can conclude that $R(\mathbf{G}_1, true) \leq R(\mathbf{G}_2, true)$.

2. Let $q \models CR(\mathbf{G}_1, true)$. If $q \in Q_{m1} \subseteq Q_{m2}$, then $q \models CR(\mathbf{G}_2, \text{true})$. Suppose $q \notin Q_{m1}$. Therefore, $\exists q_1, \ldots,$ $q_{m-1} \in Q_1$, $q_m \in Q_{m1}$, and $\sigma_0 \ldots \sigma_{m-1} \in \Sigma_1^*$ such that $\delta_1(q, \sigma_0) = q_1$, $\delta_1(q_l, \sigma_l) = q_{l+1}$ $(l \in \{1, \ldots, m-1\})$, and $q, q_l \models CR(\mathbf{G}_1, true)$ for $l \in \{1, \ldots, m\}$. Since $\mathbf{G}_1 \subseteq \mathbf{G}_2$, then $Q_{m1} \subseteq Q_{m2}$, $q, q_1, \ldots, q_m \in Q_2$, $\sigma_0 \ldots \sigma_{m-1} \in \Sigma_2^*$, $\delta_2(q, \sigma_0) = q_1$, and $\delta_2(q_l, \sigma_l) = q_{l+1}$ $(l \in \{1, \ldots, m-1\})$. Therefore, $q \models CR(\mathbf{G}_2, true)$ and we conclude that $CR(\mathbf{G}_1, true) \leq CR(\mathbf{G}_2, true)$.

□

We prove that under the conditions defined below, the relation between the reachability functions of two automata is not affected under the supervision of SFBC.

**Lemma 3.7.** *Consider $\mathbf{G}_1 = (Q_1, \Sigma_1, \delta_1, q_0, Q_{m1})$ and $\mathbf{G}_2 = (Q_2, \Sigma_2, \delta_2, q_0, Q_{m2})$. Suppose they are MR and $\mathbf{G}_1$ is a sub-automaton of $\mathbf{G}_2$. Assume $P \in \text{Pred}(Q)$ $(Q = Q_1 \cup Q_2 = Q_2)$, $P \neq false$ and $q_0 \models P$. Moreover, $P$ is controllable with respect to $\mathbf{G}_2$ and $f : Q_2 \to \Gamma$ a SFBC such that $R(\mathbf{G}_2^f, true) = P$. Then*

$$R(\mathbf{G}_1^f,\ true) \leq R(\mathbf{G}_2^f, true), \tag{3.8}$$

$$R(\mathbf{G}_1^f, true) = R(\mathbf{G}_2^f, true) \wedge R(\mathbf{G}_1, true), \tag{3.9}$$

$$R(\mathbf{G}_1^f, true) = R(\mathbf{G}_1, P). \tag{3.10}$$

*Proof.*

1. Since $\mathbf{G}_1$ is a sub-automaton of $\mathbf{G}_2$, $\mathbf{G}_1^f$ is a sub-automaton of $\mathbf{G}_2^f$. Hence, (3.8) follows.

2. We prove that (i) $R(\mathbf{G}_1^f, true) \leq R(\mathbf{G}_2^f, true) \wedge R(\mathbf{G}_1, true)$ and (ii) $R(\mathbf{G}_2^f, true) \wedge R(\mathbf{G}_1, true) \leq R(\mathbf{G}_1^f, true)$.

   i. We know that $R(\mathbf{G}_1^f, true) \leq R(\mathbf{G}_1, true)$ and we proved that $R(\mathbf{G}_1^f, true) \leq R(\mathbf{G}_2^f, true)$; therefore, we have $R(\mathbf{G}_1^f, true) \leq R(\mathbf{G}_2^f, true) \wedge R(\mathbf{G}_1, true)$.

ii. Assume $q \models R(\mathbf{G}_2^f, true) \wedge R(\mathbf{G}_1, true)$. Therefore, we have $q \models R(\mathbf{G}_2^f, true)$ and $q \models R(\mathbf{G}_1, true)$. We claim that $\exists s \in \Sigma_1^*$, such that $q = \delta_1(q_0, s)$ in $\mathbf{G}_1$ and $q = \delta_2^f(q_0, s)$ in $\mathbf{G}_2^f$, where $\delta_2^f(.,.)$ represents transitions in $\mathbf{G}_2$ under the supervision of $f$. If that is not the case, for every $s_1 \in L(\mathbf{G}_1)$ and $s_2 \in L(\mathbf{G}_2^f)$ such that $q = \delta_1(q_0, s_1)$, $q = \delta_2^f(q_0, s_2)$, $s_1 \notin L(\mathbf{G}_2^f)$, and $s_2 \notin L(\mathbf{G}_1)$. Since $L(\mathbf{G}_2^f) \subseteq L(\mathbf{G}_2)$, in $\mathbf{G}_2$, $q = \delta_2(q_0, s_2)$. But $\mathbf{G}_1$ and $\mathbf{G}_2$ are MR and this is not possible. So let $q_0, \ldots, q_{n-1}, q$ $(n \geq 1)$ be the sequence of states in $Q_1$ when $s$ is executed. Since the sequence is enabled under the supervision of $f$ (in $\mathbf{G}_2^f$), it remains enabled in $\mathbf{G}_1^f$. Therefore, we can conclude that $q \models R(\mathbf{G}_1^f, true)$.

Thus, we have proved that $R(\mathbf{G}_1^f, true) = R(\mathbf{G}_2^f, true) \wedge R(\mathbf{G}_1, true)$.

3. We know that $R(\mathbf{G}_2^f, true) = P$; therefore,

$$R(\mathbf{G}_1^f, true) = P \wedge R(\mathbf{G}_1, true) \qquad \text{(by (3.9))}$$

We prove that (i) $P \wedge R(\mathbf{G}_1, true) \leq R(\mathbf{G}_1, P)$ and (ii) $R(\mathbf{G}_1, P) \leq P \wedge R(\mathbf{G}_1, true)$.

i. We use strong induction. Base case: since $q_0 \models P$ and $q_0 \models R(\mathbf{G}_1, true)$, then $q_0 \models R(\mathbf{G}_1, P)$.

Strong inductive step: now we assume that $P \wedge R(\mathbf{G}_1, true) \leq R(\mathbf{G}_1, P)$ holds for all states that are located within a distance of $n$ transitions from $q_0$. The distance of a state $q$ from $q_0$ is defined as the shortest path to that state. We need to prove that $P \wedge R(\mathbf{G}_1, true) \leq R(\mathbf{G}_1, P)$ also holds for all states that are located within $n+1$ transitions from $q_0$ $(n \geq 0)$. Suppose $q_{n+1} \models P \wedge R(\mathbf{G}_1, true)$ and is at a distance of $n+1$ from $q_0$. Since $P$ is controllable and $R(\mathbf{G}_2^f, true) = P$; therefore, $\exists t \in \Sigma_2^*$ such that $q_{n+1} = \delta_2^f(q_0, t)$ and the trajectory on the $t$ sequence satisfies $P$. We have $q_{n+1} \models R(\mathbf{G}_1, true)$; moreover, $\mathbf{G}_1$ and $\mathbf{G}_2$ are MR. Therefore, $t \in L(\mathbf{G}_1)$ and the trajectory is in $R(\mathbf{G}_1, true)$. $q_{n+1}$ is reachable from $q_0$ and all the states leading to $q_{n+1}$ satisfy $P$; therefore, $q_{n+1} \models R(\mathbf{G}_1, P)$. $q_{n+1}$ is located within $n+1$ transitions from $q_0$ and satisfies $P \wedge R(\mathbf{G}_1, true) \leq R(\mathbf{G}_1, P)$. By the strong induction, we can say that $P \wedge R(\mathbf{G}_1, true) \leq R(\mathbf{G}_1, P)$ is true.

ii. It is clear that $R(\mathbf{G}_1, P) \leq P$ and $R(\mathbf{G}_1, P) \leq R(\mathbf{G}_1, true)$. Therefore, we have $R(\mathbf{G}_1, P) \leq P \wedge R(\mathbf{G}_1, true)$.

□

Remark 3.3 considers $\mathbf{G}_1$ and $\mathbf{G}_2$ in Lemma 3.7.

**Remark 3.3.** *By* (3.9),

$$R(\boldsymbol{G}_1^f, true) = R(\boldsymbol{G}_2^f, true) \wedge R(\boldsymbol{G}_1, true) = P \wedge P_{Q_1} \qquad \text{(Recall that } \mathbf{G}_1 \text{ and } \mathbf{G}_2 \text{ are reachable by assumption.)}$$

*Thus, $R(\mathbf{G}_1^f, true)$ is satisfied on all states in $Q_1$ that satisfy $P$. Thus, if we define $P_1 = P \wedge P_{Q_1}$, then $P_1$ as a predicate on $Q_1$ is controllable. More precisely, if $f_1$ is the restriction of $f$ to $Q_1$ and $P_1|_{Q_1}$ is the restriction of $P_1$ to $Q_1$, then*

$$R(\mathbf{G}_1^{f_1}, true) = P_1|_{Q_1} = (P \wedge P_{Q_1})|_{Q_1} = P|_{Q_1}.$$

*In other words, if $P$ is controllable with respect to $\mathbf{G}_2$, $P|_{Q_1}$ is controllable with respect to $\mathbf{G}_1$.*

Results similar to those of Lemma 3.7 hold for coreachability predicate.

**Lemma 3.8.** *Consider $\mathbf{G}_1 = (Q_1, \Sigma_1, \delta_1, q_0, Q_{m1})$ and $\mathbf{G}_2 = (Q_2, \Sigma_2, \delta_2, q_0, Q_{m2})$. Suppose they are MR and $\mathbf{G}_1$ is a sub-automaton of $\mathbf{G}_2$. Assume $P \in \mathrm{Pred}(Q)$ ($Q = Q_1 \cup Q_2 = Q_2$), $P \neq false$ and $q_0 \models P$. Moreover, $P$ is controllable and nonblocking with respect to $\mathbf{G}_1$. Let $f$ be SFBC $f : Q_2 \to \Gamma$ such that $R(\mathbf{G}_2^f, true) = P$. Then*

$$CR(\mathbf{G}_1^f,\ true) \leq CR(\mathbf{G}_2^f, true), \tag{3.11}$$

$$CR(\mathbf{G}_1^f, true) \leq CR(\mathbf{G}_2^f, true) \wedge CR(\mathbf{G}_1, true), \tag{3.12}$$

$$CR(\mathbf{G}_1, P) \leq CR(\mathbf{G}_1^f, true). \tag{3.13}$$

*Proof.*

1. (3.11) follows from the fact that $\mathbf{G}_1^f$ is a sub-automaton of $\mathbf{G}_2$.
2. (3.12) follows from (3.11) and that $\mathbf{G}_1^f$ is a sub-automaton of $\mathbf{G}_1$.
3. $P$ is controllable; therefore, $P|_{Q_1}$ is controllable (Remark 3.3). $P$ is $\mathbf{G}_1$-nonblocking

   $$R(\mathbf{G}_1, P) \leq CR(\mathbf{G}_1, P).$$

   Intuitively, $R(\mathbf{G}_1, P)|_{Q_1} = R(\mathbf{G}_1, P|_{Q_1})$ and $CR(\mathbf{G}_1, P)|_{Q_1} = CR(\mathbf{G}_1, P|_{Q_1})$. Thus,

   $$R(\mathbf{G}_1, P|_{Q_1}) \leq CR(\mathbf{G}_1, P|_{Q_1}).$$

   With $f_1 = f|_{Q_1}$ and using Theorem 2.3,

   $$CR(\mathbf{G}_1, P|_{Q_1}) \leq CR(\mathbf{G}_1^{f_1}, \mathrm{true}),$$

   and thus,

   $$CR(\mathbf{G}_1, P) \leq CR(\mathbf{G}_1^f, \mathrm{true}).$$

□

**Lemma 3.9.** *Consider the set of MR models $\mathcal{G}$ and automaton $\mathbf{G}$ that are defined in Definition 3.4. Suppose $K \in Pred(Q)$, $K \leq R(\mathbf{G}, true)$ and let $K_i = \big(K \wedge R(\mathbf{G}_i, true)\big)|_{Q_i}$. If $K$ is controllable with respect to $\mathbf{G}$, then $K_i$ is controllable with respect to $\mathbf{G}_i$ ($i \in I$).*

*Proof.* Suppose $K$ is controllable with respect to $\mathbf{G}$. We have to prove that $K_i$ is controllable with respect to $\mathbf{G}_i$ ($i \in I$). From controllability of $K$, we can conclude that there exists a SFBC $f$ such that $R(\mathbf{G}^f, \text{true}) = K$. By Lemma 3.2, $\mathbf{G}$ and $\mathbf{G}_i$ are MR. Thus, applying Lemma 3.7 and Remark 3.3 to $\mathbf{G}_i$ and $\mathbf{G}$, we can conclude that $K_i = (K \wedge R(\mathbf{G}_i, \text{true}))|_{Q_i}$ is controllable with respect to $\mathbf{G}_i$. □

Now we can prove Theorem 3.1.

*Proof of Theorem 3.1.*

1. Since $K$ is controllable with respect to $\mathbf{G}$ by assumption, by Theorem 2.2, there exists a SFBC $f$ such that

$$R(\mathbf{G}^f, true) = K \tag{3.14}$$

From assumption (ii), $R(\mathbf{G}_i, K) \leq CR(\mathbf{G}_i, K)$ ($i \in I$).

$$\bigvee_{i \in I} R(\mathbf{G}_i, K) \leq \bigvee_{i \in I} CR(\mathbf{G}_i, K)$$

$$\bigvee_{i \in I} R(\mathbf{G}_i^f, \text{true}) \leq \bigvee_{i \in I} CR(\mathbf{G}_i^f, \text{true}) \qquad \text{(by Lemmas 3.7 and 3.8)}$$

$$R(\mathbf{G}^f, \text{true}) \leq CR(\mathbf{G}^f, \text{true}) \qquad \text{(by Remark 3.2)}$$

Thus, $K$ is nonblocking with respect to $\mathbf{G}$. Now we show that $f$ is a solution to RNSSCP, i.e., conditions

(1) and (2) in Problem 3.1 are true.

$$\begin{aligned}
R(\mathbf{G}_i^f, true) &= R(\mathbf{G}^f, true) \wedge R(\mathbf{G}_i, true) && \text{(by Lemma 3.7)}\\
&= K \wedge R(\mathbf{G}_i,\ true) && \text{(by (3.14))}\\
&\leq P \wedge R(\mathbf{G}_i, true)\\
&\leq \Big(P_i \vee \big[R(\mathbf{G}, true) \wedge \neg R(\mathbf{G}_i, true)\big]\Big) \wedge R(\mathbf{G}_i, true)\\
&= \big(P_i \wedge R(\mathbf{G}_i, true)\big) \vee \Big(R(\mathbf{G}, true) \wedge \neg R(\mathbf{G}_i, true) \wedge R(\mathbf{G}_i, true)\Big)\\
&= P_i \wedge R(\mathbf{G}_i, true)\\
&\leq P_i
\end{aligned}$$

Now we just need to prove that $R(\mathbf{G}_i^f, true) \leq CR(\mathbf{G}_i^f, true)$.

$$\begin{aligned}
R(\mathbf{G}_i^f, true) &= R(\mathbf{G}_i, K) && \text{(by Lemma 3.7)}\\
&\leq CR(\mathbf{G}_i, K) && (K \text{ is } \mathbf{G}_i\text{-nonblocking})\\
&\leq CR(\mathbf{G}_i^f, true) && \text{(by Lemma 3.8)}
\end{aligned}$$

2. Since $K = R(\mathbf{G}^f, true)$ and by Theorem 2.2, $K$ is controllable with respect to $\mathbf{G}$. Since $f$ solves the RNSSCP, $R(\mathbf{G}_i^f, \text{true}) \leq CR(\mathbf{G}_i^f, \text{true})$. By Lemma 3.7, $R(\mathbf{G}_i^f, \text{true}) = K \wedge R(\mathbf{G}_i, \text{true})$. Define $K_i = K \wedge R(\mathbf{G}_i, \text{true})|_{Q_i}$ and $f_i = f|_{Q_i}$. Thus, $R(\mathbf{G}_i^{f_i}, \text{true}) = K_i$ and by Theorem 2.2,

$$R(\mathbf{G}_i^{f_i}, \text{true}) = R(\mathbf{G}_i, K_i) = CR(\mathbf{G}_i, K_i). \tag{3.15}$$

Note that the domain of the above predicates are $Q_i$. From (3.15), we conclude that

$$R(\mathbf{G}_i, K \wedge R(\mathbf{G}_i, \text{true})) = CR(\mathbf{G}_i, K \wedge R(\mathbf{G}_i, \text{true})).$$

Since states that are satisfying $K$, but they are not in $Q_i$ do not satisfy the above reachability and coreachability predicates, we can conclude

$$R(\mathbf{G}_i, K) = CR(\mathbf{G}_i, K).$$

Next we will prove that $K \leq P$. Observe that

$$\begin{aligned}
P_i \vee \Big[R(\mathbf{G}, true) \wedge \neg R(\mathbf{G}_i, true)\Big] &= \Big[P_i \vee R(\mathbf{G}, true)\Big] \wedge \Big[P_i \vee \neg R(\mathbf{G}_i, true)\Big] \\
&\geq \Big[R(\mathbf{G}_i^f, \mathrm{true}) \vee R(\mathbf{G}, \mathrm{true})\Big] \wedge \Big[R(\mathbf{G}_i^f, \mathrm{true}) \vee \neg R(\mathbf{G}, \mathrm{true})\Big] \\
&\geq \Big[\Big(R(\mathbf{G}_i, true) \wedge R(\mathbf{G}^f, true)\Big) \vee R(\mathbf{G}, true)\Big] \wedge \\
&\quad \Big[\Big(R(\mathbf{G}_i, true) \wedge R(\mathbf{G}^f, true)\Big) \vee \neg R(\mathbf{G}_i, true)\Big] \qquad \text{(Lemma 3.7)} \\
&= R(\mathbf{G}, true) \wedge \Big[R(\mathbf{G}^f, true) \vee \neg R(\mathbf{G}_i, true)\Big] \\
&= R(\mathbf{G}^f, true) \vee \Big(\neg R(\mathbf{G}_i, true) \wedge R(\mathbf{G}, true)\Big) \\
&\geq R(\mathbf{G}^f, true) \\
&= K.
\end{aligned}$$

Using the above result, we can conclude that

$$\begin{aligned}
P &= \Big[\bigwedge_{j \in I} \Big(P_j \vee \Big[R(\mathbf{G}, true) \wedge \neg R(\mathbf{G}_j, true)\Big]\Big)\Big] \wedge R(\mathbf{G}, true) \\
&\geq K \wedge R(\mathbf{G}, true) \\
&= K. \qquad \text{(Since } R(\mathbf{G}, true) \geq R(\mathbf{G}^f, true) = K\text{)}
\end{aligned}$$

□

We define the set of all controllable and $\mathbf{G}_i$-nonblocking predicates of $P$ as $\mathrm{CNb_GP}(P)$,

$$\mathrm{CNb_GP}(P) = \{K \in \mathrm{Pred}(Q) \mid K \leq P \;\&\; K \text{ controllable with respect to } \mathbf{G} \text{ and } \mathbf{G}_i\text{-nonblocking } \forall\, i \in I\}. \quad (3.16)$$

**Lemma 3.10.** $\mathrm{CNb_GP}(P)$ *is nonempty, closed under disjunction operation and has a supremal element.*

*Proof.*
*Claim 1.* $\mathrm{CNb_GP}(P)$ is nonempty since $false \in \mathrm{CNb_GP}(P)$.
*Claim 2.* Suppose $\Lambda$ is the index set of $\mathrm{CNb_GP}(P)$ and $K_\lambda \in \mathrm{CNb_GP}(P)$ for all $\lambda \in \Lambda$. We have to prove that $K = \bigvee_{\lambda \in \Lambda} K_\lambda \in \mathrm{CNb_GP}(P)$, i.e., $K \leq P$, $K$ is controllable with respect to $\mathbf{G}$ and $\mathbf{G}_i$-nonblocking. It is obvious that $K \leq P$ and [74] proved that $K$ is controllable with respect to $\mathbf{G}$. Therefore, we only need to prove that $K$ is $\mathbf{G}_i$-nonblocking. In other words, we want to prove that $R(\mathbf{G}_i, K) \leq CR(\mathbf{G}_i, K)$ for all $i \in I$. Assume $q \models R(\mathbf{G}_i, K)$; therefore, $K_\lambda$ is controllable with respect to $\mathbf{G}$. It follows from Lemma 3.9 that $\big(K_\lambda \wedge R(\mathbf{G}_i, \mathrm{true})\big)|_{Q_i}$ is controllable with respect to $\mathbf{G}_i$. Thus $q$ is reachable in $\mathbf{G}_i$ using a trajectory

that lies in $\big(K_\lambda \wedge R(\mathbf{G}_i, \text{true})\big)|_{Q_i}$. Therefore, $q \models K$ and $q \models R(\mathbf{G}_i, true)$. We know that $K = \bigvee_{\lambda\in\Lambda} K_\lambda$; thus, $\exists\lambda \in \Lambda$ such that $q \models K_\lambda$. We have $q \models R(\mathbf{G}_i, true)$ and $q \models K_\lambda$; therefore, $q \models R(\mathbf{G}_i, K_\lambda)$. Since $K_\lambda$ is $\mathbf{G}_i$-nonblocking ($R(\mathbf{G}_i, K_\lambda) \leq CR(\mathbf{G}_i, K_\lambda)$), $q \models CR(\mathbf{G}_i, K_\lambda)$. $CR(.,.)$ is a monotonically increasing function and; therefore, $CR(\mathbf{G}_i, K_\lambda) \leq CR(\mathbf{G}_i, K)$. Thus, we can conclude that $q \models CR(\mathbf{G}_i, K)$ and $R(\mathbf{G}_i, K) \leq CR(\mathbf{G}_i, K)$. $\square$

Let $K^\uparrow$ denotes $\text{supCNb}_\text{G}\text{P}(P)$. $K^\uparrow$ characterizes the largest (maximally permissive) solution of the robust supervisory control problem. In the next section, we present a computational procedure for $K^\uparrow$.

## 3.4 Solution: Computational Procedure

The following theorem presents an algorithm to calculate the supremal solution of Theorem 3.1, $K^\uparrow$.

**Theorem 3.2.** *Assume that $\mathbf{G}$ is the finite state automaton introduced in Definition 3.4 and $P$ is the predicate in (3.7). Then $K^\uparrow = \text{supCNb}_\text{G}\text{P}(P)$ can be calculated using the following iterative procedure which terminates in a finite number of steps less than or equal to the number of states satisfying $P$.*

1. *Set $r = 1$ and $S_r = P$.*
2. *$L_i = CR(\mathbf{G}_i, S_r)$ for all $i \in I$.*
3. *$S'_r = \Big[\bigwedge_{i\in I} L_i\Big] \vee \Big[\bigvee_{i\in I}\Big(L_i \wedge \neg(\bigvee_{j\in I\ \&\ j\neq i} P_{Q_j})\Big)\Big]$.*
4. *$S_{r+1} = R(\mathbf{G}, \langle S'_r\rangle)$.*
5. *If $S_{r+1} \neq S_r$, set $r = r+1$ and go to step 2.*
6. *End ($S_r = K^\uparrow$).*

*Here $\langle . \rangle$ is calculated with respect to $\mathbf{G}$ and $P_{Q_j}$ is a predicate that represents the states of $\mathbf{G}_j$ ($j \in I$).*

*Proof.*
First we prove that the algorithm converges in a finite number of iterations. In this chapter, the plant models $\mathbf{G}_i$ are finite-state automata; therefore, the set of predicates $\text{Pred}(Q)$ is a finite set. Furthermore, for each iteration, $L_i \leq S_r$ ($i \in I$); therefore, $\bigwedge_{i\in I} L_i \leq S_r$ and $\bigvee_{i\in I} L_i \leq S_r$. It can easily be seen that $\bigvee_{i\in I}\Big(L_i \wedge \neg(\bigvee_{j\in I\ \&\ j\neq i} P_{Q_j})\Big) \leq \bigvee_{i\in I} L_i$. Therefore, we can conclude that $S'_r \leq S_r$. Based on step 4, we also have $S_{r+1} \leq S'_r \leq S_r$. Therefore, this algorithm is nonincreasing and will converge to either $K^\uparrow = false$ or $K^\uparrow \neq false$ in a finite number of iterations less than or equal to the number of states satisfying $S_1 = P$.

Now suppose that the algorithm converges to $S_m$ for some $m \geq 1 : S_m = S'_m = S_{m+1}$. We prove that (i) $S_m \leq P$, (ii) $S_m$ is controllable with respect to $\mathbf{G}$ and (iii) $S_m$ is $\mathbf{G}_i$-nonblocking for all $i \in I$.

i. We have proved that our algorithm produces a nonincreasing sequence of predicates; therefore, after $m$ iteration, we have $S_m \leq S_{m-1} \leq \cdots \leq S_1 = P$.

ii. Based on Lemma 2.2, $S_{m+1}$ is the supremal controllable sub-predicate of $S'_m$. Therefore, $S_{m+1}$ is controllable with respect to $\mathbf{G}$. Since $S_{m+1} = S_m$, we can conclude that $S_m$ is controllable with respect to $\mathbf{G}$.

iii. We have to prove that $R(\mathbf{G}_k, S_m) \leq CR(\mathbf{G}_k, S_m)$ for all $k \in I$. Let us fix $k$. Assume $q \models R(\mathbf{G}_k, S_m)$, then $q \models S_m \wedge R(\mathbf{G}_k, true)$. Therefore, $\exists t = \sigma_0 \ldots \sigma_{n-1} \in \Sigma_k^*$ such that starting from $q_0$, we pass through $q_0, \ldots, q_{n-1} \in Q_k$ and reach $q$, where $\delta_k(q_l, \sigma_l) = q_{l+1}$ $(l \in \mathcal{L} = \{0, \ldots, n-2\})$ and $\delta_k(q_{n-1}, \sigma_{n-1}) = q$. Moreover, $q_l,, q_{n-1}, q \models S_m$ and $q_l, q_{n-1} \models R(\mathbf{G}_k, true)$ for all $l \in \mathcal{L}$. Since $S_m = S'_m$, we can conclude that $q_0, \ldots, q_{n-1}, q \models S'_m$. Based on the step 3 of algorithm, $q_0, \ldots, q_{n-1}, q \models \bigwedge_{k \in I} L_k$ or $q_0, \ldots, q_{n-1}, q \models \bigvee_{k \in I} \left(L_k \wedge (\bigwedge_{j \in I \ \& \ j \neq k} \neg P_{Q_j})\right)$. Either way, $q_0, \ldots, q_{n-1}, q \models (L_k = CR(\mathbf{G}_k, S_m))$. Therefore, $R(\mathbf{G}_k, S_m) \leq CR(\mathbf{G}_k, S_m)$.

Now let the iterative steps (2) to (4) in Theorem 3.2 be represented by an operative $\Psi(.)$. In steps (ii) and (iii) above, we showed that every fix-point of $\Psi(P_1)$ is controllable and $\mathbf{G}_i$-nonblocking. If $S_m \neq K^{\uparrow}$, then $S_m \leq K^{\uparrow} \leq P$. Thus, $\Psi(S_m) \leq \Psi(K^{\uparrow}) \leq \Psi(P)$ and $S_m \leq K^{\uparrow} \leq \Psi(P)$. Apply $\Psi(.)$ $m-1$ times; $S_m \leq K^{\uparrow} \leq \Psi^{(m-1)}(P) = S_m$ ($\Psi^{(m)}(.)$ denotes that $\Psi(.)$ is applied $m$ times). Thus $S_m = K^{\uparrow}$. □

## 3.5 Example

Some of the fault-tolerant control problems can be formulated as robust supervisory control problems. For instance, suppose we have a plant G and assume it starts in normal (N) mode. Later, the plant may face a failure (for the sake of simplicity, assume one failure), and it enters the failure mode (F). The overall model of the plant (in normal and faulty modes) is $\mathbf{G}_{NF}$. For each mode, the plant has different sets of marked states (for the sake of simplicity, assume one for each, $Q_{mN}$ and $Q_{mNF}$) and different specifications. In the fault-tolerant control problem, we want to find a supervisor (here a state feedback law $f$) such that

1. The nonblocking property of the normal and the faulty modes are guaranteed.

$$R(\mathbf{G}_N^f, \text{true}) \leq CR(\mathbf{G}_N^f, \text{true}),$$

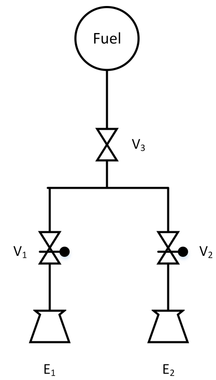



Figure 3.5: A propulsion system of a monopropellant rocket.

Table 3.1: All the events in Figure 3.6 and their controllability status.

| Label | Event | Controllable |
| --- | --- | --- |
| $a_i$ | Open valve $i$ | Yes |
| $b_3$ | Close valve 3 | Yes |
| $f$ | Valve 1 stuck-closed | No |
| $u_i$ | Engine i thrust up | No |
| $d_i$ | Engine i thrust down | No |

$$R(\mathbf{G}_{NF}^{f}, \text{true}) \leq CR(\mathbf{G}_{NF}^{f}, \text{true}).$$

2. The specifications are satisfied.

$$R(\mathbf{G}_{N}^{f}, \text{true}) \leq P_N,$$

$$R(\mathbf{G}_{NF}^{f}, \text{true}) \leq P_{NF},$$

where $P_N$ and $P_{NF}$ are predicates that represent the safe states in $\mathbf{G}_N$ (normal mode) and $\mathbf{G}_{NF}$ (normal and faulty mode).

This problem can be solved as a robust supervisory control problem for $\mathcal{G} = \{\mathbf{G}_N, \mathbf{G}_{NF}\}$. In the following, we examine an example of the above fault-tolerance problem.

Consider the monopropellant propulsion system illustrated in Figure 3.5. This model has two engines, one fuel tank, three valves, fuel pumps and two combustion chambers. In this model, $V_1$ and $V_2$ are pyrovalves. These two pyrovalves are normally closed to help fuel storage and prevent leakage; however, once we open them, they will remain that way. The valves may experience failure and become stuck-closed. For simplicity, assume that only $V_1$ may fail. The model of each component is shown in Figure 3.6. For every model, consider 0 to be the initial state. All the model events are listed in Table 3.1.

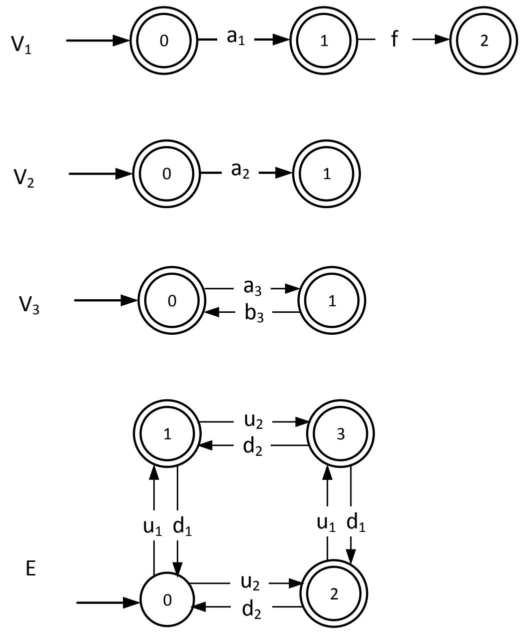



Figure 3.6: The model of system's components.

The automaton $V'$ in Figure 3.7 represents the system's components' interactions under synchronization. $V'$ is calculated by adding the necessary self-loops to the synchronous product of the three valves $V = sync(V_1, V_2, V_3)$. In Figure 3.7 ($V'$), the self-loops shows that the engine $E_1$ can be fired if and only if $V_1$ and $V_3$ are open, and the engine $E_2$ can be fired if and only if $V_2$ and $V_3$ are open. The labels of states are of the form $m_1m_2m_3$, where $m_1$, $m_2$, and $m_3$ are the states of $V_1$, $V_2$, and $V_3$.

The entire model of the system is calculated by the synchronous product of $E$ and $V'$, $sync(E, V')$. The normal and normal+faulty models of the system are shown in Figure 3.8 and 3.9. These two models are MR. The normal model $\mathbf{G}_1$ has 18 reachable states (10 marked); the normal+faulty model $\mathbf{G}_2$ has 30 reachable states (18 marked). The labels of states are of the form $n_1n_2n_3n_4$, where $n_1$ is the state of $E$, and $n_2$, $n_3$, and $n_4$ are the states of $V_1$, $V_2$, and $V_3$.

In this example, the specifications of both normal and normal+faulty models are the same; We do not want engine 1 and 2 to fire at the same time. In the DES model of the engine, Figure 3.6, only at state 3 both engines are fired. Therefore, every state in Figure 3.8 and 3.9 that has a label $3xxx$ is unsafe. The set of unsafe (illegal) states (i.e., states that do not satisfy the safety predicate) is $Q_{ill} = \{3110, 3111, 3210, 3211\}$. Since all reachable states of $\mathbf{G}_1$ and $\mathbf{G}_2$ are coreachable as well, $R(\mathbf{G}_1, true) = CR(\mathbf{G}_1, true) = P_{Q_1}$ and $R(\mathbf{G}_2, true) = CR(\mathbf{G}_2, true) = P_{Q_2}$. Moreover, we have $P_1 = P_{Q_1} \wedge \neg P_{Q_{ill}}$ and $P_2 = P_{Q_2} \wedge \neg P_{Q_{ill}}$. Since $Q_1 \subseteq Q_2$, $R(\mathbf{G}, true) = R(\mathbf{G}_2, true)$ and $Q = Q_2$. For this example, (3.7) can be simplified to $P = P_2 \wedge (P_1 \vee (P_Q \wedge \neg P_{Q_1})) =$

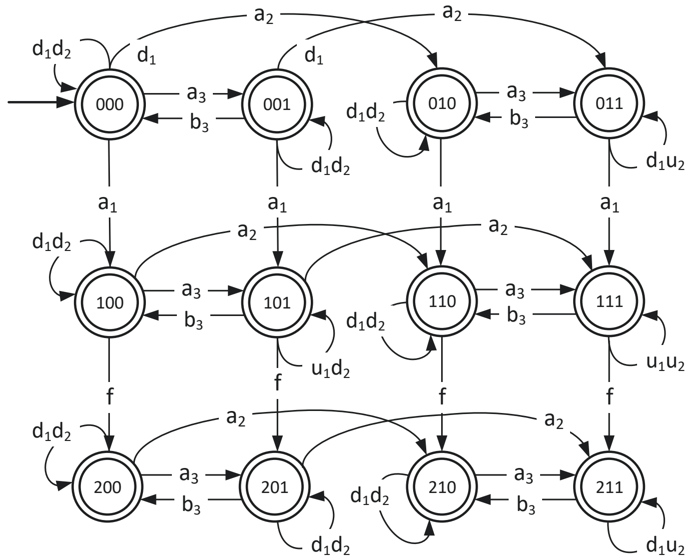



Figure 3.7: The automaton $V'$.

$P_2$. The state set represented by $P_2$ is $Q_{P_2} = Q_2 - \{3110, 3111, 3210, 3211\}$. Applying the algorithm in Proposition 3.2, we converge to the solution $K^{\uparrow}$ after 3 iterations.

In the first iteration, $S_1 = P = P_2$. At the second step, $L_1 = CR(\mathbf{G}_1, S_1)$ and $L_2 = CR(\mathbf{G}_2, S_1)$. The state sets that represents $L_1$ and $L_2$ are $Q_{L_1} = Q_1 - \{3110, 3111, 3210, 3211\}$ and $Q_{L_2} = Q_2 - \{3110, 3111, 3210, 3211\}$. At the third step, $S_1' = \left[L_1 \wedge L_2\right] \vee \left[(L_1 \wedge \neg P_{Q_2}) \bigvee (L_2 - P_{Q_1})\right]$ and the state set that represents it is $Q_{S_1'} = Q_2 - \{3110, 3111, 3210, 3211\}$. At the fourth step, $S_2 = R(\mathbf{G}, \langle S_1' \rangle)$ and $Q_{S_2} = Q_2 - \{\mathbf{0111}, \mathbf{1111}, \mathbf{2111}, \mathbf{1211}, 3110, 3111, 3210, 3211\}$. Since $S_2 \neq S_1$, the algorithm will be repeated for the second iteration.

In the second iteration, the algorithm starts from the second step, where $L_1 = CR(\mathbf{G}_1, S_2)$, $L_2 = CR(\mathbf{G}_2, S_2)$, $Q_{L_1} = Q_1 - \{\mathbf{0110}, \mathbf{0111}, \mathbf{1111}, \mathbf{2111}, \mathbf{1211}, 3110, 3111, 3210, 3211\}$ and $Q_{L_2} = Q_2 - \{\mathbf{0111}, \mathbf{1111}, \mathbf{2111}, \mathbf{1211}, 3110, 3111, 3210, 3211\}$. At the third step, $S_2' = \left[L_1 \wedge L_2\right] \vee \left[(L_1 - P_{Q_2}) \bigvee (L_2 - P_{Q_1})\right]$ and the state set that represent it is $Q_{S_2'} = Q_2 - \{\mathbf{0110}, \mathbf{0111}, \mathbf{1111}, \mathbf{2111}, \mathbf{1211}, 3110, 3111, 3210, 321\}$. At the fourth step, $S_3 = R(\mathbf{G}, \langle S_2' \rangle)$ and $Q_{S_3} = Q_2 - \{\mathbf{1110}, \mathbf{2110}, \mathbf{0110}, \mathbf{0111}, \mathbf{1111}, \mathbf{2111}, \mathbf{1211}, 3110, 3111, 3210, 3211\}$. Since $S_3 \neq S_2$, the algorithm will be repeated for the third iteration.

In the third iteration, the algorithm starts from the second step, where $L_1 = CR(\mathbf{G}_1, S_3)$, $L_2 = CR(\mathbf{G}_2, S_3)$, $Q_{L_1} = Q_1 - \{\mathbf{1110}, \mathbf{2110}, \mathbf{0110}, \mathbf{0111}, \mathbf{1111}, \mathbf{2111}, \mathbf{1211}, 3110, 3111, 3210, 3211\}$ and $Q_{L_2} = Q_2 - \{\mathbf{1110}, \mathbf{2110}, \mathbf{0110}, \mathbf{0111}, \mathbf{1111}, \mathbf{2111}, \mathbf{1211}, 3110, 3111, 3210, 3211\}$. At the third step, $S_3' = \left[L_1 \wedge L_2\right] \vee \left[(L_1 - P_{Q_2}) \bigvee (L_2 - P_{Q_1})\right]$ and the state set that represent it is $Q_{S_3'} = Q_2 - \{\mathbf{1110}, \mathbf{2110}, \mathbf{0110}, \mathbf{0111}, \mathbf{1111}, \mathbf{2111}, \mathbf{1211}, 3110, 3111, 3210, 321\}$. At the fourth step, $S_4 = R(\mathbf{G}, \langle S_3' \rangle)$ and $Q_{S_4} = Q_2 - \{\mathbf{1110}, \mathbf{2110}, \mathbf{0110}, \mathbf{0111}, \mathbf{1111}, \mathbf{2111}, \mathbf{1211}, 3110, 3111, 3210, 3211\}$. Since $S_4 = S_3$, the algorithm stops. Therefore, $K^{\uparrow} = S_4$ and there exists a SFBC $f$ such that

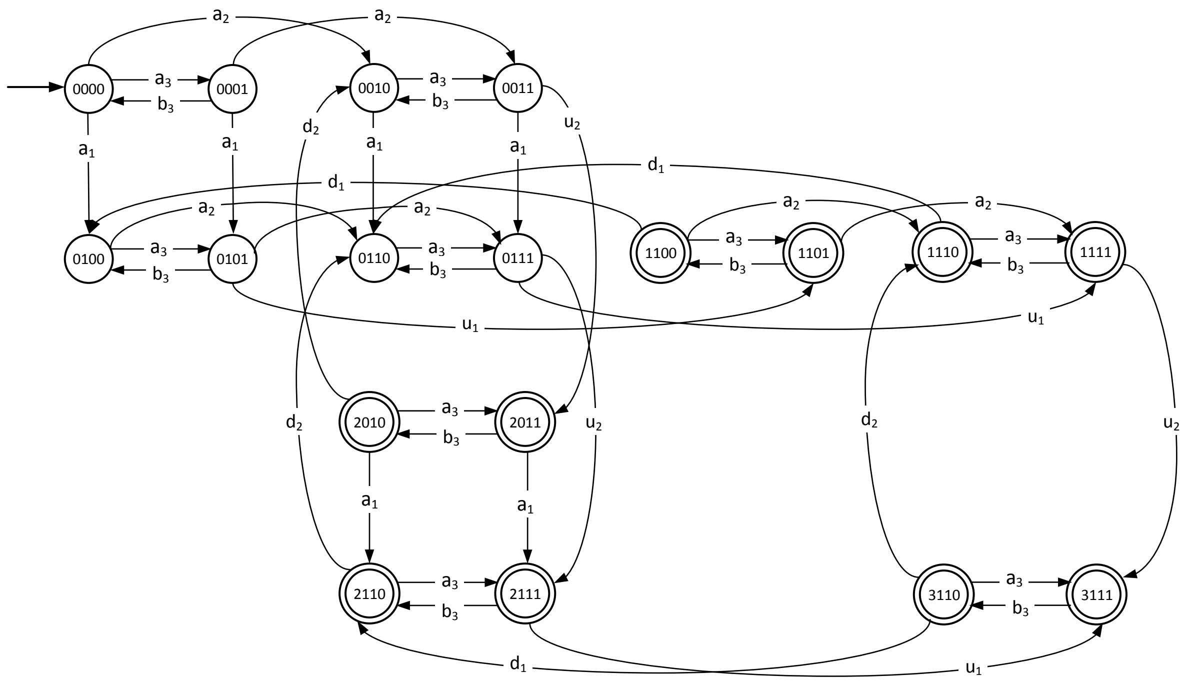



Figure 3.8: The normal model of system ($\mathbf{G}_1$).

$R(\mathbf{G}^f,\ \text{true}) = K^{\uparrow}$.

The allowed states under the supervision of $f$ are $Q^f = Q_2 - \{$**1110**, **2110**, **0110**, **0111**, **1111**, **2111**, **1211**, 3110, 3111, 3210, 3211$\}$. Since we have $Q^f$, for each $\sigma \in \Sigma$, $f_\sigma : Q \to \{0,1\}$ can easily be calculated. The set of states that we should avoid are $\{$**1110**, **2110**, **0110**, **0111**, **1111**, **2111**, **1211**, 3110, 3111, 3210, 3211$\}$; to avoid them, as one can see in Figures 3.8 and 3.9, only controllable events will be disabled. For example, in Figure 3.8, at 0110 (no engine has fired, and $V_1$, $V_2$, and $V_3$ are open, open, and closed), we will disable the controllable event $a_3$ (opening $V_3$) to avoid reaching 0111 (no engine has fired, and $V_1$, $V_2$, and $V_3$ are open). 0111 can go to 1111 (Engine 1 has fired, and $V_1$, $V_2$, and $V_3$ are open) via the uncontrollable event $u_1$ (Engine 1 fires) and 1111 can go to 3111 (both engines has fired) via the uncontrollable event $u_2$ (Engine 2 fires); therefore, we need to avoid reaching to 0111.

Since no uncontrollable event has been disabled and all states are marked, $K$ is controllable and nonblocking with respect to $\mathbf{G}_1$ and $\mathbf{G}_2$. Here, **1110**, **2110**, **0110**, **0111**, **1111**, **2111**, and **1211** are the additional states that need to be avoided besides $Q_{ill}$.

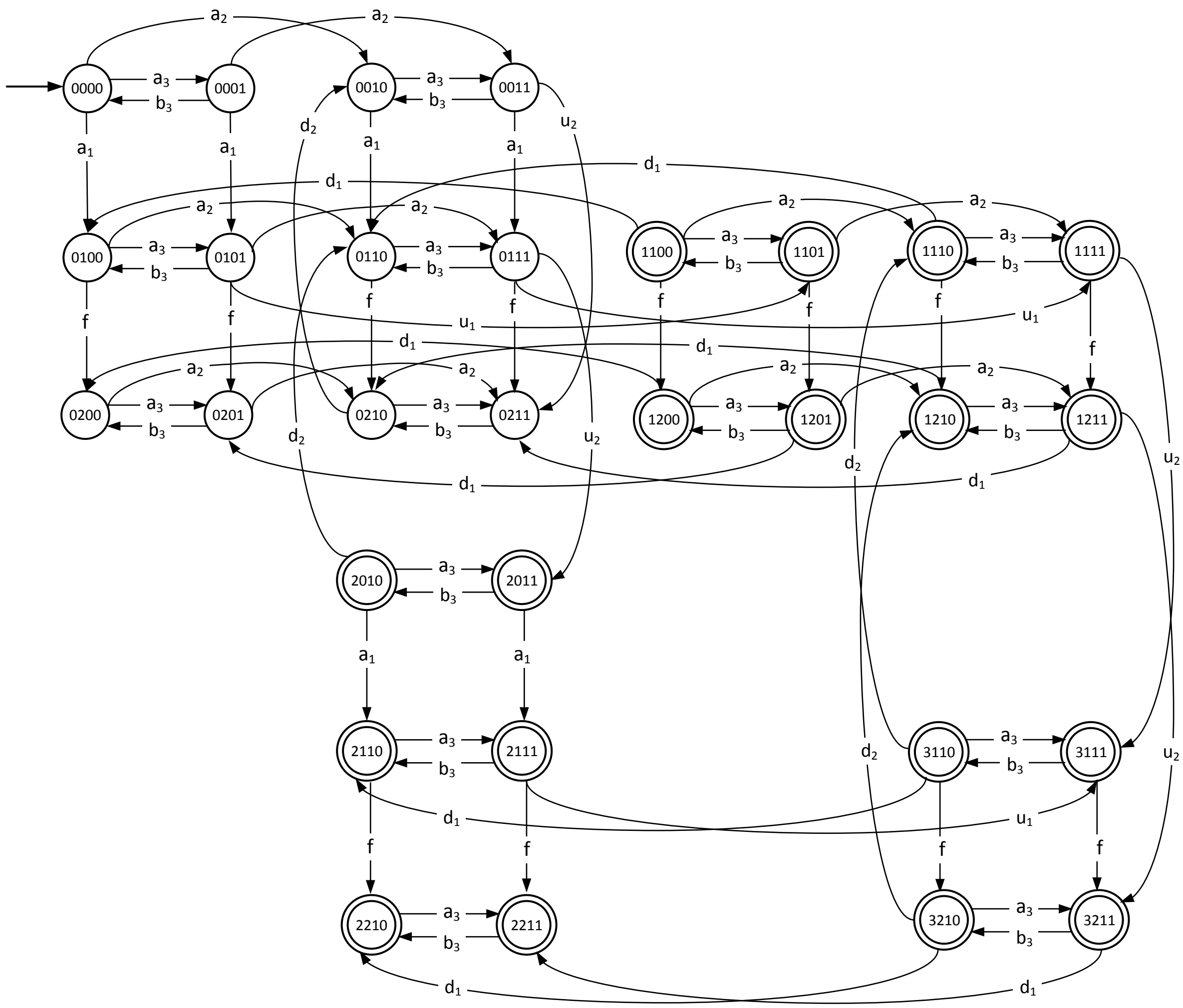



Figure 3.9: The normal+faulty model of system ($\mathbf{G}_2$).

## 3.6 Summary

In this chapter, we solved a problem of Robust Nonblocking Supervisory Control (RNSSCP) of DES in a state-based framework and characterized the corresponding solution set. Moreover, we developed an algorithm to calculate the maximally permissive solution within a finite number of iterations. This state-based framework could serve as a basis for developing design algorithms that use symbolic calculations. Such algorithms would be crucial in applying the results to industrial-size problems and are the next chapter's subject.

## 3.A Procedure to Obtain Mutually Refined Automata

In Section 3.1, we discussed the MR condition. As we mentioned before, if any automata pairs are not MR, there is a procedure that converts these automata into MR ones.

One of the operations defined over automata is *multiple biased synchronous product*. It can be regarded as an extension of the *biased synchronous product* of two automata [31].

**Definition 3.5.** *([11]) Consider a set of automata $\mathcal{G} = \{\mathbf{G}_1, \ldots, \mathbf{G}_N\}$ with $\mathbf{G}_i = (Q_i, \Sigma_i, \delta_i, q_{0i}, Q_{mi})$ $(i \in I = \{1, \ldots, N\})$. The **multiple biased synchronous product** of $\mathbf{G}_k$ for $k \in I$ is denoted by $\mathbf{G}_k \|_{mr} (\mathcal{G} - \{\mathbf{G}_k\})$ and defined as follows:*

$$reach(\mathbf{G}_k \|_{mr} (\mathcal{G} - \{\mathbf{G}_k\})) = \Big(Q_1 \times \cdots \times Q_N, \Sigma_k, \delta, \big(q_{01}, \ldots, q_{0N}\big), Q_{m1} \times \cdots \times Q_{mN}\Big), \tag{3.17}$$

*where for $q_i \in Q_i$ $(i \in I)$,*

$$\delta\big((q_1, \ldots, q_N), \sigma\big) = \begin{cases} (q_1', \ldots, q_N'), & \text{if } \sigma \in \Sigma_k \text{ and } \delta_k(q_k, \sigma)! \\ \text{undefined}, & \text{otherwise} \end{cases}$$

*with*

$$(\forall j \in I) \;\; q_j' = \begin{cases} \delta_j(q_j, \sigma), & \text{if } \sigma \in \Sigma_j \text{ and } \delta_j(q_j, \sigma)! \\ q_j, & \text{otherwise} \end{cases}.$$

**Procedure 3.1.** *([11])*

1. *Let $\mathcal{G} = \{\mathbf{G}_1, \ldots, \mathbf{G}_N\}$ and $\mathbf{G}_i = (Q_i, \Sigma_i, \delta_i, q_{i0}, Q_{mi})$ for all $i \in I = \{1, \ldots, N\}$. Add a dump state to each $\mathbf{G}_i$ $(i \in I)$ and add the self-loop of $\Sigma = \bigcup_{i \in I} \Sigma_i$ to each of dump states. Denote the updated set of automata as $\mathcal{G}' = \{\mathbf{G}_1', \ldots, \mathbf{G}_N'\}$.*
2. *Calculate $\mathbf{G}_i'' = \mathbf{G}_i \|_{mr} (\mathcal{G}' - \{\mathbf{G}_i'\})$ $(i \in I)$.*
3. *Denote the resulting set of automata as $\mathcal{G}'' = \{\mathbf{G}_1'', \ldots, \mathbf{G}_N''\}$.*

[11] proves that for the resulting set of automata $\mathcal{G}''$ derived from Procedure 3.1, $\mathbf{G}_i''$ and $\mathbf{G}_j''$ are MR $(i, j \in I)$.

# Chapter 4

# Robust Supervisory Control of Systems with State-Tree-Structure Model

In this chapter, a novel state-based approach is proposed for the robust nonblocking supervisory control problem of systems with STS models. Due to the model uncertainty, the plant model is assumed to belong to a finite set of STSs. A novel state-based supervisory control problem is formulated. A set of necessary and sufficient conditions are obtained for problem solvability. Furthermore, an algorithm is developed to calculate the supremal solution (maximally permissive) within a finite number of iterations. Finally, an illustrative example is presented in which BDDs are used to symbolically synthesize the supervisor and enhance calculation efficiency.

The rest of this chapter is organized as follows. In Section 4.1, the robust state-based supervisory control problem is defined for a system with a set of STS models and the MR property is studied for holons and STS. In Section 4.2, some implications of MR property in STS are explained. The necessary and sufficient conditions for the existence of a solution for the supervisory control of STS are given in Section 4.3. An algorithm is proposed to calculate the maximally permissive solution within a finite number of iterations in Section 4.4. In Section 4.5, the robust state-based supervisory control problem is formulated for the STS model of a Flexible Manufacturing System (FMS) with a state set of order $10^8$ and the maximally permissive solution is calculated using the proposed algorithm in Section 4.4. Finally, the summary is given in Section 4.6.

## 4.1 Problem Formulation

In this section, we define our supervisory control problem. Let us consider a DES plant and assume that due to some model uncertainty, the actual model of plant belongs to a finite set of $N$ models $\mathcal{G} = \{\mathbf{G}_1, \ldots, \mathbf{G}_N\}$ where each $\mathbf{G}_i$ ($i \in I = \{1, ..., N\}$) is a STS given below.

- $\mathbf{G}_i = (\mathrm{ST}_i, H_i, \Sigma_i, \Delta_i, \mathrm{ST}_0, \mathrm{ST}_{mi})$,
- $\mathrm{ST}_i = (X_i, x_0, \mathcal{T}_i, \varepsilon_i)$ and
- $H_i = \left\{H_i^a \mid a \in X_i,\ \mathcal{T}_i(a) = \mathrm{OR}\ \&\ H_i^a = (X_i^a, \Sigma_i^a, \delta_i^a, X_{0i}^a, X_{mi}^a)\right\}$.

For each model, the design specification (safe sub-ST) is defined by a predicate $P_i$. For $\Delta_i(.,.)$ to be sound, we assume that all the holons in $\mathbf{G}_i$ satisfy the conditions of Lemma 2.17.

**Example 4.1.** *As an example, consider a manufacturing plant with three machines ($M_1$, $M_2$, and $M_2'$) and one automated guided vehicle (AGV). The responsibility of AGV is to transfer the work-pieces between $M_1$ and $M_2$ ($M_2'$). One of the machines $M_2$ may experience a fault. The normal (normal+faulty) model of plant $\mathbf{G}_N$ ($\mathbf{G}_{NF}$), where $M_2$ does not experience any fault (may experience a fault), is shown in Figure 4.1. Assume that all the events are observable; an assumption that will be carried out for the rest of this chapter. The controllable events are $\Sigma_c = \{a_1, a_2, a_2'\}$. The marked states are shown by double circles. In this example, $\mathcal{G} = \{\mathbf{G}_N, \mathbf{G}_{NF}\}$.*

We assume that all the holons are deterministic, i.e., by any internal transition in holon, one simple state (or super-state) goes to another simple state (or super-state). Consider the STS examples in Figures 2.2 and 4.1 where all the holons are deterministic. In Figure 2.2, for example, in holon $H^B$ assigned to the super-state $B$, the transition $\alpha$ is from the simple state $x$ to the super-state $Y$ only and the transition $\gamma$ is from the super-state $Y$ to the simple state $x$ only.

Before defining the MR property for STS, in Theorem 4.1, we present an alternative set of conditions for MR property in automata and prove that they are equivalent to those in Definition 3.1. In Theorem 4.1, we change the scope of Definition 3.1 from sequences to single transitions.

**Theorem 4.1.** *Consider two automata $\mathbf{G}_1 = (Q_1, \Sigma_1, \delta_1, q_0, Q_{m1})$ and $\mathbf{G}_2 = (Q_2, \Sigma_2, \delta_2, q_0, Q_{m2})$. Let $Q_{1r}$ and $Q_{2r}$ be the reachable states of $\mathbf{G}_1$ and $\mathbf{G}_2$. Automata $\mathbf{G}_1$ and $\mathbf{G}_2$ are MR if and only if*

1. $\forall q \in Q_{1r} \cap Q_{2r}, \forall \sigma \in \Sigma_1 \cap \Sigma_2$
   $\big(\delta_1(q, \sigma)!\ \textit{and}\ \delta_2(q, \sigma)!\big) \Rightarrow \delta_1(q, \sigma) = \delta_2(q, \sigma)$,

2.

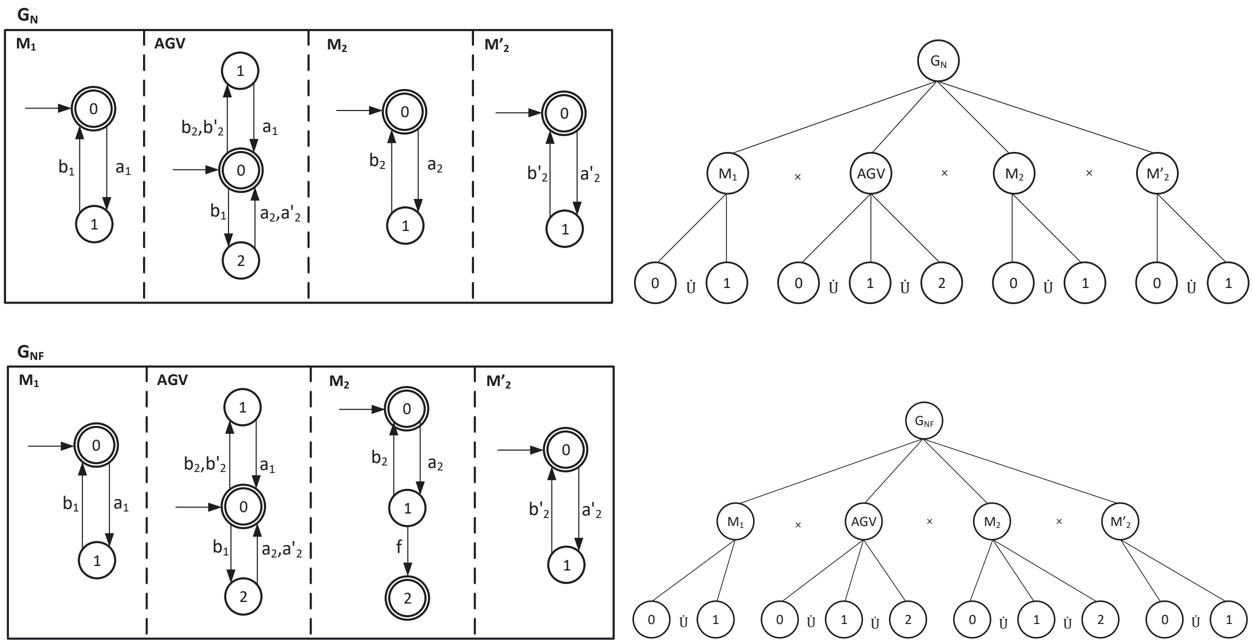



Figure 4.1: Example 4.1: the two STS models of a manufacturing plant.

*2.a.* $\forall q \in Q_{1r} \cap Q_{2r}, \forall \sigma \in \Sigma_1$
$\big(\delta_1(q,\sigma)!\ \textit{and not}\ \delta_2(q,\sigma)!\big) \Rightarrow \delta_1(q,\sigma) \in Q_{1r} - Q_{2r},$

*2.b.* $\forall q \in Q_{1r} \cap Q_{2r}, \forall \sigma \in \Sigma_2$
$\big(\textit{not}\ \delta_1(q,\sigma)!\ \textit{and}\ \delta_2(q,\sigma)!\big) \Rightarrow \delta_2(q,\sigma) \in Q_{2r} - Q_{1r},$

*3.*

*3.a.* $\forall q \in Q_{1r} - Q_{2r}, \forall \sigma \in \Sigma_1$
$\big(\delta_1(q,\sigma)!\big) \Rightarrow \delta_1(q,\sigma) \in Q_{1r} - Q_{2r},$

*3.b.* $\forall q \in Q_{2r} - Q_{1r}, \forall \sigma \in \Sigma_2$
$\big(\delta_2(q,\sigma)!\big) \Rightarrow \delta_2(q,\sigma) \in Q_{2r} - Q_{1r}.$

*Proof.*
(If) In the trivial case of $L(\mathbf{G}_1) = \emptyset$ or $L(\mathbf{G}_2) = \emptyset$, MR conditions are true. Suppose $L(\mathbf{G}_1) \neq \emptyset$ and $L(\mathbf{G}_2) \neq \emptyset$. If $q_{01} \neq q_{02}$, then it follows from condition (3.a.) and (3.b.) that $Q_{1r} \cap Q_{2r} = \emptyset$. Therefore, all three conditions of MR property (Definition 3.1) hold.

Suppose $q_{01} = q_{02}$. Therefore, the condition (1) in Definition 3.1 is true for $s = \epsilon$. Condition (1) of theorem implies condition (1) in Definition 3.1 for all strings of length $|s| = 1$. We can also use condition (1) to show that if condition (1) in Definition 3.1 holds for strings of length $|s| = k$ $(k \geq 1)$, then it holds for strings of length $|s| = k + 1$. Thus, by induction, condition (1) in Definition 3.1 is true.

Next we prove that condition (2) of Definition 3.1 is true. Suppose $q = \delta_1(q_{01}, s)$ for some $s \in L(\mathbf{G}_1) - L(\mathbf{G}_2)$ and $q_t = \delta_2(q_{02}, t)$ for some $t \in L(\mathbf{G}_2)$. In that case, $q \in Q_{1r}$ and $q_t \in Q_{2r}$. We claim $q \notin Q_{2r}$. This statement is true for any $s$ with $|s| = 1$ because of condition (2.a.). Now suppose $|s| \geq 2$. Since $s \in L(\mathbf{G}_1) - L(\mathbf{G}_2)$, $L(\mathbf{G}_2) \neq \emptyset$, and $\epsilon \in L(\mathbf{G}_2)$, there exist $s_1, s_2 \in \Sigma_1^*$ with $s = s_1 s_2$ and $s_1 \in L(\mathbf{G}_2)$. Let $s_1' \in \Sigma_1^*$ be the largest string such that $\exists s_2' \in \Sigma_1^*$, $s_1' s_2' = s$, $s_1' \in L(\mathbf{G}_2)$ and let $\delta_2(q_{02}, s_1') = q'$. It follows from condition (1) in Definition 3.1 that $\delta_1(q_{01}, s_1') = q'$. $s_2' \neq \epsilon$, otherwise $s_1' = s$ and $s \in L(\mathbf{G}_2)$ which violates the assumption $s \in L(\mathbf{G}_1) - L(\mathbf{G}_2)$. Suppose $s_2' = \sigma_1 \ldots \sigma_n$ for $n \geq 1$. Therefore, after the string $s_1'$, $\sigma_1 \ldots \sigma_n$ can only occur in $\mathbf{G}_1$. It follows from condition (2.a.) that $\delta_1(q_{01}, s_1'\sigma_1) \in Q_{1r} - Q_{2r}$ and from condition (3.a.) that $\delta_1(q_{01}, s_1'\sigma_1 \ldots \sigma_j) \in Q_{1r} - Q_{2r}$ $(2 \leq j \leq n)$. Therefore, $q = \delta_1(q_{01}, s_1' s_2') \in Q_{1r} - Q_{2r}$ and $q \neq q_t$.

Condition (3) of Definition 3.1 is shown similarly using conditions (2.b.) and (3.b.).

(Only if) If $L(\mathbf{G}_1) = \emptyset$, then $Q_{1r} = \emptyset$ and the conditions of the theorem hold. Similarly, if $L(\mathbf{G}_2) = \emptyset$, all the conditions are true.

Now suppose $L(\mathbf{G}_1) \neq \emptyset$ and $L(\mathbf{G}_2) \neq \emptyset$. Thus, $\epsilon \in L(\mathbf{G}_1) \cap L(\mathbf{G}_2)$ and from condition (1) of Definition 3.1, $q_{01} = q_{02}$. Therefore, $Q_{1r} \cap Q_{2r} \neq \emptyset$. Now we prove condition (1). Suppose $q \in Q_{1r} \cap Q_{2r}$. We claim there exists $s \in L(\mathbf{G}_1) \cap L(\mathbf{G}_2)$ such that $\delta_1(q_{01}, s) = q$ and $\delta_2(q_{02}, s) = q$. If not, there exist $s_1 \in L(\mathbf{G}_1) - L(\mathbf{G}_2)$, $s_2 \in L(\mathbf{G}_2) - L(\mathbf{G}_1)$, $\delta_1(q_{01}, s_1) = q$, and $\delta_2(q_{02}, s_2) = q$ which violates condition (2) in Definition 3.1. Thus, such $s$ exists. Now if some $\sigma$ transition is defined in $\mathbf{G}_1$ and $\mathbf{G}_2$ from state $q$, then by condition (1) in Definition 3.1, $\delta_1(q_{01}, s\sigma) = \delta_2(q_{02}, s\sigma)$ or in other words $\delta_1(q, \sigma) = \delta_2(q, \sigma)$. This proves condition (1).

Next we prove condition (2.a.). Suppose $q \in Q_{1r} \cap Q_{2r}$, $\delta_1(q, s)!$ and not $\delta_2(q, s)!$. As shown above, there exists $s \in L(\mathbf{G}_1) \cap L(\mathbf{G}_2)$ with $q = \delta_1(q_{01}, s) = \delta_2(q_{02}, s)$. Thus, $s\sigma \in L(\mathbf{G}_1) - L(\mathbf{G}_2)$ and from condition (2) in Definition 3.1, we have $\delta_1(q, \sigma) = \delta_1(q_{01}, s\sigma) \notin Q_{2r}$.

Condition (2.b.) can be shown similarly.

Next consider condition (3.a.) and suppose $q \in Q_{1r} - Q_{2r}$ and $\delta_1(q, \sigma)!$. We conclude that there exists $s \in L(\mathbf{G}_1) - L(\mathbf{G}_2)$ with $q = \delta_1(q_{01}, s)$; thus, $s\sigma \in L(\mathbf{G}_1) - L(\mathbf{G}_2)$. Now it follows from condition (2) in Definition 3.1 that $\delta_1(q, \sigma) = \delta_1(q_{01}, s\sigma)$ is not reachable in $\mathbf{G}_2$; thus, $\delta_1(q, \sigma) \notin Q_{2r}$.

Condition (3.b.) can be shown similarly. □

We consider two assumptions over the set of STS models in $\mathcal{G}$. The state set of each $\mathbf{G}_i$ is represented using a hierarchy. The following assumptions mean that the state sets of all models are represented and labeled consistently using a common hierarchy.

**Assumption 4.1.** *Considers a set of STS models $\mathcal{G} = \{\mathbf{G}_1, \ldots, \mathbf{G}_N\}$ with $\mathbf{G}_i = (ST_i, H_i, \Sigma_i, \Delta_i, ST_0, ST_{mi})$ and $ST_i = (X_i, x_0, \mathcal{T}_i, \varepsilon_i)$. We assume:*

1. *The state-trees $ST_i$ are sub-STs of a state-tree ST such that $ST = \bigvee_{i \in I} ST_i$.*

2. *The holons are defined consistently in the following sense:*

$$\text{For } i, j \in I, \text{ if } x \in X_i \cap X_j \text{ and } x \text{ is an internal state of } H_i^a \text{ in } \mathbf{G}_i \text{ and an internal state of } H_j^{a'} \text{ in } \mathbf{G}_j, \text{ then } a = a'. \tag{4.1}$$

As mentioned in Chapter 3, mutual refinement guarantees that every solution of robust state-based supervisory control can be achieved using a state feedback law.

**Definition 4.1.** *Consider the set of models in Assumption 4.1. Two STS models $\mathbf{G}_i$ and $\mathbf{G}_j$ are called MR if the corresponding flat models are MR.*

**Remark 4.1.** *Definition 4.1 states that STS $\mathbf{G}_i$ and $\mathbf{G}_j$ are MR if the transitions among basic-STs in $\mathbf{G}_i$ and $\mathbf{G}_j$ satisfy the three conditions of definition of MR property in automata (Definition 3.1) or equivalently the conditions in Theorem 4.1.*

We can use Definition 4.1 to check the MR property in STS models. For industrial-size systems calculating the flat automata and checking the MR property are computationally expensive procedures that we want to avoid. However, we observe that mutual refinement of a set of STS models can be verified from the mutual refinement of the corresponding holons in the sense defined in the following.

**Definition 4.2.** *Consider the set of STS models in Assumption 4.1. Suppose holon $H_i^a$ and $H_j^a$ belong to STSs $\mathbf{G}_i$ and $\mathbf{G}_j$, and matched to the same state $a \in X_i \cap X_j$. We say $H_i^a$ and $H_j^a$ are **MR** if*

1. $\forall x \in X_i^a \cap X_j^a$, $\forall \sigma \in \Sigma_i^a \cap \Sigma_j^a$
   $\big(\delta_i(x,\sigma)! \text{ and } \delta_j(x,\sigma)!\big) \Rightarrow \delta_i(x,\sigma) = \delta_j(x,\sigma)$,

2. 

   2.a. $\forall x \in X_i^a \cap X_j^a$, $\forall \sigma \in \Sigma_i^a$
   $\big(\delta_i(x,\sigma)! \text{ and not } \delta_j(x,\sigma)!\big) \Rightarrow \delta_i(x,\sigma) \in X_i^a - X_j^a$,

   2.b. $\forall x \in X_i^a \cap X_j^a$, $\forall \sigma \in \Sigma_j^a$
   $\big(\text{not } \delta_i(x,\sigma)! \text{ and } \delta_j(x,\sigma)!\big) \Rightarrow \delta_j(x,\sigma) \in X_j^a - X_i^a$,

3. 

   3.a. $\forall x \in X_i^a - X_j^a$, $\sigma \in \Sigma_i^a$
   $(\delta_i(x,\sigma)!) \Rightarrow \delta_i(x,\sigma) \in X_i^a - X_j^a$,

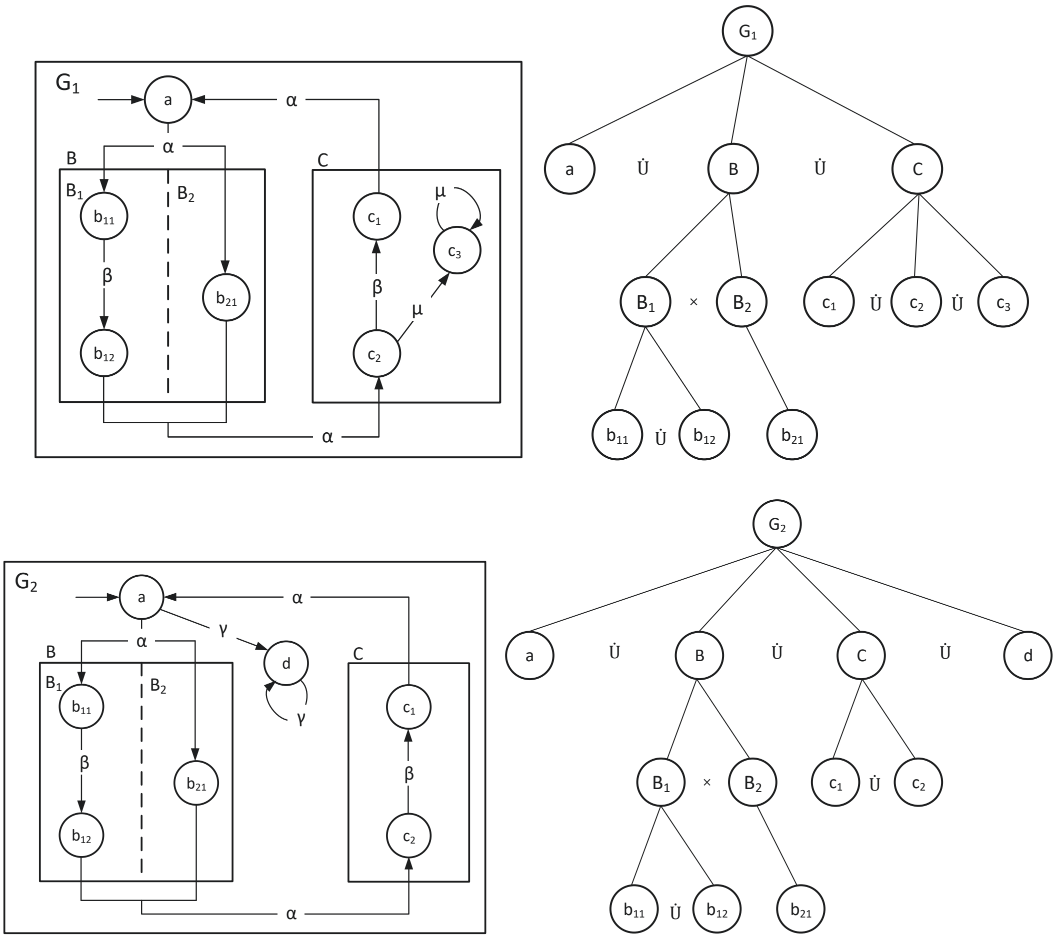


Figure 4.2: Example 4.2: the two STS models of a plant.

*3.b.* $\forall x \in X_j^a - X_i^a,\ \sigma \in \Sigma_j^a$

$(\delta_j(x,\sigma)!) \Rightarrow \delta_j(x,\sigma) \in X_j^a - X_i^a$.

**Example 4.2.** *Consider the set of STS models $\mathcal{G} = \{\boldsymbol{G}_1, \boldsymbol{G}_2\}$ shown in Figure 4.2. Here, the set of conditions in Assumption 4.1 are satisfied. In this example, the holons are all MR. The difference between these two models are the additional transitions $\mu$ and $\gamma$ that from $C_2$ to $C_3$, and a to d in $\boldsymbol{G}_1$ and $\boldsymbol{G}_2$. Since $C_3$ (d) and $\mu$ ($\gamma$) do not belong to $\boldsymbol{G}_2$ ($\boldsymbol{G}_1$); thus, the conditions of MR in Definition 4.2 are satisfied.*

*The corresponding flat automata of $\boldsymbol{G}_1$ and $\boldsymbol{G}_2$ are shown in Figure 4.3. The two automata satisfy the conditions of Theorem 4.1 (Definition 3.1). Thus, based on Definition 4.1, one can also conclude that the STS models of $\boldsymbol{G}_1$ and $\boldsymbol{G}_2$ are MR.*

**Theorem 4.2.** *Consider the set of models in Assumption 4.1. Assume for any two STS models $\boldsymbol{G}_i$ and $\boldsymbol{G}_j$, all the corresponding holons are MR. Then the STS models are MR.*

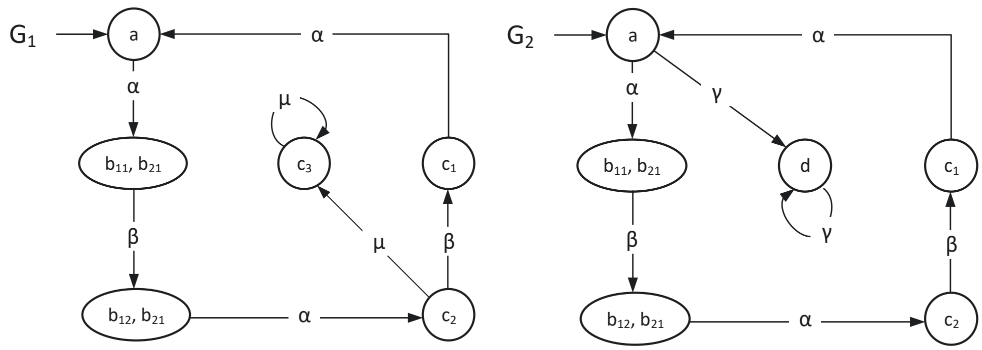



Figure 4.3: Example 4.2: the corresponding flat automata of Figure 4.2.

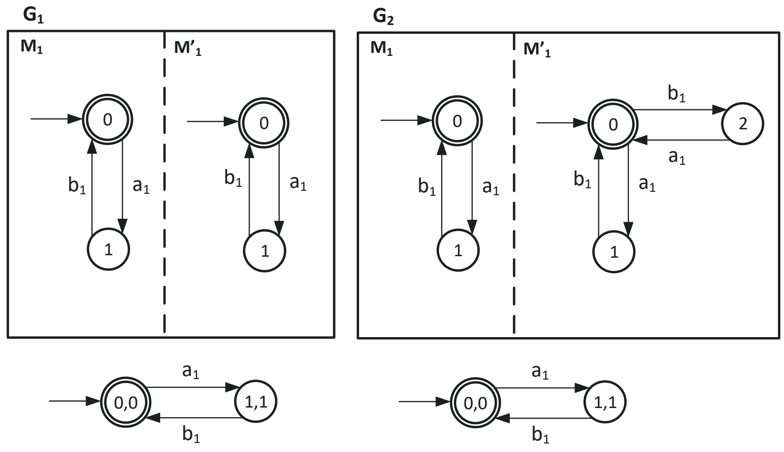
*Proof.*

In order to prove the theorem, we show that the transitions among basic-STs of $\mathbf{G}_i$ and $\mathbf{G}_j$ satisfy conditions in Theorem 4.1. Consider a reachable basic-ST of $\mathbf{G}_i$ and let $q$ denote the corresponding state of flat automaton of $\mathbf{G}_i$. The state $q$ is either (i) a simple state of a holon, say $H_i^x$ with $x \in X_i$ (when all ancestors of $q$ are OR states), or (ii) in of the form of an n-tuple $q = (q_1, \ldots, q_n)$, where $q_k$ is a simple state in some holon $H_i^{x_k}$ (when $q$ has at least one AND ancestor), $x_k \in X_i$, and $1 \le k \le n$.

Case (i). Suppose $q$ is a simple state. Assume $\sigma$ is enabled at $q$. Thus, in holon $H_i^x$ of $\mathbf{G}_i$, the target of $\sigma$ transition is either another (a) simple state, (b) an AND, or (c) an OR super-state.

(i.a). If the target of transition is a simple state $q'$, then in holon $H_i^x$ in $\mathbf{G}_i$, $q$ transitions to $q'$ by event $\sigma$. If $q$ is reachable in $\mathbf{G}_j$ and $\sigma$ is defined at $q$ in $\mathbf{G}_j$, then by condition (1) in Definition 4.2 in holon $H_j^x$ of $\mathbf{G}_j$, $q$ transitions to $q'$ by event $\sigma$. Thus, condition (1) of Theorem 4.1 holds. Similarly, condition (2) of Definition 4.2 implies condition (2.a.) of Theorem 4.1. Note that condition (2.a.) of Definition 4.2 implies that following a $\sigma$ transition in $\mathbf{G}_i$ that is not defined in $\mathbf{G}_j$, all subsequent states in the holon $H_i^x$ in $\mathbf{G}_i$ are not a state of holon $H_j^x$ in $\mathbf{G}_j$. Based on this and condition (3.a.) of Definition 4.2, we conclude that condition (3.a.) of Theorem 4.1 holds. Conditions (2.b.) and (3.b.) of Theorem 4.1 hold similarly.

(i.b). Suppose the target of $\sigma$ transition in $H_i^x$ from $q$ is an OR super-state $x'$ (the case of AND super-state is similar). This transition is represented in STS $\mathbf{G}_i$ as a transition from state $q$ to super-state $x'$ ($q \to x'$) via event $\sigma$ in $H_i^x$ and a transition from state $q$ to state $q'$ is holon $H_i^{x'}$. If $\sigma$ transition out of $q$ is enabled in $\mathbf{G}_j$, then from condition (1) in Definition 4.2 (applied to holon $H_j^x$), the target of $\sigma$ transition has to be holon $H_j^{x'}$ and from condition (1) in Definition 4.2 (applied to holon $H_j^{x'}$), from $q$, the target of $\sigma$ transition must be $q'$. This shows condition (1) of Theorem 4.1 holds. The other conditions in Theorem 4.1 similarly follow from the corresponding conditions in Definition 4.2.




(a) $\mathbf{G}_1$ (b) $\mathbf{G}_2$

Figure 4.4: Example 4.3: two STS models and their equivalent flat models.

Case (ii). Suppose $q = (q_1, q_2)$, where $q_1$ and $q_2$ are simple states of holons $H_i^{x_1}$ and $H_i^{x_2}$ in $\mathbf{G}_i$ (extension to $(q_1, \ldots, q_n)$ is straightforward). If a $\sigma$ transition is enabled at $(q_1, q_2)$, then in holons $H_i^{x_1}$ and $H_i^{x_2}$, $\sigma$ transitions are enabled at $q_1$ and $q_2$. If $\sigma$ transition is enabled from $q$ in $\mathbf{G}_j$, then it follows from condition (1) in Definition 4.2 that the targets of the $\sigma$ transition in $\mathbf{G}_i$ and $\mathbf{G}_j$ has to be the same (whether the targets are simple states or super-states). Thus, condition (1) of Theorem 4.1 holds. If the $\sigma$ transition at state $q$ is not enabled in $\mathbf{G}_j$, then it is not enabled either in $H_j^{x_1}$ or $H_j^{x_2}$. In this case, from condition (2.a.) of Definition 4.2, the target of $\sigma$ transition does not belong to either $H_j^{x_1}$ or $H_j^{x_2}$; hence, condition (2.a.) of Theorem 4.1 holds. Other conditions of Theorem 4.1 can be shown similarly to hold. □

The converse of Theorem 4.2 is not always true. Example 4.3 shows an example where STSs are MR, while the corresponding holons are not MR.

**Example 4.3.** *Consider the set of STS models $\mathcal{G} = \{\mathbf{G}_1, \mathbf{G}_2\}$ shown in Figure 4.4. In Figure 4.4, the flat models of $\mathbf{G}_1$ and $\mathbf{G}_2$ satisfy the conditions in Definition 3.1. Thus, the automata models of $\mathbf{G}_1$ and $\mathbf{G}_2$ are MR. Based on Definition 4.1, the STS models of $\mathbf{G}_1$ and $\mathbf{G}_2$ are also MR. In $\mathbf{G}_1$ and $\mathbf{G}_2$, the holons assigned to $M_1'$ does not satisfy the condition (1) in Definition 4.2; therefore, these holons are not MR.*

For the rest of this chapter, we assume that for any $\mathbf{G}_i$ and $\mathbf{G}_j$ $(i, j \in I)$, their corresponding holons $H_i^a \in H_i$ and $H_j^a \in H_j$ (for some $a \in X_i \cap X_j$) are MR. Thus, based on Theorem 4.2, all the STS models that belongs to $\mathcal{G}$ are MR with respect to one another.

Each of the STS in $\mathcal{G}$ has its own set of events $\Sigma_i$ $(i \in I)$. The controllability (uncontrollability) of shared events between different STS is consistent. Our goal is to design a SFBC $f$ such that each of these models satisfies the given specifications $P_i$ and stays nonblocking under its supervision. Our robust nonblocking supervisory control problem for STS is defined below.

**Problem 4.1.** ***Robust Nonblocking Supervisory Control Problem for State-Tree-Structure (RNSCP-STS):*** *Consider* $N$ *MR STS models* $\boldsymbol{G}_i = (ST_i, H_i, \Sigma_i, \Delta_i, ST_0, ST_{mi})$*, where* $ST_i = (X_i, x_0, \mathcal{T}_i, \varepsilon_i)$ *and* $H_i = \left\{H_i^a \mid a \in X_i,\ \mathcal{T}_i(a) = OR \text{ and } H_i^a = (X_i^a, \Sigma_i^a, \delta_i^a, X_{0i}^a, X_{mi}^a)\right\}$ *for* $i \in I = \{1, \ldots, N\}$*. There is a consistency in controllability/uncontrollability of events in STS models. Assume the STS models satisfy the Assumption 4.1. For each model, a set of safe sub-STs* $S_i \in \boldsymbol{ST}(ST_i)$ *is defined* ($ST_0 \in S_i$)*. Consider* $P_i$ *to be the predicate that defines the characteristic function of* $\bigcup_{\mathbb{S} \in S_i} B(\mathbb{S})$*. Find a SFBC* $f : \bigcup_{i \in I} B(ST_i) \to \mathit{\Pi}$ *such that*

1. $R(\boldsymbol{G}_i{}^f, true) \leq P_i$ *(safety property)*
2. $R(\boldsymbol{G}_i{}^f, true) \leq \mathrm{CR}(\boldsymbol{G}_i{}^f, true)$ *(nonblocking property)*

## 4.2 Implications of Mutually Refinement Property in State-Tree-Structure

Before presenting our results in Section 4.3, we discuss the implications of MR property in STS. We form a "union" STS called **G** by merging all the $N$ models mentioned in Definition 4.1. To make sure that our STS models in $\mathcal{G}$ are MR, we assume that all the corresponding holons in STS models are MR.

**Assumption 4.2.** *Considers a set of STS models* $\mathcal{G} = \{\boldsymbol{G}_1, \ldots, \boldsymbol{G}_N\}$ *with* $\boldsymbol{G}_i = (ST_i, H_i, \Sigma_i, \Delta_i, ST_0, ST_{mi})$ *and* $ST_i = (X_i, x_0, \mathcal{T}_i, \varepsilon_i)$*. We assume that all the corresponding holons in STS models are MR.*

**Definition 4.3.** *Consider a finite set of MR STS models defined as* $\mathcal{G} = \{\boldsymbol{G}_1, \ldots, \boldsymbol{G}_N\}$*, where* $\boldsymbol{G}_i = (ST_i, H_i, \Sigma_i, \Delta_i, ST_0, ST_{mi})$ *with* $ST_i = (X_i, x_0, \mathcal{T}_i, \varepsilon_i)$ *and* $H_i = \left\{H_i^a \mid a \in X_i,\ \mathcal{T}_i(a) = OR \text{ and } H_i^a = (X_i^a, \Sigma_i^a, \delta_i^a, X_{0i}^a, X_{mi}^a)\right\}$ $(i \in I)$*. We assume that all the corresponding holons in STS models are MR.* $\boldsymbol{G} = (ST, H, \Sigma, \Delta, ST_0, ST_m)$ *is defined below.*

1. $ST = (X, x_0, \mathcal{T}, \varepsilon) = \bigvee_{i \in I}(ST_i)$.
2. $H = \left\{H^a \mid a \in \bigcup_{i \in I} X_i,\ \mathcal{T}(a) = OR \text{ and } H^a = (\bigcup_{i \in I} X_i^a, \bigcup_{i \in I} \Sigma_i^a, \delta^a, \bigcup_{i \in I} X_{0i}^a, \bigcup_{i \in I} X_{mi}^a)\right\}$*, where* $\delta^a$ *is defined below.*
   (a) *For any* $x \in \bigcup_{i \in I} X_i^a$ *and* $\sigma \in \bigcup_{i \in I} \Sigma_i^a$*, if* $\exists j \in I$ *such that* $\delta_j^a(x, \sigma)!$*, then* $\delta^a(x, \sigma)!$*,* $\delta^a(x, \sigma) = \delta_j^a(x, \sigma)$*.*
   (b) $\delta$ *has no transition other than those described in 2a.*
3. $ST_m = \bigvee_{i \in I} ST_{mi}$.

It can easily be observed that $\text{ST}_i \in \textbf{ST}(\text{ST})$ $(i \in I)$.

**Example 4.3** (continued)**.** *The union model of Example 4.2 is shown in Figure 4.5. Note that the corresponding holons of* $\mathbf{G}_1$ *and* $\mathbf{G}_2$ *are MR.*

The following lemma enables us to show that in the union model $\mathbf{G}$, $\Delta(.,.)$ is sound.

**Lemma 4.1.** *Consider the set of MR STS models* $\mathcal{G}$ *and the union model* $\mathbf{G}$ *in Definition 4.3. Then in* $\mathbf{G}$*,*

1. *Every incoming boundary transition of any holon matched to an AND component has a unique event label.*
2. *Every outgoing boundary transition of any holon matched to an AND component has a unique event label.*

*Proof.* If $\mathbf{G}$ does not have any holon matched to an AND component, then the lemma is trivially true. Suppose that $\mathbf{G}$ has at least one holon matched to an AND component.

1. Suppose that $\mathbf{G}$ has a holon $H^x$ $(x \in \varepsilon(x_0))$ matched to the AND components $Y_1, \ldots, Y_c \in \varepsilon(x)$. Without loss of generality, suppose that $H^x$ has two holons $Y_1$ and $Y_2$ and $H^x$ also has two incoming boundary transitions labeled $\sigma \in \Sigma$ that enter $Y_1$ and $Y_2$. The $\sigma$ transitions are from external states $x_1, x_2 \in \varepsilon(x_0)$ to $(y_{11}, y_{21})$ and $(y_{12}, y_{22})$, where $y_{11}, y_{12} \in \varepsilon(Y_1)$ and $y_{21}, y_{22} \in \varepsilon(Y_2)$. These transitions cannot happen in one $\mathbf{G}_i$ $(i \in I)$, since for all $i \in I$, $\Delta_i(.,.)$ is sound. Without loss of generality, assume that these transitions happen in $\mathbf{G}_1$ and $\mathbf{G}_2$ $(\mathbf{G}_1, \mathbf{G}_2 \in \mathcal{G})$. All the possible cases are:

   (i) States $x_1$ and $x_2$ belong to both $\mathbf{G}_1$ and $\mathbf{G}_2$ or in other words, $x_1, x_2 \in \varepsilon_1(x_0) \cap \varepsilon_2(x_0)$.

   (i.a) The destination state of $x_1$ and $x_2$ are the same or in other words, $(y_{11}, y_{21}) = (y_{12}, y_{22})$.

   (i.b) The destination state of $x_1$ and $x_2$ are two distinct states or in other words, $(y_{11}, y_{21}) \neq (y_{12}, y_{22})$.

   (i.b.1) The destination state of $x_1$ and $x_2$ belong to both $\mathbf{G}_1$and $\mathbf{G}_2$.

   (i.b.2) The destination state of $x_1$ and $x_2$ do not belong to both $\mathbf{G}_1$ and $\mathbf{G}_2$.

   (ii) State $x_1$ only belongs to $\mathbf{G}_1$, or in other words $x_1 \in \varepsilon_1(x_0) - \varepsilon_2(x_0)$ and $x_2 \in \varepsilon_1(x_0) \cap \varepsilon_2(x_0)$. The other cases (e.g., $x_2$ only belongs to $\mathbf{G}_1$) are similar.

   (ii.a) The destination state of $x_1$ and $x_2$ are the same or in other words, $(y_{11}, y_{21}) = (y_{12}, y_{22})$.

   (ii.b) The destination state of $x_1$ and $x_2$ are two distinct states or in other words, $(y_{11}, y_{21}) \neq (y_{12}, y_{22})$.

   (ii.b.1) The destination state of $x_1$ and $x_2$ belong to both $\mathbf{G}_1$and $\mathbf{G}_2$.

   (ii.b.2) The destination state of $x_1$ and $x_2$ do not belong to both $\mathbf{G}_1$ and $\mathbf{G}_2$.

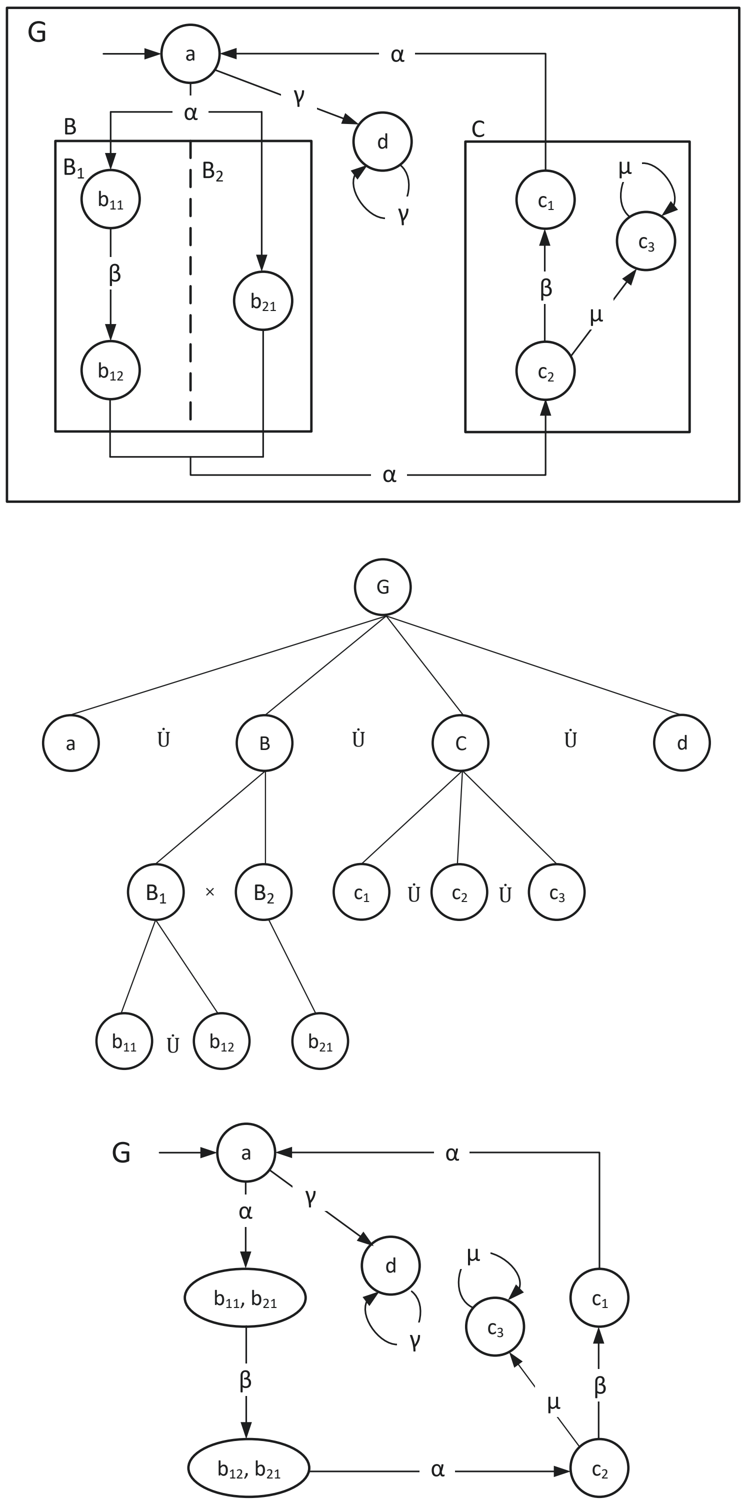



Figure 4.5: Example 4.2: **G**, the union STS model, the ST, and the equivalent flat model.

(iii) State $x_1$ and $x_2$ do not belong to $\mathbf{G}_1$ or $\mathbf{G}_2$, or in other words $x_1, x_2 \notin \varepsilon_1(x_0) \cup \varepsilon_2(x_0)$. This case cannot happen, since we assumed that the $\sigma$ transitions from external states $x_1, x_2 \in \varepsilon(x_0)$ to $(y_{11}, y_{21})$ and $(y_{12}, y_{22})$ happen in $\mathbf{G}_1$ and $\mathbf{G}_2$.

Case (i.a). Suppose the states $x_1$ and $x_2$ exist in both $\mathbf{G}_1$ and $\mathbf{G}_2$. The transition $x_1$ to $x$ happens in $\mathbf{G}_1$ and the transition $x_2$ to $x$ happens in $\mathbf{G}_2$. The destination states in $Y_1$ and $Y_2$ are the same in both $\mathbf{G}_1$ and $\mathbf{G}_2$. But, the corresponding holons in $\mathbf{G}_1$ and $\mathbf{G}_2$ are MR and that would violate the condition (2.a.) in Definition 4.2.

Cases (i.b.1) and (i.b.2) can be shown similar to Case (i.a). Note that in holons, the $\sigma$ transitions are from $x_1$ and $x_2$ to $Y$.

Case (ii.a). Suppose the states $x_1$ belongs to $\mathbf{G}_1$ and $x_2$ to $\mathbf{G}_2$. The transition $x_1$ to $x$ happens in $\mathbf{G}_1$ and the transition $x_2$ to $x$ happens in $\mathbf{G}_2$. The destination states in $Y_1$ and $Y_2$ are the same in both $\mathbf{G}_1$ and $\mathbf{G}_2$. But, the corresponding holons in $\mathbf{G}_1$ and $\mathbf{G}_2$ are MR and that would violate the condition (3.a.) in Definition 4.2.

Cases (ii.b.1) and (ii.b.2) can be shown similar to Case (ii.a).

2. It can be shown similarly.

□

**Remark 4.2.** *The transition function $\Delta(.,.)$ is sound in the union model $\boldsymbol{G}$. Thus, by Lemma 2.3 in Section 2.4, $\Delta(.,.)$ is in correspondence with the state transitions of the equivalent flat automaton of $\boldsymbol{G}$.*

The MR condition of holons helps us to form a deterministic union STS in Definition 4.3.

**Remark 4.3.** *The transitions of any holon in $\boldsymbol{G}$ is the union of the transitions of the corresponding holons in $\boldsymbol{G}_i$'s. It follows from the MR property of holons and Definition 4.3 that all the holons of $\boldsymbol{G}$ are deterministic (This result is similar to the determinism of union model $\boldsymbol{G}$ discussed in Section 3.2). The determinism of holons results in the determinism of the STS.*

The following lemma states that no new sequence of events can be found in the union model. This is the STS version of Lemma 3.1. Here we only considered sequences starting from initial State-Tree (ST) $\mathrm{ST}_0$, but the lemma can easily be generalized.

**Lemma 4.2.** *Consider the set of MR STS models $\mathcal{G}$ and the union model $\boldsymbol{G} = (ST, H, \Sigma, \Delta, ST_0, ST_m)$ in Definition 4.3 with $ST = (X, x_0, \mathcal{T}, \varepsilon)$ and $H = \left\{H^a \mid a \in X,\ \mathcal{T}(a) = OR \text{ and } H^a = (X^a, \Sigma^a, \delta^a, X_0^a, X_m^a)\right\}$. For any $s \in \Sigma^*$ and $T \in \boldsymbol{ST}(ST)$ such that $\Delta(ST_0, s) = T$, there exists $i \in I$ such that $s \in {\Sigma_i}^*$, $T \in \boldsymbol{ST}(ST_i)$ and $\Delta_i(ST_0, s) = \Delta(ST_0, s) = T$.*

*Proof.*

For $s \in \epsilon$ the lemma is trivially true. Suppose $s \neq \epsilon$ and for some $n \geq 1$, $s = \sigma_0 \ldots \sigma_{n-1}$ (with $\sigma_l \in \Sigma$, $0 \leq l \leq n-1$). We have $\mathrm{ST}_0 \in B(\mathrm{ST})$. Thus $\mathrm{ST}_0 \in B(\mathrm{ST})$. Also there exists $b_1, \ldots, b_n \in \mathbf{ST}(\mathrm{ST})$ with $\Delta(b_l, \sigma_l) = b_{l+1}$ $(1 \leq l \leq n-1)$, $\Delta(\mathrm{ST}_0, \sigma_0) = b_1$, and $b_n = T$. We claim that $b_l \in B(\mathrm{ST})$ $(1 \leq l \leq n)$. We prove our claim by induction. First, we show that $b_1 \in B(\mathrm{ST})$. If $b_1 \notin B(\mathrm{ST})$, then $count(b_1) \neq 1$ $(1 < count(b_1))$ and based on Remark 4.2, $b_1$ represents at least two distinct states in the flat automaton model of $\mathbf{G}$. Thus, in the flat automaton of $\mathbf{G}$, the initial state goes to at least two distinct destination states via event $\sigma_1$, but this violates $\mathbf{G}$ being deterministic. Thus, $b_1 \in B(\mathrm{ST})$. Suppose $b_k \in B(\mathrm{ST})$ $(1 \leq k \leq n)$. We now prove that $b_{k+1} \in B(\mathrm{ST})$. Based on assumption $\Delta(b_k, \sigma_k) = b_{k+1}$. If $b_{k+1} \notin B(\mathrm{ST})$, then $count(b_{k+1}) \neq 1$ $(1 < count(b_{k+1}))$ and based on Remark 4.2, $b_{k+1}$ represents at least two distinct states in the flat automaton model of $\mathbf{G}$. Thus, in the flat automaton of $\mathbf{G}$, there exists a state that goes to at least two distinct destination states via event $\sigma_k$, but this violates $\mathbf{G}$ being deterministic. Thus, $b_{k+1} \in B(\mathrm{ST})$. Therefore, we have proved that $b_l \in B(\mathrm{ST})$ $(1 \leq l \leq n)$.

Consider the $\mathrm{ST}_0, b_1, \ldots, b_n \in B(\mathrm{ST})$ and let $q_0, q_1, \ldots, q_n$ denote the corresponding state of flat automaton of $\mathbf{G}$. The state $q_k$ $(1 \leq k \leq n)$ is either a simple state of a holon, say for some $x_k \in X$, $H^{x_k}$ (where all ancestors of $q_t$ are OR states), or in of the form of an $m_k$-tuple $q_k = (q_{k,1}, \ldots, q_{k,m_k})$, where $q_{k,t}$ is a simple state in some holon $H_i^{x_{k,t}}$ (when $q_k$ has at least one AND ancestor), $x_{k,t} \in X$, and $1 \leq t \leq m_k$. Consider $\Delta(b_k, \sigma_k) = b_{k+1}$ $(0 \leq k \leq n-1$ and $b_0 = \mathrm{ST}_0)$. All the possible cases are:

(i) $b_k$ and $b_{k+1}$ are both simple states.

(ii) $b_k$ is a simple state and $b_{k+1}$ is a tuple.

(iii) $b_k$ is a tuple and $b_{k+1}$ is a simple state.

(iv) $b_k$ and $b_{k+1}$ are both tuple.

First we show our claim for a special case where for all $1 \leq k \leq n$, $q_k$ is a simple state of a holon $H^{x_k}$. Based on Definition 4.3, $\mathrm{ST}_0$ belongs to all $\mathbf{G}_i$ $(i \in I)$. Thus, $q_0$ also belongs to the equivalent flat automaton of $\mathbf{G}_i$ $(i \in I)$. We claim that $\exists j \in I$ such that $s = \sigma_0 \ldots \sigma_{n-1}$ (with $\sigma_l \in \Sigma_j$, $0 \leq l \leq n-1$), $b_1, \ldots, b_n \in B(\mathrm{ST}_j)$ with $\Delta_j(b_l, \sigma_l) = b_{l+1}$ $(1 \leq l \leq n-1)$, $\Delta_j(\mathrm{ST}_0, \sigma_0) = b_1$, and $b_n = T$. We prove our claim using induction.

Consider $\Delta(\mathrm{ST}_0, \sigma_0) = b_1$, where $q_0$ and $q_1$ are simple states of holons $H^{x_0}$ and $H^{x_1}$ respectively $(x_0, x_1 \in X)$. Thus, we have $\delta^{x_1}(q_0, \sigma_0) = q_1$, where $q_0$ is the boundary state of holon $H^{x_1}$. Based on Definition 4.3, there exists $i_1 \in I$ such that $\delta_{i_1}^{x_1}(q_0, \sigma_0) = q_1$. In $\mathbf{G}$, $q_0$ and $q_1$ are simple states. Based on Definition 4.3, $q_0$ and $q_1$ are also simple states in $\mathbf{G}_{i_1}$ and state $q_0$ goes to $q_1$ via $\sigma_0$ in the equivalent flat model of $\mathbf{G}_{i_1}$ (since $\Delta_{i_1}(.,.)$ is sound). Now consider $\Delta(b_1, \sigma_1) = b_2$, where $q_1$ and $q_2$ are simple states of holons $H^{x_1}$ and $H^{x_2}$ respectively $(x_1, x_2 \in X)$. Thus, we have $\delta^{x_2}(q_1, \sigma_1) = q_2$, where $q_1$ is the boundary state of Holon $H^{x_2}$. Based on Definition

4.3, there exists $i_2 \in I$ such that $\delta_{i_2}^{x_2}(q_1, \sigma_1) = q_2$. In $\mathbf{G}$, $q_1$ and $q_2$ are simple states. Based on Definition 4.3, $q_1$ and $q_2$ are also simple states in $\mathbf{G}_{i_2}$ and state $q_1$ goes to $q_2$ via $\sigma_1$ in the equivalent flat model of $\mathbf{G}_{i_2}$ (since $\Delta_{i_2}(.,.)$ is sound). We claim that in the flat model of $\mathbf{G}_{i_2}$, $q_0$ goes to $q_1$ via event $\sigma_0$. If that is not the case, then it would violate the assumption that $\mathbf{G}_{i_1}$ and $\mathbf{G}_{i_2}$ are MR. Thus, in flat model of $\mathbf{G}_{i_2}$, $\sigma_0$ is enabled at $q_0$ and we have $\Delta_{i_2}(\mathrm{ST}_0, \sigma_0\sigma_1) = b_2$.

Now assume that $\Delta(\mathrm{ST}_0, \sigma_0 \ldots \sigma_r) = \Delta_{i_r}(\mathrm{ST}_0, \sigma_0 \ldots \sigma_{r-1}) = b_r$ ($2 \le r \le n$ and $i_r \in I$). Starting from the simple state $q_0$, we pass through $q_1, \ldots, q_{r-1}$ and reach to $q_r$ via events $\sigma_0 \ldots \sigma_{r-1}$ ($2 \le r \le n$). We show that $\Delta(\mathrm{ST}_0, \sigma_0 \ldots \sigma_{r-1}\sigma_r) = \Delta_{i_{r+1}}(\mathrm{ST}_0, \sigma_0 \ldots \sigma_{r-1}\sigma_r) = b_{r+1}$ ($2 \le r \le n$ and $i_{r+1} \in I$). The equivalent flat state of $b_r$ and $b_{r+1}$ are $q_r$ and $q_{r+1}$. $q_r$ and $q_{r+1}$ are simple states of holons $H^{x_r}$ and $H^{x_{r+1}}$ respectively ($x_r, x_{r+1} \in X$). Thus, we have $\delta^{x_{r+1}}(q_r, \sigma_r) = q_{r+1}$, where $q_r$ is the boundary state of holon $H^{x_{r+1}}$. Based on Definition 4.3, there exists $i_{r+1} \in I$ such that $\delta_{i_{r+1}}^{x_{r+1}}(q_r, \sigma_r) = q_{r+1}$. In $\mathbf{G}$, $q_r$ and $q_{r+1}$ are simple states. Based on Definition 4.3, $q_r$ and $q_{r+1}$ are also simple states in $\mathbf{G}_{i_{r+1}}$ and state $q_r$ goes to $q_{r+1}$ via $\sigma_r$ in the equivalent flat model of $\mathbf{G}_{i_{r+1}}$ (since $\Delta_{i_{r+1}}(.,.)$ is sound). We claim that in the flat model of $\mathbf{G}_{i_{r+1}}$, starting from the simple state $q_0$, we pass through $q_1, \ldots, q_{r-1}$ and reach to $q_r$ via events $\sigma_0 \ldots \sigma_{r-1}$. If that is not the case, then it would violate the assumption that $\mathbf{G}_{i_r}$ and $\mathbf{G}_{i_{r+1}}$ are MR. Thus, in the flat model of $\mathbf{G}_{i_{r+1}}$, we have $\Delta_{i_{r+1}}(\mathrm{ST}_0, \sigma_0 \ldots \sigma_r) = b_{r+1}$.

Now we prove our claim for the general case where for $1 \le k \le n$, $q_k$ is either a simple state of a holon $H^{x_k}$ or in the form of an $m_k$-tuple $q_k = (q_{k,1}, \ldots, q_{k,m_k})$, where $q_{k,t}$ is a simple state in some holon $H^{x_{k,t}}$ (when $q_k$ has at least one AND ancestor), $x_{k,t} \in X$, and $1 \le t \le m_k$. Based on Definition 4.3, $\mathrm{ST}_0$ belongs to all $\mathbf{G}_i$ ($i \in I$). Thus, $q_0$ also belongs to the equivalent flat automaton of $\mathbf{G}_i$ ($i \in I$). We claim that $\exists j \in I$ such that $s = \sigma_0 \ldots \sigma_{n-1}$ (with $\sigma_l \in \Sigma_j$, $0 \le l \le n-1$), $b_1, \ldots, b_n \in B(\mathrm{ST}_j)$ with $\Delta_j(b_l, \sigma_l) = b_{l+1}$ ($1 \le l \le n-1$), $\Delta_j(\mathrm{ST}_0, \sigma_0) = b_1$, and $b_n = T$. We prove our claim using induction.

Without loss of generality, consider $\Delta(\mathrm{ST}_0, \sigma_0) = b_1$, where $q_0$ is a simple state of a holon $H^{x_0}$ and $q_1$ is an $m_1$-tuple $q_1 = (q_{1,1}, \ldots, q_{1,m_1})$. Thus, for holon $H^{x_0}$, we have an outgoing boundary transition that maps $q_0$ to $q_1$ via event $\sigma_0$. Based on Definition 4.3, there exists $i_0 \in I$ such that $\delta_{i_0}^{x_0}(q_0, \sigma_0) = q_1$. Now consider $\Delta(b_1, \sigma_1) = b_2$, where $q_2$ is $m_2$-tuple $q_2 = (q_{2,1}, \ldots, q_{2,m_2})$. Thus, there are boundary transitions between holons that maps $q_1$ to $q_2$. All these transitions should belong to a $\mathbf{G}_{i_1}$ ($i_1 \in I$); otherwise, since the starting(ending) state and the event are the same, it would violate the MR property of holons. Thus, $\Delta_{i_1}(b_1, \sigma_1) = b_2$.

Now assume that $\Delta(\mathrm{ST}_0, \sigma_0 \ldots \sigma_r) = \Delta_{i_r}(\mathrm{ST}_0, \sigma_0 \ldots \sigma_{r-1}) = b_r$ ($2 \le r \le n$ and $i_r \in I$). Starting from $q_0$, we pass through $q_1, \ldots, q_{r-1}$ and reach to $q_r$ via events $\sigma_0 \ldots \sigma_{r-1}$ ($2 \le r \le n$). We show that $\Delta(\mathrm{ST}_0, \sigma_0 \ldots \sigma_{r-1}\sigma_r) = \Delta_{i_{r+1}}(\mathrm{ST}_0, \sigma_0 \ldots \sigma_{r-1}\sigma_r) = b_{r+1}$ ($2 \le r \le n$ and $i_{r+1} \in I$). The equivalent flat state of $b_r$ and $b_{r+1}$ are $q_r$ and $q_{r+1}$. Without loss of generality, assume that $q_r$ is $m_r$-tuple $q_r = (q_{r,1}, \ldots, q_{r,m_r})$ and $q_{r+1}$ is a simple state of holon $H^{x_{r+1}}$ ($x_{r+1} \in X$). Thus, we have $\delta^{x_{r+1}}(q_r, \sigma_r) = q_{r+1}$, where $q_r$ is the boundary state of holon $H^{x_{r+1}}$. Based on Definition 4.3, there exists $i_{r+1} \in I$ such that $\delta_{i_{r+1}}^{x_{r+1}}(q_r, \sigma_r) = q_{r+1}$. In $\mathbf{G}$, $q_r$ and $q_{r+1}$ are simple states. $q_r$ goes to $q_{r+1}$ via $\sigma_r$ in the equivalent flat model of $\mathbf{G}_{i_{r+1}}$ (since $\Delta_{i_{r+1}}(.,.)$ is sound). We claim that in the flat model of $\mathbf{G}_{i_{r+1}}$, starting from $q_0$, we pass through $q_1,\ldots,q_{r-1}$ and reach to $q_r$ via events $\sigma_0\ldots\sigma_{r-1}$. If that is not the case, then it would violate the assumption that $\mathbf{G}_{i_r}$ and $\mathbf{G}_{i_{r+1}}$ are MR. Thus, in the flat model of $\mathbf{G}_{i_{r+1}}$, we have $\Delta_{i_{r+1}}(\mathrm{ST}_0,\sigma_0\ldots\sigma_r) = b_{r+1}$.

For the rest of the combinations of states (e.g., $q_0$ is a tuple, $q_1$ is a simple state), the lemma can be shown to be true similarly.

□

**Remark 4.4.** *We have shown that the union model* $\mathbf{G}$ *is deterministic and* $\Delta(.,.)$ *is sound. Moreover, we proved that no new sequence of events are generated in* $\mathbf{G}$*. Therefore,* $L(\mathbf{G}) = \bigcup_{i\in I} L(\mathbf{G}_i)$ *and the union model formed in Definition 3.4 is the equivalent flat automaton of STS model* $\mathbf{G}$ *in Definition 4.3.*

For the rest of this chapter, the proofs of lemmas and theorem are similar to the proofs given in Chapter 3. We prove that all the corresponding holons in $\mathbf{G}$ and $\mathbf{G}_i$ $(i \in I)$ are MR.

**Lemma 4.3.** *Consider the set of MR models* $\mathcal{G}$ *and the STS* $\mathbf{G}$ *defined in Definition 4.3. For any* $i \in I$, $a \in X_i \cap X$ *and* $\mathcal{T}_i(a) = \mathcal{T}(a) = OR$, $H_i^a = (X_i^a, \Sigma_i^a, \delta_i^a, X_{0i}^a, X_{mi}^a)$ *and* $H^a = (X^a, \Sigma^a, \delta^a, X_0^a, X_m^a)$ *are MR.*

*Proof.* Assume $a \in X \cap X_i$ such that $\mathcal{T}(a) = \mathcal{T}_i(a) = OR$. $H^a = (X^a, \Sigma^a, \delta^a, X_0^a, X_m^a)$ and $H_i^a = (X_i^a, \Sigma_i^a, \delta_i^a, X_{0i}^a, X_{mi}^a)$ are the two holons assigned to state $a$ in $\mathbf{G}$ and $\mathbf{G}_i$ respectively. It follows from the Definition 4.3 and the assumption that all the corresponding holons in $\mathcal{G}$ are MR that $H^a$ and $H_i^a$ $(i \in I)$ are also MR. □

**Corollary 4.1.** *Consider the set of MR models* $\mathcal{G}$ *and the STS* $\mathbf{G}$ *defined in Definition 4.3.* $\mathbf{G}$ *and* $\mathbf{G}_i$ *are MR* $(i \in I)$*.*

**Lemma 4.4.** *Consider the set of MR models* $\mathcal{G}$ *and the STS* $\mathbf{G}$ *defined in Definition 4.3. Let* $f : B(ST) \to \Pi$ *be an SFBC defined over* $\mathbf{G}$*. For any* $i, j \in I$, ${\mathbf{G}_i}^f$ *and* ${\mathbf{G}_j}^f$ *are MR.*

*Proof.*
Similar to Lemma 3.4 we can prove that ${\mathbf{G}_i}^f$ and ${\mathbf{G}_j}^f$ are MR. Here, the SFBC domain instead of state set is the set of basic-ST. □

The reachability and coreachability of union model can be calculated using those of the $\mathbf{G}_i$'s $(i \in I)$.

**Lemma 4.5.** *Let* $\mathbf{G}$ *be the STS defined in Definition 4.3. Then we have*

$$R(\mathbf{G}, true) = \bigvee_{i\in I} R(\mathbf{G}_i,\ true), \tag{4.2}$$

$$\mathrm{CR}(\boldsymbol{G}, true) = \bigvee_{i\in I} \mathrm{CR}(\boldsymbol{G}_i, true). \tag{4.3}$$

*Proof.*

1. We prove that, (i) $\bigvee_{i\in I} R(\mathbf{G}_i, true) \le R(\mathbf{G}, true)$ and (ii) $R(\mathbf{G}, true) \le \bigvee_{i\in I} R(\mathbf{G}_i, true)$.

   i. Assume $b \models \bigvee_{i\in I} R(\mathbf{G}_i, true)$, then $\exists j \in I$ such that $b \models R(\mathbf{G}_j, true)$. State-tree $b$ is reachable in $\mathbf{G}_j$; thus, $\exists s \in \Sigma_j^*$ such that $\Delta_j(\mathrm{ST}_0, s) = b$. Based on Definition 4.3, $s \in \Sigma^*$ and $\Delta(\mathrm{ST}_0, s) = b$. Therefore, $b$ is also reachable in $\mathbf{G}$ and $b \models R(\mathbf{G}, true)$. We have proven that $\bigvee_{i\in I} R(\mathbf{G}_i, true) \le R(\mathbf{G}, \mathrm{true})$.

   ii. Assume $b \models R(\mathbf{G}, true)$. Therefore, $\exists t \in \Sigma^*$ such that in $\mathbf{G}$, $\Delta(\mathrm{ST}_0, t) = b$. Based on Lemma 4.2, $\exists j \in I$ such that $\Delta_j(\mathrm{ST}_0, t) = \Delta(\mathrm{ST}_0, t) = b$. Thus, $b \models R(\mathbf{G}_j, \mathrm{true})$ and $b \models \bigvee_{i\in I} R(\mathbf{G}_i, true)$.

      We proved that $R(\mathbf{G},\ true) \le \bigvee_{i\in I} R(\mathbf{G}_i, true)$.

2. We prove that (i) $\bigvee_{i\in I} CR(\mathbf{G}_i,\ true) \le CR(\mathbf{G},\ true)$ and (ii) $CR(\mathbf{G}, true) \le \bigvee_{i\in I} CR(\mathbf{G}_i, true)$.

   i. The proof will be similar to section (i) in part 1 above.

   ii. Assume $b \models CR(\mathbf{G}, true)$, then $\exists b_m \in B(\mathrm{ST}_m)$ and $t = \sigma_0, \ldots, \sigma_{n-1} \in \Sigma^*$ $(n \ge 1)$ such that $\Delta(b, t) = b_m$. Similar to the proof of section (ii) in part 1, it can be shown that $\exists j \in I$ such that $\Delta_j(b, t) = b_m$. Therefore, $b \models CR(\mathbf{G}_j, true)$ and $b \models \bigvee_{i\in I} CR(\mathbf{G}_i, true)$.

      We proved that $CR(\mathbf{G}, true) \le \bigvee_{i\in I} CR(\mathbf{G}_i, true)$.

□

**Remark 4.5.** *Using Lemma 4.4, we can easily show that the results of Lemmas 4.2 and 4.5 also hold for the STS under the supervision of a SFBC $f : B(ST) \to \Pi$. In particular,*

$$R(\boldsymbol{G}^f, true) = \bigvee_{i\in I} R(\boldsymbol{G}_i^f,\ true) \tag{4.4}$$

$$\mathrm{CR}(\boldsymbol{G}^f, true) = \bigvee_{i\in I} \mathrm{CR}(\boldsymbol{G}_i^f,\ true). \tag{4.5}$$

## 4.3 Solution: Necessary and Sufficient Conditions

Theorem 4.3 is our main result. It presents a set of necessary and sufficient conditions for having a solution for RNSCP-STS (Problem 4.1).

**Theorem 4.3.** *Consider RNSCP-STS in Problem 4.1 and $\boldsymbol{G} = (ST, H, \Sigma, \Delta, ST_0, ST_m)$ in Definition 4.3. Suppose Assumption 4.2 holds. Define the predicate $P$ as*

$$P = \left[ \bigwedge_{j \in I} \left( P_j \vee \left[ R(\boldsymbol{G}, true) \wedge \neg R(\boldsymbol{G}_j, true) \right] \right) \right] \wedge R(\boldsymbol{G}, true). \tag{4.6}$$

1. *If there exists a predicate $K \leq P$ with $K \neq$ false such that*

   *(a) $K$ is controllable with respect to $\boldsymbol{G}$,*

   *(b) $K$ is $\boldsymbol{G}_i$-nonblocking for all $i \in I$,*

   *then RNSCP-STS has a solution $f$ with $R(\boldsymbol{G}^f, true) = K$.*

2. *Conversely, if $f$ is a solution of problem, then $K = R(\boldsymbol{G}^f, true)$ is controllable with respect to $\boldsymbol{G}$, $\boldsymbol{G}_i$-nonblocking for all $i \in I$ and $K \leq P$.*

Before proving Theorem 4.3, we define the notion of *sub-STS* below.

**Definition 4.4.** *Consider $\boldsymbol{G}_1 = (ST_1, H_1, \Sigma_1, \Delta_1, ST_{01}, ST_{m1})$ and $\boldsymbol{G}_2 = (ST_2, H_2, \Sigma_2, \Delta_2, ST_{02}, ST_{m2})$, where $ST_1 = (X_1, x_0, \mathcal{T}_1, \varepsilon_1)$, $H_1 = \left\{ H_1^a \mid a \in X_1,\ \mathcal{T}_1(a) = OR\ \&\ H_1^a = (X_1^a, \Sigma_1^a, \delta_1^a, X_{01}^a, X_{m1}^a) \right\}$, $ST_2 = (X_2, x_0, \mathcal{T}_2, \epsilon_2)$ and $H_2 = \left\{ H_2^a \mid a \in X_2,\ \mathcal{T}_2(a) = OR\ \&\ H_2^a = (X_2^a, \Sigma_2^a, \delta_2^a, X_{02}^a, X_{m2}^a) \right\}$. We say $\boldsymbol{G}_1$ is a **sub-STS** of $\boldsymbol{G}_2$ and write $\boldsymbol{G}_1 \subseteq \boldsymbol{G}_2$ if the following conditions are true.*

1. *$ST_{01} = ST_{02}$ and $ST_{m1} \leq ST_{m2}$.*

2. *$ST_1$ is a sub-ST of $ST_2$: $ST_1 \in \boldsymbol{ST}(ST_2)$.*

3. *For all $a \in X_1 \cap X_2$ such that $\mathcal{T}_1(a) = \mathcal{T}_2(a)$, the following propositions are true.*

   *(a) $X_{01}^a \subseteq X_{02}^a$, $X_{m1}^a \subseteq X_{m2}^a$ and $X_1^a \subseteq X_2^a$.*

   *(b) $\forall x \in (X_i^a \cap X_j^a)$ and $x \in X_{Ii}^a$, then $x \in X_{Ij}^a$ and vice versa. The same condition should also be true for boundary states.*

   *(c) $\Sigma_1^a \subseteq \Sigma_2^a$ and for all $s \in \Sigma_1^a$, $\delta_1^a(x_{01}^a, s) = \delta_2^a(x_{02}^a, s)$.*

4. *$\Sigma_1 \subseteq \Sigma_2$ and for all $s \in \Sigma_1$, $\Delta_1(ST_{01}, s) = \Delta_2(ST_{02}, s)$.*

To prove Theorem 4.3, we need the results in Lemmas 4.6 to 4.9.

**Lemma 4.6.** *Consider two STS $\boldsymbol{G}_1 = (ST_1, H_1, \Sigma_1, \Delta_1, ST_{01}, ST_{m1})$ and $\boldsymbol{G}_2 = (ST_2, H_2, \Sigma_2, \Delta_2, ST_{02}, ST_{m2})$. Assume $\boldsymbol{G}_1$ is a sub-STS of $\boldsymbol{G}_2$. Then $R(\boldsymbol{G}_1, true) \leq R(\boldsymbol{G}_2, true)$ and $CR(\boldsymbol{G}_1, true) \leq CR(\boldsymbol{G}_2, true)$.*

*Proof.*

1. Let $b \models R(\mathbf{G}_1, true)$. If $b = \mathrm{ST}_{01}$ (initial ST), then obviously $b \models R(\mathbf{G}_2, \mathrm{true})$ ($\mathrm{ST}_{01} = \mathrm{ST}_{02}$). Suppose $b \neq \mathrm{ST}_{01}$. Therefore, $b \in B(\mathrm{ST}_1)$, $\exists b_1, \ldots, b_{n-1} \in B(\mathrm{ST}_1)$, and $\sigma_0 \ldots \sigma_{n-1} \in \Sigma_1^*$ ($n \geq 1$) such that $\Delta_1(b_l, \sigma_l) = b_{l+1}$ ($0 \leq l \leq n-2$), $\Delta_1(\mathrm{ST}_{01}, \sigma_0) = b_1$, $\Delta_1(b_{n-1}, \sigma_{n-1}) = b$, and $b_l \models R(\mathbf{G}_1, true)$ for $l \in \{1, \ldots, n-1\}$. Based on the definition of sub-STS in Definition 4.4, since $\mathbf{G}_1 \subseteq \mathbf{G}_2$, then $b_1, \ldots, b_{n-1}, b \in B(\mathrm{ST}_2)$, $\sigma_0 \ldots \sigma_{n-1} \in \Sigma_2^*$, $\Delta_2(\mathrm{ST}_{01}, \sigma_0) = b_1$ ($\mathrm{ST}_{01} = \mathrm{ST}_{02}$), $\Delta_2(b_l, \sigma_l) = b_{l+1}$ ($1 \leq l \leq n-2$), and $\Delta_2(b_{n-1}, \sigma_{n-1}) = b$. Therefore, $b_l \models R(\mathbf{G}_2, true)$ ($1 \leq l \leq n-1$) and $b \models R(\mathbf{G}_2, \mathrm{true})$. We can conclude that $R(\mathbf{G}_1, true) \leq R(\mathbf{G}_2, true)$.

2. Let $b \models CR(\mathbf{G}_1, true)$. If $b \in B(\mathrm{ST}_{m1}) \subseteq B(\mathrm{ST}_{m2})$, then $b \models CR(\mathbf{G}_2, \mathrm{true})$. Suppose $b \notin B(\mathrm{ST}_{m1})$. Therefore, $\exists b_1, \ldots, b_{m-1} \in B(\mathrm{ST}_1)$, $b_m \in B(\mathrm{ST}_{m1})$, and $\sigma_0 \ldots \sigma_{m-1} \in \Sigma_1^*$ such that $\Delta_1(b, \sigma_0) = b_1$, $\Delta_1(b_l, \sigma_l) = b_{l+1}$ ($l \in \{1, \ldots, m-1\}$), and $b, b_l \models CR(\mathbf{G}_1, true)$ for $l \in \{1, \ldots, m\}$. Since $\mathbf{G}_1 \subseteq \mathbf{G}_2$, then $\mathrm{ST}_{m1} \subseteq \mathrm{ST}_{m2}$, $b, b_1, \ldots, b_m \in B(\mathrm{ST}_2)$, $\sigma_0 \ldots \sigma_{m-1} \in \Sigma_2^*$, $\Delta_2(b, \sigma_0) = b_1$, and $\Delta_2(b_l, \sigma_l) = b_{l+1}$ ($l \in \{1, \ldots, m-1\}$). Therefore, $b \models CR(\mathbf{G}_2, true)$ and we conclude that $CR(\mathbf{G}_1, true) \leq CR(\mathbf{G}_2, true)$.

□

We prove that under the conditions defined below, the relation between the reachability functions of two STS is not affected under the supervision of SFBC.

**Lemma 4.7.** *Consider $\mathbf{G}_1 = (ST_1, H_1, \Sigma_1, \Delta_1, ST_0, ST_{m1})$ and $\mathbf{G}_2 = (ST_2, H_2, \Sigma_2, \Delta_2, ST_0, ST_{m2})$. Suppose they are MR and $\mathbf{G}_1$ is a sub-STS of $\mathbf{G}_2$. Assume $P \in \mathrm{Pred}(ST)$ ($ST = ST_1 \vee ST_2$), $P \neq false$, and $ST_0 \models P$. Moreover, $P$ is controllable with respect to $\mathbf{G}_2$ and $f : B(ST) \to \Pi$ a SFBC such that $R(\mathbf{G}_2{}^f, true) = P$. Then*

$$R(\mathbf{G}_1{}^f,\ true) \leq R(\mathbf{G}_2{}^f, true), \tag{4.7}$$

$$R(\mathbf{G}_1{}^f, true) = R(\mathbf{G}_2{}^f, true) \wedge R(\mathbf{G}_1, true), \tag{4.8}$$

$$R(\mathbf{G}_1^f, true) = R(\mathbf{G}_1, P). \tag{4.9}$$

*Proof.*

1. Since $\mathbf{G}_1$ is a sub-automaton of $\mathbf{G}_2$, $\mathbf{G}_1{}^f$ is a sub-automaton of $\mathbf{G}_2{}^f$. Hence, (4.7) follows.

2. We prove that (i) $R(\mathbf{G}_1{}^f, true) \leq R(\mathbf{G}_2{}^f, true) \wedge R(\mathbf{G}_1, true)$ and (ii) $R(\mathbf{G}_2{}^f, true) \wedge R(\mathbf{G}_1, true) \leq R(\mathbf{G}_1{}^f, true)$.

   i. We know that $R(\mathbf{G}_1{}^f, true) \leq R(\mathbf{G}_1, true)$ and we proved that $R(\mathbf{G}_1{}^f, true) \leq R(\mathbf{G}_2{}^f, true)$; therefore, we have $R(\mathbf{G}_1{}^f, true) \leq R(\mathbf{G}_2{}^f, true) \wedge R(\mathbf{G}_1, true)$.

ii. Assume $b \models R(\mathbf{G_2}^f, true) \wedge R(\mathbf{G}_1, true)$. Therefore, we have $b \models R(\mathbf{G_2}^f, true)$ and $b \models R(\mathbf{G}_1, true)$. We claim that $\exists s \in \Sigma_1^*$, such that $b = \Delta_1(ST_0, s)$ in $\mathbf{G}_1$ and $b = \Delta_2^f(ST_0, s)$ in $\mathbf{G_2}^f$, where $\Delta_2^f(.,.)$ represents transitions in $\mathbf{G}_2$ under the supervision of $f$. If that is not the case, for every $s_1 \in \Sigma_1^*$ such that $b = \Delta_1(ST_0, s_1)$, $\Delta_2^f(ST_0, s_1)$ does not exists and $s_2 \in \Sigma_2^*$ such that $b = \Delta_2^f(ST_0, s_2)$ and $\Delta_1(ST_0, s_2)$ does not exists. Since every transition in $\mathbf{G}_2^f$ also exists in $\mathbf{G}_2$, $b = \Delta_2(ST_0, s_2)$. But $\mathbf{G}_1$ and $\mathbf{G}_2$ are MR and this is not possible. So let $b_0, b_1 \ldots, b_{n-1}, b$ ($n \geq 1$, $\mathrm{ST}_0 = b_0$) be the sequence of ST in $\mathbf{G}_1$ when $s$ is executed. Since the sequence is enabled under the supervision of $f$ (in $\mathbf{G_2}^f$), it remains enabled in $\mathbf{G_1}^f$. Therefore, we can conclude that $b \models R(\mathbf{G_1}^f, true)$.

Thus, we have proved that $R(\mathbf{G_1}^f, true) = R(\mathbf{G_2}^f, true) \wedge R(\mathbf{G}_1, true)$.

3. We know that $R(\mathbf{G}_2^f, true) = P$; therefore,

$$R(\mathbf{G}_1^f, true) = P \wedge R(\mathbf{G}_1, true) \qquad \text{(by (4.8))}$$

We prove that (i) $P \wedge R(\mathbf{G}_1, true) \leq R(\mathbf{G}_1, P)$ and (ii) $R(\mathbf{G}_1, P) \leq P \wedge R(\mathbf{G}_1, true)$.

i. We use strong induction. Base case: since $\mathrm{ST}_0 \models P$ and $\mathrm{ST}_0 \models R(\mathbf{G}_1, true)$, then $\mathrm{ST}_0 \models R(\mathbf{G}_1, P)$.

Strong inductive step: now we assume that $P \wedge R(\mathbf{G}_1, true) \leq R(\mathbf{G}_1, P)$ holds for all states that are located within a distance of $n$ transitions from $\mathrm{ST}_0$. The distance of a state $b$ from $\mathrm{ST}_0$ is defined as the shortest path to that state. We need to prove that $P \wedge R(\mathbf{G}_1, true) \leq R(\mathbf{G}_1, P)$ also holds for all states that are located within $n+1$ transitions from $\mathrm{ST}_0$ ($n \geq 0$). Suppose $b_{n+1} \models P \wedge R(\mathbf{G}_1, true)$ and is at a distance of $n+1$ from $\mathrm{ST}_0$. Since $P$ is controllable and $R(\mathbf{G}_2^f, true) = P$; therefore, $\exists t \in \Sigma_2^*$ such that $b_{n+1} = \Delta_2^f(\mathrm{ST}_0, t)$ and the trajectory on the $t$ sequence satisfies $P$. We have $b_{n+1} \models R(\mathbf{G}_1, true)$; moreover, $\mathbf{G}_1$ and $\mathbf{G}_2$ are MR. Therefore, $t \in L(\mathbf{G}_1)$ and the trajectory is in $R(\mathbf{G}_1, true)$.

$b_{n+1}$ is reachable from $\mathrm{ST}_0$ and all the states leading to $b_{n+1}$ satisfy $P$; therefore, $b_{n+1} \models R(\mathbf{G}_1, P)$. $b_{n+1}$ is located within $n+1$ transitions from $\mathrm{ST}_0$ and satisfies $P \wedge R(\mathbf{G}_1, true) \leq R(\mathbf{G}_1, P)$. By the strong induction, we can say that $P \wedge R(\mathbf{G}_1, true) \leq R(\mathbf{G}_1, P)$ is true.

ii. It is clear that $R(\mathbf{G}_1, P) \leq P$ and $R(\mathbf{G}_1, P) \leq R(\mathbf{G}_1, true)$. Therefore, we have $R(\mathbf{G}_1, P) \leq P \wedge R(\mathbf{G}_1, true)$.

□

Remark 4.6 considers $\mathbf{G}_1$ and $\mathbf{G}_2$ in Lemma 4.7.

**Remark 4.6.** *Similar to Remark 3.3, we can interpret the result of Lemma 4.7 as follows.*

*$P$ is controllable with respect to $\mathbf{G}_2$. Then $P|_{B(ST_1)}$ (the restriction of $P$ to $B(ST_1)$) is controllable with respect to $\mathbf{G}_1$.*

Results similar to those of Lemma 4.7 hold for coreachability predicate.

**Lemma 4.8.** *Consider $\mathbf{G}_1 = (ST_1, H_1, \Sigma_1, \Delta_1, ST_0, ST_{m1})$ and $\mathbf{G}_2 = (ST_2, H_2, \Sigma_2, \Delta_2, ST_0, ST_{m2})$. Suppose they are MR and $\mathbf{G}_1$ is a sub-STS of $\mathbf{G}_2$. Assume $P \in \mathrm{Pred}(ST)$ ($ST = ST_1 \vee ST_2$), $P \neq false$, and $ST_0 \models P$. Moreover, $P$ is controllable and nonblocking with respect to $\mathbf{G}_1$. Let $f : B(ST) \to \Pi$ to be a SFBC such that $R(\mathbf{G}_2{}^f, true) = P$. Then*

$$CR(\mathbf{G}_1{}^f,\ true) \le CR(\mathbf{G}_2{}^f, true), \tag{4.10}$$

$$CR(\mathbf{G}_1{}^f,\ true) \le CR(\mathbf{G}_2{}^f, true) \wedge CR(\mathbf{G}_1,\ true), \tag{4.11}$$

$$CR(\mathbf{G}_1, P) \le CR(\mathbf{G}_1{}^f, true). \tag{4.12}$$

*Proof.*

1. (4.10) follows from the fact that $\mathbf{G}_1^f$ is a sub-automaton of $\mathbf{G}_2$.

2. (4.11) follows from (4.10) and that $\mathbf{G}_1^f$ is a sub-automaton of $\mathbf{G}_1$.

3. $P$ is controllable; therefore, $P|_{B(\mathrm{ST}_1)}$ is controllable (Remark 4.6). $P$ is $\mathbf{G}_1$-nonblocking

   $$R(\mathbf{G}_1, P) \le CR(\mathbf{G}_1, P).$$

   Intuitively, $R(\mathbf{G}_1, P)|_{B(\mathrm{ST}_1)} = R(\mathbf{G}_1, P|_{B(\mathrm{ST}_1)})$ and $CR(\mathbf{G}_1, P)|_{B(\mathrm{ST}_1)} = CR(\mathbf{G}_1, P|_{B(\mathrm{ST}_1)})$. Thus,

   $$R(\mathbf{G}_1, P|_{B(\mathrm{ST}_1)}) \le CR(\mathbf{G}_1, P|_{B(\mathrm{ST}_1)}).$$

   With $f_1 = f|_{B(\mathrm{ST}_1)}$ and using Theorem 2.3,

   $$CR(\mathbf{G}_1, P|_{B(\mathrm{ST}_1)}) \le CR(\mathbf{G}_1^{f_1}, \mathrm{true}),$$

   and thus,

   $$CR(\mathbf{G}_1, P) \le CR(\mathbf{G}_1^{f}, \mathrm{true}).$$

∎

**Lemma 4.9.** *Consider the set of MR models $\mathcal{G}$ and the STS $\mathbf{G} = (ST, H, \Sigma, \Delta, ST_0, ST_m)$ defined in Definition 4.3. Suppose $K \in Pred(ST)$, $K \le R(\mathbf{G}, true)$ and let $K_i = \big(K \wedge R(\mathbf{G}_i, true)\big)|_{B(ST_i)}$. If $K$ is controllable with respect to $\mathbf{G}$, then $K_i$ is controllable with respect to $\mathbf{G}_i$ ($i \in I$).*

*Proof.* Suppose $K$ is controllable with respect to $\mathbf{G}$. We have to prove that $K_i$ is controllable with respect to $\mathbf{G}_i$ ($i \in I$). From controllability of $K$, we can conclude that there exists a SFBC $f$ such that $R(\mathbf{G}^f, \text{true}) = K$. By Remark 4.1, $\mathbf{G}$ and $\mathbf{G}_i$ are MR. Thus, applying Lemma 4.7 and Remark 4.6 to $\mathbf{G}_i$ and $\mathbf{G}$, we can conclude that $K_i = (K \wedge R(\mathbf{G}_i, \text{true}))|_{B(\text{ST}_i)}$ is controllable with respect to $\mathbf{G}_i$. □

Now we can prove Theorem 4.3.

*Proof of Theorem 4.3.*

1. Since $K$ is controllable with respect to $\mathbf{G}$ by assumption, by Theorem 2.2, there exists a SFBC $f$ such that

$$R(\mathbf{G}^f, true) = K \tag{4.13}$$

From assumption (ii), $R(\mathbf{G}_i, K) \le CR(\mathbf{G}_i, K)$ ($i \in I$).

$$\begin{aligned}
\bigvee_{i\in I} R(\mathbf{G}_i, K) &\le \bigvee_{i\in I} CR(\mathbf{G}_i, K) \\
\bigvee_{i\in I} R(\mathbf{G}_i^f, \text{true}) &\le \bigvee_{i\in I} CR(\mathbf{G}_i^f, \text{true}) && \text{(by Lemmas 4.7 and 4.8)} \\
R(\mathbf{G}^f, \text{true}) &\le CR(\mathbf{G}^f, \text{true}) && \text{(by Remark 4.5)}
\end{aligned}$$

Thus, $K$ is nonblocking with respect to $\mathbf{G}$. Now we show that $f$ is a solution to RNSSCP, i.e., conditions (1) and (2) in Problem 4.3 are true.

$$\begin{aligned}
R(\mathbf{G}_i^f, true) &= R(\mathbf{G}^f, true) \wedge R(\mathbf{G}_i, true) && \text{(by Lemma 4.7)} \\
&= K \wedge R(\mathbf{G}_i,\ true) && \text{(by (4.13))} \\
&\le P \wedge R(\mathbf{G}_i, true) \\
&\le \Big(P_i \vee \big[R(\mathbf{G}, true) \wedge \neg R(\mathbf{G}_i, true)\big]\Big) \wedge R(\mathbf{G}_i, true) \\
&= \big(P_i \wedge R(\mathbf{G}_i, true)\big) \vee \big(R(\mathbf{G}, true) \wedge \neg R(\mathbf{G}_i, true) \wedge R(\mathbf{G}_i, true)\big) \\
&= P_i \wedge R(\mathbf{G}_i, true) \\
&\le P_i
\end{aligned}$$

Now we just need to prove that $R(\mathbf{G}_i^f, true) \le CR(\mathbf{G}_i^f, true)$.

$$
\begin{aligned}
R(\mathbf{G}_i^f, true) &= R(\mathbf{G}_i, K) && \text{(by Lemma 4.7)}\\
&\le CR(\mathbf{G}_i, K) && \text{($K$ is $\mathbf{G}_i$-nonblocking)}\\
&\le CR(\mathbf{G}_i^f, true) && \text{(by Lemma 4.8)}
\end{aligned}
$$

2. Since $K = R(\mathbf{G}^f, true)$ and by Theorem 2.2, $K$ is controllable with respect to $\mathbf{G}$. Since $f$ solves the RNSSCP, $R(\mathbf{G}_i^f, \text{true}) \le CR(\mathbf{G}_i^f, \text{true})$. By Lemma 4.7, $R(\mathbf{G}_i^f, \text{true}) = K \wedge R(\mathbf{G}_i, \text{true})$. Define $K_i = K \wedge R(\mathbf{G}_i, \text{true})|_{B(\text{ST}_i)}$ and $f_i = f|_{B(\text{ST}_i)}$. Thus, $R(\mathbf{G}_i^{f_i}, \text{true}) = K_i$ and by Theorem 2.2,

$$
R(\mathbf{G}_i^{f_i}, \text{true}) = R(\mathbf{G}_i, K_i) = CR(\mathbf{G}_i, K_i). \tag{4.14}
$$

Note that the domain of the above predicates are $Q_i$. From (4.14), we conclude that

$$
R(\mathbf{G}_i, K \wedge R(\mathbf{G}_i, \text{true})) = CR(\mathbf{G}_i, K \wedge R(\mathbf{G}_i, \text{true})).
$$

Since states that are satisfying $K$, but they are not in $Q_i$ do not satisfy the above reachability and coreachability predicates, we can conclude

$$
R(\mathbf{G}_i, K) = CR(\mathbf{G}_i, K).
$$

Next we will prove that $K \le P$.

$$
\begin{aligned}
P_i \vee \Big[R(\mathbf{G}, true) \wedge \neg R(\mathbf{G}_i, true)\Big] &= \Big[P_i \vee R(\mathbf{G}, true)\Big] \wedge \Big[P_i \vee \neg R(\mathbf{G}_i, true)\Big]\\
&\ge \Big[\big(R(\mathbf{G}_i, true) \wedge R(\mathbf{G}^f, true)\big) \vee R(\mathbf{G}, true)\Big] \wedge\\
&\quad \Big[\big(R(\mathbf{G}_i, true) \wedge R(\mathbf{G}^f, true)\big) \vee \neg R(\mathbf{G}_i, true)\Big] && \text{(Lemma 4.7)}\\
&= R(\mathbf{G}, true) \wedge \Big[R(\mathbf{G}^f, true) \vee \neg R(\mathbf{G}_i, true)\Big]\\
&= R(\mathbf{G}^f, true) \vee \big(\neg R(\mathbf{G}_i, true) \wedge R(\mathbf{G}, true)\big)\\
&\ge R(\mathbf{G}^f, true)\\
&= K.
\end{aligned}
$$

Using the above result, we can conclude that

$$
\begin{aligned}
P &= \Big[\bigwedge_{j\in I}\Big(P_j \vee \big[R(\mathbf{G}, true) \wedge \neg R(\mathbf{G}_j, true)\big]\Big)\Big] \wedge R(\mathbf{G}, true) \\
&\geq K \wedge R(\mathbf{G}, true) \\
&= K. && \text{(Since } R(\mathbf{G}, true) \geq R(\mathbf{G}^f, true) = K\text{)}
\end{aligned}
$$

□

We define the set of all controllable and $\mathbf{G}_i$-nonblocking predicates of $P$ as $\mathrm{CNb_GP}(P) = \{K \in \mathrm{Pred}(Q) \mid K \leq P \;\&\; K$ controllable with respect to $\mathbf{G}$ and $\mathbf{G}_i$-nonblocking $\forall\, i \in I\}$.

**Lemma 4.10.** $\mathrm{CNb_GP}(P)$ *is nonempty, closed under disjunction operation and has a supremal element.*

*Proof.*
*Claim 1.* $\mathrm{CNb_GP}(P)$ is nonempty since $false \in \mathrm{CNb_GP}(P)$.
*Claim 2.* Suppose $\Lambda$ is the index set of $\mathrm{CNb_GP}(P)$ and $K_\lambda \in \mathrm{CNb_GP}(P)$ for all $\lambda \in \Lambda$. We have to prove that $K = \bigvee_{\lambda\in\Lambda} K_\lambda \in \mathrm{CNb_GP}(P)$, i.e., $K \leq P$, $K$ is controllable with respect to $\mathbf{G}$ and $\mathbf{G}_i$-nonblocking. It is obvious that $K \leq P$ and [74] proved that $K$ is controllable with respect to $\mathbf{G}$. Therefore, we only need to prove that $K$ is $\mathbf{G}_i$-nonblocking. In other words, we want to prove that $R(\mathbf{G}_i, K) \leq CR(\mathbf{G}_i, K)$ for all $i \in I$. Assume $q \models R(\mathbf{G}_i, K)$; therefore, $K_\lambda$ is controllable with respect to $\mathbf{G}$. It follows from Lemma 4.9 that $\big(K_\lambda \wedge R(\mathbf{G}_i, \mathrm{true})\big)|_{Q_i}$ is controllable with respect to $\mathbf{G}_i$. Thus $q$ is reachable in $\mathbf{G}_i$ using a trajectory that lies in $\big(K_\lambda \wedge R(\mathbf{G}_i, \mathrm{true})\big)|_{Q_i}$. Therefore, $q \models K$ and $q \models R(\mathbf{G}_i, true)$. We know that $K = \bigvee_{\lambda\in\Lambda} K_\lambda$; thus, $\exists \lambda \in \Lambda$ such that $q \models K_\lambda$. We have $q \models R(\mathbf{G}_i, true)$ and $q \models K_\lambda$; therefore, $q \models R(\mathbf{G}_i, K_\lambda)$. Since $K_\lambda$ is $\mathbf{G}_i$-nonblocking ($R(\mathbf{G}_i, K_\lambda) \leq CR(\mathbf{G}_i, K_\lambda)$), $q \models CR(\mathbf{G}_i, K_\lambda)$. $CR(.,.)$ is a monotonically increasing function and; therefore, $CR(\mathbf{G}_i, K_\lambda) \leq CR(\mathbf{G}_i, K)$. Thus, we can conclude that $q \models CR(\mathbf{G}_i, K)$ and $R(\mathbf{G}_i, K) \leq CR(\mathbf{G}_i, K)$. □

Let $K^{\uparrow}$ denotes $\mathrm{supCNb_GP}(P)$. $K^{\uparrow}$ characterizes the largest (maximally permissive) solution of the robust supervisory control problem. In the next section, we present a computational procedure for $K^{\uparrow}$.

## 4.4 Solution: Computational Procedure

The following theorem defines an algorithm to calculate the supremal solution of Theorem 4.3, $K^{\uparrow}$.

**Theorem 4.4.** *Assume that* $\mathbf{G}$ *is the STS introduced in Definition 4.3 and* $P$ *is the predicate in* (4.6). *Then* $K^{\uparrow} = \mathrm{supCNb_G P}(P)$ *can be calculated using the following iterative procedure which terminates in a finite number of steps less than or equal to the number of states satisfying* $P$.

1. *Set* $r = 1$ *and* $S_r = P$.
2. $L_i = CR(\mathbf{G}_i, S_r)$ *for all* $i \in I$.
3. $S'_r = \left[\bigwedge_{i\in I} L_i\right] \vee \left[\bigvee_{i\in I}\left(L_i \wedge \neg(\bigvee_{j\in I\ \&\ j\neq i} P_{Q_j})\right)\right]$.
4. $S_{r+1} = R(\mathbf{G}, \langle S'_r\rangle)$.
5. *If* $S_{r+1} \neq S_r$, *set* $r = r+1$ *and go to step 2.*
6. *End* ($S_r = K^{\uparrow}$).

*where* $\langle . \rangle$ *is calculated with respect to* $\mathbf{G}$ *and* $P_{Q_j}$ *is a predicate that represents the states of* $\mathbf{G}_j$ $(i, j \in I)$.

*Proof.*
First we prove that the algorithm converges in a finite number of iterations. In this chapter, the plant models $\mathbf{G}_i$ are finite-state; therefore, the set of predicates $\mathrm{Pred}(Q)$ is a finite set. Furthermore, for each iteration, $L_i \leq S_r$ $(i \in I)$; therefore, $\bigwedge_{i\in I} L_i \leq S_r$ and $\bigvee_{i\in I} L_i \leq S_r$. It can easily be seen that $\bigvee_{i\in I}\left(L_i \wedge \neg(\bigvee_{j\in I\ \&\ j\neq i} P_{Q_j})\right) \leq \bigvee_{i\in I} L_i$. Therefore, we can conclude that $S'_r \leq S_r$. Based on step 4, we also have $S_{r+1} \leq S'_r \leq S_r$. Therefore, this algorithm is nonincreasing and will converge to either $K^{\uparrow} = \mathit{false}$ or $K^{\uparrow} \neq \mathit{false}$ in a finite number of iterations less than or equal to the number of states satisfying $S_1 = P$.

Now suppose Assume that the algorithm converges to $S_m$ for some $m \geq 1 : S_m = S'_m = S_{m+1}$. We prove that (i) $S_m \leq P$, (ii) $S_m$ is controllable with respect to $\mathbf{G}$ and (iii) $S_m$ is $\mathbf{G}_i$-nonblocking for all $i \in I$.

i. We have proved that our algorithm produces a nonincreasing sequence of predicates; therefore, after $m$ iteration, we have $S_m \leq S_{m-1} \leq \cdots \leq S_1 = P$.

ii. Based on Lemma 2.2, $S_{m+1}$ is the supremal sub-predicate of $S'_m$. Therefore, $S_{m+1}$ is controllable with respect to $\mathbf{G}$. Since $S_{m+1} = S_m$, we can conclude that $S_m$ is controllable with respect to $\mathbf{G}$.

iii. We have to prove that $R(\mathbf{G}_k, S_m) \leq CR(\mathbf{G}_k, S_m)$ for all $k \in I$. Let us fix $k$. Assume $b \models R(\mathbf{G}_k, S_m)$, then $b \models S_m \wedge R(\mathbf{G}_k, \mathit{true})$. Therefore, $\exists t = \sigma_0, \ldots, \sigma_{n-1} \in \Sigma_k^*$ such that starting from $\mathrm{ST}_0$, we pass through $\mathrm{ST}_0, \ldots, b_{n-1} \in B(\mathrm{ST})$ and reach $b$, where $\delta_k(b_l, \sigma_l) = b_{l+1}$ $(l \in \mathcal{L} = \{0, \ldots, n-2\})$ and $\delta_k(b_{n-1}, \sigma_{n-1}) = b$. Moreover, $\mathrm{ST}_0, \ldots, b_{n-1}, b \models S_m$ and $b_l \models R(\mathbf{G}_k, \mathit{true})$ for all $l \in \mathcal{L}$. Since $S_m = S'_m$, we can conclude that

$ST_0, \ldots, b_{n-1}, b \models S'_m$. Based on the step 3 of algorithm, $ST_0, \ldots, b_{n-1}, b \models \bigwedge_{k \in I} L_k$ or $ST_0, \ldots, b_{n-1}, b \models \bigvee_{k \in I} \left( L_k \wedge (\bigwedge_{j \in I \text{ \& } j \neq k} \neg P_{Q_j}) \right)$. Either way, $ST_0, \ldots, b_{n-1}, b \models (L_k = CR(\mathbf{G}_k, S_m))$.

Therefore, $R(\mathbf{G}_k, S_m) \leq CR(\mathbf{G}_k, S_m)$.

Now let the iterative steps (2) to (4) in Theorem 4.4 be represented by an operative $\Psi(.)$. In steps (ii) and (iii) above, we showed that every fix-point of $\Psi(P_1)$ is controllable and $\mathbf{G}_i$-nonblocking. If $S_m \neq K^{\uparrow}$, then $S_m \leq K^{\uparrow} \leq P$. Thus, $\Psi(S_m) \leq \Psi(K^{\uparrow}) \leq \Psi(P)$ and $S_m \leq K^{\uparrow} \leq \Psi(P)$. Apply $\Psi(.)$ $m-1$ times; $S_m \leq K^{\uparrow} \leq \Psi^{(m-1)}(P) = S_m$ ($\Psi^{(m)}(.)$ denotes that $\Psi(.)$ is applied $m$ times). Thus $S_m = K^{\uparrow}$. □

## 4.5 Example

A Flexible Manufacturing System (FMS) is an automated system that receives raw materials as input, and after performing multiple processes on the received items, delivers the processed materials in the output. A limited number of machines perform the processes on the input workpieces. Therefore, there is a competition on allocating resources (machines) between workpieces. A deadlock can happen in this system if the robots want to upload a machine with more than its capacity. To avoid deadlocks in the system, different approaches have been proposed to design a supervisory control for FMS. In this thesis, we model the FMS by STS, define the RNSCP-STS for that, and calculate the solution (supervisor). We adopt the model of FMS from [10].

As shown in Figure 4.6, we assume that the FMS has 4 machines ($M_1$, $M_2$, $M_3$, and $M_4$), 4 input/output pairs ($I_1/O_1$, $I_2/O_2$, $I_3/O_3$, and $I_4/O_4$), and 3 robots ($R_1$, $R_2$, and $R_3$) that helps to move the workpieces. The STS model of FMS is shown in Figure 4.7. Initially, only $M_1$, $M_2$, and $M_3$ are active during the production process. If the workload of $M_1$ increases, $M_4$ is activated. Figure 4.7 and Figure 4.8 show the two models of FMS.

The $FMS_1$ (Figure 4.7) has 3 production processes that are shown in Figure 4.9. In the first process, the raw material is received through $I_1$, $R_2$ delivers the material to $M_2$, and after the $M_1$ process, $R_2$ transfers it to $O_1$. In the second process, the material is received through $I_2$, $R_3$ delivers it to $M_1$, then $R_2$ hands it over to $M_3$, and finally, $R_1$ transfers it to $O_2$. In the third process, the material is received through $I_3$, $R_1$ delivers it to $M_1$, then $R_2$ hands it over to $M_2$, and finally $R_3$ transfers it to $O_3$.

As shown in Figure 4.10, $FMS_2$ (Figure 4.8) has the same first 2 production processes described for $FMS_1$ except for the third process. In $FMS_2$, the third process is done in a different way. The material is received through $I_4$, $R_1$ delivers it to $M_3$, then $R_2$ hands it over to $M_4$, and finally $R_3$ transfers it to $O_4$.

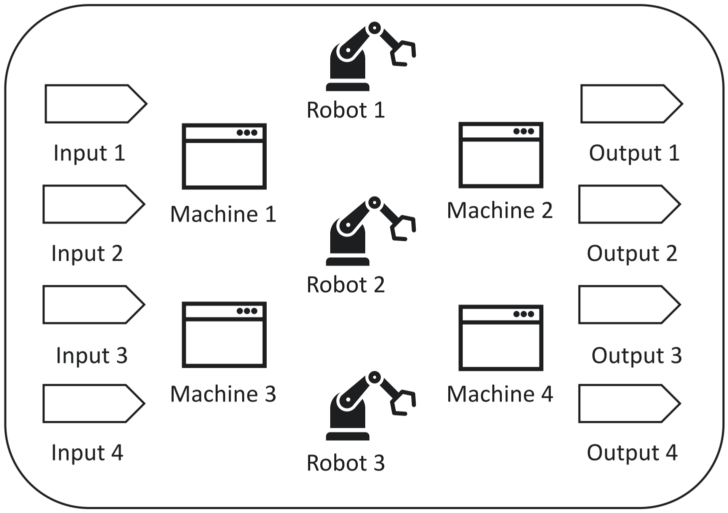



Figure 4.6: The layout of FMS components.

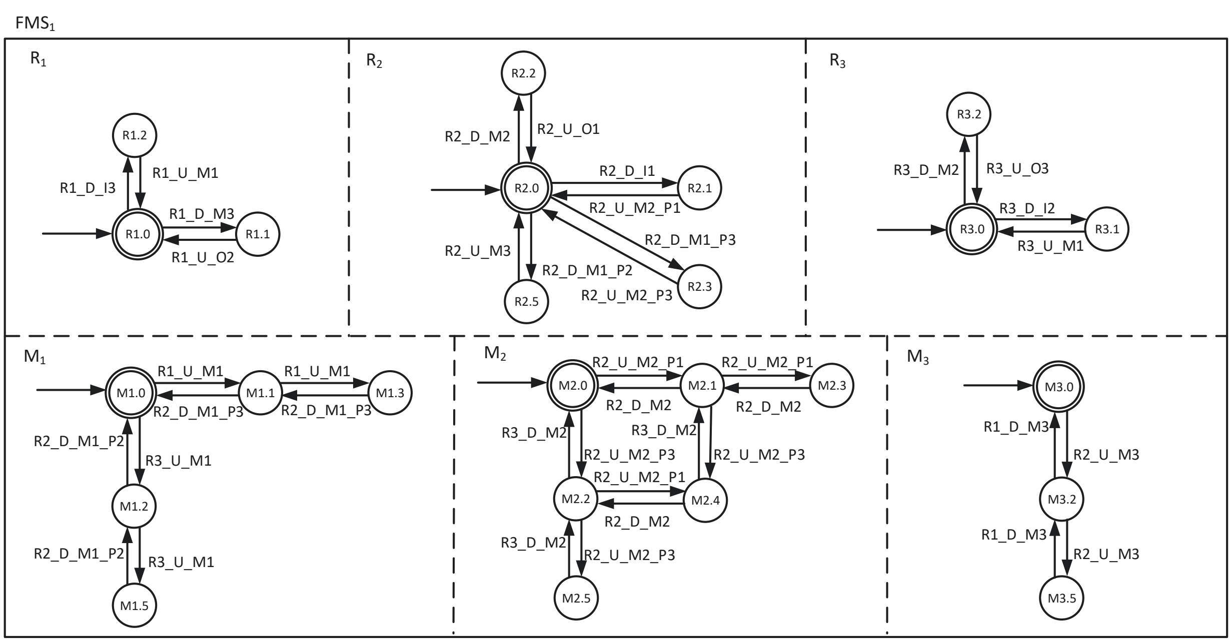



Figure 4.7: The STS model of $\mathrm{FMS}_1$.

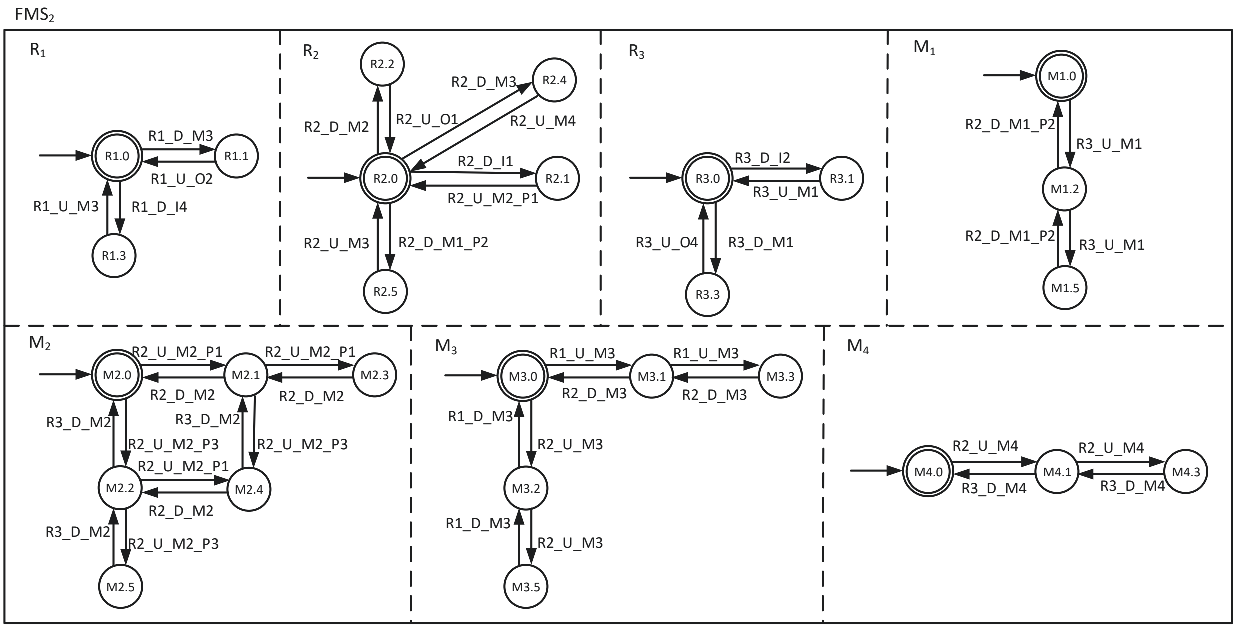



Figure 4.8: The STS model of $FMS_2$.

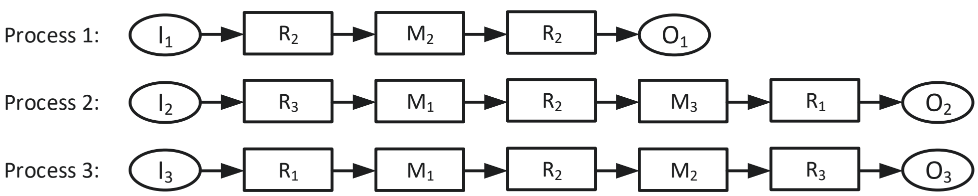



Figure 4.9: The production processes of $FMS_1$.

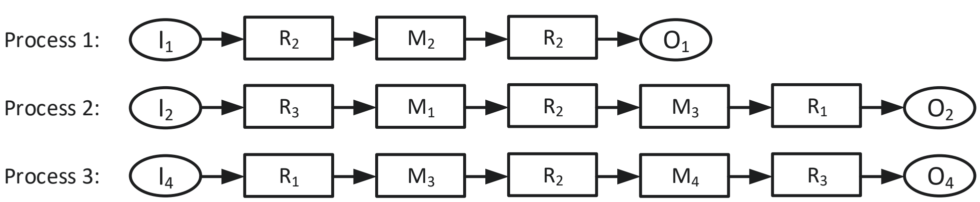



Figure 4.10: The production processes of $FMS_2$.

Table 4.1: The list of events of Figure 4.7 and 4.8.

| Label | Event |
|---|---|
| $R1\_D\_I3$ | $R1$ downloads from $I3$ |
| $R1\_D\_I4$ | $R1$ downloads from $I4$ |
| $R1\_U\_Mi$ | $R1$ uploads to $Mi$ ($i = 1$ and 3) |
| $R1\_D\_M3$ | $R1$ downloads from $M3$ |
| $R1\_U\_O2$ | $R1$ uploads to $O2$ |
| $R2\_D\_M1\_Pi$ | $R2$ downloads from $M1$ ($i = 2$ and 3) |
| $R2\_D\_Mi$ | $R2$ downloads from $Mi$ ($i = 2$ and 3) |
| $R2\_D\_I1$ | $R2$ downloads from $I1$ |
| $R2\_U\_O1$ | $R2$ uploads to $O1$ |
| $R2\_U\_M2\_Pi$ | $R2$ uploads $Pi$-type workpiece to $M2$ for $i = 1$ and 3 |
| $R2\_U\_Mi$ | $R2$ uploads to $Mi$ ($i = 3$, and 4) |
| $R3\_D\_I2$ | $R3$ downloads from $I2$ |
| $R3\_U\_M1$ | $R3$ uploads to $M1$ |
| $R3\_U\_O3$ | $R3$ uploads to $O3$ |
| $R3\_U\_O4$ | $R3$ uploads to $O4$ |
| $R3\_D\_Mi$ | $R3$ downloads from $Mi$ ($i = 2$ and 4) |

The list and description of all the events in both models are shown in Table 4.1. All the events are assumed to be controllable. Besides avoiding deadlocks, $FMS_1/FMS_2$ has to satisfy the following five specifications.

SP.1 Each input/output pair has a fixed buffer size. The buffer sizes of $I_1/O_1$, $I_2/O_2$, $I_3/O_3$, and $I_4/O_4$ are 3, 7, 11, and 11 respectively.

SP.2 The buffers should neither overflow nor underflow.

SP.3 Each machine can only handle a maximum of two workpieces simultaneously.

SP.4 If the machines $M_1$ and $M_3$ are processing a specific product type, they cannot be uploaded by another product type. In other words, $M_1$ and $M_3$ cannot be active in more than one production process.

SP.5 $FMS_1$ and $FMS_2$ should follow the production processes shown in Figure 4.9 and 4.10 respectively.

The STS models of $FMS_1$ and $FMS_2$ satisfy SP.5. To satisfy SP.1, we alter the STS models and add four buffers $B_1$, $B_2$, $B_3$ and $B_4$. The updated models of $FMS_1$ and $FMS_2$ are shown in Figure 4.11 and 4.12. The union model FMS is illustrated in Figure 4.13. The union flat automaton has a state set with a size of order[1] $10^8$.

[1] The order of the size of the plant state set is calculated by multiplying the sizes of state sets of all the components.

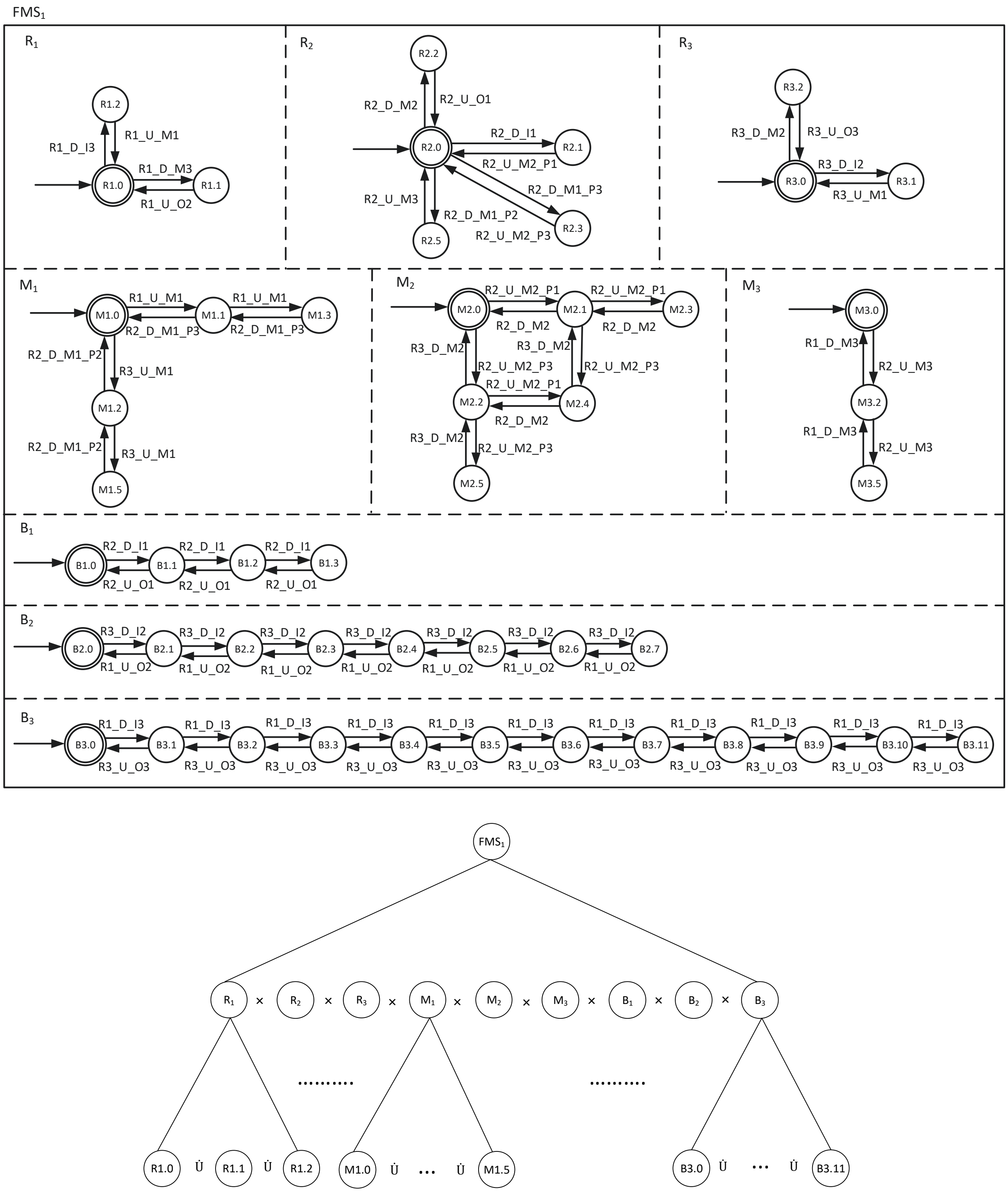



Figure 4.11: The STS model of $FMS_1$ with buffers.

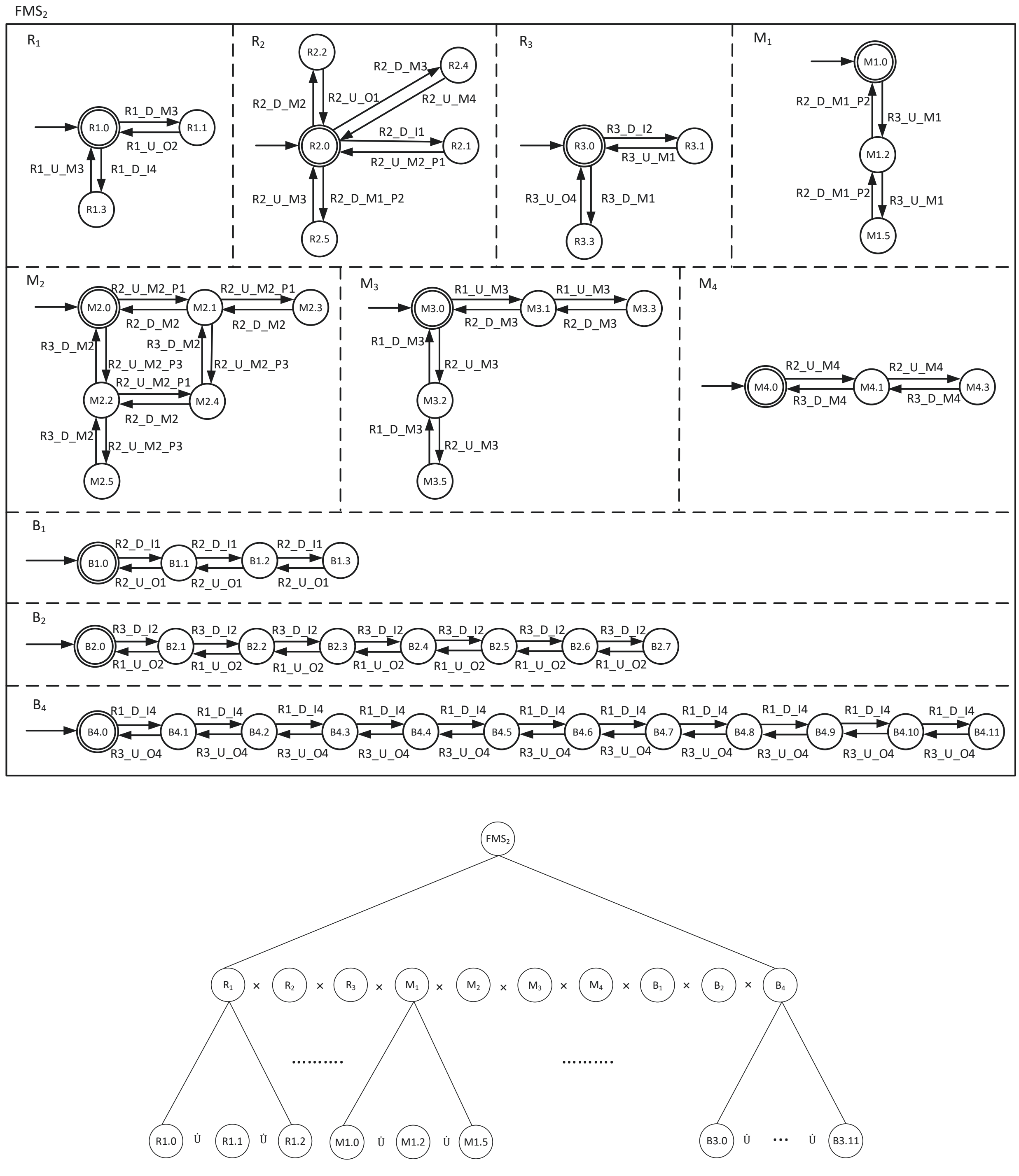



Figure 4.12: The STS model of $FMS_2$ with buffers.

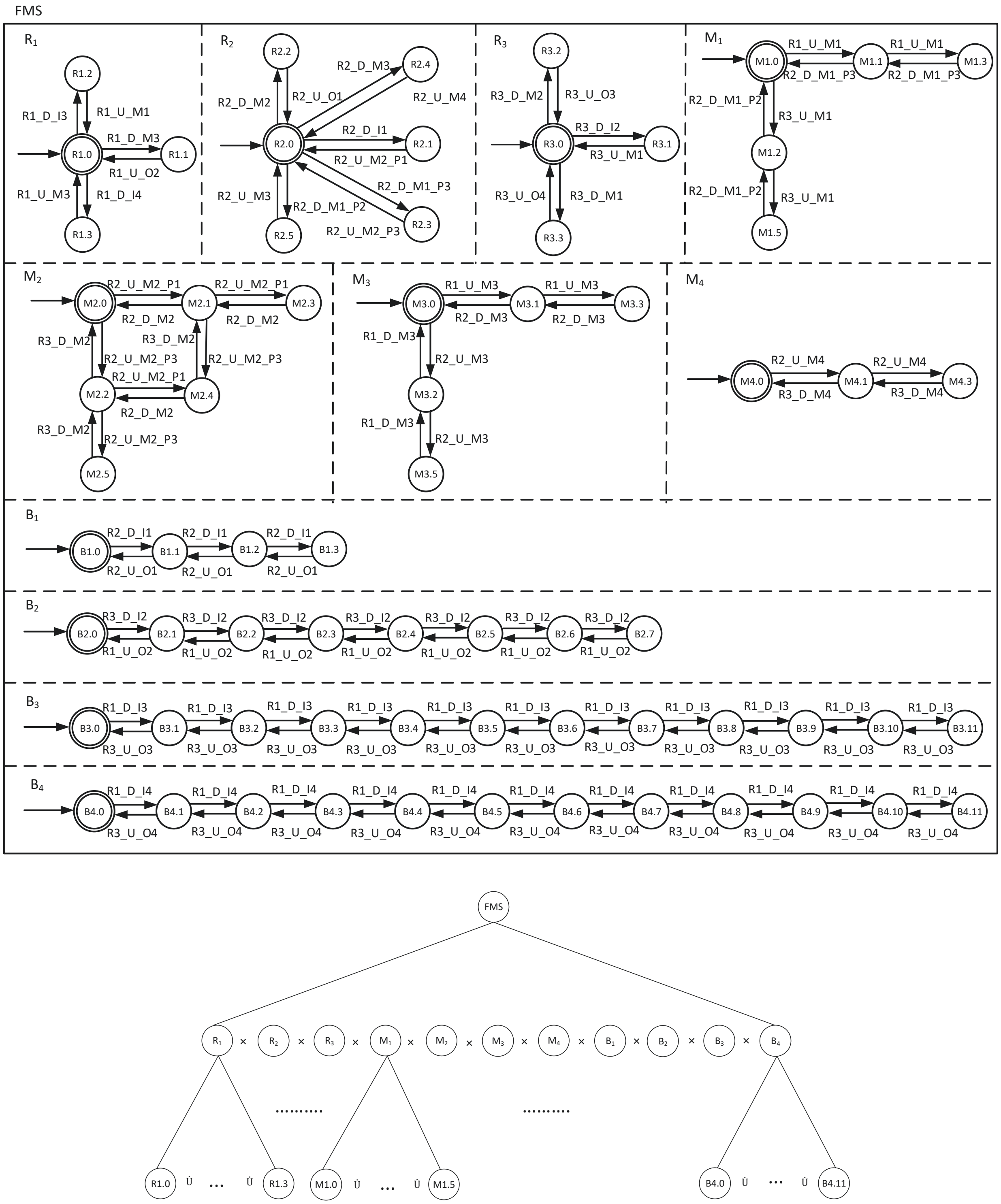



Figure 4.13: The STS model of FMS with buffers.

Table 4.2: The list of events that should be disabled at some states of $FMS_1$ (Figure 4.11) to satisfy SP.2.

| State | Event | Reason |
|---|---|---|
| $B1.0$ | $R2\_U\_O1$ | $B1$ Underflows |
| $B1.3$ | $R2\_D\_I1$ | $B1$ Overflows |
| $B2.0$ | $R1\_U\_O2$ | $B2$ Underflows |
| $B2.7$ | $R3\_D\_I2$ | $B2$ Overflows |
| $B3.0$ | $R3\_U\_O3$ | $B3$ Underflows |
| $B3.11$ | $R1\_D\_I3$ | $B3$ Overflows |

Table 4.3: The list of events that should be disabled at some states of $FMS_2$ (Figure 4.12) to satisfy SP.2.

| State | Event | Reason |
|---|---|---|
| $B1.0$ | $R2\_U\_O1$ | $B1$ Underflows |
| $B1.3$ | $R2\_D\_I1$ | $B1$ Overflows |
| $B2.0$ | $R1\_U\_O2$ | $B2$ Underflows |
| $B2.7$ | $R3\_D\_I2$ | $B2$ Overflows |
| $B3.0$ | $R3\_U\_O3$ | $B3$ Underflows |
| $B3.11$ | $R1\_D\_I3$ | $B3$ Overflows |
| $B4.0$ | $R3\_U\_O4$ | $B4$ Underflows |
| $B4.11$ | $R1\_D\_I4$ | $B4$ Overflows |

To satisfy SP.2, Tables 4.2 (for $FMS_1$) and 4.3 (for $FMS_2$), and to satisfy SP.3, and SP.4, Tables 4.4 (for $FMS_1$) and 4.5 (for $FMS_2$) list all the states and the related events that should be disabled at those states.

The set of unsafe sub-STs for $FMS_1$ and $FMS_2$ are denoted by $S_1$ and $S_2$. The $S_1$ and $S_2$ identify the predicates $\neg P_1$ and $\neg P_2$. The predicates corresponding to the safe sub-ST in $FMS_1$ and $FMS_2$ are $P_1$ and $P_2$.

Using (4.6), we calculate predicate $P$ in Theorem 4.3 as follows.

$$P = \Big[\big(P_1 \vee \big[R(\mathrm{FMS},\mathrm{true}) \wedge \neg R(\mathrm{FMS}_1,\mathrm{true})\big]\big) \wedge \big(P_2 \vee \big[R(\mathrm{FMS},\mathrm{true}) \wedge \neg R(\mathrm{FMS}_2,\mathrm{true})\big]\big)\Big] \wedge R(\mathrm{FMS},\mathrm{true}) \quad (4.15)$$

Algorithm 4.4 converges to the maximally permissive solution $K^{\uparrow}$ after 5 iterations. Using a a personal computer with 8 GB RAM and Intel(R) Core(TM) i5-3470, 3.20GHz CPU, it takes 0.473609 seconds to synthesize $K^{\uparrow}$. We used BuDDY and STSLib libraries in C++ to run the simulation. The BDD size of $K^{\uparrow}$ is 5942. The BDD size of all control functions are 2105. The maximum and minimum BDD sizes of control functions are 375 and 2. Recall that the number of states of the plant is of order $10^8$.

The BDD sizes of all control functions are listed in Table 4.6. In Figures 4.14 and 4.15, three examples of generated control functions are illustrated. The first buffer $B_1$ has 4 states; therefore, the BDD size of $B_1$ is

Table 4.4: The list of events that should be disabled at some states in $FMS_1$ to satisfy SP.3.

| State | Event | Reason |
|---|---|---|
| $M1.0, M1.1, M1.3$ | $R2\_D\_M1\_P2$ | $M1$ Underflows |
| $M1.0, M1.2, M1.5$ | $R2\_D\_M1\_P3$ | $M1$ Underflows |
| $M1.2, M1.3, M1.5$ | $R1\_U\_M1$ | $M1$ Overflows |
| $M1.1, M1.3, M1.5$ | $R3\_U\_M1$ | $M1$ Overflows |
| $M2.0, M2.2, M2.5$ | $R2\_D\_M2$ | $M2$ Underflows |
| $M2.0, M2.1, M2.3$ | $R3\_D\_M2$ | $M2$ Underflows |
| $M2.3, M2.4, M2.5$ | $R2\_U\_M2\_P1, R2\_U\_M2\_P3$ | $M2$ Overflows |
| $M3.0$ | $R1\_D\_M3$ | $M3$ Underflows |
| $M3.5$ | $R2\_U\_M3$ | $M3$ Overflows |

Table 4.5: The list of events that should be disabled at some states in $FMS_2$ to satisfy SP.3.

| State | Event | Reason |
|---|---|---|
| $M1.0$ | $R2\_D\_M1\_P2$ | $M1$ Underflows |
| $M1.2$ | $R3\_U\_M1$ | $M1$ Overflows |
| $M2.0, M2.2, M2.5$ | $R2\_D\_M2$ | $M2$ Underflows |
| $M2.0, M2.1, M2.3$ | $R3\_D\_M2$ | $M2$ Underflows |
| $M2.3, M2.4, M2.5$ | $R2\_U\_M2\_P1, R2\_U\_M2\_P3$ | $M2$ Overflows |
| $M3.0, M3.1, M3.3$ | $R1\_D\_M3$ | $M3$ Underflows |
| $M3.0, M3.2, M3.5$ | $R2\_D\_M3$ | $M3$ Underflows |
| $M3.3, M3.4, M3.5$ | $R1\_U\_M3, R2\_U\_M3$ | $M3$ Overflows |
| $M4.0$ | $R3\_D\_M4$ | $M4$ Underflows |
| $M4.2$ | $R2\_U\_M4$ | $M4$ Underflows |

Table 4.6: The BDD size of all control functions in FMS.

| Controllable Event | BDD size of control function |
|---|---|
| $R1\_D\_I3$ | 203 |
| $R1\_D\_I4$ | 375 |
| $R1\_U\_M1$ | 175 |
| $R1\_U\_M3$ | 185 |
| $R1\_D\_M3$ | 3 |
| $R1\_U\_O2$ | 3 |
| $R2\_D\_M1\_P2$ | 334 |
| $R2\_D\_M1\_P3$ | 91 |
| $R2\_D\_M2$ | 4 |
| $R2\_D\_M3$ | 62 |
| $R2\_D\_I1$ | 189 |
| $R2\_U\_O1$ | 2 |
| $R2\_U\_M2\_P1$ | 5 |
| $R2\_U\_M2\_P3$ | 5 |
| $R2\_U\_M3$ | 118 |
| $R2\_U\_M4$ | 2 |
| $R3\_D\_I2$ | 270 |
| $R3\_U\_M1$ | 66 |
| $R3\_U\_O3$ | 4 |
| $R3\_U\_O4$ | 4 |
| $R3\_D\_M2$ | 3 |
| $R3\_D\_M4$ | 2 |

of size 2 (has 2 bits). The two bits of $B_1$ are named $B1\_0$ and $B1\_1$ where $B1\_0$ and $B1\_1$ identify the first and the second bits respectively. Figure 4.14a indicates that in $B_1$, $R2\_U\_O1$ is only disabled when $B1\_0 = 0$ and $B1\_1 = 0$. Therefore, to avoid underflowing $B_1$, $R2\_U\_O1$ is disabled at state 0 in $B1$ ($B1\_0 = 0$ and $B1\_1 = 0$). The machine $M2$ has 6 states; therefore, it is modeled by a BDD of size 3. The labels of 3 bits are $M2\_0$, $M2\_1$, and $M2\_2$ respectively. In $M_2$ (Figure 4.14b), $R2\_U\_M2\_P1$ is only enabled at states 0, 1, and 2. The same logic can be applied for the analysis of Figure 4.15.

## 4.6 Summary

In this chapter, we formulated a novel robust supervisory control problem for systems modeled by State-Tree-Structure (STS). We found a sufficient condition in STSs that guarantees the mutual refinement prop-

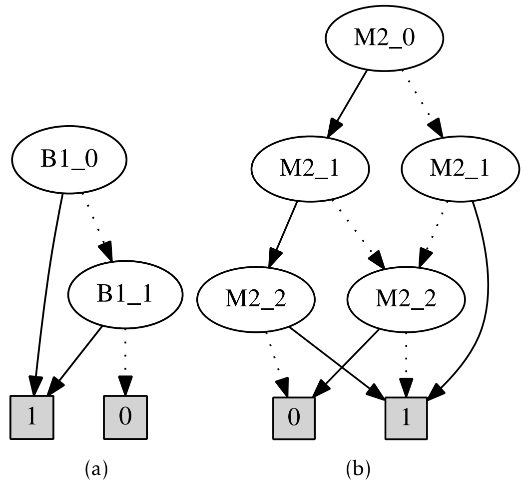



Figure 4.14: The control function of controllable events (a) $R2\_U\_O1$ and (b) $R2\_U\_M2\_P1$.

erty. This sufficient condition can be verified by examining the holons of STSs. A set of necessary and sufficient conditions for the existence of a solution are also presented. An algorithm was designed to calculate the maximally permissive solution. We proved that this algorithm converges to the solution within a finite number of iterations. We applied our robust supervisory control problem to a Flexible Manufacturing System (FMS) with a state set with a size of order $10^8$. Using a a personal computer, it took less that 0.5 seconds to converge to the solution.

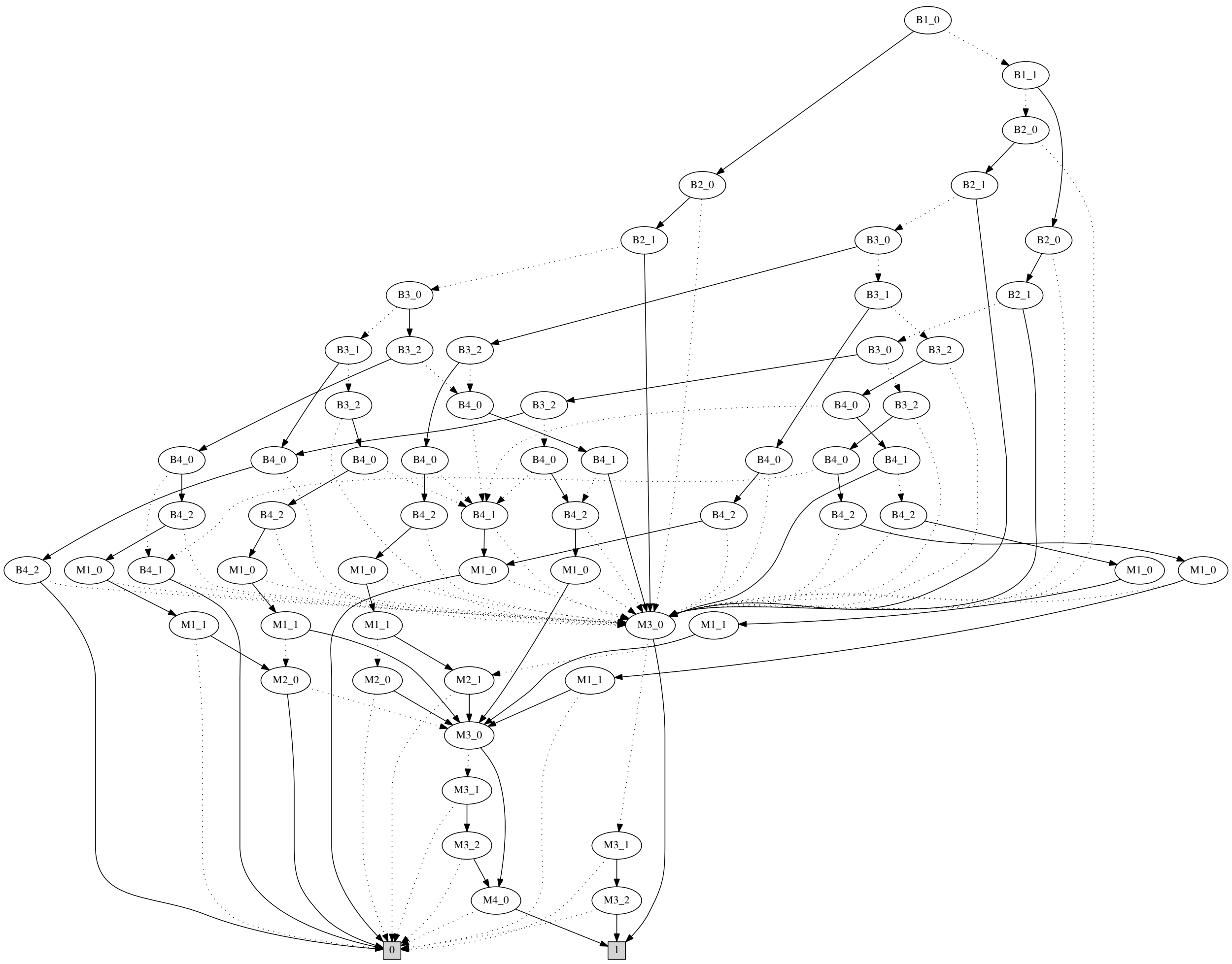



Figure 4.15: The control function of controllable event $R2\_D\_M3$.

# Chapter 5

# Conclusion

## 5.1 Summary

In this thesis, we studied two issues of robustness and symbolic calculations in the problem of supervisory control. We assumed that our models satisfy the mutually refined condition to ensure that we can characterize the robust supervisory control solutions via state feedback control laws. The mutually refined property of automata does not cause any restrictions on the problem formulation. We formulated a novel robust state-based nonblocking supervisory control problem and developed an algorithm to calculate the maximally permissive solution within a finite number of iterations.

Describing state sets by predicates enables us to store their information more efficiently. For a structured model, sets' predicate representation can be significantly simpler than a roster representation. Moreover, structured models are visually more comprehensible and synthesizing a supervisor for them is less time-consuming. State-Tree-Structure (STS) with both modular and hierarchical structures are suitable for handling state explosion. We considered systems that are modeled by STS. A large range of manufacturing, process control, and aerospace systems lend themselves to the hierarchical models of STS.

We extended the mutually refined property definition from automata to STSs. Moreover, we found a sufficient condition to verify the mutually refined property of STS models without constructing their flat models. We formulated a robust supervisory control problem for systems modeled with STS and developed an algorithm to calculate the maximally permissive solution in a finite number of iterations. We illustrated our results on a flexible manufacturing system model with a state set of order $10^8$. For this case-study, we synthesized the supervisor using our algorithm and a BDD-based program in C++.

As technology evolves, the complexity of systems also increases. The systems can have multiple operational procedures. Moreover, the systems may need to be functional in the presence of minor faults to avoid disruption and economic loss. While our work is not the ultimate solution for such systems, we provide a stepping-stone towards it. In our work, we have assumed that all the events are observable. An assumption that is not necessarily true in all systems. Furthermore, we have assumed that the corresponding holons are mutually refined in the STSs. An assumption that may not be true as well.

## 5.2 Future Work

We found a condition to verify the mutually refined property in STS. However, we did not present any solutions for STS models that are not mutually refined. Similar to automata, a procedure can be developed to convert the non-mutually refined STSs to mutually refined ones. Since holons being mutually refined is a sufficient condition for STSs being mutually refined, one can simply develop a procedure for conversion of holons from non-mutually refined to mutually refined.

In this thesis, we assumed that all the events are observable. The results of [69] and [23] can be used to extend our work for automata and STS models to the case of partial observation.

In this thesis, we developed a robust nonblocking state-based supervisory control to handle model uncertainty. Besides robust supervisory control, adaptive supervisory control is also used to handle model uncertainty in DES. The concept of predicates can be extended to adaptive supervisory control as well. So far, no state-based adaptive supervisory control has been proposed for systems modeled by STS.

# References


[1] S. B. Akers, “Binary decision diagrams,” *IEEE Trans. Comput.*, vol. c-27, no. 6, pp. 509–516, June 1978.

[2] S. Balemi, G. J. Hoffmann, P. Gyugyi, H. Wong-Toi, and G. F. Franklin, “Supervisory control of a rapid thermal multiprocessor,” *IEEE Trans. Automat. Control*, vol. 38, no. 7, pp. 1040–1059, July 1993.

[3] F. Boroomand and S. Hashtrudi-Zad, “A limited lookahead policy in robust nonblocking supervisory control of discrete event systems,” *Proc. American Control Conference (ACC), Washington, DC, USA*, pp. 935–939, June 2013.

[4] S. E. Bourdon, M. Lawford, and W. M.Wonham, “Robust nonblocking supervisory control of discrete-event systems,” *IEEE Trans. Automat. Control*, vol. 50, no. 12, pp. 2015–2021, December 2005.

[5] Y. Brave and M. Heymann, “Control of discrete-event systems modeled as hierarchical state machine,” *IEEE Trans. Automat. Control*, vol. 38, no. 12, pp. 1803–1819, December 1993.

[6] R. E. Bryant, “Graph-based algorithms for boolean function manipulation,” *IEEE Trans. Comput.*, vol. 35, no. 8, pp. 677–691, August 1986.

[7] L. K. Carvalho, J. C. Basilio, and M. V. Moreira, “Robust diagnosis of discrete event systems against intermittent loss of observations,” *Automatica*, vol. 48, no. 9, pp. 2068–2078, 2012.

[8] W. Chao, Y. Gan, Z. Wang, and W. Wonham, “Representation of supervisory controls using state tree structures, binary decision diagrams, automata, and supervisor reduction,” *Proc. of 24th Chinese Control and Decision Conference (CCDC), Taiyuan, China*, pp. 45–50, May 2012.

[9] ——, “Modular supervisory control and coordination of state tree structures,” *International Journal of Control*, vol. 86, no. 1, pp. 9–21, 2013.

[10] W. Chao, Y. Gan, W. M. Wonham, and Z. Wang, *Nonblocking supervisory control of flexible manufacturing systems based on state tree structures*, ser. A volume in the Advances in Civil and Industrial Engineering (ACIE) Book Series. Engineering Science Reference, 2013, ch. 1, pp. 1–19.

[11] X. Y. Chen, *Online robust nonblocking supervisory control of discrete-event systems*. M.A.Sc. Thesis, Dept. Electrical and Computer Eng., Concordia Univ., 2007.

[12] X. Y. Chen and S. Hashtrudi-Zad, "A direct approach to robust supervisory control of discrete-event systems," *2008 Canadian Conference on Electrical and Computer Engineering, Niagara Falls, Ont., Canada*, pp. 957–962, May 2008.

[13] S.-L. Chung and S. Lafortune, "Limited lookahead policies in supervisory control of discrete event systems," *IEEE Trans. Automat. Control*, vol. 37, no. 12, pp. 1921–1935, December 1992.

[14] J. E. R. Curry and B. H. Krogh, "Robustness of supervisors for discrete-event systems," *IEEE Trans. Automat. Control*, vol. 44, no. 2, pp. 376–379, February 1999.

[15] M. H. De Queiroz, J. E. R. Cury, and W. M. Wonham, "Multitasking supervisory control of discrete-event systems," *Discrete Event Dynamic Systems*, vol. 15, no. 4, pp. 375–395, 2005.

[16] K. Dong, Q. Quan, and W. M. Wonham, "Failsafe mechanism ddesign for autonomous aerial refueling using state tree structures," *Unmanned Systems*, vol. 7, no. 4, pp. 261–279, June 2019.

[17] P. A. eite, F. L. Baldissera, and J. E. Cury, "State-based supervisory control with restrictions on the supervisor realization," *Discrete Event Dyn Syst*, vol. 30, no. 4, pp. 671–693, 2020.

[18] Z. Fei, S. Miremadi, K. Akesson, and B. Lennartson, "Efficient symbolic supervisor synthesis for extended finite automata," *IEEE Trans. Control Syst. Technol.*, vol. 22, no. 6, pp. 2368–2375, November 2014.

[19] Z. Fei, S. Reveliotis, S. Miremadi, and K. Akesson, "A BDD-based approach for designing maximally permissive deadlock avoidance policies for complex resource allocation systems," *IEEE Trans. Autom. Sci. Eng.*, vol. 12, no. 3, pp. 990–1006, July 2015.

[20] B. Gaudin and H. Marchand, "Supervisory control of product and hierarchical discrete event systems," *European Journal of Control*, vol. 10, no. 2, pp. 131–145, December 2004.

[21] J. Geurts, *Supervisory control of MRI subsystems*. M.A.Sc. Thesis, Dept. Mechanical Eng., System Eng. Group, Eindhoven Univ. Technol., 2012.

[22] C. Gu, X. Wang, and Z. Li, "Synthesis of supervisory control with partial observation on normal state-tree structures," *IEEE Trans. Autom. Sci. Eng.*, vol. 16, no. 2, pp. 984–997, April 2019.

[23] C. Gu, X. Wang, Z. Li, and N. Wu, "Supervisory control of state-tree structures with partial observation," *Information Sciences*, vol. 465, pp. 523–544, 2018.

[24] J. Gunnarsson, *Symbolic methods and tools for discrete event dynamic systems*. PhD. Thesis, Division of Automatic Control Department of Electrical Engineering, Linkoping Studies in Science and Technology, Sweden, 1997.

[25] N. B. Hadj-Alouane, S. Lafortune, and F. Lin, "Variable lookahead supervisory control with state information," *IEEE Trans. Automat. Control*, vol. 39, no. 12, pp. 2398–2410, December 1994.

[26] D. Harel, "Statecharts: a visual formalism for complex systems," *Science of Computer Programing*, vol. 8, pp. 231–274, June 1987.

[27] L. E. Holloway, B. H. Krogh, and A. Giua, "A survey of petri net methods for controlled discrete event systems," *Discrete Event Dynamic Systems*, vol. 7, no. 2, pp. 151–190, Apr 1997.

[28] R. Kamphuis, *Design and real-time implementation of a supervisory controller for baggage handling at Veghel Airport*. M.A.Sc. Thesis, Dept. Mechanical Eng., System Eng. Group, Eindhoven Univ. Technol., 2013.

[29] J. Komenda, J. van Schuppen, B. Gaudin, and H. Marchand, "Supervisory control of modular systems with global specification languages," *Automatica*, vol. 44, pp. 1127–1134, 2008.

[30] T. Korssen, V. Dolk, J. Van De Mortel-Fronczak, M. Reniers, and M. Heemels, "Systematic model-based design and implementation of supervisors for advanced driver assistance systems," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 2, pp. 533–544, February 2018.

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

[32] R. J. Leduc, P. Dai, and R. Song, "Synthesis method for hierarchical interface-based supervisory control," *IEEE Trans. Automat. Control*, vol. 54, no. 7, pp. 1548–1560, July 2009.

[33] R. J. Leduc, M. Lawford, and W. M. Wonham, "Hierarchical interface-based supervisory control—part ii: Parallel case," *IEEE Trans. Autom. Control*, vol. 50, no. 9, September 2005.

[34] C. Y. Lee, "Representation of switching circuits by binary decision programs," *Bell System Technical Journal*, vol. 38, no. 4, pp. 985–999, July 1959.

[35] Y. Li and W. M. Wonham, "Controllability and observability in the state-feedback control of discrete-event systems," *Proc. of the 27th Conference on Decision and Control (CDC), Austin, Texas, USA*, pp. 203–208, December 1988.

[36] ——, "Control of vector discrete-event systems I-the base mode," *IEEE Trans. Automat. Control*, vol. 38, no. 8, pp. 1214–1227, August 1993.

[37] F. Lin, “Robust and adaptive supervisory control of discrete event systems,” *IEEE Trans. Automat. Control*, vol. 38, no. 12, pp. 1848–1852, December 1993.

[38] H. Liu, R. J. Leduc, and S. L. Ricker, “Hierarchical interface-based decentralized supervisory control,” *Proc. of 54th Conference on Decision and Control (CDC), Osaka, Japan*, pp. 1693–1700, December 2015.

[39] C. Ma and W. Wonham, “Nonblocking supervisory control of state-tree structures,” *IEEE Trans. Automat. Control*, vol. 51, no. 5, pp. 782–793, May 2006.

[40] C. Ma and W. M. Wonham, *Nonblocking supervisory control of state tree structures*, ser. Lecture Notes in Control and Information Science. Springer Berlin Heidelberg, 2005, vol. 317.

[41] ——, “STSLib and its application to two benchmarks,” *Proc. of the 9th International Workshop on Discrete Event Systems (WODES’08), Goteborg, Sweden*, pp. 119–124, May 2008.

[42] S. Miremadi, K. A. Z. Fei, and B. Lennartson, “Symbolic computation of reduced guards in supervisory control,” *IEEE Trans. Autom. Sci. Eng.*, vol. 8, no. 4, pp. 754–765, October 2011.

[43] ——, “Symbolic representation and computation of timed discrete-event systems,” *IEEE Trans. Autom. Sci. Eng.*, vol. 11, no. 1, pp. 6–19, January 2014.

[44] S. Miremadi and B. Lennartson, “Symbolic on-the-fly synthesis in supervisory control theory,” *IEEE Trans. Control Syst. Technol.*, vol. 24, no. 5, pp. 1705–1716, September 2016.

[45] S. Miremadi, B. Lennartson, and K. Akesson, “A BDD-based approach for modeling plant and supervisor by extended finite automata,” *IEEE Trans. Control Syst. Technol.*, vol. 20, no. 6, pp. 1421–1435, November 2012.

[46] R. Mohammadi and S. Hashtrudi-Zad, “A recursive algorithm for diagnosis in hierarchical finite-state machines,” *2007 IEEE International Conference on Systems, Man and Cybernetics, Montreal, Que., Canada*, pp. 1345–1350, Oct 2007.

[47] A. Mohammadi-Idghamishi and S. Hashtrudi-Zad, “Hierarchical fault diagnosis: Application to an ozone plant,” *IEEE Trans. Syst., Man, Cybern., Syst.: Part C*, vol. 37, no. 5, pp. 1040–1047, September 2007.

[48] A. Paoli and S. Lafortune, “Diagnosability analysis of a class of hierarchical state machines,” *Discrete Event Dynamic Systems*, vol. 18, no. 3, pp. 385–413, Sep. 2008. [Online]. Available: http://dx.doi.org/10.1007/s10626-008-0044-5

[49] S.-J. Park and J.-T. Lim, “Robust and nonblocking supervisory control of nondeterministic discrete event systems using trajectory models,” *IEEE Trans. Automat. Control*, vol. 47, no. 4, pp. 655–658, April 2002.

[50] S.-J. Park and K.-H. Cho, "Decentralized supervisory control of nondeterministic discrete event systems: The existence condition of a robust and nonblocking supervisor," *Automatica*, vol. 43, pp. 377–383, 2007.

[51] S. Rahnamoon and W. M. Wonham, "State-based control of timed discrete-event systems," in *Proc. of 54th Conference on Decision and Control (CDC), Miami Beach, FL, USA*. IEEE, December 2018, pp. 4833–4838.

[52] P. J. Ramadge and W. M. Wonham, "Modular feedback logic for discrete event systems," *SIAM J. Control Optim.*, vol. 25, no. 5, pp. 1202–1218, September 1987.

[53] ——, "Supervisory control of a class of discrete event processes," *SIAM J. Control Optim*, vol. 25, no. 1, pp. 206–230, January 1987.

[54] F. F. Reijnen, M. A. Reniers, J. M. van de Mortel-Fronczak, and J. E. Rooda, "structured synthesis of fault-tolerant supervisory controllers," in *10th IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes SAFEPROCESS 2018: Warsaw, Poland*, vol. 51, no. 24. IFAC-PapersOnLine, August 2018, pp. 894–901.

[55] A. Saadatpoor, *Timed state-tree-structures: supervisory control and fault diagnosis*. PhD. Thesis, Dept. Electrical Eng., Univ. of Toronto, 2009.

[56] A. Saadatpoor and W. Wonham, "State based control of timed discrete event systems using binary decision diagrams," *Systems & Control Letters*, vol. 56, no. 1, pp. 62–74, January 2007.

[57] A. Saboori and S. Hashtrudi-Zad, "Robust nonblocking supervisory control of discrete-event systems under partial observation," *Systems & Control Letters*, vol. 55, pp. 839–848, 2006.

[58] A. Schumann, Y. Pencole, and S. Thiebaux, "Diagnosis of discrete-event systems using BDDs." *15th International Workshop on Principles of Diagnosis (DX-04)*, pp. 197–202, 2004.

[59] J. Sztipanovits and A. Misra, "Diagnosis of discrete-event systems using ordered binary decision diagrams," *7th Intl. Workshop on Principles of Diagnosis (DX-96), Val Morin, Canada*, 1996.

[60] S. Takai, "Robust supervisory control of a class of timed discrete event systems under partial observation," *Systems & Control Letters*, vol. 39, pp. 267–273, April 2000.

[61] ——, "Synthesis of maximally permissive and robust supervisors for prefix-closed language specifications," *IEEE Trans. Automat. Control*, vol. 47, no. 1, pp. 132–136, January 2002.

[62] ——, "Maximizing robustness of supervisors for partially observed discrete event systems," *Automatica*, vol. 40, pp. 531–535, March 2004.

[63] ——, "Robust failure diagnosis of partially observed discrete event systems," *10th IFAC Workshop on Discrete Event Systems (WODES'10), Berlin, Germany*, vol. 43, pp. 205–510, August 2010.

[64] ——, "Robust prognosability for a set of partially observed discrete event systems," *Automatica*, vol. 51, pp. 123–130, 2015.

[65] J. H. A. Tomola, F. G. Cabral, L. K. Carvalho, and M. V. Moreira, "Robust disjunctive-codiagnosability of discrete-event systems against permanent loss of observations," *IEEE Trans. Automat. Control*, vol. 62, no. 11, pp. 5808–5815, December 2017.

[66] A. Vahidi, M. Fabian, and B. Lennartson, "Efficient supervisory synthesis of large systems," *Systems & Control Letters*, vol. 14, pp. 1157–1167, April 2006.

[67] J. M. van de Mortel-Fronczak, M. H. van der Heijden, R. G. Huisman, and M. A. Reniers, "Supervisor synthesis in model-based automotive systems engineering," *Proc. 2014 ACM/IEEE Conference on Cyber-Physical Systems, ICCPS, Berlin, Germany*, pp. 187–198, April 2014.

[68] B. Wang, *Top-down design for RW supervisory control theory*. M.A.Sc. Thesis, Dept. Electrical and Computer Eng., Univ. of Toronto, 1995.

[69] D. Wang, L. Lin, Z. Li, and W. M. Wonham, "State-based control of discrete-event systems under partial observation," *IEEE Access*, vol. 6, pp. 42 084–42 093, 2018.

[70] D. Wang, X. Wang, and Z. Li, "Nonblocking supervisory control of state-tree structures with conditional-preemption matrices," *IEEE Trans. Ind. Informat.*, vol. 16, no. 6, pp. 3744–3756, June 2020.

[71] X. Wang, Z. Li, and W. M. Wonham, "Real-time scheduling based on nonblocking supervisory control of state-tree structures," *IEEE Trans. Automat. Control*, doi:10.1109/TAC.2020.3031023.

[72] W. M. Wonham and P. J. Ramadge, "On the supremal controllable sublanguage of a given language," *SIAM J. Control Optim.*, vol. 25, no. 3, p. 637–659, May 1987.

[73] ——, "Modular supervisory control of discrete-event systems," *Math. Control Signals Systems*, vol. 1, no. 1, pp. 13–30, 1988.

[74] W. M. Wonham and K. Cai, *Supervisory control of discrete-event systems*, ser. Monograph Series Communications and Control Engineering. Springer Berlin Heidelberg, 2018.

[75] C. Xiao and F. Liu, "Robust fault prognosis of discrete-event systems against loss of observations," *IEEE Transactions on Automation Science and Engineering*, pp. 1–12, 2021.

[76] F. Yari, S. Hashtrudi-Zad, and S. Tafazoli, "Robust supervisory control of a spacecraft propulsion system," *20th IFAC Symposium on Automatic Control in AerospaceACA 2016, Sherbrooke, Que., Canada*, vol. 49, no. 17, pp. 200–205, August 2016.

[77] X. Yin, J. Chen, Z. Li, and S. Li, "Robust fault diagnosis of stochastic discrete event systems," *IEEE Trans. Automat. Control*, vol. 64, no. 10, pp. 4237–4244, October 2019.