Hierarchical Robust Supervisory Control of Discrete-Event Systems

Mohsen Zamani Fekri

A Thesis
in
The Department
of
Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy at
Concordia University
Montreal, Quebec, Canada

Your file Votre référence
ISBN: 978-0-494-67355-3
Our file Notre référence
ISBN: 978-0-494-67355-3

Abstract

Hierarchical Robust Supervisory Control of Discrete-Event Systems

Mohsen Zamani Fekri, PhD

Concordia University, 2010

The problem of Robust Supervisory Control (RSC) of Discrete-Event Systems (DES) is concerned with situations in which the DES plant model has dynamics uncertainty. A main challenge in the development of solutions for supervisory control problems (including RSC) is the issue of complexity of resulting solutions. Hierarchical approaches to supervision have been found to be effective in mitigating the above issue. In hierarchical control, a high-level supervisor designed based on a simplified high-level model of the plant, receives information about important events in the plant and issues high-level supervisory commands.

In this thesis, the problem of hierarchical robust supervisory control under partial observation is studied. First, the setup of Zhong-Wonham for hierarchical control is extended to the case of control under partial observation. A Factorization property is derived that the reporting map must satisfy so that the reports sent to the high-level supervisor rely only on the low-level observable sequences. Furthermore, the three properties of Unobservable-and-Unique-Controllability (UUC), Unobservable-and-Uncontrollable-Prefixes-for-Observability (UUPO) and Partially-Observable-Strict-Output-Control-Consistency (PO-SOCC) are introduced and showed to ensure hierarchical consistency. Algorithms for modification of the plant model and reporting map (if necessary) to satisfy the Factorization, UUC, UUPO and PO-SOCC properties have also been developed.

Next, the problem of robust supervisory control of a finite family of discrete-event plants is studied. Each plant has a separate closed specification language. A hierarchical solution is developed assuming full observation and then extended to the case of partial observation, following the approach in the thesis for hierarchical control under partial observation.

Finally, a case study involving a flexible manufacturing system production line is studied where a machine is prone to failure. Following the approach developed in this thesis, a hierarchical robust supervisory control is designed to solve the control problem.

Acknowledgements

I would like to express my heartfelt gratitude to my supervisor, Dr. Shahin Hashtrudi Zad, whose vision and knowledge in systems and control theory guided me throughout this work. This thesis would not have been possible if he had not thoroughly reviewed every part of it and had not given me his helpful suggestions. I am extremely grateful for having his support and will never forget the knowledge, humbleness and patience he displayed throughout the course of this research.

It is an honor for me to thank all of my previous teachers and friends who have supported and encouraged me to continue my studies. Special thanks go to Siamak whose definite comments and suggestions served to rule out any further discussion on subjects. I would also like to express my sincere and respectful thanks to Amin and Elham for their true friendship and the support they have had for me in different ways during these years.

I am deeply indebted to my family who have always supported me. I owe my deepest gratitude to my wife and best friend, Fatemeh, for the love she has for me and the hope she has given me from the day she has entered my life.

Contents

List of Abbreviations

DES	Discrete-Event Systems
ES Algorithm	Event-Split Algorithm
FP Procedure	Factorization Property Procedure
FP Test	Factorization Property Test
FMS	Flexible Manufacturing Systems
HRSC	Hierarchical Robust Supervisory Control
HRSCPO	Hierarchical Robust Supervisory Control under Partial Observation
HSC	Hierarchical Supervisory Control
Joint-OCC	Joint-Output-Control-Consistency
Joint-OOC	Joint-Output-Observation-Consistency
Joint-PO-SOCC	Joint-Partially-Observable-Strict-Output-Control-Consistency
Joint-SOCC	Joint-Strict-Output-Control-Consistency
Joint-UUC	Joint-Unobservability-and-Unique-Controllability
Joint-UUPO	Joint-Unobservable-Uncontrollable-Prefixes-of-Observability
MSPSS	Multi State Projected String Split
NRSC	Nonblocking Robust Supervisory Control
NRSC	Nonblocking Robust Supervisory Control
OCC	Output-Control-Consistency
OOC	Output-Observation-Consistency
PO-SOCC	Partially-Observable-Strict-Output-Control-Consistency
RSC	Robust Supervisory Control
RT	Reachability Tree
RW	Ramadge-Wonham
SCE Algorithm	Single-Control-Event Algorithm
SOCC	Strict-Output-Control-Consistency
SSPSS	Single State Projected String Split
UET Algorithm	Uncontrollable-Exiting-Transitions Algorithm
UUC	Unobservability-and-Unique-Controllability
UUPO	Unobservable-Uncontrollable-Prefixes-of-Observability

List of Figures

List of Tables

Chapter 1

Introduction

Discrete-event systems (DES) represent an abstract level of modeling in which instead of the continuous-variable dynamics of the interactions among the components in the system, a simplified form of dynamics dealing with the order of occurrence of such interactions is of importance. In other words, the order of occurrence of actions (events), regardless of the continuous-variable dynamics with which they happen, and the state to which the system reaches as a result of each action, regardless of how long the system stays there, are the two main characteristics of DES models. Therefore, in a DES model the state of system evolves to another state following the occurrence of an event.

Different methodologies for DES modeling and its related supervision issues have been suggested in the literature. In particular, robust supervisory control (RSC) has been studied for dealing with cases in which the plant model or the specification change due to the conditions under which the plant is operated or due to the existence of uncertainty in the models.

The computational complexity of supervisor synthesis is polynomial in the number of system states which in turn increase exponentially with the number of the components of system. Gohari and Wonham [18] showed that the problem of supervisor synthesis is NP-hard and it is unlikely that any algorithm can be found to circumvent state space explosion (which is exponential in the number of system components). Therefore, structured frameworks in which supervisor synthesis is performed in a horizontal or vertical fashion rather than in a central fashion, have gained considerable attention in the literature. Modular (also decentralized) and hierarchical supervisory control methodologies exercise horizontal and vertical modularity in supervisor synthesis. Two important advantages to be gained by exploiting such structured supervisor synthesis are reduced computational complexity in supervisor synthesis and a more manageable and transparent design and interface. In this thesis, which is built on the Ramadge-Wonham (RW) supervisory control theory of DES, we solve an RSC problem in a hierarchical framework. A review of the related works follows.

1.1 Supervisory Control of Discrete-Event Systems

Discrete-event systems represent a wide range of systems in which the dynamics of the system is discrete in time and value and the system is event-deriven. Consequently dynamics of DES systems can be described by the sequences of events which are generated asynchronously. Examples of DES systems can be found in automation and management systems including manufacturing systems, chemical plants, power plants and transportation systems. The management of such systems requires the knowledge of the state of system and the next set of possible actions at each state. Petri nets [24, 34] and Automata [41, 57] are among the favorite DES models. Max-Plus Algebra [2, 14] is also another methodology for modeling a type of DES. In this thesis we have chosen automata to model DES. Two types of actions (events) are defined in DES: events which can be disabled and are called *controllable* (e.g. firing an engine), and events which cannot be disabled and are called *uncontrollable* (e.g. a sensor measurement output). It is usually desired to restrict the system behavior through disabling controllable events so that the dynamics of system remains within certain patterns of event evolution. In Automata Theory, the behavior of system is represented by the language of an automaton consisting of the sequences of events generated by the system. Therefore, controlling or restricting the behavior of the system means to requiring the system to talk a specific language. In such framework, specifications which are the requirements for or the demands from the system, are expressed in terms of languages over the event set of the system. To meet the specifications, controllable events in the system should be disabled (whenever necessary) so that the behavior of the system under supervision is restricted to either the specification language or a subset of it. The task described here is simply referred to as *supervisory control.*

The RW supervisory control theory [41] was the first control theory for DES, modeled by Automata Theory. The supervisory control problem defined in [57] is *optimal* in the sense that the supervisor is *minimally restrictive.* In other words, the controlled system is the largest possible sublanguage of the specification which can be achieved under supervision. The original supervisory control problem in [41, 57] is a centralized problem. Nevertheless, numerous extensions of [57] which deal with uncertainty, exponential state explosion and partial observation have been studied in the literature. We will review some of these extensions here.

1.2 Robust Supervisory Control

A system whose dynamics change due to the occurrence of fault or due to new conditions it is operated under which, requires a supervisor which is robust with respect to the model uncertainty or changes. Fault recovery [44] is a problem that can be dealt with using robust supervisory control. In a fault recovery problem, the system has a normal mode and several faulty modes, and in each mode a certain set of requirements are expected from the system (e.g. being deadlock free). Then, the main goal will be to have a single supervisor that regardless of the current mode of the system (i.e. normal or faulty) can ensure the operation of the system satisfies all the requirements.

Different approaches to robust control problems have been explored. Few research has been done on robust supervisory control in Petri net framework. [10] presents a robust supervisory control for production systems in Petri net framework to avoid deadlock and achieve liveness in the presence of multiple resource failures. In the framework of finite state machines represented with automata, robust supervisory control has extensively been studied. In one category of RSC problems, the robustness is defined in terms of the ability to reach a set of states [11, 35, 36]. There, the RSC problem focuses on resilience or error recovery properties of the (fault-tolerant) system. Specifically, the controlled system should return to a specified set of states E after being subjected to failure or error (*E-stability*). A second approach to robustness was initiated by Lin [30] in which a supervisory control problem is solved for a finite set of plants G_i $(i = 1, \cdots, n)$. [30] considers a case that the exact model of the plant is not known precisely; however, it is known that the plant model is among a finite set of possibilities. The problem in [30] specifies that whatever the plant is, the behavior of the system under supervision should not change, i.e. the specification is the same for all of the plants G_i. Furthermore, nonblocking is not considered. Takai [48] extends [30] to the case that the specification for a plant G_i $(i = 1, \cdots, n)$ is obtained as a subset of G_i's closed behavior and another unique language E. Bourdon et al. extends [30, 48] even further and presents an RSC problem [4] for a finite set of plants where each plant is required to satisfy its own specification. Within the aforementioned category, [4] solves a nonblocking RSC (NRSC) problem in its most general form, i.e. an NRSC problem for a finite set of pairs (plant[i], specification[i]) $(i = 1 \cdots n)$. [4] also gives an algorithm for finding the maximally permissible solution of the NRSC problem. However, all events are assumed to be observable in [4]. Saboori and Hashtrudi-Zad [45] extend [4] to include partial observation. The authors in [45] also showed the necessary and sufficient conditions that guarantee nonblocking in RSC problem.

Cury and Krogh [12, 13] initiated another type of RSC problem where for a given nominal model of the system dynamics, a controller is synthesized that maximizes the set of plants for which the closed-loop behavior is within specified bounds. This family of plants should necessarily include the nominal plant model. Other RSC problems in DES framework which more or less fall in the above categories include the works of [22, 47, 26, 39, 37]. The uncertainty in [47] has been associated with the internal and unobservable events denoted by Δ-*transitions*. This is comparable with the results in [4, 45] where uncertainty results from the different modes under which the system operates. In this thesis we have chosen the RSC problem of [4, 45] as the basis for our Hierarchical RSC problem. We will show how the RSC problem of [4, 45] can be solved in a hierarchical framework. Next, we review hierarchical supervisory control.

1.3 Hierarchical Supervisory Control

To mitigate exponential state explosion in DES, and hence decrease the computational complexity of supervisor synthesis, and to construct a consistent design structure that captures the component-based structure or modular aspect of the system, modular [43, 58, 55, 23] and hierarchical supervisory control [5, 7, 27, 28, 29, 52, 59, 49, 20] have been widely studied in the literature.
Supervisor synthesis in both of modular and hierarchical approaches is done in a structural fashion rather than in a simply centralized fashion. In brief, in structural supervisor synthesis, a system is broken into several modules (in modular control) or levels (in hierarchical control) which interact with each other. Then, supervisor synthesis is performed in modules or certain levels where later a coordinator or a translation map relates different supervisors in different modules or levels with each other to yield a overall consistency in the system. In this thesis, we have shown how an RSC problem can be solved in a hierarchical framework. Namely, we have used the hierarchical approach to solve a robust supervisor synthesis problem. A review of different hierarchical approaches follows.
[49] and [20] have studied hierarchical supervisory control in Petri nets. [49] proposes a hierarchical modeling in which the assembly planning level represents the high-level and the control planning level represents the low-level. The tasks at the high-level are defined in terms of assembly parts while they are defined in terms of production system resources at the low-level. [49] gives the condition that should be satisfied so that the hierarchical system achieves liveness, however it has not addressed the control and resource allocation problems. The modeling in [49] is, in part, comparable to the work of [59] in automata theory, although that of [59] has been given in a more general framework and addresses supervision. In [20], *ICs* (Integrated Components) which

consist of sensors and actuators form the low-level and *agents* which consist of a number of ICs, have their own controller and have communication units form the high-level (e.g. robots and automated guided vehicles). Component assembly modeling has been studied with Object Oriented (OO) techniques. Two perspectives have been studied: static plant modeling and dynamic plant modeling. Class diagram from Unified Modeling Language (UML) has been employed for static modeling. On the other hand, dynamic plant modeling describes the internal behavior of components which is represented by Petri nets. Functional logic issue has been addressed by using predicate logic and control synthesis has been discussed for both the low-level and high-level. The agents communicate with each other (if necessary) through the input and output places which are added to the model and the the conditions for achieving liveness at the high-level and low-level have been derived. The work of [20] is developed for manufacturing systems and uses a hierarchical modular design comparable to those based on automata theory [46, 15].

Hierarchical supervisory control (HSC) has been studied using finite-state automata from different viewpoints: top-down design, bottom-up design, state aggregation and recently interfaced-based design. The first category of hierarchical frameworks is the top-down design which is largely influenced by the statecharts proposed by Harel [21]. A statechart is a medium for state aggregation in which a finite state machine is characterized with two main capabilities: orthogonality (concurrency) and hierarchy. A key concept is superstate which is a state that consists of substates. The relation between a superstate and its substates defines a hierarchy in the system. Furthermore, the concurrency is modeled using superstate that are formed from parallel product of a number of components. In this top-down design, the system model is recursively defined from the top (highest level) to the bottom (lowest level). Brave and Heymann

[5] proposed a hierarchical supervisory control in this setup. Marchand and Gaudin [16, 33, 17] and Ma and Wonham [32, 31] later continued this work. Wang [50] introduced the state-tree structure (STS). A STS is a tree of states where each superstate is expanded in either AND or OR substates.
In the context of RW supervisory control theory, [59] proposes a bottom-up hierarchical approach in which the low-level behavior is summarized and reported to the high-level through a causal reporting map. Here, it is assumed all plant events are observable. A procedure is also proposed to refine (if necessary) the information sent to the high-level so that the plant satisfies a Strict-Output-Control-Consistency (SOCC) property. It is shown that SOCC guarantees *hierarchical consistency.* Loosely speaking, hierarchical consistency means that the behavior of the plant under supervision, as reported to the high-level, matches the expectation of the high-level supervisor.
Extensions of the results of [59] especially to include nonblocking property and time are discussed in [52, 53]. The authors in [52], in a more general framework, have derived sufficient conditions under which the bottom-up hierarchical approach of [59] is nonblocking. The important observer property, introduced in [52], together with the consistency between the marked states at the low-level and high-level ensure the nonblocking property at the high-level carries over to the low-level. One important aspect of the observer property which was later studied in [51, 54] is that if the reporting map is observer, the state-size of the high-level model would be smaller than that of the minimal canonical representation of the low-level model. This implies if the reporting map is observer, the computational complexity of supervisor synthesis decreases at the high-level. [19] has presented a reachability tree-based algorithm for modifying the reporting map so that it becomes observer. A more efficient algorithm which is based on automata and congruences is given in [54]. The interesting

observer property in cases where the reporting map is a natural projection, has further been investigated and used in [15, 46, 40] due to its potential to reduce computational complexity.

Including partial observation in the bottom-up hierarchical framework has been studied in [25]. There, it is shown that hierarchical consistency can be achieved if (i) the high-level specification is controllable and H-observable with respect to the high-level model and (ii) the supremal controllable sublanguage of the inverse image of high-level specification is observable with respect to low-level model. H-observability follows from observability with respect to the high-level models. No course of action has been provided if either of the two conditions above are not met. In other words, it is not known how the H-observability of the high-level specification with respect to the high-level model and the observability of the low-level supremal controllable language with respect to the low-level model can be ensured at the same time.

State aggregation is the third type of hierarchical framework which was introduced by Caines et al. [6, 8]. There, the hierarchy is achieved by aggregating the states of the model. Controllability is also considered in terms of the existence of the paths between the states. The last and most recent hierarchical approach is found in the works of Leduc, Lawford and Wonham [27, 28]. In their *master-slave* approach, the high-level system requests services from the low-level system through an interface, modeled as DES. Leduc has extended this work to the parallel case in which more than one low-level model exist. The authors in [28] have defined a hierarchical consistency, referred to as level-wise consistency, which can be done separately for each low-level model.

[38] also presents a hierarchical framework for a system which has uncertainty in modeling. [38] considers one high-level specification for the entire system, where, and in contrast to [38], we assume each plant has its own specifica-

tion. Furthermore, we present sufficient and necessary conditions for achieving Joint-SOCC property which ensures hierarchical consistency in the system consisting of several models. From the modeling point of view, our proposed framework has one more advantage over that of [38]; we exploit automata structure to present our results and models, however, the results in [38] are presented on a language basis and it is not known how a control structure is guaranteed for the system which consists of several model (Joint-OCC property in Chapter 4).

1.4 A Motivational Example

Figure 1.1 shows a flexible manufacturing system (FMS) which consists of two machines and two buffers. The machines work independently and can be run in parallel. Machine 1 can perform Operations 1 and 2 and Machine 2 can perform Operations 2 and 3. Workpieces are fed into the machines through two *feed-lines.* When a machine completes a process it will deposit the workpiece to either a buffer or the conveyor depending on the product recipes. The buffers have a capacity of one and they have access to both machines. Therefore, a transfer from one machine to another is possible using buffers. Three product recipes $\mathbf{A} = OP1 + OP3 + OP2$, $\mathbf{B} = OP3 + OP2$ and $\mathbf{C} = OP3 + OP1$ are available. Suppose due to safety regulation (or other planning considerations) it is required that after two consecutive production of A, one B and finally one C be produced. Furthermore, suppose the system is prone to fault and Machine 1 might fail to perform Operation 2 at any time. In order to start the process, the FMS should receive a *start* signal. The system could also be shut down at any time (*stop* signal). It is required that the system satisfy the safety regulation in both the normal and faulty modes, i.e. after every two A's,

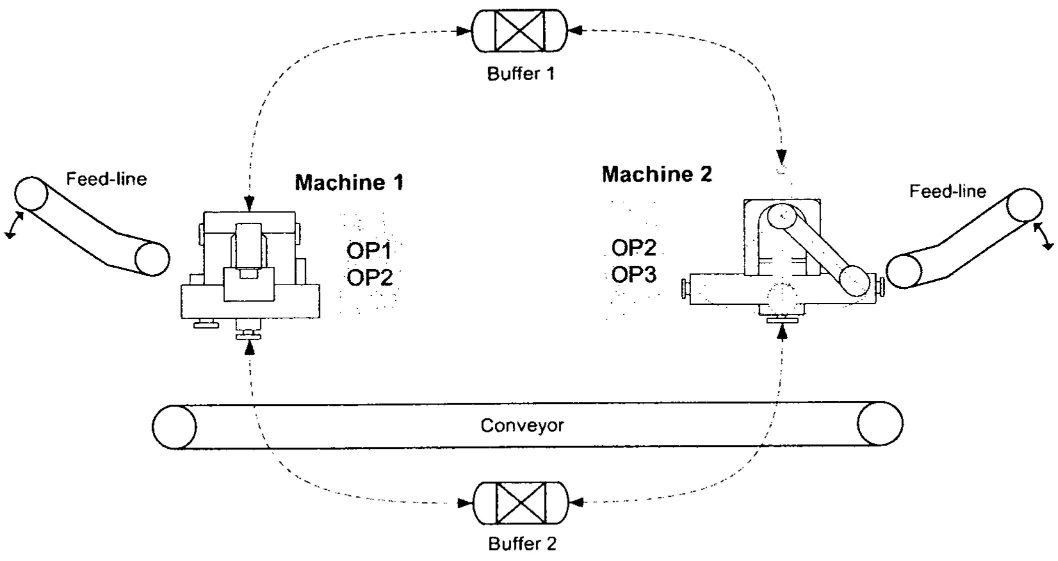


Figure 1.1: A flexible manufacturing system, consisting of two machines and two buffers

one B and one C are produced, and the maximum capability of the machines be used at any time (i.e. machines can be employed simultaneously). Finally a *repair* action is possible only when the system has stopped.

In Chapter 6 we will show how this problem can be solved as an HRSC problem, resulting in a transparent two-level supervisory control.

1.5 Complexity Issues

One main advantage of hierarchical supervisory control is its potential to reduce the computational complexity of supervisor synthesis. Wong [54] notes that in a hierarchical problem the high-level model can be obtained as the natural projection of a low-level Mealy model once the outputs at the low-level are assigned to transitions. Given the fact that natural projection could result in an exponential state explosion in the worst case, it is concluded that the state-size of the high-level model might increase exponentially, in the worst case, in contrast to our primary intention to reduce computational complexity in the system. However, [54] has shown if the reporting map is observer, the state-size of the high-level model will be less than or equal to that of the minimal canonical representation of the low-level model. This implies that if the reporting map is observer, the computational complexity of supervisor synthesis will not increase at the high-level. Therefore, it can be concluded that the state-size of the high-level model, hence the high-level supervisor synthesis computational complexity, in the worst case is exponential and in the best case is less than or equal to that of the low-level model. It should be noted that, the observer property is a sufficient condition in achieving the best case above. In other words, a high-level model, with a state-size smaller than that of the low-level model, might be achieved even if the reporting map is not observer. While the implications of the observer property is that transferring the low-level problem to the high-level does not increase the complexity, in particular, the adoption of a hierarchical approach has reduced computational complexity significantly. For instance, in the case study given in [28] the state-size of the final (high-level) model decreases by an order of eleven. In the simple case study we present in Chapter 6 the state-size of the high-level model is less than the size of the low-level model by an order of three.

1.6 Thesis Objectives

In this thesis we develop a bottom-up hierarchical solution to the RSC problem of [45]. The objective is to derive the conditions which ensure an RSC problem can be solved in a hierarchical framework, specifically in the bottom-up approach of [59] and under partial observation. The primary motivation for this work has been the RSC problems which are encountered in fault recovery problems. The importance of the recently investigated fault recovery problems in DES and the computational complexity of RSC problem motivates the development of a framework which can capture the structure of the RSC problem and reduce the supervisor synthesis computational complexity.

The RSC problem in [45] is formulated for a finite number of plant models, each matched with a specification. The goal of solving an RSC problem is to find a unique supervisor that regardless of which plant is currently active, it can disable events to keep the overall behavior of the system within the bounds set by specifications. The task of such supervisor becomes more crucial when the plant models share sequences of events and where some of these sequences need to be disabled to meet various specifications in the system. The supervision of such systems, due to the state-size explosion problem, is not usually easy.

The bottom-up hierarchical approach that we have used in this thesis can also be seen as a medium which can capture the natural hierarchy that one can find in or attribute to the behavior of a system. For example, a sequence of events which does a certain task is referred to by the task it does, rather than by the events it consists of. Therefore, if a certain task is done in different ways, i.e. different sequences of events perform the same task, then, it will be clearer to refer to the performed task rather than the different patterns of events which can do the task. Therefore, as another benefit, using a hierarchical framework, we can expect to gain a more user-friendly and transparent design in the system.

1.7 Thesis Outline and Contributions

The contributions of this thesis are in extending the bottom-up hierarchical framework of [59] to include partial observation and in development of a hierarchical solution for the problem of robust supervisory control. The outline of the thesis is as follows.

Chapter 2 reviews the preliminary mathematics and the theory of supervisory control which we use throughout the thesis.

Chapter 3 **extends the bottom-up hierarchical framework to include partial observation**. To study event observability at the high-level, we have introduced an output-observation-consistency condition which is the dual of the output-control-consistency of [59]. We show that the reporting map of the system should satisfy a feasibility condition which we refer to as Factorization Property. This condition is necessary for the construction of a reporting map that respects the observation limitations of the low-level. Specifically, it implies that low-level sequences with the same natural projection (look-alike sequences) must be reported and observed the same by the high-level supervisor (i.e. they must have the same natural projection at the high-level). We have also derived three conditions that ensure hierarchical consistency under partial observation. Three conditions that prevent unintentional disablement are Unobservability-and-Unique-Controllability (UUC), Unobservable-and-Uncontrollable-Prefixes-for-Observability (UUPO) and Partially-Observable-Strict-Output-Control-Consistency (PO-SOCC). The PO-SOCC property can be regarded as the extension of SOCC property [59] to the case of partial observation. If these conditions are satisfied, it is ensured that every high-level controllable and observable specification can be implemented at and recovered from the low-level. At the end of Chapter 3, we show how by proper modifications in the low-level models, the Factorization Property, UUC, UUPO and PO-SOCC can be ensured.

Chapter 4 **introduces the hierarchical robust supervisory control problem**. We propose a framework for solving the RSC problem in a hierarchical fashion. We show in a hierarchical RSC (HRSC) problem, the reporting map should meet a feasibility condition with respect to all system models. Furthermore, we show how hierarchical consistency is achieved in a system with multiple models (robust hierarchical consistency). To achieve robust hierarchical consistency, we have shown the plant models should satisfy a Joint-Strict-Output-Controllable-Consistency (Joint-SOCC) property, as the extension of SOCC property [59] to the case of robust problem.

Chapter 5 combines and links our previous results in Chapters 3 and 4 with each other. In Chapter 5, we solve HRSC problem under partial observation (HRSCPO). Specifically, we have shown how robust hierarchical consistency property can be achieved under partial observation. In order to do so, we show that it suffices to combine PO-SOCC property, which extends SOCC property [59] to the case of control under partial observation in Chapter 3, with Joint-SOCC property, which extends SOCC property to RSC problem in Chapter 4. This results in a more general property that is called Joint-PO-SOCC property.

Chapter 6 presents a case study. The theory in Chapter 4 is illustrated through a fault recovery problem in a flexible manufacturing system (FMS) consisting of two machines and two buffers. There, each machine is able to do several tasks and a *product* is a combination of a subset of the tasks that the system can perform in a certain order. Furthermore, the machines are prone to fault, the process starts in the beginning by an operator and can be shut down by the operator at any time. It is required that the system produce two products A and B and keep the production order in the faulty condition. We show how solving an HRSC problem will be a natural and transparent solution for the automation of the above manufacturing system.

Chapter 7 summarizes our results and proposes new research directions which can further extend and improve our results in this thesis.

Chapter 2

Background

Advancements in computerization and the trend towards process automation in industry, for years, have demanded a mathematical platform which can help experts and researchers in industry to come up with comprehensive and reliable tools for automation and supervision. In these applications, the type of the signals which is used is discrete both in time and value, e.g. acknowledge signal, start signal, shut-down signal. Here the commands, issued at some point in the system, or the actions which take place in the system are referred to as *events*. Upon the occurrence of any event, the system reaches a new state or keeps its current state unchanged. Therefore, as the dynamics of the systems evolves (i.e. certain events occur in the system) the state of the system changes from one to another or remains unchanged. This implies that the discrete behavior of a system can be modeled by a pattern of sequence of events. The automata theory is one of the most comprehensive and useful tools which enables us to capture the behavior of discrete-event systems as sequences of events (actions) and represent that behavior in a compact and computationally efficient form. Every automaton generates a language which in this concept would represent the behavior of the system. A language consists of finite or infinite set of sequences of events. In the theory of supervisory control, the system behavior modeled as languages can be restricted or (technically) controlled. The control action over the language of a system is exercised through the enablement and disablement of a subset of events which are *controllable.* Different methodologies have been developed so far for this control or supervision action. In the following sections of this chapter, we first review useful concepts from algebra and automata theory and then we will study the two main control frameworks and methodologies which are relevant to the subject of this thesis.

2.1 Algebra

A short review of the set theory follows. The reader is referred to [3, 56] for more details. Let X be a set. A binary relation $R : X \longmapsto X$ which can also be considered as a subset of $X \times X$ is said to be a *partial order* if it is

- reflexive: $(\forall x \in X)\ xRx$;
- transitive: $(\forall x, y, z \in X)\ xRy$ and $yRz \Longrightarrow xRz$;
- antisymmetric: $(\forall x, y \in X)\ xRy$ and $yRx \Longrightarrow x = y$

where by xRy we mean $R(x) = y$. A partial order R is denoted by $\leq$. We refer to the pair $(X, \leq)$ as a *poset* (or partially order set). Let X be a nonempty set, and $E \subseteq X \times X$ a binary relation on X. E is an **equivalence relation** if it is

- reflexive: $(\forall x \in X)\ xEx$;
- symmetric: $(\forall x, x' \in X)\ xEx' \Longrightarrow x'Ex$;
- transitive: $(\forall x, x', x'' \in X)\ xEx'$ and $x'Ex'' \Longrightarrow xEx''$.

If E is an equivalence relation and xEx' we write $x \equiv x'(mod\ E)$. Furthermore, for $x \in X$, the set $[x]$ denotes the subset of elements x' which are equivalent to x:

$$[x] = \{x' \in X \mid x'Ex\}.$$

Each subset $[x]$ is called a *coset* (or equivalently class) of x with respect of the equivalence relation E. It can easily be shown that any two cosets are either the same or disjoint.

Let $\mathcal{P}$ be a family of subsets of X induced by α in some index set A:

$$\mathcal{P} = \{C_\alpha \mid \alpha \in A\}, \;\; C_\alpha \subseteq X.$$

The family $\mathcal{P}$ is a partition of X if each subset C_α is nonempty, each x belongs to some C_α and subsets with distinct indices are pairwise disjoint. The subsets C_α are the cells of $\mathcal{P}$. Therefore, it can easily be shown that the cosets $[x]$ are a partition of X. Inversely it can be shown that every partition $\mathcal{P}$ of X corresponds to an equivalence relation E on X which is defined properly as:

$$xEy \;\text{ iff }\; (\exists\alpha \in A)x \in C_\alpha \;\text{ and }\; y \in C_\alpha.$$

Then let $\mathcal{E}(X)$ or simply $\mathcal{E}$ denote the class of all equivalence relations on (partitions of) X. A partial order $\leq$ can be assigned to $\mathcal{E}$ as follows:

$$(\forall E_1, E_2 \in \mathcal{E})E_1 \leq E_2 \;\text{ iff }\; (\forall x, y \in X)xE_1y \Longrightarrow xE_2y.$$

Here E_1 is said to be finer than E_2. If $E_1 \leq E_2$ then every coset of E_1 will be a subset of some (and therefore exactly one) coset of E_2.

One important equivalence relation on $\mathcal{E}(X)$ that we refer to in this thesis is the **equivalence kernel** of a function $f : X \longmapsto Y$ which is denoted by *ker f* and is defined as follows:

$$(\forall x, x' \in X) x \equiv x' (\text{mod } ker\ f) \text{ iff } f(x) = f(x').$$

Lemma 2.1. *[3] Suppose* $\mathrm{f} : X \longmapsto Y$ *and* $\mathrm{g} : X \longmapsto Z$ *and let* $ker\ \mathrm{g} \leq ker\ \mathrm{f}$*. Then there exists a unique map* $\mathrm{h} : Z \longmapsto Y$ *such that* $\mathrm{f} = \mathrm{h} \circ \mathrm{g}$*. Here* f *is said to factor through* g *(see Figure 2.1).*

$$\begin{tikzcd} & X \arrow[dl, "g"'] \arrow[dr, "f"] & \\ Z \arrow[rr, dashed, "h"] & & Y \end{tikzcd} \qquad \ker g \leq \ker f$$

Figure 2.1: If $ker\ g \leq ker\ f$, a unique map $h : Z \longmapsto Y$ exists and f is said to factor through g.

2.2 Automata Theory

A deterministic automaton is a 5-tuple $G = (Q, \Sigma, \delta, q_o, Q_m)$ where Q is the set of states, Σ the set of events, $\delta : Q \times \Sigma \longmapsto Q$ the partial transition function, q_o the initial state and Q_m the set of marked states. Σ is the event set and assumed to be nonempty and finite. Σ is also called an alphabet. In our notation, we will use Σ^+ to denote the set of all finite sequences of symbols (strings) over Σ. We also use $\Sigma^* = \Sigma^+ \cup \{\epsilon\}$, where $\epsilon \notin \Sigma$ is the empty string.

Definition 2.2. ***Catenation*** *of two sequences* s *and* t *is defined as* $cat(s, t) = st$.

Definition 2.3. *A* ***language*** *over alphabet* Σ *is any subset of* Σ^*.

The language $L(G) \subseteq \Sigma^*$, the closed behavior of G, [56] is defined as $L(G) = \{s \in \Sigma^* | \delta(q_o, s)!\}$ where $\delta(q_o, s)!$ means $\delta(q_o, s)$ is defined. $L(G)$ in fact represents the dynamics of the system modeled by G. On the other hand, the marked language $L_m(G) \subseteq L(G)$ which is the set of sequences which end in Q_m is defined as $L_m(G) = \{s \in L(G) | \delta(q_\circ, s) \in Q_m\}$. $L_m(G)$ is considered as the set of sequences which flag the accomplishment of certain tasks in the system denoted by the states of Q_m.

Outputs can be attributed to transitions or states of an automaton such that upon executing any event or reaching a state of that automaton, an output is generated. An automaton G is a Mealy automaton if its transitions are assigned outputs and is called a Moore automaton if the outputs are assigned to its states.

A **Mealy automaton** is a 7-tuple $G = (Q, \Sigma, T, \delta, \eta, q_\circ, Q_m)$ where Q, Σ, δ, $q_\circ$ and Q_m are defined as before, T is the set of outputs and $\eta : Q \times \Sigma \longmapsto T$ is the output map satisfying the following

$$\begin{cases} \eta(q, \sigma) \in T, & \text{if } \delta(q, \sigma)!; \\ undefined, & \text{otherwise.} \end{cases}$$

Therefore, different transitions reaching a state q' might produce different outputs. On the other hand, a Moore automata generates a unique output at each state:

A **Moore automaton** is a 7-tuple $G = (Q, \Sigma, T, \delta, \omega, q_\circ, Q_m)$ where Q, Σ, δ, $q_\circ$ and Q_m are defined as before, T is the set of outputs and $\omega : Q \longmapsto T$ is the output map. Mealy automata can be converted into Moore automata and vice versa [9]. The properties given in the following are general and valid regardless of associating outputs to the model or not.

Definition 2.4. *For every $s \in \Sigma^*$, the set $pre(s) = \{s' \in \Sigma^* | \exists s'' \in \Sigma^* : s = s's''\}$ denotes the* ***prefixes*** *of s. If $s' \in pre(s)$ we write $s' \leq s$.*

Definition 2.5. *Let $L \subseteq \Sigma^*$ be a language. The* ***prefix-closure*** *of L is* *defined by $\overline{L} = \{s' \in \Sigma^* | \exists s \in L : s' \leq s\}$. $\overline{L}$ denotes all sequences which have an extension in L.*

Definition 2.6. *An automaton G is* ***nonblocking*** *if we have $\overline{L_m(G)} = L(G)$.*

If there is a path s from the initial state $q_\circ$ to some state q, i.e. $q = \delta(q_\circ, s)$, q is said to be **reachable** from $q_\circ$ or simply reachable. We obtain the reachable part of an automaton by deleting the unreachable states and the transitions which are attached to them. If there is a path s from the state q to a marked state, i.e. $\delta(q, s) \in Q_m$, we say q is **coreachable** to Q_m or simply coreachable. We can obtain the coreachable part of an automaton by deleting the states which are not coreachable. Coreachability is closely related to the concept of blocking. An automaton is nonblocking if and only if every reachable state is coreachable.

Definition 2.7. *An automaton G is said to be* ***trim*** *if its states are reachable and coreachable.*

Let $trim(G)$ denote the trim part of G. For every automaton $G = (Q, \Sigma, \delta, q_\circ, Q_m)$ with the event set Σ which marks the language $L_m(G)$, we can build another automaton G^{co} which marks the language $\Sigma^* - L_m(G)$. Automaton G^{co} is called the **complement** of automaton G with respect to the set Σ.
Several operations such as intersection, synchronization and union can be performed on languages. Such operations can be performed on automata which mark these languages. These operations are known as compositional operations. We review some of them which are used in this thesis.

Definition 2.8. *Consider two automata $G_i = (Q_i, \Sigma_i, \delta_i, q_{i,\circ}, Q_{i,m})$ $(i = 1, 2)$. The* ***Product*** *of G_1 and G_2 is shown by $G_1 \times G_2$ and is defined as the reachable* *part of the automaton* $(Q, \Sigma, \delta, q_o, Q_m)$ *where*

$$\begin{aligned} Q &= Q_1 \times Q_2 \\ \Sigma &= \Sigma_1 \cap \Sigma_2 \\ q_o &= (q_{1,o}, q_{2,o}) \\ Q_m &= Q_{1,m} \times Q_{2,m} \end{aligned}$$

and partial transition function $\delta : Q \times \Sigma \longmapsto Q$ *is defined as*

$$\delta((q_1, q_2), \sigma) = \begin{cases} (\delta_1(q_1, \sigma), \delta_2(q_2, \sigma)), & \textit{if } \delta_1(q_1, \sigma)! \textit{ and } \delta_2(q_2, \sigma)!; \\ \textit{undefined}, & \textit{otherwise.} \end{cases} \tag{2.1}$$

The product $G_1 \times G_2$ represents the common behavior of two systems G_1 and G_2. In can easily be shown that $L(G_1 \times G_2) = L(G_1) \cap L(G_2)$ and $L_m(G_1 \times G_2) = L_m(G_1) \cap L_m(G_2)$.

Natural projection is another useful operation.

Definition 2.9. *Let* Σ *be a set and* $\Sigma_o \subseteq \Sigma$ *be a subset of it. The map* $P : \Sigma^* \longmapsto \Sigma_o^*$ *is called the* ***natural projection*** *and is defined as follows.*

$$\begin{aligned} P(\epsilon) &= \epsilon \\ P(\sigma) &= \begin{cases} \sigma, & \textit{if } \sigma \in \Sigma_o; \\ \epsilon, & \textit{otherwise.} \end{cases} \\ P(s\sigma) &= P(s)P(\sigma) \quad \textit{for } s \in \Sigma^*, \sigma \in \Sigma. \end{aligned} \tag{2.2}$$

The inverse natural projection denoted by $P^{-1} : \Sigma_o^* \longmapsto 2^{\Sigma^*}$ is defined as follows. For a sequence $t \in \Sigma_o^*$

$$P^{-1}(t) = \{s \in \Sigma^* | P(s) = t\}. \tag{2.3}$$

While the product of two automata represents the common behavior of two system's components, it is usually the joint behavior of two system's components which is of importance in modeling the behavior of the total system. In joint operation, two components synchronize on common events only and operate independently when performing their private (uncommon) events. A mathematical definition for parallel behavior follows.

Definition 2.10. *Let $G_i = (Q_i, \Sigma_i, \delta_i, q_{i,o}, Q_{i,m})$ $(i = 1, 2)$ be two automata. The* ***Synchronous product or parallel composition*** *of G_1 and G_2 is shown by $G_1||G_2$ and is defined as the reachable part of $(Q, \Sigma, \delta, q_o, Q_m)$ where*

$$\begin{aligned} Q &= Q_1 \times Q_2 \\ \Sigma &= \Sigma_1 \cup \Sigma_2 \\ q_o &= (q_{1,o}, q_{2,o}) \\ Q_m &= Q_{1,m} \times Q_{2,m} \end{aligned}$$

and the partial transition function $\delta : Q \times \Sigma \longmapsto Q$ is defined as

$$\delta((q_1, q_2), \sigma) = \begin{cases} (\delta_1(q_1, \sigma), \delta_2(q_2, \sigma)), & \text{if } \sigma \in \Sigma_1 \cap \Sigma_2 \text{ and } \delta_1(q_1, \sigma)! \\ & \text{and } \delta_2(q_2, \sigma)!; \\ (\delta_1(q_1, \sigma), q_2), & \text{if } \sigma \in \Sigma_1 - \Sigma_2 \text{ and } \delta_1(q_1, \sigma)!; \\ (q_1, \delta_2(q_2, \sigma)), & \text{if } \sigma \in \Sigma_2 - \Sigma_1 \text{ and } \delta_2(q_2, \sigma)!; \\ \text{undefined}, & \text{otherwise.} \end{cases} \tag{2.4}$$

The first row in (2.4), describes the common behavior (event) of G_1 and G_2 which occurs at states in which both automata are able to perform it. On the other hand, the second and the third row show the independent behaviors (private events) of G_1 and G_2 that can occur regardless of the state of the other automaton.

Let $P_1 : \Sigma^* \longmapsto \Sigma_1^*$ and $P_2 : \Sigma^* \longmapsto \Sigma_2^*$ be the natural projections with Σ_1 and Σ_2 as the event sets of G_1 and G_2 and $\Sigma = \Sigma_1 \cup \Sigma_2$. It can be shown that $L(G_1||G_2)$ is obtained as

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

Next we discuss a procedure for constructing an automaton to generate and make the union of the closed behaviors and the union of the marked behaviors of two automata. This operation does not seem to be standard in the literature. However, in this thesis, we will find it useful in our discussions.

Definition 2.11. ***Union ($\cup$) of Automata:*** *Consider two automata $G_i = (Q_i, \Sigma_i, \delta_i, q_{i,\circ}, Q_{i,m})$ $(i = 1, 2)$. The union of G_1 and G_2, denoted by $G = G_1 \cup G_2$, is the reachable part of the automaton $(Q, \Sigma, \delta, q_\circ, Q_m)$ where*

$$\begin{aligned} Q &= (Q_1 \dot{\cup} \{d_1\}) \times (Q_2 \dot{\cup} \{d_2\}) - \{(d_1, d_2)\}, \\ \Sigma &= \Sigma_1 \cup \Sigma_2, \\ q_\circ &= (q_{1,\circ}, q_{2,\circ}), \\ Q_m &= (Q_{1,m} \dot{\cup} \{d_1\}) \times (Q_{2,m} \dot{\cup} \{d_2\}) - \{(d_1, d_2)\}, \end{aligned}$$

$\dot{\cup}$ denotes the disjoint union operation, d_1 and d_2 are new labels that do not exist in Q_1 and Q_2, and the partial transition function $\delta : Q \times \Sigma \longmapsto Q$ is

defined as

$$\delta((q_1,q_2),\sigma) = \begin{cases} (\delta_1(q_1,\sigma),\delta_2(q_2,\sigma)), & \textit{if } \delta_1(q_1,\sigma)! \textit{ and } \delta_1(q_2,\sigma)!; \\ (\delta_1(q_1,\sigma),d_2), & \textit{if only } \delta_1(q_1,\sigma)!\ ; \\ (d_1,\delta_2(q_2,\sigma)), & \textit{if only } \delta_2(q_2,\sigma)!; \\ \textit{undefined} & \textit{otherwise.} \end{cases}$$

$$\delta((q_1,d_2),\sigma) = \begin{cases} (\delta_1(q_1,\sigma),d_2), & \textit{if } \delta_1(q_1,\sigma)!\ ; \\ \textit{undefined} & \textit{otherwise.} \end{cases}$$

$$\delta((d_1,q_2),\sigma) = \begin{cases} (d_1,\delta_2(q_2,\sigma)), & \textit{if } \delta_2(q_2,\sigma)!\ ; \\ \textit{undefined} & \textit{otherwise.} \end{cases}$$

Note that $L(G) = L(G_1) \cup L(G_2)$ and $L_m(G) = L_m(G_1) \cup L_m(G_2)$. Also $\delta((q_{1,o}, q_{2,o}), s) = (q_1, d_2)$ if and only if $s \in L(G_1) - L(G_2)$ and $\delta((q_{1,o}, q_{2,o}), s) = (d_1, q_2)$ if and only if $s \in L(G_2) - L(G_1)$. Furthermore, it can be seen that union operation is associative. Alternatively, $G = G_1 \cup \cdots \cup G_n$ with $G_i = (Q_i, \Sigma_i, \delta_i, q_{i,o}, Q_{i,m})$, can be constructed as

$$G = trim[(G_1^{co} \times \cdots \times G_n^{co})^{co}] \tag{2.5}$$

where $trim(.)$ denotes the trim operation.

Definition 2.12. *Consider a model $G = (Q, \Sigma, \delta, q_o, Q)$ and let $\Sigma = \Sigma_o \dot{\cup} \Sigma_{uo}$ and $P : \Sigma^* \longmapsto \Sigma_o^*$ be a natural projection map. The automaton*

$$\tilde{G} = (\tilde{Q}, \Sigma_o, \tilde{\delta}, \tilde{q}_o, \tilde{Q})$$

is called the (reachable) ***observer automaton*** *of G and characterized as follows.*

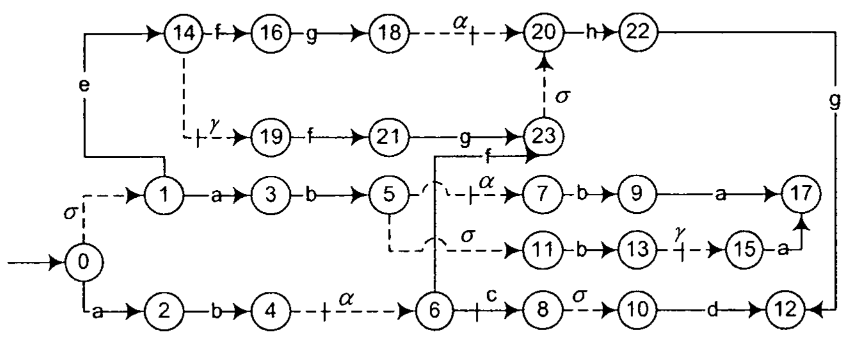


Figure 2.2: An automaton G with unobservable transitions denoted by dashed lines

$\tilde{Q} \subseteq 2^Q - \{\emptyset\}$ *is the state set and*

$$\tilde{q}_o = \{q \mid \exists s \in L(G) : \ q = \delta(q_o, s) \text{ and } P(s) = \epsilon\}.$$

The transition function $\tilde{\delta}$ *is defined as follows: For* $\tilde{q}_1 \in \tilde{Q}$ *and* $\sigma \in \Sigma_o$,

$$\tilde{q}_2 = \tilde{\delta}(\tilde{q}_1, \sigma) = \{q_2 \mid \exists q_1 \in \tilde{q}_1 \text{ and } s \in P^{-1}(\sigma) : q_2 = \delta(q_1, s) \ \}.$$

$\tilde{G}$ generates $L(\tilde{G}) = P(L(G))$; i.e. it generates the projection of the closed behavior of G. If a state $\tilde{q}$ is reached in $\tilde{G}$ by a sequence $\mathbf{S} \in \Sigma_o^*$ ($\tilde{q} = \tilde{\delta}(\tilde{q}_o, \mathbf{S})$), then $\tilde{q}$ would be the estimate of the states of G given observation $\mathbf{S}$. We will denote this state estimate by $est(\mathbf{S})$ as well. Thus

$$est(\mathbf{S}) = \tilde{\delta}(\tilde{q}_o, \mathbf{S}) = \{q \mid \exists s' \in P^{-1}(\mathbf{S}) \cap L(G) : q = \delta(q_o, s')\}. \tag{2.6}$$

Therefore, the partial transition function $\tilde{\delta}$ in the observer automaton $\tilde{G}$ has the property that for every $\mathbf{S}, \mathbf{S}' \in L(\tilde{G})$

$$\tilde{\delta}(\tilde{q}_o, \mathbf{S}) = \tilde{\delta}(\tilde{q}_o, \mathbf{S}') \Longleftrightarrow est(\mathbf{S}) = est(\mathbf{S}').$$

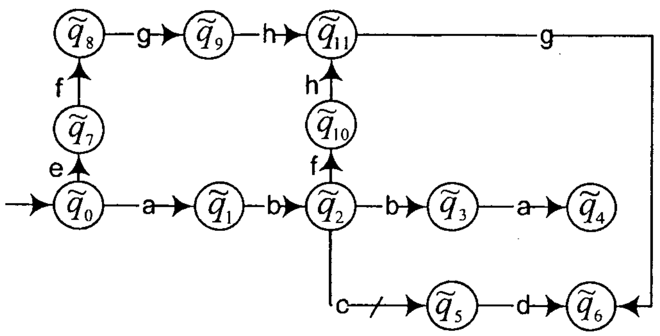


Figure 2.3: Observer automaton $\tilde{G}$ for model G in Fig. 2.2

Example 2.13. *Fig. 2.2 shows an automaton* $G = (Q, \Sigma, \delta, q_o, Q)$ *with the event set* $\Sigma = \{a,\ b,\ c,\ d,\ e,\ f,\ g,\ h\} \cup \{\alpha, \gamma, \sigma\}$ *and observable events* $\Sigma_o = \{a,\ b,\ c,\ d,\ e,\ f,\ g,\ h\}$. *Unobservable events have been shown by dashed lines. The observer automaton* $\tilde{G}$ *has been given in Fig. 2.3. Each state* $\tilde{q}_i = \delta(\tilde{q}_o, \boldsymbol{S})$ *of* $\tilde{G}$ *corresponds to a state estimate* $est(\boldsymbol{S})$. *Examples include* $\tilde{q}_o = est(\epsilon) = \{0, 1\}$, $\tilde{q}_8 = est(e.f) = \{16, 21\}$, $\tilde{q}_9 = est(e.f.g) = \{18, 20, 23\}$ *and* $\tilde{q}_{10} = est(a.b.f) = \{20, 23\}$. *This shows the state estimates are not necessarily disjoint as can be seen for* $est(e.f.g)$ *and* $est(a.b.f)$. *Furthermore, since* $est(e.f.g) \neq est(a.b.f)$, *(2.6) implies* $e.f.g$ *and* $a.b.f$ *would reach two different states in the observer automaton* $\tilde{G}$, *namely, states* $\tilde{q}_9$ *and* $\tilde{q}_{10}$.

2.3 Supervisory control of Discrete-Event Systems

Consider an automaton G modeling a system. The closed behavior of G, $L(G)$, represents all possible behavior (sequences) which the system can experience. However, it is often needed that the system's behavior be restricted (by a supervisor) such that the controlled system's behavior satisfies certain characteristics. In supervisory control of DES, the language of the system under supervision must be contained in some specification language. Here, the specification language is the desired language which is obtained from the design requirements. The supervision which is studied in this thesis is language-based; however, the computations for such supervision are carried out on automata, representing the languages.

Let an automaton $G = (Q, \Sigma, \delta, q_o, Q_m)$ describe a plant model. It is assumed that the event set Σ can be partitioned into disjoint sets of **controllable** events Σ_c and **uncontrollable** events Σ_{uc} as $\Sigma = \Sigma_c \dot{\cup} \Sigma_{uc}$. The above partitioning follows from the assumption that only a subset of events (Σ_c) can be disabled (and controlled) by supervisor. In the following we give a formal definition of supervisor. Let

$$\Gamma = \{\gamma \in 2^{\Sigma} | \gamma \supseteq \Sigma_{uc}\}$$

be the set of *control patterns* and define a map $S : L(G) \longmapsto \Gamma$. Then S is called a **supervisor** for G if for every sequence $s \in L(G)$, $S(s)$ is a subset of Σ which at least includes Σ_{uc} and is enabled after s.

Definition 2.14. *A Supervisor S is called* ***admissible*** *if it does not disable any uncontrollable event.*

Admissibility can be regarded as a physical restriction and can be ensured through the controllability property (to be discussed later). Furthermore, we use the notation S/G to denote the controlled system which is also called the *system under supervision.* The language of the system under supervision, i.e. $L(S/G)$, by definition is calculated as follows.

(i) $\epsilon \in L(S/G)$

(ii) If $s \in L(S/G), \sigma \in S(s)$, and $s\sigma \in L(G)$ then $s\sigma \in L(S/G)$.

(iii) No other string belongs to L(S/G).

The marked language of a system under supervision is also obtained by $L_m(S/G) = L(S/G) \cap L_m(G)$.

In a supervisory control problem the desired behavior is described in terms of a specification language. The sublanguage of the marked behavior that belongs to the specification language and satisfies the specification, is called the **legal marked behavior**. Let $E \subseteq L_m(G)$ denote the legal behavior. A control problem is considered fully solved if a supervisor, as elaborated above, exists so that the language of the system under supervision is contained in the specification language, namely, it meets the specifications and that the system under supervision is nonblocking.

Definition 2.15. *For a plant G and a legal language E, we say the nonblocking supervisory control problem is solved if there exists an admissible supervisor S such that*

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

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

*Furthermore, a supervisor S is called **maximally permissive** if and only if for any other supervisor S' which solves the supervisory control problem for G and E we have $L_m(S'/G) \subseteq L_m(S/G)$ and $L(S'/G) \subseteq L(S/G)$.*

Since supervisor admissibility property is always required, not every legal language can be generated by the system under supervision. The admissibility of supervisor is related to the property of controllability.

Definition 2.16. *A language $K \in \Sigma^*$ is said to be **controllable with respect to another closed language** M if $\overline{K}\Sigma_{uc} \cap M \subseteq \overline{K}$.*

It can be shown that *controllability* is closed with respect to the union operation. Furthermore, the empty language is always controllable. Therefore, for every language E there exists a supremal sublanguage of E which is controllable with respect to $L(G)$.

Notation 2.17. *We use $E_G^{\uparrow}$ to denote the supremal controllable sublanguage of E with respect to $L(G)$.*

The system under supervision will be nonblocking if we have $\overline{L_m(S/G)} = L(S/G)$. This is achieved if the (controllable) legal behavior (or a controllable sublanguage of it) is $L_m(G)$-closed.

Definition 2.18. *[56] A language K is said to be **L-closed** if $\overline{K} \cap L = K$.*

The existence of a nonblocking supervisor for a given plant model G, a specification E, the set of uncontrollable events Σ_{uc} is examined in Theorem 2.19 [41]:

Theorem 2.19. *[41] (**Nonblocking Supervisory Control**): Consider a plant G and a legal behavior $E \subseteq L_m(G)$ and a language $K \subseteq E$, $K \neq \emptyset$. There exists a nonblocking supervisor S for G such that $L_m(S/G) = K$ and $L(S/G) = \overline{K}$ if and only if*

(i) K is controllable w.r.t $L(G)$.

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

Theorem 2.19 indicates for any sublanguage K of E which is $L_m(G)$-closed and controllable with respect to $L(G)$, there exists a supervisor S which can synthesize K.

Remark 2.20. *Supervisor S can be implemented by the automaton which generates $\overline{K}$.*

Due to observation limitations, a subset of event set Σ may be unobservable. That is, the supervisor would not recognize if they have occurred. Observable events are those events whose occurrence are detected (observed) by the supervisor. Therefore, in a general case, event set Σ is partitioned as $\Sigma = \Sigma_o \dot{\cup} \Sigma_{uo}$ where Σ_o and Σ_{uo} are respectively the subset of **observable** and **unobservable** events. This implies, two sequences of events corresponding to two different patterns of behavior, in general, may seem the same to supervisor, i.e. they have the same pattern of observable events. More precisely, if $P : \Sigma^* \longmapsto \Sigma_o^*$ is a natural projection, for $s, s' \in \Sigma^*$, $s \neq s'$, we would have $P(s) = P(s')$. In such a case, the decision of supervisor for the two look-alike sequences s and s' must be the same. In other words, $S(s) = S(s')$. Such a supervisor is called **feasible**. Feasibility of supervisor can be ensured for cases involving *observable languages.*

Definition 2.21. *Let $G = (Q, \Sigma, \delta, q_o, Q_m)$ be an automaton, $\Sigma = \Sigma_o \dot{\cup} \Sigma_{uo}$ and $P : \Sigma^* \longmapsto \Sigma_o^*$ be the natural projection. A language E is said to be $(L(G), P)$-observable if for every two sequences $s, s' \in L(G)$ with $P(s) = P(s')$ and for every $\sigma \in \Sigma$*

$$s\sigma \in E \;\; \text{and} \;\; s'\sigma \in L(G) \;\; \text{and} \;\; s' \in E \Longrightarrow s'\sigma \in E.$$

The **observability** property requires two sequences s, s' that have the same natural projection and are extended by the same event σ, to be treated the same by the supervisor S; i.e. if both s and s' belong to E then $\sigma \in S(s) \Longleftrightarrow \sigma \in S(s')$ for every $\sigma \in \Sigma$. Observability is not closed under union operation of languages. Therefore, a supremal observable sublanguage of E with respect to $L(G)$ might not exist. An alternative property which is sometimes used is *normality.*

Definition 2.22. *Let $G = (Q, \Sigma, \delta, q_o, Q_m)$ be an automaton, $\Sigma = \Sigma_o \dot{\cup} \Sigma_{uo}$ and $P : \Sigma^* \longmapsto \Sigma_o^*$ be a natural projection. A language $K \subseteq L(G)$ is said to be $(L(G), P)$-**normal** if $\overline{K} = L(G) \cap P^{-1}P(\overline{K})$.*

Normal languages are closed under union operation and $\emptyset$ is a normal language. Thus a supremal normal sublanguage of a given language exists. While every $(L(G), P)$-normal language K is $(L(G), P)$-observable, the reverse is not necessarily true. The following proposition [9] describes a set of conditions under which observability implies normality.

Proposition 2.23. *[9] Assume $\Sigma_c \subseteq \Sigma_o$. Then if K is controllable w.r.t. $L(G)$ and is $(L(G), P)$-observable, then it is $(L(G), P)$-normal.*

It follows from Proposition 2.23 that when the assumption $\Sigma_c \subseteq \Sigma_o$ holds (all the controllable events are observable) a supremal observable and controllable sublanguage of a given language E exists and is equal to the supremal normal and controllable sublanguage of E. In the rest of this thesis, unless explicitly stated, observability property will be used. Note that, normality is stronger than observability.

Theorem 2.24. *[9] **(Nonblocking Supervisory Control Under Partial Observation)**: Consider a plant G and a legal behavior $E \subseteq L_m(G)$ and a language $K \subseteq E$, $K \neq \emptyset$. There exists a feasible admissible nonblocking supervisor S for G such that $L_m(S/G) = K$ and $L(S/G) = \overline{K}$ if and only if*

(i) K is controllable w.r.t $L(G)$.

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

(iii) K is $(L(G), P)$-observable.

2.4 Robust Supervisory Control (RSC) Problem

Robustness of control system against changes in plant dynamics or changes of specification is an important and desired property. Robustness in supervision of DES has been developed in the literature from different viewpoints [35, 36, 30, 4, 45]. A review of the RSC problem in [4, 45] follows.

Consider a family of DES consisting of N models $G_1, \cdots, G_N$ and the corresponding legal behaviors $E_1, \cdots, E_N$. Let Σ_i be the event set of G_i and $\Sigma_{i,o}$ and $\Sigma_{i,uc}$ the subsets of observable and uncontrollable events in Σ_i. Although Σ_i's are not necessarily the same, it is assumed they agree on the controllability and observability of their common events. In other words, an event $\sigma \in \Sigma_i \cap \Sigma_j$ $(i \neq j)$ is controllable (resp. observable) in Σ_i if and only if it is controllable (resp. observable) in Σ_j $(i \neq j)$.

It is desired that each model G_i under supervision meets its corresponding legal language E_i; In a sense, this implies that N separate supervisory control problems be solved and N separate supervisors be obtained. However, in a case that the actual model of a system is unknown and is only known to be among the finite set of models G_i $(i = 1, \cdots, N)$, solving supervisory control problem requires a unique nonblocking and feasible supervisor that the system under its supervision can meet the specifications regardless of what the current model actually is. A formal definition of RSC-PO problem follows.

Definition 2.25. *[45]* ***Robust nonblocking supervisory control problem under partial observation (RNSC-PO)****: Given plants G_i and specifications E_i for $i = 1, \cdots, N$, let $\Sigma = \bigcup_{1 \leq i \leq N} \Sigma_i$ and $\Sigma_{uc} = \bigcup_{1 \leq i \leq N} \Sigma_{i,uc}$ and define $\Gamma = \{\gamma \subseteq \Sigma | \gamma \supseteq \Sigma_{uc}\}$. S is said to be a robust feasible nonblocking supervisor if for $i = 1, \cdots, N$ we have*

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

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

Definition 2.26. *[45] (G_i-nonblocking): Let G be a DES with event set Σ, and marked and closed languages $L_m(G)$ and $L(G)$. Then a language $K \subseteq \Sigma^*$*

*is called **G-nonblocking** if* $\overline{K \cap L_m(G)} = \overline{K} \cap L(G)$.

Theorem 2.27. *[45](*Solution to RNSC-PO problem*) Given plants G_i and specifications E_i for $i = 1, \cdots, N$, let Σ, Γ and S be defined as in Definition 2.25. Let G be an automaton for which we have $L_m(G) = \bigcup_{1 \le i \le N} L_m(G_i)$ and $L(G) = \bigcup_{1 \le i \le N} L(G_i)$ and define $E = \bigcap_{1 \le i \le N} (E_i \cup (\Sigma^* - L_m(G_i))) \cap L_m(G)$. Then there exists a feasible supervisor S which solves the RNSC-PO problem if and only if there exits a language $K \subseteq E$, $K \neq \emptyset$ which is*

(i) controllable w.r.t. $L(G)$;

(ii) $(L(G), P)$-observable;

(iii) $L_m(G)$-closed;

(iv) G_i-nonblocking for $i = 1, \cdots, N$.

If such S exists, then $L_m(S/G) = K$ and $L(S/G) = \overline{K}$ and

(1) $L_m(S/G_i) = K \cap L_m(G_i)$

(2) $L(S/G_i) = \overline{K} \cap L(G_i)$

The results (1) and (2) in Theorem 2.27 are in fact the implications of Lemma 2.28 [12] applied to plant models G_i ($i = 1, \cdots, N$) and G.

Lemma 2.28. *[12] Consider two automata G and H with $L(H) \subseteq L(G)$. Let S/G and S/H denote respectively G and H under the supervision of a supervisor S. Then*

(i) $L(S/H) = L(S/G) \cap L(H)$

(ii) $L_m(S/H) = L_m(S/G) \cap L_m(H)$

The reader is referred to [4, 45] for the computation methods using which the conditions $(i) - (iv)$ in Theorem 2.27 can be achieved.

Remark 2.29. *In cases, where nonblocking property is not studied in supervisory control problem, then $Q_i = Q_{m,i}$, and in Theorem 2.27, conditions (iii)-(iv) can be eliminated (since they will always hold). The automaton G in Theorem 2.27 can be obtained from the union of the plant models G_i $(i = 1, \cdots, N)$. Therefore, RNSC-PO problem, can be seen as a supervisory control problem for plant G and the specification E which is obtained from E_i $(i = 1, \cdots, N)$ as given. Once $L(S/G)$ is obtained, the individual languages $L(S/G_i)$ can be obtained using Lemma 2.28.*

2.5 Hierarchical Supervisory Control (HSC)

The primary goal in controlling a system is to shape the behavior of the system such that the system satisfies certain characteristics; however, different priorities in different levels of management demand controlling systems from different viewpoints. For example, a system might be controlled to avoid buffer overflow- or underflowing at the lowest level while the manager at a higher level, assured of proper working of the system and the buffers under normal conditions, may only be concerned with the cost of repairs. Moreover, at a higher level of management not all of the details of the system are necessarily needed. In other words, some details of the dynamics of the system can be ignored by the manager since they are not needed or are irrelevant to the specifications at the high-level. This suggests the dynamics of the systems can be aggregated to certain order, depending on the relation between the plant and the high-level specifications, resulting in lower computational complexity for supervisor design. Improvement in the transparency of user interface and design can also be mentioned as additional benefits of such aggregation. Model aggregation has been studied in the literature from

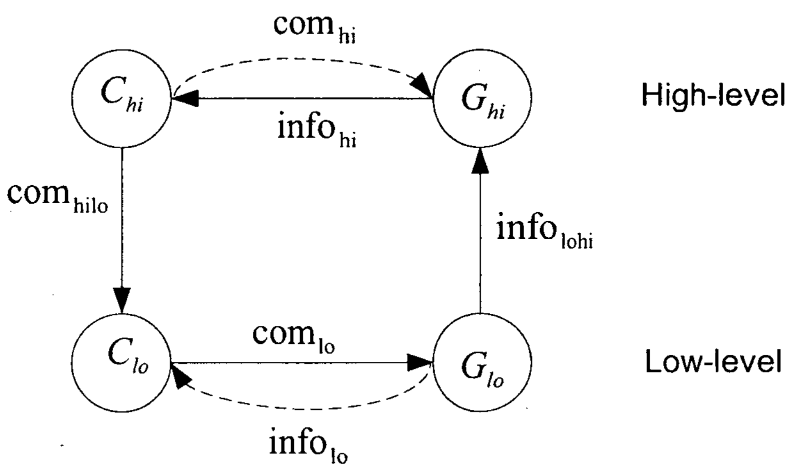


Figure 2.4: General overview of a hierarchical supervisory control system

different viewpoints [5, 7, 27, 28, 52, 59]. The bottom-up hierarchical framework of [Zhong-Wonham] [59] is a language-based platform which aggregates sequences of events of an alphabet Σ into another alphabet T. This is compatible with the robust supervisory framework we explained in previous section. A review of the bottom-up hierarchical framework of [59] follows.

In the hierarchical framework of [59], Fig. 2.4, two levels of hierarchy, low-level and high-level, are identifiable. In brief, a map summarizes the behavior of the low-level model G_{lo} and reports it to the high-level to generate the high-level model G_{hi}. Let E_{hi} be the high-level expectation (specification). Then a supervisory control problem is solved for G_{hi} and E_{hi}. The dashed lines in Fig. 2.4 indicate virtual transmissions of information or commands which are not implemented directly. Fig. 2.4 shows that the actual feedback system is closed not through the interaction between the high-level controller C_{hi} and the high-level model G_{hi} but rather through first, a translation channel com_{hilo} which translates the high-level commands to the low-level, then, a low-level feedback channel com$_{lo}$ which communicates to G_{lo} and finally a reporting map info$_{lohi}$ which summarizes the latest low-level model's behavior evolution for G_{hi}. This is a bottom-up hierarchical supervisory control framework [59].

If the reporting map info$_{lohi}$ is chosen properly, then HSC approach can reduce the computational complexity of supervisor synthesis. A short review of the synthesis of the reporting map info$_{lohi}$ which is denoted by θ follows.

The low-level model is assumed to be a finite state deterministic automaton $G_{lo} = (Q, \Sigma, q_o, \delta, Q)$ that represents the actual plant model which is to be controlled in the real world. The abstraction channel info$_{lohi}$ is a causal map $\theta : L(G_{lo}) \longmapsto T^*$ where T is the set of events at the high-level. The causal map θ would have the following property:

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

$$\theta(s\sigma) = \begin{cases} \text{either} & \theta(s) \\ \text{or} & \theta(s)\tau, \quad \text{for some } \tau \in T \end{cases} \tag{2.7}$$

Therefore, the sequences of events in $L(G_{lo})$ are abstracted into events of T and more generally into sequences of T^*. For a sequence s, $\theta(s)$ carries the important information that is transmitted to the high-level. Next, a finite state deterministic automaton G_{hi} can be realized to model the high-level language $L(G_{hi}) = \theta(L(G_{lo}))$. An output map $\hat{\omega} : L(G_{lo}) \longmapsto T \cup \{\tau_\circ\}$ where $\tau_\circ$ is called the silent event, is defined over $L(G_{lo})$ such that for every $s \in L(G_{lo})$ we have

$$\hat{\omega}(s\sigma) = \begin{cases} \tau_\circ & \text{if } \theta(s\sigma) = \theta(s) \\ \tau & \text{if } \theta(s\sigma) = \theta(s)\tau. \end{cases} \tag{2.8}$$

Therefore the output map $\hat{\omega}$ generates the silent event $\tau_\circ$ whenever the map θ outputs nothing and generates a new high-level event τ otherwise. Furthermore, a new output map $\omega : Q \longmapsto T \cup \{\tau_\circ\}$ can be incorporated into the low-level model G_{lo} to satisfy the following: $\omega(q) = \hat{\omega}(s)$ for any s such that $q = \delta(q_\circ, s)$. This yields a Moore automaton $G_{lo} = (Q, \Sigma, T \cup \{\tau_\circ\}, \delta, \omega, q_\circ, Q)$.

In this way, the output map ω generates the information transmitted to the high-level.

A state $q \in Q$ of G_{lo} is called **vocal** if $\omega(q) \neq \tau_{\circ}$. A sequence $s \in L(G_{lo})$ for which $q = \delta(q_{\circ}, s)$ is vocal, is called **vocal.** A sequence s that connects the initial state $q_{\circ}$ to a vocal state or a vocal state to another vocal state through a set of nonvocal states is called a **silent path**.

Notation 2.30. *The empty string and the set of sequences leading to a vocal state are denoted by $L_{voc} = \{s \in L(G_{lo}) | s = \epsilon \text{ or } \hat{\omega}(s) \neq \tau_{\circ}\}$. The nonempty silent extensions of a sequence $s \in L(G_{lo})$ that lead to a vocal state are denoted by $L_{voc}(s) = \{s' \in \Sigma^{+} | ss' \in L_{voc} \text{ and } (\exists \tau \in T : \ \ \theta(ss') = \theta(s)\tau)\}$. Note that the sequence $s \in L(G_{lo})$ in the definition of $L_{voc}(s)$ is not necessarily vocal.*

For a sequence $s \in \Sigma^{*}$, let the map $\Sigma^{c} : \Sigma^{*} \longmapsto 2^{\Sigma_{c}}$.

$$\Sigma^{c}(s) = \{\sigma \mid \sigma \in \Sigma_{c}, \ \exists v, v' \in \Sigma^{*}, \ \ v\sigma v' = s\} \tag{2.9}$$

denote the controllable events in s. A silent path s is **controllable** if it includes at least one controllable event (i.e. $|\Sigma^{c}(s)| \geq 1$), and is **uncontrollable** otherwise (i.e. $|\Sigma^{c}(s)| = 0$). A controllable (uncontrollable) sequence is defined similarly.

To equip the high-level model G_{hi} with control structure, we need to know how the controllability is inherited at the high-level. The following property (if holds) clarifies the issue of the controllability of events in T.

Definition 2.31. *[59] The Moore automaton G_{lo} is said to satisfy the **Output-Control-Consistency (OCC)** property if every vocal state in it is reachable either by only controllable silent paths or by only uncontrollable silent paths.*

Alternatively, let T_c and T_{uc} denote the controllable and uncontrollable subsets of T. Suppose $s = s'r$ where $s' \in L_{voc}$ and $r \in L_{voc}(s')$ and define the

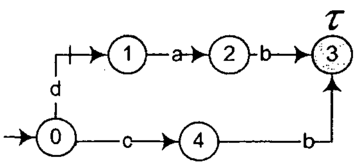


Figure 2.5: OCC property is violated at state 3; silent paths *d.a.b* and *c.b* which reach the vocal state 3 and generate the same output τ are respectively controllable and uncontrollable

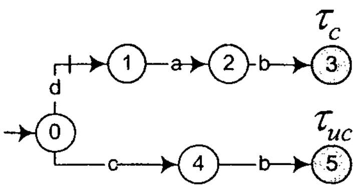


Figure 2.6: OCC property holds by splitting state 3 and generating controllable (τ_c) and uncontrollable (τ_{uc}) copies of the output τ.

maps $\chi^c : L_{voc} - \{\epsilon\} \longmapsto \{red, green\}$ according to

$$\chi^c(s) = \begin{cases} red & \hat{\omega}(s) \in T_c \\ green & \hat{\omega}(s) \in T_{uc} \end{cases} \tag{2.10}$$

and $color^c : L_{voc} - \{\epsilon\} \longmapsto \{red, green\}$ as follows

$$color^c(s) = \begin{cases} red & \Sigma^c(r) \neq \emptyset \\ green & \Sigma^c(r) = \emptyset. \end{cases} \tag{2.11}$$

The map χ^c shows if the assigned label to the state reached by s belongs to the controllable set T_c or the uncontrollable set T_{uc}. On the other hand, the map $color^c$ determines if the last silent segment of s is controllable or uncontrollable. The OCC property can therefore be rephrased as follows.

Definition 2.32. *The Moore automaton* $G_{lo} = (Q, \Sigma, T \cup \{\tau_o\}, \delta, \omega, q_o)$ *is OCC if for all* $s \in L_{voc} - \{\epsilon\}$

$$\chi^c(s) = color^c(s). \tag{2.12}$$

If the OCC property does not hold, using the OCC algorithm in [59], G_{lo} can be modified to satisfy the OCC property. Fig. 2.5 shows an example where the OCC property is violated. There, event d is controllable. Thus controllable silent path $d.a.b$ and uncontrollable silent path $c.b$ generate the same output τ which implies a violation of the OCC property. Fig. 2.6 shows a possible modification by OCC algorithm where $\tau_c \in T_c$ and $\tau_{uc} \in T_{uc}$. It can be seen that each output is chosen in Fig. 2.6 to match the controllability of its corresponding silent path. This enables us to have a well-defined high-level model for the plant. If the OCC property holds in the system, the high-level event alphabet T can be partitioned as $T = T_c \dot{\cup} T_{uc}$. Therefore, a high-level control structure can be synthesized for the system. Let S_{hi} be a solution of supervisory control problem for high-level model G_{hi} and legal behavior E_{hi}. The high-level supervisor $S_{hi} : L(G_{hi}) \times T \longmapsto \{0, 1\}$ is implemented through a *disabled-event map* $\Delta_{hi} : L(G_{hi}) \longmapsto 2^{T_c}$ as follow. For every $t \in L(G_{hi})$ and $\tau \in T$, we have

$$S_{hi}(t, \tau) = \begin{cases} 0, & \text{if } \tau \in \Delta_{hi}(t); \\ 1, & \text{otherwise.} \end{cases}$$

where 0 and 1 respectively indicate the disablement and enablement of τ after t. A low-level version of *disabled-event map* Δ_{hi} is defined by $\Delta_{lo} : \Sigma^* \times T \longrightarrow 2^{\Sigma_c}$ according to

$$\Delta_{lo}(s, t) = \{\sigma \in \Sigma_c | (\exists s' \in \Sigma_{uc}^*) s\sigma s' \in L(G_{lo}) \ \textit{and} \ \hat{\omega}(s\sigma s') \in \Delta_{hi}(t) \\ \textit{and} \ \forall s'' < s', \hat{\omega}(s\sigma s'') = \tau_\circ\}. \quad (2.13)$$

Finally, the control at the low-level is implemented through a low-level supervisor $S_{lo} : L(G_{lo}) \times \Sigma \longmapsto \{0, 1\}$ which is defined similar to S_{hi}.

$$S_{lo}(s, \sigma) = \begin{cases} 0, & \text{if } \sigma \in \Delta_{lo}(s); \\ 1, & \text{otherwise.} \end{cases} \quad (2.14)$$

Therefore a virtual command flow from high-level to low-level can be considered as

$$S_{hi} \Longleftarrow \Delta_{hi} \Longleftrightarrow \Delta_{lo} \Longrightarrow S_{lo}.$$

Now let $E_{hi} \subseteq L(G_{hi})$ the legal language at the high-level be controllable. S_{hi} can be synthesized so that $L(S_{hi}/G_{hi}) = E_{hi}$. The high-level commands are translated and implemented at the low-level by S_{lo} as explained before. This yields a system under supervision S_{lo}/G_{lo}. Let $E_{lo} = \theta^{-1}(E_{hi}) \subseteq L(G_{lo})$ be the translation of E_{hi} at the low-level. E_{lo} is interpreted as the low-level legal language. Theorem 2.33 states the relation between the language of the system under supervision $L(S_{lo}/G_{lo})$ and the low-level legal language E_{lo}.

Theorem 2.33. *[59] Assuming the plant model G_{lo} is OCC, we have $L(S_{lo}/G_{lo})$ $= E_{lo}^{\uparrow}$ where $E_{lo}^{\uparrow}$ is the supremal controllable sublanguage of E_{lo} w.r.t $L(G_{lo})$ and Σ_{uc}.*

Theorem 2.33 states that the language the system under supervision S_{lo}/G_{lo} generates is in fact the supremal controllable sublanguage of E_{lo}. However, the relation between the system under supervision at the high-level and that at the low-level is yet to be established. It is expected that the high-level controlled language E_{hi} (the high-level expectation) be recovered through the abstraction $\theta(E_{lo}^{\uparrow})$. However, it turns out that in cases we have $\theta(E_{lo}^{\uparrow}) \subset E_{hi}$. The strict inclusion would imply the occurrence of unintended disablement in the system. This unintentional disablement is addressed in the following.

Definition 2.34. *[59, 25] Consider two vocal nodes n_1 and n_2 with different controllable outputs in the reachability tree of G_{lo} which are reached by respectively silent paths $s_1\sigma s_2 s_3$ and $s_1\sigma s_2 s_4$ that share an initial segment $s_1\sigma$. The nodes n_1 and n_2 are said to be* ***partners*** *if their corresponding silent paths start either at the root state or at the same vocal state, $\sigma \in \Sigma_c$, $s_2 \in \Sigma_{uc}^*$ and*

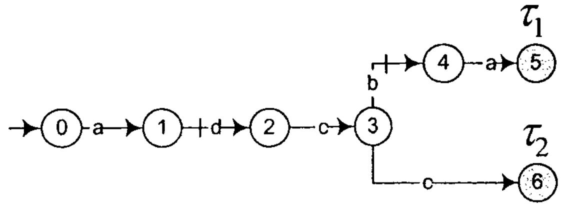


Figure 2.7: Nodes 5 and 6 are partners and thus disabling τ_1 and τ_2 are not independent from each other.

at least one of the strings s_3 and s_4 belongs to Σ_{uc}^*.

Definition 2.35. *[59] A plant model G_{lo} is said to be **Strictly-Output-Control-Consistent (SOCC)** if (i) it is OCC and (ii) no two vocal nodes with different controllable outputs are partners in it.*

Figure 2.7 shows a model which is not SOCC; nodes 5 and 6 are partners there. In Figure 2.7 if the high-level event τ_2 is to be disabled, the low-level event d should be disabled as a result of which the high-level event τ_1 will also be disabled. This unintentional disablement would not occur if node 2 or 3 were vocalized. The SOCC algorithm developed in [59] can be applied to modify the system to render it SOCC. Therefore, we assume G_{lo} is SOCC and thus no unintentional disablement would occur at the low-level. This implies E_{hi} can be recovered as $\theta(L(S_{lo}/G_{lo}))$ in the system under supervision.

Definition 2.36. *[59] We say hierarchical consistency holds in the system if $\theta(L(S_{lo}/G_{lo})) = E_{hi}$.*

Theorem 2.37 summarizes the result.

Theorem 2.37. *[59] Assume that the low-level model G_{lo} is SOCC. Also let θ, E_{hi}, E_{lo} and S_{lo} be as defined previously. Then we have $\theta(L(S_{lo}/G_{lo})) = E_{hi}$.*

Theorem 2.37 implies that if the system is SOCC, the desired high-level controllable specification E_{hi} can be recovered through the application of low-level supervisor and thus the system is hierarchically consistent.

2.6 Conclusion

In this chapter we reviewed background theory which we use in this thesis. We reviewed results from algebra, automata theory and supervisory control of discrete event systems. In particular, robust supervisory control of DES was discussed as a technique for dealing with model uncertainty or changes of system dynamics. Also, hierarchical supervisory control of DES was discussed as a framework in which supervision can be done at higher-level of abstraction resulting in a more transparent and partially less complex solution.

Chapter 3

Hierarchical Supervisory Control under Partial Observation

This chapter extends the hierarchical framework of Zhong-Wonham [59] to the case of control under partial observation. The hierarchical framework of Zhong-Wonham [59] considers a full observation case. Here we have relaxed this assumption about observability of the events and extended the setup of [59] to the case of control under partial observation. Not surprisingly, the results we obtain for hierarchical supervisory control under partial observation reduce to their counterparts given in [59], for the case of full observation.

3.1 Problem Formulation and Reporting Map

Problem Formulation: In this chapter, we extend the hierarchical setup of [59] (see Section 2.5 for details) to the case of control under partial observation. Consider the Moore automaton $G = (Q, \Sigma, T \cup \{\tau_o\}, \delta, \omega, q_o, Q)$. It follows from the partial observation assumption that the event set of Σ can be partitioned as $\Sigma = \Sigma_o \dot{\cup} \Sigma_{uo}$ where Σ_o and Σ_{uo} are observable and unobservable event sets.

The output symbols in T generated by G_{lo} announce the completion of sequences which are important for high-level supervision. Because some of the low-level events are assumed unobservable, some of the output symbols become unobservable. Therefore, we should partition T into observable subset T_o and unobservable subset T_{uo}. Towards this end, we define a silent path **unobservable** if it contains only unobservable events; otherwise it will be **observable**. The following definition characterizes the high-level observable and unobservable event sets.

Definition 3.1. *Suppose $\tau \in T$ is the output generated from the sequence ss' with $s \in L_{voc}$ and $s' \in L_{voc}(s)$; i.e. $\tau = \hat{\omega}(ss')$. Then $\tau \in T_{uo}$ if $P_{lo}(s') = \epsilon$; otherwise $\tau \in T_o$.*

Suppose q is the state reached by ss' in Definition 3.1 ($q = \delta(q_o, ss')$). Then Definition 3.1 is well-defined if G_{lo} is Output-Observation-Consistent as defined below.

Definition 3.2. *A model G_{lo} is **Output-Observation-Consistent (OOC)** if every vocal state $q \in Q$ is reached either by only unobservable silent paths or by only observable silent paths.*

Note that an observable silent path may also contain low-level unobservable transitions. OOC property can be regarded as the dual of OCC property. If G_{lo} is not OOC, then an algorithm identical to the OCC algorithm with controllable (resp. uncontrollable) events replaced with observable (resp. unobservable) events can be used to modify G_{lo} to satisfy OOC. From now on, we assume G_{lo} is OOC and OCC. Therefore T can be partitioned according

$$\begin{tikzcd}
G_{hi} \arrow[r, "P_{hi}"] & \tilde{G}_{hi} \\
G_{lo} \arrow[u, "\theta"] \arrow[r, "P_{lo}"] & \tilde{G}_{lo} \arrow[u, dashed, "\tilde{\theta}"']
\end{tikzcd}$$

Figure 3.1: If $ker(P_{lo}) \leq ker(P_{hi} \circ \theta)$, then map $\tilde{\theta}$ exists.

to $T = T_o \dot{\cup} T_{uo}$. Let

$$P_{hi} : T^* \longmapsto T_o^*$$

denote the high-level natural projection. With the above arrangement, the generation of symbols in T_{uo} will be unobservable to the high-level supervisor. Therefore a reporting map that maps the low-level observation to high-level information must be a map $\tilde{\theta} : P_{lo}(L(G_{lo})) \longmapsto T_o^*$ such that for every $s \in L(G_{lo})$,

$$\tilde{\theta}(P_{lo}(s)) = P_{hi}(\theta(s))$$

or in other words,

$$\tilde{\theta} \circ P_{lo} = P_{hi} \circ \theta.$$

Lemma 2.1 implies such a map exists (and is unique) if and only if

$$ker(P_{lo}) \leq ker(P_{hi} \circ \theta). \tag{3.1}$$

In other words, $P_{hi} \circ \theta$ must factor through P_{lo}. We refer to (3.1) as the **Factorization Property**. This is shown as the commutative diagram in Fig. 3.1. Note that in the proposed hierarchy under partial observation, $\tilde{\theta}$ (not θ) is the reporting map which will be implemented.

As mentioned in Section 2.5, initially in a hierarchical supervisory control problem the output symbols T and the reporting map θ (and equivalently the output map ω) are chosen so that the occurrence of significant events or accomplishment of important tasks are captured in the high-level model and also reported to the high-level supervisor. If after the problem is set up, Factorization Property (3.1) turns out not to be satisfied, then it is possible to enhance θ (through vocalizing some of silent states and refining G_{lo}) to ensure Factorization Property (3.1). The details are given in Section 3.7 (Appendix) at the end of this chapter. Therefore, in the rest of this chapter we assume the reporting map is such that Factorization Property (3.1) is satisfied.

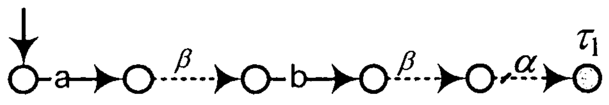
Definition 3.1 and Factorization Property (3.1) have implications for silent paths which are described in the following proposition.

Proposition 3.3. *Consider a nonempty vocal sequence $s \in L_{voc}$. If $\hat{\omega}(s) \in T_o$ (is observable), then $s = s_1\sigma$ for some $\sigma \in \Sigma_o$ and $s_1 \leq s$.*

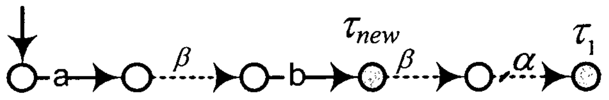
Proof: Assume for a sequence $s \in L_{voc}$, the output $\hat{\omega}(s) = \tau \in T_o$ is observable. There exist sequences $s' \in L_{voc}$ and $s'' \in L_{voc}(s')$ such that $s = s's''$. Now since $\hat{\omega}(s) \in T_o$ is observable, Definition 3.1 requires $P_{lo}(s'') \neq \epsilon$. This implies for some $\sigma \in \Sigma_o$ and sequences $s_1 \in \Sigma^*$ and $u \in \Sigma_{uo}^*$ we can write $s'' = s_1\sigma u$. Now suppose $s = s's_1\sigma u$ does not end in an observable event. That means $u \in \Sigma_{uo}^+$. Next note that $P_{lo}(s's_1\sigma u) = P_{lo}(s's_1\sigma)$. However, $P_{hi}(\theta(s's_1\sigma u)) = P_{hi}(\theta(s's_1\sigma)\tau) = P_{hi}(\theta(s's_1\sigma))\tau \neq P_{hi}(\theta(s's_1\sigma))$ which violates (3.1). Therefore $u = \epsilon$ and $s = s's_1\sigma \in \Sigma^*\Sigma_o$. □

Proposition 3.3 states that if a silent path generates a high-level *observable* event, the last event of that silent path must be *observable*. In Example 3.4 we show how satisfying Factorization Property can affect the vocalization, specifically, holding Proposition 3.3.

Example 3.4. *Fig. 3.2 shows two models in which $\Sigma = \{a, b, \alpha, \beta\}$ is the set of low-level events, $\Sigma_o = \{a, b\}$ is the set of observable events and the dashed*



(a): G_1



(b): G_2

Figure 3.2: (a) Factorization Property (3.1) does not hold and Proposition 3.3 is not ensured; (b) Factorization Property (3.1) and Proposition 3.3 both hold.

lines denote the unobservable events. Let $s_1 = a.\beta.b$, $s_2 = a.\beta.b.\beta.\alpha$ *and* $\boldsymbol{S} = P_{lo}(s_1) = P_{lo}(s_2) = a.b$. *In model* G_1 *in Fig. 3.2.(a),* $T = \{\tau_1\}$ *is the set of high-level events and we have* $T_o = T$. *Model* G_1 *does not satisfy Factorization Property (3.1) since we have* $\boldsymbol{S} = P_{lo}(s_1) = P_{lo}(s_2)$, $\theta(s_1) = \epsilon$, $\theta(s_2) = \tau_1$ *and* $P_{hi}(\tau_1) \neq \epsilon$. *While* $\omega(s_2) = \tau_1$ *is considered an observable high-level event, the last event of* s_2 *is not observable. That means Proposition 3.3 does not hold in Fig. 3.2.(a). On the other hand, model* G_2 *in Fig. 3.2.(b) illustrates model* G_1 *with a modified reporting map. Model* G_2 *vocalization in Fig. 3.2.(b) implies* $T = \{\tau_{new}, \tau_1\}$ *is the updated high-level event set in which* $T_o = \{T_{new}\}$ *is the set of observable events. Note that* τ_1 *is considered an unobservable high-level event here. Model* G_2 *satisfies Factorization Property (3.1) where we have* $P_{hi}(\theta(s_1)) = P_{hi}(\tau_{new}) = P_{hi}(\tau_{new}.\tau_1) = P_{hi}(\theta(s_2))$. *Therefore, Proposition 3.3 should hold as it is also the case in Fig. 3.2.(b). Note that, in Fig. 3.2.(b),* $b \in \Sigma_o$ *is the last event of* s_1 *where we have* $\theta(s_1) = \tau_{new} \in T_o$.

3.2 Vocalizing Observer Automaton

Observer automaton $\tilde{G}_{lo} = (\tilde{Q}, \Sigma_o, \tilde{\delta}, \tilde{q}_o)$ plays an important role in establishing hierarchical consistency under partial observation. In this section we show how $\tilde{G}_{lo}$ can be vocalized.

Definition 3.5. *Consider an automaton $G = (Q, \Sigma, \delta, q_o)$. For sequences $s_1, s_2 \in L(G)$ we say $s_1 \equiv s_2 (mod\, G)$ if and only if $\delta(q_o, s_1) = \delta(q_o, s_2) = q$ for some $q \in Q$.*

The binary relation $\equiv$ (mod G) is an equivalence relation on $L(G)$ and partitions $L(G)$ into $|Q|$ equivalence classes.

Lemma 3.6. *Let $\tilde{G}_{lo} = (\tilde{Q}, \Sigma_o, \tilde{\delta}, \tilde{q}_o)$ be the observer automaton for G_{lo}. For every two sequences $\boldsymbol{S}, \boldsymbol{S}' \in L(\tilde{G}_{lo})$ define $\mathfrak{S} = \{s \mid s \in P_{lo}^{-1}(\boldsymbol{S}) \cap L(G_{lo})\}$ and $\mathfrak{S}' = \{s \mid s \in P_{lo}^{-1}(\boldsymbol{S}') \cap L(G_{lo})\}$. Then we have $\boldsymbol{S} \equiv \boldsymbol{S}' (mod\, \tilde{G}_{lo})$ if and only if $(\forall s \in \mathfrak{S}, \ \exists s' \in \mathfrak{S}' : \quad s \equiv s'(mod\, G_{lo}))$ and $(\forall s' \in \mathfrak{S}', \ \exists s \in \mathfrak{S} : \quad s \equiv s'(mod\, G_{lo}))$.*

Proof: The proof is not difficult and omitted for brevity. □

Two subsets of $\tilde{q} \in \tilde{Q}$ (that are not necessarily disjoint) are defined in the following.

Definition 3.7. *Let $\tilde{q} = \tilde{\delta}(\tilde{q}_o, \boldsymbol{S})$ be a state of $\tilde{G}_{lo}$ with $\boldsymbol{S} \in \Sigma_o^+$. We refer to the set $in_{\boldsymbol{S}}(\tilde{q}) = \{q \mid \exists s' \in \Sigma^*, \sigma \in \Sigma_o : \ \boldsymbol{S} = P_{lo}(s'\sigma)$ and $q = \delta(q_o, s'\sigma)\}$ as the* ***incoming subset*** *of state $\tilde{q}$ w.r.t the sequence $\boldsymbol{S}$. We also denote the union of incoming subsets of $\tilde{q}$ as $in(\tilde{q}) = \{q \mid q \in \tilde{q}$ and $(\exists \boldsymbol{S} \in \Sigma_o^+ : q \in in_{\boldsymbol{S}}(\tilde{q}))\}$.*

The set $in(\tilde{q})$ consists of those states in $\tilde{q}$ that can be reached by sequences $s \in L(G_{lo})$ whose last events are observable.

Definition 3.8. *Let $\tilde{q} = \tilde{\delta}(\tilde{q}_o, \boldsymbol{S})$ be a state of $\tilde{G}_{lo}$ and $\sigma \in \Sigma_o$ be an observable event. We refer to the set $out_\sigma(\tilde{q}) = \{q \in \ \tilde{q} \mid \ \delta(q, \sigma)!\}$ as the* ***outgoing***

subset *of $\tilde{q}$ w.r.t σ. We also denote the union of outgoing subsets of $\tilde{q}$ as $out(\tilde{q}) = \{q | q \in \tilde{q}$ and $(\exists \sigma \in \Sigma_o : \; q \in out_\sigma(\tilde{q}))\}$.*

Thus $out(\tilde{q})$ contains the states $q \in \tilde{q}$ from which an observable transition exits.

Example 3.9. *Consider the model in Fig. 2.2. There, we have $in(\tilde{q}_2) = \{4, 5\}$, $out(\tilde{q}_2) = \{6, 7, 11\}$ and $in(\tilde{q}_8) = out(\tilde{q}_8) = \{16, 21\}$. As it can be observed here, incoming and outgoing subsets are not necessarily disjoint from each other.*

Next, we show how a state $\tilde{q} = \tilde{\delta}(\tilde{q}_o, \mathrm{S})$ of $\tilde{G}_{lo}$ can inherit the output of its incoming subset of states $in(\tilde{q})$.

Definition 3.10. *A state $\tilde{q} \in \tilde{Q}$ with $\tilde{q} \neq \tilde{q}_o$ is called **P-vocal** if it contains some vocal state (i.e. there exists $q \in \tilde{q}$ with $\omega(q) \neq \tau_o$).*

Lemma 3.11. *Let $\tilde{q} \neq \tilde{q}_o$ be a state of $\tilde{G}_{lo}$. Then $\tilde{q}$ is P-vocal if and only if for every $\boldsymbol{S} \in \Sigma_o^+$ that $\tilde{q} = \tilde{\delta}(\tilde{q}_o, \boldsymbol{S})$, there exists a state $q \in in_{\boldsymbol{S}}(\tilde{q})$ with $\omega(q) \neq \tau_o$.*

***Proof*:** (If) Since $\tilde{q}$ is P-vocal, there exists $p \in \tilde{q}$ with $\omega(p) \neq \tau_o$. For every $\mathrm{S} \in \Sigma_o$ that $\tilde{q} = \delta(\tilde{q}_o, \mathrm{S})$, there exist $s_1 \in \Sigma^*$, $\sigma \in \Sigma_o$ and $s' \in \Sigma_{uo}^*$ such that $P_{lo}(s_1 \sigma s') = \mathrm{S}$ and $p = \delta(q_o, s_1 \sigma s')$. If $s' = \epsilon$, $p \in in_{\mathrm{S}}(\tilde{q})$ and the lemma holds. Suppose $s' \neq \epsilon$. Let $q = \delta(q_o, s_1 \sigma)$. Clearly we have $q \in in_{\mathrm{S}}(\tilde{q})$. On the path from q to p, using the sequence s', let q_1 be the first vocal state reached with $q_1 = \delta(q, s'')$, $s'' \leq s'$. Since $s'' \in \Sigma_{uo}^+$, by Proposition 3.3, $\omega(q_1) \in T_{uo}$. Now if $\omega(q) = \tau_o$ then the sequence $\sigma s''$ which includes an observable event has resulted in the generation of an unobservable output $\omega(q_1)$ which violates Definition 3.3. Therefore $\omega(q) \neq \tau_o$.

(Only if) Follows from Definition 3.10. □

Next, we extend the results of the previous lemma to show that all of the incoming states of a vocal state $\tilde{q}$ are vocal and furthermore they generate the same observable output.

Proposition 3.12. *Let $\tilde{q} \in \tilde{Q}$ ($\tilde{q} \neq \tilde{q}_o$) be a P-vocal state in $\tilde{G}_{lo}$. Then there exists $\tau \in T_o$ such that for all $q \in in(\tilde{q})$, $\omega(q) = \tau$.*

Proof: Consider $q_1, q_2 \in in(\tilde{q})$. Suppose $q_i = \delta(q_o, s_i)$ for some sequences $s_i \in L(G_{lo})$ ($i = 1, 2$). Let $\hat{\omega}(s_i) = \tau_i$ and $\mathbf{S}_i = P_{lo}(s_i)$ ($i = 1, 2$). We distinguish the following two cases:

Case 1: $\mathbf{S}_1 = \mathbf{S}_2$. Let $s \in P_{lo}^{-1}(\mathbf{S}_1) \cap L(G_{lo})$ be the sequence in Lemma 3.11 for which we have $q = \delta(q_o, s)$, $q \in in_{\mathbf{S}_1}(\tilde{q})$ and $\hat{\omega}(s) \neq \tau_o$. Let $\hat{\omega}(s) = \boldsymbol{\tau}$. Definition 3.1 implies $\boldsymbol{\tau} \in T_o$ is observable. Since $P_{lo}(s) = P_{lo}(s_i) = \mathbf{S}_1$ ($i = 1, 2$), Factorization Property (3.1) implies that for some $t \in T_o^+$ we have

$$t = P_{hi}(\theta(s)) = P_{hi}(\theta(s_i)). \tag{3.2}$$

On the other hand, since $q, q_i \in in_{\mathbf{S}_1}(\tilde{q})$, for some $\sigma \in \Sigma_o$ and $s', s'_i \in L(G_{lo})$ we have $s = s'\sigma$ and $s_i = s'_i\sigma$ where $P_{lo}(s') = P_{lo}(s'_i)$. By Factorization Property (3.1), $P_{hi}(\theta(s')) = P_{hi}(\theta(s'_i))$. Now it follows from (3.2) that $P_{hi}(\theta(s')\boldsymbol{\tau}) = P_{hi}(\theta(s'_i)\tau_i)$ and since $\boldsymbol{\tau} \in T_o$, $P_{hi}(\theta(s'))\boldsymbol{\tau} = P_{hi}(\theta(s'_i))P_{hi}(\tau_i)$. But $P_{hi}(\theta(s')) = P_{hi}(\theta(s'_i)$. Therefore $\boldsymbol{\tau} = \tau_i \in T_o$. This proves that for every state $q_i \in in_{\mathbf{S}_1}(\tilde{q})$ we have $\omega(q_i) = \boldsymbol{\tau} \in T_o$.

Case 2: $\mathbf{S}_1 \neq \mathbf{S}_2$. Lemma 3.6 implies for every $r \in P_{lo}^{-1}(\mathbf{S}_1) \cap L(G_{lo})$, there exists $r' \in P_{lo}^{-1}(\mathbf{S}_2) \cap L(G_{lo})$ such that $r \equiv r' (\mathrm{mod}\ G_{lo})$. Especially for $r = s_1$ and some $r' \in P_{lo}^{-1}(\mathbf{S}_2) \cap L(G_{lo})$ we have $s_1 \equiv r'$ (mod G_{lo}). Therefore we have

$$\hat{\omega}(s_1) = \hat{\omega}(r'). \tag{3.3}$$

In the course of proving case 1 we showed $\tau_i = \hat{\omega}(s_i) \in T_o$. Since $s_1 \equiv$

r' (mod G_{lo}) and $q_1 \in in(\tilde{q})$, $\delta(q_o, r') \in in(\tilde{q})$. Therefore from case 1 we have

$$\hat{\omega}(s_2) = \hat{\omega}(r'). \tag{3.4}$$

(3.3) and (3.4) imply $\hat{\omega}(s_2) = \hat{\omega}(s_1)$ or equivalently $\omega(q_1) = \omega(q_2)$. This completes the proof. □

3.3 Output Map Properties

The uniqueness of the outputs of $in(\tilde{q})$, shown in Proposition 3.12, justifies the definition of an output map $\tilde{\omega} : L(\tilde{G}_{lo}) \longmapsto T_\circ \cup \{\tau_\circ\}$ for $\tilde{G}_{lo}$.

Definition 3.13. *We define $\tilde{\omega} : L(\tilde{G}_{lo}) \longmapsto T_\circ \cup \{\tau_\circ\}$ as follows. Let $\boldsymbol{S} \in L(\tilde{G}_{lo})$, $\tilde{q} = \tilde{\delta}(\tilde{q}_\circ, \boldsymbol{S})$ and $I = \{s \in L(G_{lo}) |\ \delta(q_\circ, s) \in in(\tilde{q})\}$. Then $\tilde{\omega}(\boldsymbol{S}) = \tau_\circ$ if $\boldsymbol{S} = \epsilon$ and*

$$\tilde{\omega}(\boldsymbol{S}) = \hat{\omega}(s) \qquad \textit{for any } s \in I. \tag{3.5}$$

□

Proposition 3.12 implies that the output map $\tilde{\omega}$ is well-defined and every state $\tilde{q}$ is either silent, $\tilde{\omega}(\mathbf{S}) = \tau_o$, or P-vocal with an unique output $\tilde{\omega}(\mathbf{S}) = \tau \in T_o$. Furthermore the observer model $\tilde{G}_{lo}$ can be equipped with an output map $\varpi : \tilde{Q} \longmapsto T_\circ \cup \{\tau_\circ\}$ where for every $\tilde{q} \in \tilde{Q}$ that $\tilde{q} = \tilde{\delta}(\tilde{q}_\circ, \mathbf{S})$, we have $\varpi(\tilde{q}) = \tilde{\omega}(\mathbf{S})$. Once again Proposition 3.12 ensures ϖ is well-defined. So finally $\tilde{G}_{lo} = (\tilde{Q}, \Sigma_o, T_o \cup \{\tau_o\}, \tilde{\delta}, \varpi, \tilde{q}_o)$ becomes a Moore automaton.

Theorem 3.14. *The reporting map $\tilde{\theta} : L(\tilde{G}_{lo}) \longmapsto T_o^*$, defined by $\tilde{\theta} \circ P_{lo} = P_{hi} \circ \theta$, has the property that for every $\boldsymbol{S} \in L(\tilde{G}_{lo})$, $\tilde{\theta}(\boldsymbol{S}) = \epsilon$ if $\boldsymbol{S} = \epsilon$, and $\tilde{\theta}(\boldsymbol{S}) = \tilde{\theta}(\boldsymbol{S}')\tilde{\omega}(\boldsymbol{S}'\sigma)$ if $\boldsymbol{S} = \boldsymbol{S}'\sigma$ for some $\boldsymbol{S}' \in \Sigma_\circ^*$ and $\sigma \in \Sigma_\circ$.*

Proof: First note that for $\mathbf{S} \in L(\tilde{G}_{lo})$, $\tilde{\theta}(\mathbf{S}) = P_{hi}(\theta(s))$ for every $s \in P_{lo}^{-1}(\mathbf{S}) \cap L(G_{lo})$. If $\mathbf{S} = \epsilon$, then $\epsilon \in P_{lo}^{-1}(\mathbf{S}) \cap L(G_{lo})$ and hence

$$\tilde{\theta}(\mathbf{S}) = P_{hi}(\theta(\epsilon)) = P_{hi}(\epsilon) = \epsilon. \tag{3.6}$$

Now, consider $\mathbf{S} = \mathbf{S}'\sigma$ with $\mathbf{S}' \in \Sigma_o^*$ and $\sigma \in \Sigma_o$. Note that for every $s \in P_{lo}^{-1}(\mathbf{S}) \cap L(G_{lo}) = P_{lo}^{-1}(\mathbf{S}'\sigma) \cap L(G_{lo})$ there exists $s' \in P_{lo}^{-1}(\mathbf{S}') \cap L(G_{lo})$ such that $s = s'\sigma$. Therefore we have

$$\begin{aligned} \tilde{\theta}(\mathbf{S}) = P_{hi}(\theta(s)) &= P_{hi}(\theta(s'\sigma)) \\ &= P_{hi}(\theta(s')\hat{\omega}(s'\sigma)) \\ &= P_{hi}(\theta(s'))P_{hi}(\hat{\omega}(s'\sigma)). \end{aligned} \tag{3.7}$$

The definitions of $\tilde{\theta}$ and $\tilde{\omega}$, (3.7) imply

$$P_{hi}(\theta(s'))P_{hi}(\hat{\omega}(s'\sigma)) = \tilde{\theta}(\mathbf{S}')P_{hi}(\tilde{\omega}(\mathbf{S}'\sigma)). \tag{3.8}$$

Considering the fact that $\tilde{\omega}(\mathbf{S}) \in T_o$, we will have

$$\tilde{\theta}(\mathbf{S}')P_{hi}(\tilde{\omega}(\mathbf{S}'\sigma)) = \tilde{\theta}(\mathbf{S}')\tilde{\omega}(\mathbf{S}'\sigma) \tag{3.9}$$

Therefore (3.7), (3.8) and (3.9) yield $\tilde{\theta}(\mathbf{S}) = \tilde{\theta}(\mathbf{S}')\tilde{\omega}(\mathbf{S}'\sigma)$. □

The following definitions will be useful in the next section in the study of hierarchical consistency.

Definition 3.15. *Let $\tilde{q}$ be a state in $\tilde{G}_{lo}$ and $\boldsymbol{S} \in \Sigma_o^+$ such that $\tilde{\delta}(\tilde{q}, \boldsymbol{S})!$. We say $\boldsymbol{S}$ is* $\mathbf{P}_{\mathbf{lo}}$***-controllable*** *from $\tilde{q}$ if for every $q \in out(\tilde{q})$ and every $s \in P_{lo}^{-1}(\boldsymbol{S})$ that $\delta(q, s)!$, we have $|\Sigma^c(s)| \geq 1$.*

Recall from (2.9) that $\Sigma^c(s)$ is the set of controllable events of the string

s. Definition 3.15 states that given a transition $\tilde{q} \xrightarrow{\mathbf{S}}$, the sequence $\mathbf{S}$ is P_{lo}-controllable from $\tilde{q}$ if every sequence $s \in P_{lo}^{-1}(\mathbf{S})$ exiting from an outgoing state $q \in \tilde{q}$, is controllable (i.e. contains at least a controllable events).

Definition 3.16. *Let $\tilde{q}$ be a state in $\tilde{G}_{lo}$ and $\boldsymbol{S} \in \Sigma_o^+$ such that $\tilde{\delta}(\tilde{q}, \boldsymbol{S})!$. We say $\boldsymbol{S}$ is $\mathbf{P_{lo}}$-**uncontrollable** from $\tilde{q}$ if there exists $q \in out(\tilde{q})$ and $s \in P_{lo}^{-1}(\boldsymbol{S}) \cap \Sigma_{uc}^+$ with $\delta(q, s)!$.*

Definition 3.16 states that given a transition $\tilde{q} \xrightarrow{\mathbf{S}}$, the sequence $\mathbf{S}$ is P_{lo}-uncontrollable from $\tilde{q}$ if there exists at least one sequence $s \in P_{lo}^{-1}(\mathbf{S})$ which exits from an outgoing state $q \in out(\tilde{q})$ and consists of uncontrollable events only.

Definition 3.17. *An observable sequence $\boldsymbol{S} \in \Sigma_o^+$ in $\tilde{G}_{lo}$ which connects the root state $\tilde{q}_\circ$ to a P-vocal state or two immediate P-vocal states to each other is called a $\mathbf{P_{lo}}$-**silent** path.*

Therefore a P_{lo}-silent path $\mathbf{S}$, from a P-vocal state $\tilde{p}$ to a P-vocal state $\tilde{q}$, is P_{lo}-controllable if all the sequences $s \in P_{lo}^{-1}(\mathbf{S})$ which connect an outgoing state of $\tilde{p}$ to an incoming state of $\tilde{q}$ are controllable (i.e. contain at least one controllable event) and is P_{lo}-uncontrollable otherwise.

Example 3.18. *Consider the state $\tilde{q}_2 = \tilde{\delta}(\tilde{q}_\circ, a.b)$ of $\tilde{G}_{lo}$ in Fig. 2.3 and let $\boldsymbol{S}_1 = c.d$ and $\boldsymbol{S}_2 = b.a$ be two sequences which start from $\tilde{q}_2$. Fig. 2.2 shows that $out(\tilde{q}_2) = \{6, 7, 11\}$. It is observed that $\boldsymbol{S}_1$ is P_{lo}-controllable from $\tilde{q}_2$ since the sequence $c.\sigma.d$ from state 6 is the only transition out of $\{6, 7, 11\}$ that has the projection $c.d$ and we have $\Sigma^c(c.\sigma.d) = \{c\} \neq \emptyset$. On the other hand, $\boldsymbol{S}_2$ is P_{lo}-uncontrollable from $\tilde{q}_2$ since for $7 \in out(\tilde{q}_2)$ we have $\delta(7,\ b.d)!$ and $\Sigma^c(b.d) = \emptyset$.*

3.4 Supervision Implementation At The Low-level

Let G_{lo} describe the low-level model and $\tilde{G}_{lo}$ be the observer automaton for G_{lo}. Let G_{hi} be the abstracted model at the high-level with $L(G_{hi}) = \theta(L(G_{lo}))$ and E_{hi} be a high-level controllable specification which is also observable w.r.t. G_{hi}. A virtual high-level supervisor S_{hi} can be obtained by solving a supervisory control problem under partial observation for the pair (G_{hi}, E_{hi}) such that $L(S_{hi}/G_{hi}) = E_{hi}$. Let $E_{lo} = \theta^{-1}(E_{hi})$ be the low-level image of E_{hi}. Hierarchical supervisory control is performed by a low-level supervisor S_{lo} implementing a controllable and observable sublanguage of E_{lo} at the low-level with $L(S_{lo}/G_{lo}) \subseteq E_{lo}$. It is desired to achieve hierarchical consistency [59] under partial observation. Hierarchical consistency requires that $\theta(L(S_{lo}/G_{lo})) = E_{hi}$; i.e. high-level specification E_{hi} is recovered through the actual low-level supervisory action. In [59], hierarchical consistency is achieved by ensuring that in two silent paths which have a common prefix and generate different outputs, disablement of one silent path does not (unintentionally) disable the other silent path. In the case of control under partial observation, we have to consider look-alike sequences.

3.4.1 UUC Property

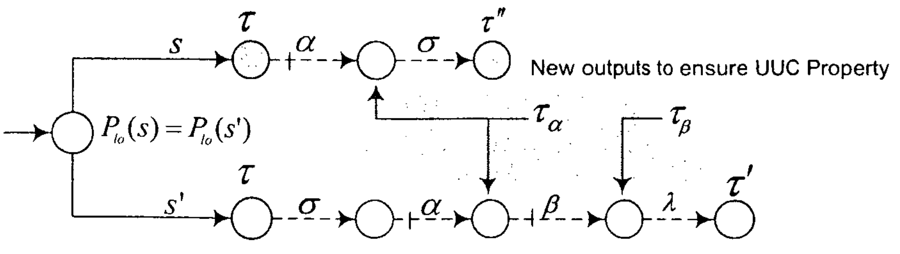


Figure 3.3: Motivational example for UUC property: Event α cannot be disabled independently.

Fig. 3.3 shows a model in which $\Sigma_c = \{\alpha, \beta\}$ and the dashed lines denote the low-level unobservable events Σ_{uo}. Also $T = \{\tau, \tau', \tau''\}$ is the set of high-level events and $T_c = \{\tau', \tau''\}$ the controllable subset. The system is prone to unintentional disablement if either of τ' or τ'' is to be disabled through disabling α. To resolve this issue, we include the information about the unobservable-controllable events in G_{hi}. Specifically, we propose the following property to be imposed.

Property 3.19. ***Unobservable-and-Uniquely-Controllable (UUC)*** *sile nt paths: For each $s \in L_{voc}$ and $s' \in L_{voc}(s)$ that we have $\hat{\omega}(ss') \in T_{uo} \cap T_c$, we should have $|\Sigma^c(s')| = 1$.*

If UUC is not satisfied, we can use the Single-Controllable-Event (SCE) Algorithm, discussed in the Appendix, to ensure it. Briefly, SCE Algorithm proceeds as follows. Define a new set of labels

$$T_c^{uo} = \{\tau_\sigma \mid \sigma \in \Sigma_{uo} \cap \Sigma_c\} \tag{3.10}$$

and add them to the set of high-level events. Also by definition we let $T_c^{uo} \subseteq T_{uo} \cap T_c$. To satisfy UUC, we assign output from the set T_c^{uo} to the destination node of every unobservable-controllable transition of every unobservable-controllable silent path (If the label of transition is σ, the assigned label will be $\tau_\sigma \in T_c^{uo}$). In this way, by choosing labels from T_c^{uo}, we include the information about the unobservable-controllable events in the high-level model G_{hi}. UUC and the modification performed in the above procedure ensure that if an unobservable silent path is controllable, then its controllability originates from only a single controllable event, say $\sigma \in \Sigma_c$, and furthermore the occurrence of σ is shown in G_{hi} with a unique symbol τ_σ. The result of applying the above procedure is shown in Fig. 3.3 where the new outputs are indicated with shaded area.

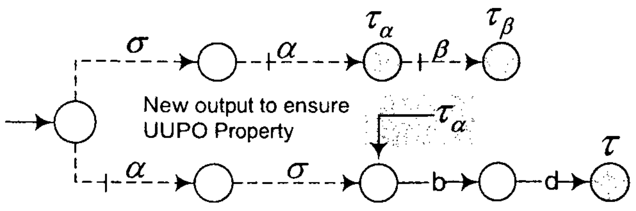


Figure 3.4: Motivational example for UUPO property: Event α cannot be disabled independently.

3.4.2 UUPO Property

Fig. 3.4 shows another model which satisfies UUC. Here $\Sigma_c = \{\alpha, \beta\}$ and as usual unobservable low-level events have been shown with dashed lines. $T = \{\tau_\alpha, \tau_\beta, \tau\}$ is the set of high-level events. It can be checked that the system is still prone to unintended disablement if either of τ_α or τ is to be disabled through disabling α. We propose Property 3.20 to be imposed as the remedy to this problem. For a path $r \in \Sigma^*$, let

$$\eta(r) = \{u \mid u \in \Sigma_{uo}^*,\ \exists w \in \Sigma_o^+\Sigma^* \cup \{\epsilon\} :\ r = uw\} \tag{3.11}$$

be the largest unobservable prefix of r.

Property 3.20. ***Unobservable-and-Uncontrollable-Prefixes-for-Obse rvable (UUPO)*** *silent paths: For every sequence $s \in L_{voc}$ and its silent path extension $r \in L_{voc}(s)$, it should be the case that*

$$P_{lo}(r) \neq \epsilon \ \ and\ \ \hat{\omega}(sr) \in T_c \Longrightarrow \Sigma^c(\eta(r)) = \emptyset. \tag{3.12}$$

Property UUPO means if an observable-controllable silent path starts with an unobservable segment, that segment should be uncontrollable. (3.12) can also be rewritten as

$$\Sigma^c(\eta(r)) \neq \emptyset \Longrightarrow P_{lo}(r) = \epsilon. \tag{3.13}$$

In Fig. 3.4, if the state reached by $\alpha\sigma$ is vocalized, then $\alpha\sigma$ will become an unobservable silent path and UUPO will be ensured. In general, by vocalizing some states and refining the structure of $\tilde{G}_{lo}$ UUC (and UUPO) can be ensured. The details are given in UET Algorithm in the Appendix.

If a P_{lo}-silent path is P_{lo}-controllable, then the output generated by the silent path is controllable. However, the reverse is not generally true. Proposition 3.21 discusses the condition under which the reverse is ensured. Let $\tilde{L}_{voc} = \{\mathbf{S} \in L(\tilde{G}_{lo}) \mid \mathbf{S} = \epsilon \text{ or } \tilde{\omega}(\mathbf{S}) \neq \tau_0\}$ contain the null sequence ϵ and the sequences which end in P-vocal states.

Proposition 3.21. *Assume UUPO holds in the system and let $\boldsymbol{S} \in \tilde{L}_{voc}$ be a P-vocal sequence in $\tilde{G}_{lo}$ and $R \in \Sigma_o^+$ be a P_{lo}-silent path following $\boldsymbol{S}$. In this case, if $\hat{\omega}(\boldsymbol{SR})$ is controllable ($\hat{\omega}(\boldsymbol{SR}) \in T_c$), then $\boldsymbol{R}$ is P_{lo}-controllable from $\tilde{q} = \tilde{\delta}(\tilde{q}_o, \boldsymbol{S})$.*

Proof: Direct result of Proposition 3.12 and property UUPO. □

Proposition 3.21 implies that a P_{lo}-silent path is P_{lo}-controllable if and only if the the output which is generated upon completion of that P_{lo}-silent path is controllable. A similar conclusion follows from Proposition 3.21 for P_{lo}-uncontrollable P_{lo}-silent paths. *In the rest of this thesis we assume the map θ is such that UUC and UUPO properties are satisfied.*

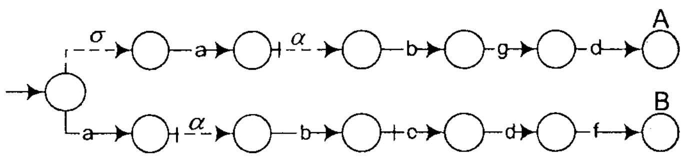


Figure 3.5: Motivational example for PO-SOCC property: Event α cannot be disabled independently.

3.4.3 PO-SOCC Property

Fig. 3.5 shows a model whose reporting map satisfies Factorization Property (3.1) as well as UUC and UUPO properties. There, $\Sigma = \{\text{a, b, c, d, f, g}, \alpha, \sigma\}$, $\Sigma_c = \{\alpha, \text{c}\}$ and the dashed lines denote the low-level unobservable events $\Sigma_{uo} = \{\alpha, \sigma\}$. $T = T_o = \{\text{A, B}\}$ is the set of high-level events. It can be verified that the system is prone to unintended disablement which occurs if for example event A is desired to be disabled at the high-level. In this case, event α is disabled after an observable event "a" is generated. If the low-level supervisor is feasible, this leads to unintended disablement of B.

For a more general discussion consider Fig. 3.6.(a). Fig. 3.6.(a) shows

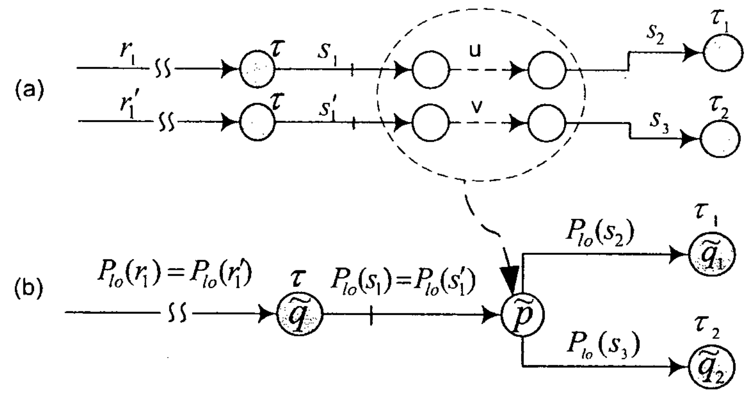


Figure 3.6: PO-SOCC property inspiration, (a): two branches in reachability tree of G_{lo}, (b): corresponding projected branches *with common segments* in reachability tree of $\tilde{G}_{lo}$.

two transitions $r_1 s_1 u s_2$, $r_1' s_1' v s_3 \in L(G_{lo})$ which generate high-level events $\tau_1, \tau_2 \in T_c$. We assume $P_{lo}(r_1) = P_{lo}(r_1')$, $P_{lo}(u) = P_{lo}(v) = \epsilon$, $P_{lo}(s_1) = P_{lo}(s_1') \neq \epsilon$ and $P_{lo}(s_2), P_{lo}(s_3) \neq \epsilon$. Fig. 3.6.(b) shows the corresponding projections $P_{lo}(r_1)P_{lo}(s_1 u)P_{lo}(s_2)$, $P_{lo}(r_1')P_{lo}(s_1' v)P_{lo}(s_3) \in L(\tilde{G}_{lo})$ share a common segment $P_{lo}(r_1)P_{lo}(s_1) = P_{lo}(r_1')P_{lo}(s_1') \in \Sigma_o^+$. Let $\tilde{q}$ be P-vocal and $P_{lo}(s_1 u)P_{lo}(s_2)$ and $P_{lo}(s_1' v)P_{lo}(s_3)$ be two P_{lo}-silent paths which start from $\tilde{q}$ and lead respectively to $\tilde{q}_1$ and $\tilde{q}_2$ with $\varpi(\tilde{q}_1) = \tau_1$ and $\varpi(\tilde{q}_2) = \tau_2$. Also let the state $\tilde{p}$ be the last state which is reached by the common segment $P_{lo}(r_1)P_{lo}(s_1u) = P_{lo}(r_1')P_{lo}(s_1'v)$. If $P_{lo}(s_2)$ or $P_{lo}(s_3)$ is P_{lo}-uncontrollable from $\tilde{p}$, disabling τ_1 or τ_2 might not be done independently from each other (This is comparable with partnership status in the full observation case). The PO-SOCC property (which will turn out to be a generalization of SOCC for the case of control under partial observation) is introduced in the following to prevent unintended disablement in these cases. First we define P-partners.

Definition 3.22. *Two P-vocal nodes n_1 and n_2 with different controllable outputs in the reachability tree of $\tilde{G}_{lo}$ are said to be* ***P-partners*** *if their P_{lo}-silent paths start either at the root node or at the same P-vocal node, share an initial segment $\boldsymbol{S}_1 \in \Sigma_o^+$ which is followed in turn by segments $\boldsymbol{S}_2 \in \Sigma_o^+$ and $\boldsymbol{S}_3 \in \Sigma_o^+$, where $\boldsymbol{S}_1\boldsymbol{S}_2$ and $\boldsymbol{S}_1\boldsymbol{S}_3$ are P_{lo}-controllable and at least one of $\boldsymbol{S}_2$ or $\boldsymbol{S}_3$ is P_{lo}-uncontrollable.* □

If two nodes in the reachability tree of $\tilde{G}_{lo}$ are P-partners, then disabling the high-level event associated with one of them might unintentionally disable the other one.

Definition 3.23. *A Moore automaton G_{lo} is* ***Partially-Observable-Strictly-Output-Control-Consistent (PO-SOCC)*** *if it is (i) OCC and OOC and (ii) no two P-vocal nodes with controllable outputs are P-partners in the reachability tree of its observer automaton $\tilde{G}_{lo}$.* □

Remark 3.24. *In Proposition 3.26 we show that PO-SOCC is stronger than SOCC. However, note that in the case of full observation, the PO-SOCC property reduces to SOCC (and properties UUC and UUPO hold trivially).*

Remark 3.25. *Similar to SOCC, if PO-SOCC does not hold in the system, by vocalizing new states and refining the structure of G_{lo} we can ensure PO-SOCC. The details are given in PO-SOCC Algorithm in the Appendix.*

Proposition 3.26. *If G_{lo} is UUC, UUPO and PO-SOCC, then it is SOCC.*

Proof: By contradiction suppose G_{lo} is not SOCC and thus there exist two vocal nodes n_1 and n_2 with controllable outputs that are partners in the reachability tree of G_{lo} (see Fig. 3.7.(a)) and there exist $s \in L_{voc}$, $s_1 \in \Sigma^*$, $\sigma \in \Sigma_c$ and $s_2 \in \Sigma^*$, $s_3, s_4 \in \Sigma^+$ of which $s_2 \in \Sigma_{uc}^*$ and s_3 or $s_4 \in \Sigma_{uc}^+$ and $\delta(q_o,\ ss_1\sigma s_2 s_3) = n_1$ and $\delta(q_o,\ ss_1\sigma s_2 s_4) = n_2$. First note that if $\hat{\omega}(n_1)$ is unobservable, $\hat{\omega}(n_2)$ is unobservable and vice versa. To see why, without loss of generality assume $\hat{\omega}(ss_1\sigma s_2 s_3) = \hat{\omega}(n_1) \in T_{uo}$ is unobservable which implies $\sigma \in \Sigma_{uo}$, $s_1, s_2, s_3 \in \Sigma_{uo}^*$ (by Definition 3.1). On the other hand, since $\sigma \in \Sigma_c$ is controllable, (3.13) implies $s_1\sigma s_2 s_4$ will be unobservable too. Furthermore, property UUC and vocalization rule in the SCE Procedure require that $\hat{\omega}(ss_1\sigma s_2 s_3) = \hat{\omega}(ss_1\sigma s_2 s_4) = \tau_\sigma$ are the same for $\sigma \in \Sigma_{uo} \cap \Sigma_c$. Therefore reaching n_1 or n_2 generates the same high-level event τ_σ and takes us to the same state in G_{hi} which implies n_1 and n_2 are not partners. Thus without loss of generality we assume both n_1 and n_2 have observable outputs.

Therefore assume $\hat{\omega}(ss_1\sigma s_2 s_3) = \tau \in T_o$ and $\hat{\omega}(ss_1\sigma s_2 s_4) = \tau' \in T_o$ are both

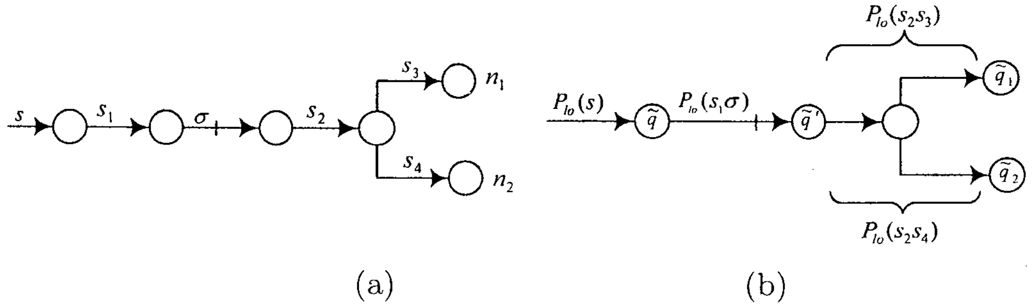


Figure 3.7: Proposition 3.26: if (a): states n_1 and n_2 are partners, (b): states $\tilde{q}_1$ and $\tilde{q}_2$ will be P-partners.

observable and let the reachability tree of observer automaton $\tilde{G}_{lo}$ be given in Fig. 3.7.(b). Also let $\mathbf{S} = P_{lo}(s)$, $\mathbf{S}_1 = P_{lo}(s_1\sigma s_2 s_3)$ and $\mathbf{S}_2 = P_{lo}(s_1\sigma s_2 s_4)$ and $\tilde{q}$, $\tilde{q}_1$ and $\tilde{q}_2$ be states in reachability tree of $\tilde{G}_{lo}$ that are respectively reached

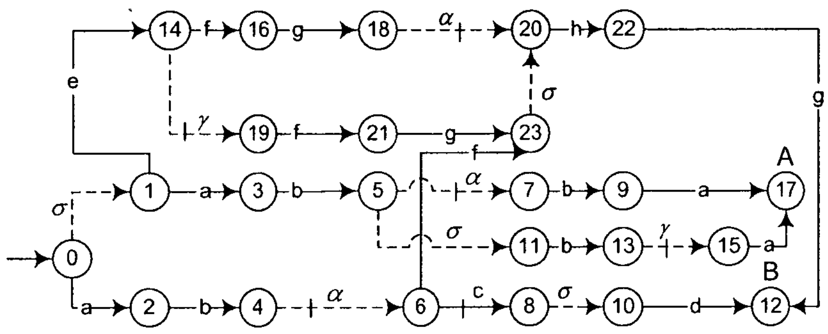


Figure 3.8: Automaton G_{lo} with unobservable transitions denoted by dashed lines and outputs A and B

by $\mathbf{S}$, $\mathbf{SS}_1$ and $\mathbf{SS}_2$. First note that $s \in L_{voc}$ which implies $\mathbf{S} \in \tilde{L}_{voc}$. Therefore $\tilde{q}$ is either the root state $\tilde{q}_o$ or a P-vocal state. Let $\tilde{q}'$ be the state which is reached by the sequence $\mathbf{S}P_{lo}(s_1\sigma)$. Next note that $\sigma \in \Sigma_c$ and since $\tau, \tau' \in T_c \cap T_o$, property UUPO requires $\Sigma^c(\eta(s_1\sigma)) = \emptyset$. This implies $P_{lo}(s_1\sigma) \neq \epsilon$ which means states $\tilde{q}$ and $\tilde{q}'$ are distinct. Furthermore, since $\tau, \tau' \in T_o$ are observable, by Proposition 3.3 we can see that $\delta(q_o, ss_1\sigma s_2 s_3) \in in(\tilde{q}_1)$ and $\delta(q_o, ss_1\sigma s_2 s_4) \in in(\tilde{q}_2)$. Therefore by (3.5) we have $\tilde{\omega}(\mathbf{SS}_1) = \tau \in T_c \cap T_o$ and $\tilde{\omega}(\mathbf{SS}_2) = \tau' \in T_c \cap T_o$. It follows from Proposition 3.21 that $\mathbf{S}_1$ and $\mathbf{S}_2$ are P_{lo}-controllable from $\tilde{q}$. On the other hand, by assumption, either $s_2 s_3 \in \Sigma_{uc}^+$ or $s_2 s_4 \in \Sigma_{uc}^+$ which implies either $P_{lo}(s_2 s_3)$ or $P_{lo}(s_2 s_4)$ is P_{lo}-uncontrollable from $\tilde{q}'$. If $\tau \neq \tau'$, the above facts imply $\tilde{q}_1$ and $\tilde{q}_2$ are P-partners in the reachability tree of $\tilde{G}_{lo}$ which contradicts the assumption that G_{lo} is PO-SOCC. Therefore G_{lo} is SOCC. $\square$

Example 3.27. *Consider the plant G_{lo} in Fig. 3.8. G_{lo} is SOCC since the sequences leading to states 12 and 17 (and thus generating controllable high-level events A and B) have the common initial segment of* $0 \xrightarrow{\sigma} 1$, *and all sequences from state 1 to states 12 and 17 are controllable. Now we observe disabling high-level event A also disables B since to disable A (following observable sequence $\sigma.a.b$), event α in state* 5 *must be disabled as a result of which*

α at state 4 will also be disabled leading to an unintended disablement of B. In fact, it can be verified that G_{lo} is not PO-SOCC. Specifically, in the reachability tree of $\tilde{G}_{lo}$ (see Fig. 2.3 for details), the nodes which are reached by sequences $\boldsymbol{R}_1 = a.b.b.a$ and $\boldsymbol{R}_2 = a.b.c.d$ will be P-partners. To see why, note that $\boldsymbol{R}_1$ and $\boldsymbol{R}_2$ are P_{lo}-controllable P_{lo}-silent paths which share a segment $\boldsymbol{S}_1 = a.b$ which can be extended to the P-vocal state $\tilde{q}_4$ and $\tilde{q}_6$ by a P_{lo}-uncontrollable sequence $\boldsymbol{S}_2 = b.a$ and a P_{lo}-controllable sequence $\boldsymbol{S}_3 = c.d$ respectively. $\boldsymbol{S}_2$ is P_{lo}-uncontrollable since there is a transition $17 = \delta(7, b.a)$ with $b.a \in \Sigma_{uc}^{+}$ in Fig. 3.8 and $7 \in out(\tilde{q}_2)$, the sequence $\boldsymbol{S}_2$ is P_{lo}-uncontrollable.

Next we show how G_{lo} can be modified so that it satisfies PO-SOCC (as well as the Factorization Property, UUC and UUPO properties). The change is done in two steps shown in Fig. 3.9 and 3.11. In Fig. 3.9, states 4 and 5 are vocalized to turn the P_{lo}-uncontrollable segment $\boldsymbol{S}_2$ of the P_{lo}-silent path $\boldsymbol{R}_1$ into a P_{lo}-silent path. Next, states 6 and 7 need to be vocalized to satisfy UUC property. However, it can be observed that of the two newly-formed silent paths $b.a$, starting from state 7, and $\sigma.b.\gamma.a$, starting from state 5, the former is uncontrollable but the latter is controllable. Therefore, OCC is violated here. If we split state 17, say into 17a and 17b, to have controllable and uncontrollable versions of A (as required by OCC algorithm), then the Factorization Property would be violated since the sequences leading to states 17a and 17b have the same natural projection and generate different outputs. As shown in Appendix, the remedy is to vocalize states 13 and 15 so that the information of the low-level unobservable-controllable transition γ from state 13 to 15 is included in G_{hi}. Following this, state 9 is also vocalized with the same output as 13 to ensure the Factorization Property. Finally state 12 is split into 12 and 27 to ensure OCC, as shown in Fig. 3.11. The model in Fig. 3.11 is PO-SOCC and satisfies the Factorization Property, UUC and UUPO properties.

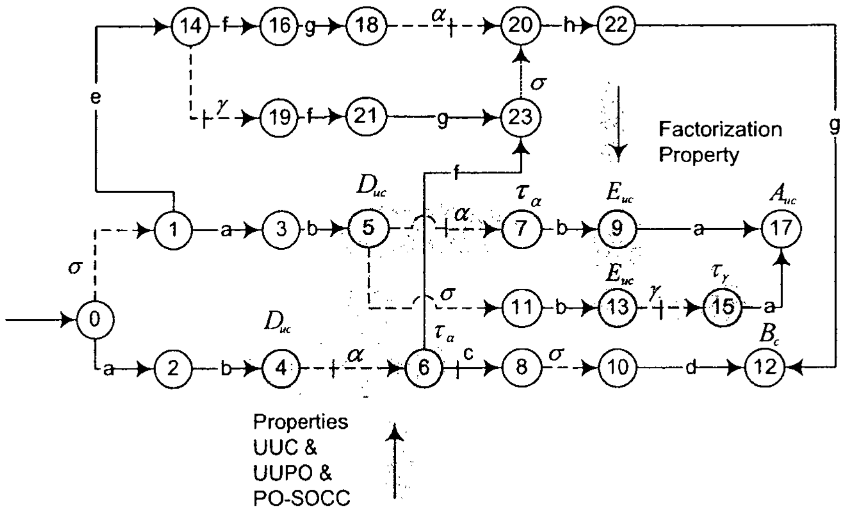


Figure 3.9: States enclosed in shaded area should be vocal so that G_{lo} is UUC, UUPO and PO-SOCC

Suppose the high-level specification E_{hi} is controllable and observable. The supremal controllable sublanguage of $E_{lo} = \theta^{-1}(E_{hi})$ which we denote by $E_{lo}^{\uparrow}$ would not necessarily be observable with respect to the low-level model G_{lo}. Below we show $E_{lo}^{\uparrow}$ can be guaranteed to be observable in a more relaxed sense which we call (G, P_{lo}, θ)-observability.

Let a map $act_\theta : L(G_{lo}) \longmapsto 2^T$ be defined as

$$act_\theta(s) = \begin{cases} \{\hat{\omega}(s)\}, & \text{if } s \in L_{voc}, \\ \{\tau \mid \exists u \in L_{voc}(s),\ \hat{\omega}(su) = \tau\}, & \text{otherwise.} \end{cases}$$

$act_\theta(s)$ returns the generated high-level event if s is vocal. Otherwise, it returns the set of next possible high-level events.

Definition 3.28. *A language $E \subseteq L(G)$ is said to be (G, P_{lo}, θ)-**observable** if for any $s, s' \in E$ that $P_{lo}(s) = P_{lo}(s')$, and for any $\sigma \in \Sigma$ that $act_\theta(s\sigma) - act_\theta(s'\sigma) \neq \emptyset$ and $act_\theta(s'\sigma) \neq \emptyset$, we have*

$$s\sigma \in E \text{ and } s'\sigma \in L(G) \implies s'\sigma \in E.$$

Next, we show (G, P_{lo}, θ)-observability of $E_{lo}^{\uparrow}$ is ensured by the PO-SOCC property.

Proposition 3.29. *Consider a high-level controllable specification E_{hi}. Let $E_{lo} = \theta^{-1}(E_{hi})$ be the translation of E_{hi} at the low-level. Then if G_{lo} is PO-SOCC, $E_{lo}^{\uparrow}$ is (G_{lo}, P_{lo}, θ)-observable.*

Proof: We show if $E_{lo}^{\uparrow}$ is not (G_{lo}, P_{lo}, θ)-observable, then G_{lo} cannot be PO-SOCC. Suppose there exist $s, s' \in E_{lo}^{\uparrow}$ with $P_{lo}(s) = P_{lo}(s')$, and $\sigma \in \Sigma$ such that $act_\theta(s\sigma) - act_\theta(s'\sigma) \neq \emptyset$, $act_\theta(s'\sigma) \neq \emptyset$ and $s\sigma \in E_{lo}^{\uparrow}$ and $s'\sigma \in L(G_{lo})$ but $s'\sigma \notin E_{lo}^{\uparrow}$. First note that if $\sigma \in \Sigma_{uc}$, then since $s' \in E_{lo}^{\uparrow}$ and $s'\sigma \in L(G_{lo})$, from the controllability of $E_{lo}^{\uparrow}$ we conclude that $s'\sigma \in E_{lo}^{\uparrow}$ contradicting the assumption. Therefore from now, we assume σ is controllable ($\sigma \in \Sigma_c$).

Now let $\theta(s) = t$ and $\theta(s') = t'$ for some $t,\ t' \in T^*$. From the Factorization

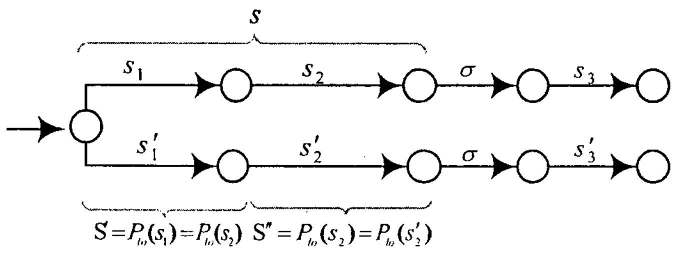

s
s_1
s_2
σ
s_3
s_1'
s_2'
σ
s_3'
$S'=P_{lo}(s_1)=P_{lo}(s_1')$ $S''=P_{lo}(s_2)=P_{lo}(s_2')$

Figure 3.10: Proposition 3.29: s_1 and s_1' are the largest vocal prefixes respectively of s and s' whose natural projections $P_{lo}(s_1) = P_{lo}(s_1') = \mathbf{S}'$ are the same. If (G_{lo}, P_{lo}, θ)-observability is violated, then nodes corresponding to $s\sigma s_3$ and $s'\sigma s_3'$ must be P-partners.

Property (3.1) it follows that $P_{hi}(t) = P_{hi}(t') = \mathbf{t}$ for some $\mathbf{t} \in T_o^*$. Let $\mathbf{S} = P_{lo}(s) = P_{lo}(s')$.

Thus $\tilde{\theta}(\mathbf{S}) = \mathbf{t}$. Let $\mathbf{S}'$ denote the shortest prefix of $\mathbf{S}$ such that $\tilde{\theta}(\mathbf{S}') = \tilde{\theta}(\mathbf{S})$. Then for some $\mathbf{S}'' \in \Sigma_o^*$, $\mathbf{S} = \mathbf{S}'\mathbf{S}''$. Furthermore, let s_1 and s_1' be the largest prefixes of s and s' such that $P_{lo}(s_1) = P_{lo}(s_1') = \mathbf{S}'$ and $s_1, s_1' \in L_{voc}$. Thus there exist s_2 and s_2' with $s = s_1 s_2$ and $s' = s_1' s_2'$ and $P_{lo}(s_2) = P_{lo}(s_2') = \mathbf{S}''$ (Fig. 3.10). By assumption, $act_\theta(s\sigma) - act_\theta(s'\sigma) \neq \emptyset$ and $act_\theta(s'\sigma) \neq \emptyset$. So suppose $\tau \in act_\theta(s\sigma) - act_\theta(s'\sigma) \neq \emptyset$ and $\tau' \in act_\theta(s'\sigma) \neq \emptyset$. Obviously we must have $\tau \neq \tau'$. Also since $\sigma \in \Sigma_c$, by OCC, $\tau, \tau' \in T_c$. Three cases are possible.

Case (1): $\tau \in T_{uo}$. Then for some $s_3 \in \Sigma_{uo}^*$ we have $\hat{\omega}(s_1 s_2 \sigma s_3) = \tau \in T_{uo}$, which in turn implies $s_2 \in \Sigma_{uo}^*$ and $\sigma \in \Sigma_{uo}$. Since $P_{lo}(s_2) = P_{lo}(s_2') = \mathbf{S}''$, we can conclude that $s_2' \in \Sigma_{uo}^*$, and thus $s_2'\sigma \in \Sigma_{uo}^+$. Now let $s_3' \in \Sigma^*$ be a sequence such that $\hat{\omega}(s_1' s_2' \sigma s_3') = \tau'$. From (3.13), $s_2'\sigma s_3' \in L_{voc}(s_1')$, $s_2'\sigma \in \Sigma_{uo}^+$ and $\sigma \in \Sigma_c$, it follows that $s_3' \in \Sigma_{uo}^*$ and therefore $\hat{\omega}(s_1' s_2' \sigma s_3') = \tau' \in T_{uo}$. Now from property UUC and the following vocalization rule in SCE Procedure (when applied to $s_1, s_1' \in L_{voc}$), we can conclude that $\hat{\omega}(s_1 s_2 \sigma s_3) = \hat{\omega}(s_1' s_2' \sigma s_3') = \tau_\sigma$ with $\tau_\sigma \in T_c^{uo}$ defined in (3.39). This means $\tau = \tau' = \tau_\sigma$ which contradicts the assumption $\tau \neq \tau'$.

Case (2): $\tau \in T_o$ and $s\sigma \in L_{voc}$. Since $P_{lo}(s\sigma) = P_{lo}(s'\sigma)$, by Factorization Property (3.1) we have:

$$\begin{array}{rrcl} & P_{hi}[\theta(s\sigma)] & = & P_{hi}[\theta(s'\sigma)] \\ \Longrightarrow & P_{hi}[t\hat{\omega}(s\sigma)] & = & P_{hi}[t'\hat{\omega}(s'\sigma)] \\ \Longrightarrow & P_{hi}(t)P_{hi}(\hat{\omega}(s\sigma)) & = & P_{hi}(t')P_{hi}(\hat{\omega}(s'\sigma)) \\ \Longrightarrow & \mathbf{t}\,\tau & = & \mathbf{t}\,P_{hi}(\hat{\omega}(s'\sigma)). \end{array}$$

This implies $s'\sigma \in L_{voc}$ and $\tau = P_{hi}(\hat{\omega}(s'\sigma)) = \tau'$ which contradicts the assumption $\tau \neq \tau'$.

Case (3): $\tau \in T_o$ and $s\sigma \notin L_{voc}$. There exists $s_3 \in \Sigma^+$ such that $\hat{\omega}(s_1 s_2 \sigma s_3) = \tau$. Now $\tau \in T_o$, $\sigma \in \Sigma_c$ and (3.12) imply $\Sigma^c(\eta(s_2\sigma s_3)) = \emptyset$ and $\mathbf{S}''P_{lo}(\sigma) = P_{lo}(s_2\sigma) = P_{lo}(s_2'\sigma) \neq \epsilon$. Then note that $s'\sigma \notin L_{voc}$ otherwise, since $P_{lo}(s_2'\sigma) \neq \epsilon$, we would have $\omega(s'\sigma) = \tau' \in T_o$, and for sequences $s\sigma$ and $s'\sigma$, $P_{lo}(s\sigma) = P_{lo}(s'\sigma)$ and $P_{hi}(\theta(s\sigma)) = \mathbf{t} \neq \mathbf{t}\tau' = P_{hi}(\theta(s'\sigma))$ which would violate Factoriza-

tion Property (3.1). Therefore there exists $s_3' \in \Sigma^+$ such that $\hat{\omega}(s_1's_2'\sigma s_3') = \tau'$ and $\tau' \in T_o$. Furthermore, Proposition 3.3 implies $s_3, s_3' \in \Sigma_o^+$. Next by (3.5), $\tilde{\omega}(\mathbf{S}'\mathbf{S}'' P_{lo}(\sigma)P_{lo}(s_3)) = \hat{\omega}(s\sigma s_3) = \tau \in T_c$ and $\tilde{\omega}(\mathbf{S}'\mathbf{S}'' P_{lo}(\sigma)P_{lo}(s_3')) = \hat{\omega}(s'\sigma s_3') = \tau' \in T_c$. Therefore by Proposition 3.21, both $\mathbf{S}'' P_{lo}(\sigma)P_{lo}(s_3)$ and $\mathbf{S}'' P_{lo}(\sigma)P_{lo}(s_3')$ are P_{lo}-controllable from state $\tilde{q}' = \tilde{\delta}(\tilde{q}_o, \mathbf{S}')$. Now consider the sequence $s'\sigma$. By assumption, $s'\sigma \in L(G_{lo})$ and $s'\sigma \notin E_{lo}^{\uparrow}$. Any extension of $s'\sigma$, that is $s'\sigma s''$, which does not lead to a vocal state is a legal string since $\theta(s'\sigma s'') = \theta(s'\sigma) \in \theta(E_{lo}^{\uparrow}) \subseteq E_{hi}$. Therefore it follows from $s'\sigma \notin E_{lo}^{\uparrow}$ that there exists an uncontrollable sequence $s'' \in \Sigma_{uc}^+$ such that $s'\sigma s'' \in L_{voc}$. Therefore without loss of generality we will assume s_3', introduced earlier in Case (3), is an uncontrollable sequence ($s_3' \in \Sigma_{uc}^+$). This requires that $P_{lo}(s_3')$ to be P_{lo}-uncontrollable from state $\tilde{q} = \tilde{\delta}(\tilde{q}_o, \mathbf{S}'\mathbf{S}'' P_{lo}(\sigma))$. Since $\tau \neq \tau'$, the nodes in the reachability tree of $\tilde{G}_{lo}$ reached by $\mathbf{S}'\mathbf{S}'' P_{lo}(\sigma)P_{lo}(s_3)$ and $\mathbf{S}'\mathbf{S}'' P_{lo}(\sigma)P_{lo}(s_3')$ are P-partners, and thus G_{lo} cannot be PO-SOCC, contradicting the hypothesis assumption. This completes the proof that $E_{lo}^{\uparrow}$ is (G, P_{lo}, θ)-observable. $\square$
Next, it is shown how supervisor can be implemented at the low-level.

3.4.4 Hierarchical Consistency under Partial Observation

Let $S_{hi} : T^* \times T_c \longmapsto \{0, 1\}$ be the high-level supervisor which implements a controllable and observable specification E_{hi}. A disabled-event map $\Delta_{hi} : L(G_{hi}) \longmapsto 2^{T_c}$ can be derived for S_{hi} as

$$\Delta_{hi}(t) = \{\tau \in T_c | \; S_{hi}(t, \tau) = 0\}.$$

A high-level disablement is implemented at the low-level by a low-level disablement law $\tilde{\Delta}_{lo} : \Sigma^* \times T^* \longmapsto 2^{\Sigma_c}$ given by

$$\tilde{\Delta}_{lo}(s,t) = \bigcup\{\Delta_{lo}(s',t')| \quad s' \in P_{lo}^{-1}P_{lo}(s) \cap L(G_{lo}), \\ t' \in P_{hi}^{-1}P_{hi}(t) \cap L(G_{hi})\}, \tag{3.14}$$

where $\Delta_{lo}(s,t)$ is defined similar to [59]:

$$\Delta_{lo}(s,t) = \{\sigma \in \Sigma_c| \ \exists u \in \Sigma_{uc}^*, \ \ s\sigma u \in L(G_{lo}) \ \text{ and } \\ \hat{\omega}(s\sigma u) \in \Delta_{hi}(t) \ \text{ and } \forall s'' < u, \ \hat{\omega}(s\sigma s'') = \tau_o\}.$$

Then low-level supervisor $\tilde{S}_{lo} : \Sigma^* \times T \longmapsto \{0,1\}$ is defined based on $\tilde{\Delta}_{lo}$ as follows:

$$\tilde{S}_{lo}(s,\sigma) = \begin{cases} 0 & \text{if } \sigma \in \tilde{\Delta}_{lo}(s,\theta(s)) \\ 1 & \text{otherwise.} \end{cases} \tag{3.15}$$

To study the properties of $\tilde{S}_{lo}$, we compare it with the low-level supervisor $S_{lo} : \Sigma^* \times T \longmapsto \{0,1\}$ defined based on Δ_{lo} for the case of full observation [59]:

$$S_{lo}(s,\sigma) = \begin{cases} 0 & \text{if } \sigma \in \Delta_{lo}(s,\theta(s)) \\ 1 & \text{otherwise.} \end{cases} \tag{3.16}$$

Recall that in supervisory control under partial observation, a supervisor is said to be *feasible* if it acts the same in response to two look-alike sequences (two sequences with the same natural projection).

Proposition 3.30. *Low-level supervisor $\tilde{S}_{lo}$, given in (3.15), is feasible.*

Proof: We must show for every $s, s' \in L(G_{lo})$ that $P_{lo}(s) = P_{lo}(s')$ we have $\tilde{\Delta}_{lo}(s, \theta(s)) = \tilde{\Delta}_{lo}(s', \theta(s'))$. First note that

$$\tilde{\Delta}_{lo}(s, \theta(s)) = \bigcup \{\Delta_{lo}(r, t) |\ r \in P_{lo}^{-1} P_{lo}(s) \cap L(G_{lo}),\quad t \in P_{hi}^{-1} P_{hi}(\theta(s)) \cap L(G_{hi})\}. \tag{3.17}$$

Since $P_{lo}(s) = P_{lo}(s')$, Factorization Property (3.1) implies $P_{hi}(\theta(s)) = P_{hi}(\theta(s'))$. Therefore (3.17) can be written as

$$\begin{aligned} \tilde{\Delta}_{lo}(s, \theta(s)) &= \bigcup \{\Delta_{lo}(r, t) |\ r \in P_{lo}^{-1} P_{lo}(s') \cap L(G_{lo}), \\ &\qquad\qquad t \in P_{hi}^{-1} P_{hi}(\theta(s')) \cap L(G_{hi})\} \\ &= \tilde{\Delta}_{lo}(s', \theta(s')). \end{aligned}$$

Hence $\tilde{S}_{lo}$ is feasible. □

The following theorem shows that UUC, UUPO and PO-SOCC guarantee hierarchical consistency.

Theorem 3.31. *If G_{lo} is UUC, UUPO and PO-SOCC and the high-level specification E_{hi} is controllable and observable with respect to G_{hi}, then $\theta(L(\tilde{S}_{lo}/G_{lo})) = E_{hi}$*

Proof: Since by Theorem 2.37, $\theta(L(S_{lo}/G_{lo})) = E_{hi}$, it suffices to show that $\theta(L(S_{lo}/G_{lo})) = \theta(L(\tilde{S}_{lo}/G_{lo}))$.

($\supseteq$): From the definition of $\tilde{\Delta}_{lo}$, for every $s \in L(G_{lo})$, $\Delta_{lo}(s, \theta(s)) \subseteq \tilde{\Delta}_{lo}(s, \theta(s))$; therefore the set of events disabled by S_{lo} is a subset of those disabled by $\tilde{S}_{lo}$ and therefore $\theta(L(\tilde{S}_{lo}/G_{lo})) \subseteq \theta(L(S_{lo}/G_{lo}))$.

($\subseteq$): To show $\theta(L(S_{lo}/G_{lo})) \subseteq \theta(L(\tilde{S}_{lo}/G_{lo}))$ we prove

$$\theta(L(G_{lo})) - \theta(L(\tilde{S}_{lo}/G_{lo})) \subseteq \theta(L(G_{lo})) - \theta(L(S_{lo}/G_{lo})). \tag{3.18}$$

If $\theta(L(G_{lo}))-\theta(L(\tilde{S}_{lo}/G_{lo})) = \emptyset$, (3.18) holds. Suppose $\theta(L(G_{lo}))-\theta(L(\tilde{S}_{lo}/G_{lo}$ $)) \neq \emptyset$. Then for any $t_1 \in \theta(L(G_{lo})) - \theta(L(\tilde{S}_{lo}/G_{lo}))$, there exists $s_1 \in L(G_{lo}) - L(\tilde{S}_{lo}/G_{lo})$ such that $\theta(s_1) = t_1 \notin \theta(L(\tilde{S}_{lo}/G_{lo}))$. Let $s_2 < s_1$ be the largest prefix of s_1 in $L(\tilde{S}_{lo}/G_{lo})$. Thus there exists $\sigma \in \Sigma_c$ and $s_3 \in \Sigma^*$ such that $s_1 = s_2\sigma s_3$ with $s_2 \in L(\tilde{S}_{lo}/G_{lo})$, $\theta(s_2) \in \theta(L(\tilde{S}_{lo}/G_{lo}))$ and $\sigma \in \tilde{\Delta}_{lo}(s_2, \theta(s_2))$. Since $\theta(s_2) \neq \theta(s_1)$, the sequence σs_3 goes through at least one vocal state. Thus there exists $s_4 \leq s_3, \tau \in T$ such that $\sigma s_4 \in L_{voc}(s_2)$, $\hat{\omega}(s_2\sigma s_4) = \tau$. Let $t_2 = \theta(s_2)$. Clearly $t_2\tau \notin \theta(L(\tilde{S}_{lo}/G_{lo}))$. We show $t_2\tau \notin \theta(L(S_{lo}/G_{lo}))$.

Case (1) $\sigma \in \tilde{\Delta}_{lo}(s_2, \theta(s_2))$ is disabled by $\tilde{S}_{lo}$ but $\sigma \notin \Delta_{lo}(s_2, \theta(s_2))$ remains enabled by S_{lo}: Recall from Theorem 2.33 that $L(S_{lo}/G_{lo}) = E_{lo}^{\uparrow}$. Thus $\sigma \notin \Delta_{lo}(s_2, \theta(s_2))$ implies

$$s_2\sigma \in E_{lo}^{\uparrow}. \tag{3.19}$$

Next, note that since $\sigma \in \tilde{\Delta}_{lo}(s_2, \theta(s_2))$, by (3.14), there should exist $s' \in P_{lo}^{-1}P_{lo}(s_2) \cap L(G_{lo})$ and $t' \in P_{hi}^{-1}P_{hi}(\theta(s_2)) \cap L(G_{hi})$ such that $\sigma \in \Delta_{lo}(s', t')$. Factorization Property (3.1) for s_2 and s' with $P_{lo}(s') = P_{lo}(s_2)$ implies $P_{hi}(\theta(s')$ $) = P_{hi}(\theta(s_2))$ and since $P_{hi}(t') = P_{hi}(\theta(s_2))$, we have $P_{hi}(t') = P_{hi}(\theta(s'))$. Then, since S_{hi} is feasible it is concluded that $\Delta_{hi}(t') = \Delta_{hi}(\theta(s'))$. This implies $\sigma \in \Delta_{lo}(s', \theta(s'))$ and hence

$$s'\sigma \notin E_{lo}^{\uparrow}. \tag{3.20}$$

Let $u' \in \Sigma_{uc}^*$ be the corresponding sequence in the definition of Δ_{lo} for which we have $\sigma u' \in L_{voc}(s')$ and $\hat{\omega}(s'\sigma u') \in \Delta_{hi}(\theta(s'))$. Let $\hat{\omega}(s'\sigma u') = \tau'$ for some $\tau' \in T$. Therefore, we should have

$$\tau' \in \Delta_{hi}(\theta(s')) \tag{3.21}$$

and $\theta(s')\tau' \notin \theta(L(S_{lo}/G_{lo}))$. Recall that we had assumed $\hat{\omega}(s_2\sigma s_4) = \tau$. We have $P_{lo}(s_2) = P_{lo}(s')$ and that

$$s_2\sigma \in E_{lo}^{\uparrow} \;\;\&\;\; s'\sigma \in L(G_{lo}) \;\;\&\;\; s'\sigma \notin E_{lo}^{\uparrow} \tag{3.22}$$

where the first and the last terms were given respectively in (3.19) and (3.20). Note that by Proposition 3.26 the system is SOCC; hence we have $act_\theta(s'\sigma) = \{\tau'\}$. Now if $\tau \neq \tau'$, then $act_\theta(s_2\sigma) - act_\theta(s'\sigma) \neq \emptyset$ which together with (3.22) imply $E_{lo}^{\uparrow}$ is not (G, P_{lo}, θ)-observable. This cannot be the case since G_{lo} is PO-SOCC; thus by Proposition 3.29, $E_{lo}^{\uparrow}$ is (G_{lo}, P_{lo}, θ)-observable. This implies $\tau = \tau'$. Besides, S_{hi} is feasible which implies $\Delta_{hi}(\theta(s_2)) = \Delta_{hi}(\theta(s'))$. Therefore, (3.21) implies $\tau \in \Delta_{hi}(\theta(s_2))$ or equivalently $\theta(s_2)\tau \notin E_{hi}$. Finally, since by Theorem 2.37 $\theta(L(S_{lo}/G_{lo})) = E_{hi}$, we have

$$\theta(s_2)\tau = t_2\tau \notin \theta(L(S_{lo}/G_{lo})). \tag{3.23}$$

Case (2): $\sigma \in \tilde{\Delta}_{lo}(s_2, \theta(s_2))$ and $\sigma \in \Delta_{lo}(s_2, \theta(s_2))$ is disabled by both S_{lo} and $\tilde{S}_{lo}$: In this case, $\theta(s_2)\tau = t_2\tau \notin \theta(L(S_{lo}/G_{lo}))$.

Therefore in both cases we have shown $t_2\tau \notin \theta(L(S_{lo}/G_{lo}))$. Finally, since $t_2\tau \notin \theta(L(S_{lo}/G_{lo}))$, $t_2\tau \leq t_1$ and $\theta(L(S_{lo}/G_{lo}))$ is closed, we conclude $t_1 \notin \theta(L(S_{lo}/G_{lo}))$. This proves (3.18) and thus $\theta(L(S_{lo}/G_{lo})) \subseteq \theta(L(\tilde{S}_{lo})/G_{lo})$ which completes the proof. □

Example 3.32. *Let us continue Example 3.27. The low-level model after the modifications to ensure Factorization Property, UUC, UUPO and PO-SOCC properties is shown in Fig. 3.11. The high-level model G_{hi} is shown in Fig. 3.12. The high-level event set is $T = T_c \cup T_{uc}$ with $T_c = \{B_c, \tau_\alpha, \tau_\gamma\}$ and $T_{uc} = \{A_{uc}, B_{uc}, D_{uc}, E_{uc}\}$ where we have $\tau_\alpha, \tau_\gamma \in T_c^{uo}$ with T_c^{uo} defined in (3.39). Let the design specification E_{hi}' at the high-level be described as* "Event **A** cannot

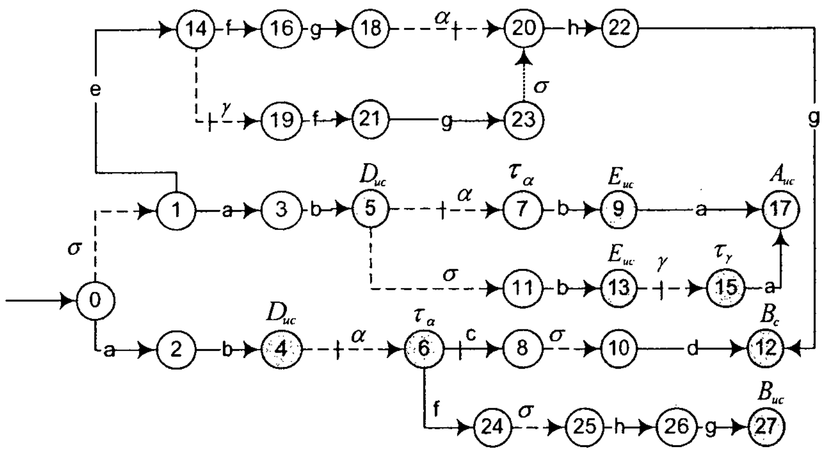


Figure 3.11: Final model G_{lo} after satisfying properties UUC, UUPO and PO-SOCC. Colored states have been assigned new outputs.

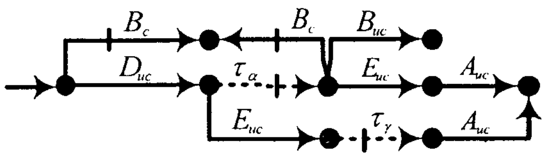


Figure 3.12: Final high-level model G_{hi}

occur". *This is formally written as* $E'_{hi} = \{t \in L(G_{hi}) \mid \forall r \leq t,\ rA_{uc} \nleq t\}$. *The automaton generating an observable and controllable sublanguage of* E'_{hi} *is given in Fig. 3.13. Let* $E_{hi} = \overline{B_c + D_{uc}E_{uc}}$ *denote this sublanguage of* E'_{hi} *in the following. Then, consider a feasible supervisor* $\tilde{S}_{lo}$ *which implements* $E_{lo} = \theta^{-1}(E_{hi})$ *at the low-level. The low-level system under supervision* $\tilde{S}_{lo}/G_{lo}$ *is given in Fig. 3.14. It can easily be verified that* $L(\tilde{S}_{lo}/G_{lo})$ *in Fig. 3.14 is hierarchically consistent with the high-level specification* E_{hi}*, i.e. we have* $\theta(L(\tilde{S}_{lo}/G_{lo})) = E_{hi}$*. Recall from Example 3.27 that some sequences, not leading to high-level event* A*, were unintentionally disabled there. Here, by the refinements applied to the reporting map as explained in Example 3.27 hierarchical consistency has been achieved. Note that how the UUC, UUPO and PO-SOCC properties have ensured hierarchical consistency in this system by relabeling in which* α *or* γ *at the low-level respectively after states 4, 5 and*

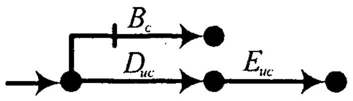


Figure 3.13: Controllable and observable specification E_{hi} at the high-level

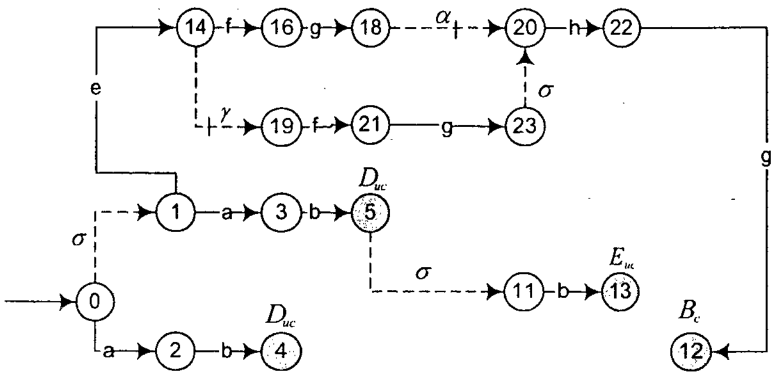


Figure 3.14: $\tilde{S}_{lo}/G_{lo}$: System under supervision which is hierarchically consistent with E_{hi} in Fig. 3.13.

13 are disabled if and only if τ_α or τ_γ are disabled at the high-level.

Next we show how (G_{lo}, P_{lo}, θ)-observability, which is induced by UUC, UUPO and PO-SOCC properties, and is weaker than observability, suffices to ensure hierarchical consistency.

Example 3.33. *Consider the Moore model G_{lo} in Fig. 3.15 where $\Sigma = \{a, b, c, d, \sigma, \beta, \gamma\}$, $\Sigma_c = \{c, \sigma\}$, $\Sigma_{uo} = \{\sigma, \alpha, \beta, \gamma\}$, $T = T_\circ = \{\tau_1, \tau_2\}$ and $T_c = \{\tau_2\}$. The Factorization Property holds in the system and the high-level model G_{hi} is given in Fig. 3.16. Let $E_{hi} = \overline{\tau_1}$. E_{hi} is controllable*

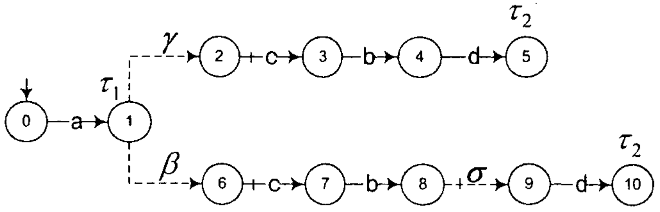


Figure 3.15: G_{lo} is UUC, UUPO and PO-SOCC.

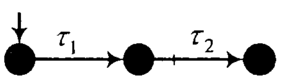


Figure 3.16: High-level model G_{hi} which corresponds to model G_{lo} in Fig. 3.15.

and observable with respect to G_{hi}. $E_{lo} = \theta^{-1}(E_{hi}) = \overline{a.(\gamma.c.b + \beta.c.b.\sigma)}$ and $E_{lo}^{\uparrow} = \overline{a.(\gamma + \beta.c.b)}$. It can be seen that $E_{lo}^{\uparrow}$ is not observable with respect to G_{lo}. It can be verified that the properties UUC, UUPO and PO-SOCC hold in the system. Hence, there exits a low-level feasible supervisor $\tilde{S}_{lo}$ such that $\theta(L(\tilde{S}_{lo}/G_{lo})) = \overline{\tau_1}$; i.e. hierarchical consistency holds in the system. In particular, $\tilde{S}_{lo}$ (following (3.15)) will disable event c once G_{lo} reaches state 1. Therefore, the closed behavior of the system under supervision will be $L(\tilde{S}_{lo}/G_{lo}) = \overline{a.(\gamma + \beta)} \subset E_{lo}^{\uparrow}$. Note that $E_{lo}^{\uparrow}$ is (G_{lo}, P_{lo}, θ)-observable.

Example 3.33 shows that even if $E_{lo}^{\uparrow}$ is not observable with respect to G_{lo}, we are still able to ensure hierarchical consistency, provided that G_{lo} is UUC, UUPO and PO-SOCC. In this case, $E_{lo}^{\uparrow}$ will be (G_{lo}, P_{lo}, θ)-observable.

3.5 Comparison with H-observability

Kim et al. [25] have investigated partial observability in the hierarchical supervisory control framework of [59]. In this regard, [25] introduces the notion of H-observability.

Definition 3.34. *[25] A language $E_{hi} \subseteq L(G_{hi})$ is said to be H-observable w.r.t. (A, G_{hi}) if for all $s, s' \in \overline{A}(\subseteq L(G_{lo}))$ s.t. $P(s) = P(s')$,*

$$(\forall \tau \in T_c)\theta(s)\tau \in E_{hi} \text{ and } \theta(s') \in E_{hi} \text{ and } \theta(s')\tau \in L(G_{hi}) \Longrightarrow \theta(s')\tau \in E_{hi}. \tag{3.24}$$

To achieve hierarchical consistency under partial observation, Theorem 1 of [25] states that assuming G_{lo} is SOCC, for a controllable specification E_{hi}, hierarchical consistency is achieved, i.e. $\theta(L(S_{lo}/G_{lo})) = E_{hi}$, if

(i) E_{hi} is H-observable w.r.t. $(E_{lo}^{\uparrow}, G_{hi})$;

(ii) $E_{lo}^{\uparrow}$ is observable w.r.t. G_{lo}.

As discussed earlier in this chapter, partial observation at the low-level, has two important implications for the supervisory control problem at the high-level. First, some of the high-level events in T become unobservable. To ensure consistency in the observability status of the high-level events, we introduced the OOC condition. Secondly, in order to be able to have a feasible reporting map (one that reports to the high-level based on the low-level observations $P_{lo}(s)$), the Factorization Property must be satisfied. None of the above two implications have been considered in [25].
Now, let us assume OOC and the Factorization Property hold. The following proposition shows that observability of E_{hi} implies H-observability (w.r.t. $(E_{lo}^{\uparrow}, G_{hi})$).

Proposition 3.35. *If G_{lo} satisfies Factorization Property (3.1) and E_{hi} is observable w.r.t G_{hi}, then E_{hi} is H-observable w.r.t. $(E_{lo}^{\uparrow}, G_{hi})$.*

Proof: Consider two sequences $s, s' \in E_{lo}^{\uparrow}$ such that $P_{lo}(s) = P_{lo}(s')$. Then, $E_{lo}^{\uparrow} \subseteq L(G_{lo})$ and thus Factorization Property implies that

$$P_{hi}(\theta(s)) = P_{hi}(\theta(s')).$$

Next, since E_{hi} is observable w.r.t G_{hi}, for sequences $\theta(s)$ and $\theta(s')$ and for every $\tau \in T$ we have:

$$\theta(s)\tau \in E_{hi} \text{ and } \theta(s') \in E_{hi} \text{ and } \theta(s')\tau \in L(G_{hi}) \Longrightarrow \theta(s')\tau \in E_{hi}. \quad (3.25)$$

Comparing (3.24) and (3.25), it is concluded that E_{hi} is H-observable w.r.t. $(E_{lo}^{\uparrow}, G_{hi})$. □

Therefore, Theorem 1 of [25] essentially guarantees hierarchical consistency if

$E_{lo}^{\uparrow}$ is observable.

While it is known how the SOCC property can be satisfied [59], no course of action or algorithm has been given in [25] to deal with cases where the first or the second conditions of Theorem 1 of [25] is not met. Furthermore, the conditions given in [25] are specification-based, i.e. depend on E_{hi}, and even if they are satisfied, hierarchical consistency would only be valid for a certain E_{hi}.

In contrast to the solution of [25], our solution guarantees hierarchical consistency for any controllable and observable specification E_{hi}. Finally, it should be noted that as shown in Example 3.33, in the approach proposed in this thesis, to achieve hierarchical consistency, $E_{lo}^{\uparrow}$ does not have to be observable; satisfying the weaker assumption of (G_{lo}, P_{lo}, θ)-observability would be enough.

3.6 Conclusion

In this chapter, the hierarchical supervisory control setup of Zhong-Wonham was extended to the case of control under partial observation. The construction of a reporting map was discussed. A feasible low-level supervisor was proposed to implement the commands of the high-level supervisor. The UUC, UUPO and Partially-Observable-SOCC (PO-SOCC) properties were introduced to ensure hierarchical consistency under partial observation.

3.7 Appendix

3.7.1 Necessary and Sufficient Conditions for the Satisfaction of the Factorization Property

In this appendix we show how the output map ω can be refined such that the Factorization Property, the UUC, UUPO and PO-SOCC properties hold in the system. We begin with the Factorization Property. Let $G_{lo} = (Q, \Sigma, T \cup \{\tau_o\}, \delta, \omega, q_o)$ be the low-level model and $\tilde{G}_{lo} = (\tilde{Q}, \Sigma_o, \tilde{\delta}, \tilde{q}_o)$ the observer automaton for G_{lo}. Consider two sequences $s, s' \in L(G_{lo})$ for which we have

$$P_{lo}(s) = P_{lo}(s').$$

It is desired by the Factorization Property (3.1) to have $P_{hi}(\theta(s)) = P_{hi}(\theta(s'))$. Let

$$\theta(s) = \varepsilon_1\tau_1 \cdots \varepsilon_n\tau_n\varepsilon_{n+1} \quad \text{and} \quad \theta(s') = \varepsilon'_1\tau'_1 \cdots \varepsilon'_m\tau'_m\varepsilon'_{m+1}$$

where $\tau_i, \tau'_j \in T_o$ and $\varepsilon_i, \varepsilon'_j \in T^*_{uo}$ ($i = 1, \cdots, n$ and $j = 1, \cdots, m$). However, since $\varepsilon_{n+1}, \varepsilon'_{m+1} \in T^*_{uo}$ and $P_{lo}(\varepsilon_{n+1}) = P_{lo}(\varepsilon'_{m+1}) = \epsilon$, without loss of generality, we assume $\varepsilon_{n+1} = \varepsilon'_{m+1} = \epsilon$. This yields

$$\theta(s) = \varepsilon_1\tau_1 \cdots \varepsilon_n\tau_n \quad \text{and} \quad \theta(s') = \varepsilon'_1\tau'_1 \cdots \varepsilon'_m\tau'_m.$$

Thus satisfying (3.1) requires that

$$\tau_1 \cdots \tau_n = P_{hi}(\theta(s)) = P_{hi}(\theta(s')) = \tau'_1 \cdots \tau'_m$$

which implies $n = m$ and $\tau_i = \tau'_i$ for $i = 1, \cdots, n$. Furthermore, by Theorem 3.14 we have $\tilde{\theta}(P_{lo}(s)) = \tilde{\theta}(P_{lo}(s')) = \tau_1 \cdots \tau_n$. Note that $P_{lo}(s) = P_{lo}(s')$ consists of n sequential P_{lo}-silent paths in $L(\tilde{G}_{lo})$; i.e. we have $P_{lo}(s) = P_{lo}(s') = \xi_1 \cdots \xi_n$ where each ξ_i is a P_{lo}-silent path in $L(\tilde{G}_{lo})$.

Consider the segmentations

$$s = s_n = s_{n-1}\nu_n r_n \quad \text{and} \quad s' = s'_n = s'_{n-1}\nu'_n r'_n \tag{3.26}$$

where we have $s_{n-1}, s'_{n-1} \in L_{voc}$, $P_{lo}(\nu_n) = P_{lo}(\nu'_n) = \epsilon$, $r_n \in L_{voc}(s_{n-1}\nu_n)$, $r'_n \in L_{voc}(s'_{n-1}\nu'_n)$, $\theta(s_{n-1}\nu_n) = \varepsilon_1\tau_1 \cdots \tau_{n-1}\varepsilon_n$ and $\theta(s'_{n-1}\nu'_n) = \varepsilon'_1\tau'_1 \cdots \tau'_{n-1}\varepsilon'_n$. Clearly we have

$$P_{lo}(s_{n-1}) = P_{lo}(s'_{n-1}) = \xi_1 \cdots \xi_{n-1} \quad \text{and} \quad P_{lo}(r_n) = P_{lo}(r'_n) = \xi_n$$

which imply

$$\tilde{\theta}(P_{lo}(s_{n-1})) = \tilde{\theta}(P_{lo}(s'_{n-1})) = \tau_1 \cdots \tau_{n-1}$$
$$\tilde{\omega}(P_{lo}(s_n)) = \tilde{\omega}(P_{lo}(s'_n)) = \tau_n.$$

In other words, all the incoming states $q \in in(\tilde{q})$ where $\tilde{q} = est(P_{lo}(s_n))$ have the same output. Furthermore, the segmentation in (3.26) can be applied to the sequences s_{n-1} and s'_{n-1} such that $s_{n-1} = s_{n-2}\nu_{n-1}r_{n-1}$ and $s'_{n-1} = s'_{n-2}\nu'_{n-1}r'_{n-1}$ where s_{n-2}, s'_{n-2}, ν_{n-1}, ν'_{n-1}, r_{n-1} and r'_{n-1} are defined as before. Therefore,

$$\tilde{\omega}(P_{lo}(s_i)) = \tilde{\omega}(P_{lo}(s'_i)) = \tau_i \quad \text{for} \quad i = 1, \cdots, n.$$

This discussion agrees with Proposition 3.12 which states that if Factorization Property holds, then all the incoming states of a P-vocal state have the same output. We show the reverse is also true. Next, for any P_{lo}-silent path $\xi \in \Sigma_o^+$

entering a P-vocal state $\tilde{q}$, let

$$A_{\tilde{q}}(\xi) = \{r \in P_{lo}^{-1}(\xi) \mid \exists s \in L_{voc} : r \in L_{voc}(s) \text{ and } \delta(q_\circ, sr) \in in(\tilde{q})\} \quad (3.27)$$

be the set of silent paths in $L(G_{lo})$ whose natural projections is ξ and enter the incoming states set $in(\tilde{q})$. Note that by Definition 3.1 the outputs generated by $A_{\tilde{q}}(\xi)$ are observable.

Proposition 3.36. *The Factorization Property holds if and only if for every P-vocal state $\tilde{q}$ and for every incoming state $q \in in(\tilde{q})$, we have $\omega(q) = \boldsymbol{\tau}$ for some fixed $\boldsymbol{\tau} \in T_o$.*

Proof: **(If)**: Assume the Factorization Property (3.1) holds. Then Proposition 3.12 states all the incoming states in a P-vocal state $\tilde{q}$ have the same output $\boldsymbol{\tau} \in T_o$.

(Only if): Assume for every P-vocal state $\tilde{q}$ and for every incoming state $q \in in(\tilde{q})$, the output $\omega(q) \in T_\circ$ is unique and fixed. Now consider sequences $s, s' \in L(G_{lo})$ with the same projection $P_{lo}(s) = P_{lo}(s')$. Therefore, if for some sequence $r \leq s$, we have $\hat{\omega}(r) \in T_\circ$, then (i) $r \in \Sigma^*\Sigma_\circ^+$ leads to an incoming state and (ii) there exists a sequence $r' \leq s'$ such that $\hat{\omega}(r') = \hat{\omega}(r)$ and $r' \in \Sigma^*\Sigma_\circ^+$ (i.e., r' leads to an incoming state). This implies $P_{lo}(s)$ and $P_{lo}(s')$ consist of the same number of P_{lo}-silent paths. Let $n \in \mathbb{N}$ be the largest number for which we have $\xi_1 \cdots \xi_n \leq P_{lo}(s) = P_{lo}(s')$ where ξ_i are P_{lo}-silent paths in $L(\tilde{G}_{lo})$. It then follows that s and s' can respectively be split as $s = \nu_1 r_1 \cdots \nu_n r_n \vartheta$ and $s' = \nu_1' r_1' \cdots \nu_n' r_n' \vartheta'$ where $\nu_i, \nu_i' \in \Sigma_{uo}^*$ are unobservable sequences, $r_i, r_i' \in A_{\tilde{q}_i}(\xi_i)$ are silent paths which enter appropriate $\tilde{q}_i$'s to which ξ_i's enter, and $\vartheta, \vartheta' \in \Sigma^*$ are sequences in Σ^* which might only generate unobservable outputs. We will have $P_{hi}(\theta(\nu_1 r_1 \cdots \nu_n r_n \vartheta)) = P_{hi}(\theta(\nu_1 r_1 \cdots \nu_n r_n))$ and $P_{hi}(\theta(\nu_1' r_1' \cdots \nu_n' r_n' \vartheta')) = P_{hi}(\theta(\nu_1' r_1' \cdots \nu_n' r_n'))$. Therefore and without loss of generality assume $\vartheta = \vartheta' = \epsilon$. Hence, we can write $s = \nu_1 r_1 \cdots \nu_n r_n$ and $s' = \nu_1' r_1' \cdots \nu_n' r_n'$. Next, note that by splitting $s_i = s_{i-1} r_i$ and $s_i' = s_{i-1}' r_i'$ $(i = n, \cdots, 1)$, where $s_n = s$ and $s_n' = s'$ and

$$s_{i-1} = \nu_1 r_1 \cdots \nu_{i-1} r_{i-1} \nu_i \quad \text{and} \quad s_{i-1}' = \nu_1' r_1' \cdots \nu_{i-1}' r_{i-1}' \nu_i', \tag{3.28}$$

we have

$$\theta(s_i) = \theta(s_{i-1}) \varepsilon_i \, \hat{\omega}(s_i) \quad \text{and} \quad \theta(s_i') = \theta(s_{i-1}') \varepsilon_i' \, \hat{\omega}(s_i').$$

Here, $\varepsilon_i, \varepsilon_i' \in T_{uo}^*$ are generated by ν_i and ν_i' transitions. Now since $r_i, r_i' \in A_{\tilde{q}_i}(\xi_i)$ enter the incoming states of the P-vocal state $\tilde{q}_i$ and by assumption all the incoming states of P-vocal state $\tilde{q}_i$ have the same output, say τ_i, it can easily be seen that $\hat{\omega}(s_i) = \hat{\omega}(s_i') = \tau_i \in T_\circ$ which implies

$$\theta(s_i) = \theta(s_{i-1}) \varepsilon_i \, \tau_i \quad \text{and} \quad \theta(s_i') = \theta(s_{i-1}') \varepsilon_i' \, \tau_i.$$

Therefore by recursively splitting s_i, s_i', as given in (3.28), we will have

$$\theta(s) = \varepsilon_1 \tau_1 \cdots \varepsilon_n \tau_n \quad \text{and} \quad \theta(s') = \varepsilon_1' \tau_1 \cdots \varepsilon_n' \tau_n. \tag{3.29}$$

Equation (3.29) implies

$$P_{hi}(\theta(s)) = P_{hi}(\theta(s')) = \tau_1 \cdots \tau_n.$$

This proves that by ensuring silent paths $r \in A_{\tilde{q}}(\xi)$ which enter P-vocal state $\tilde{q}$ have the same output, the Factorization Property (3.1) can be ensured. □

3.7.2 Initialization Procedure

If the the conditions of Proposition 3.36 do not hold, in order to modify the reporting map θ, we need to examine the sequences of events in the system. Alternatively, if the Moore automaton which represents the closed behavior of the hierarchical system lends itself to certain characteristics, we can examine the states of the Moore automaton. In this section, we present one possible way of using the information of the states instead of examining the sequences. Let $G_{lo} = (Q, \Sigma, T \cup \{\tau_\circ\}, \delta, \omega, q_\circ)$ be the low-level model, $\tilde{G}_{lo} = (\tilde{Q}, \Sigma_\circ, \tilde{\delta}, \tilde{q}_\circ)$ the observer automaton for G_{lo} and $\tilde{Q}_{pvoc} \subseteq \tilde{Q}$ the set of P-vocal states in $\tilde{G}_{lo}$.

Step 1: (Defining a virtual output map μ) Define a new label set $T' = \{\mathrm{a}, \mathrm{b}, \alpha, \beta\}$ with $T'_\circ = \{\mathrm{a}, \mathrm{b}\}$ as the observable subset and $T'_c = \{\mathrm{a}, \alpha\}$ as the controllable subset. b and β are regarded as the uncontrollable versions of a and α. Also, α and β are regarded as unobservable versions of a and b. Let a new output map $\mu : Q \longmapsto T' \cup \{\tau_\circ\}$ be defined over the state set Q of G_{lo} with the following order of priorities in assignment:

(i) $\mu(q) = a$ if $q \in in(\tilde{q})$ for some $\tilde{q} \in \tilde{Q}_{pvoc}$.

(ii) $\mu(q) = \alpha$ if $q \in out(\tilde{q})$ for some $\tilde{q} \in \tilde{Q}_{pvoc}$.

(iii) $\mu(q) = \tau_\circ$, otherwise.

Therefore, if q is both an incoming state and an outgoing state, then q by the first priority is assigned $\mu(q) = \mathrm{a}$. Apply OCC and OOC Algorithms to the model G_{lo}, considering $\mu : Q \longmapsto T' \cup \{\tau_\circ\}$ as the output map, with the following conversions in those algorithms as needed:

- OCC: $a \longleftrightarrow b$ or $\alpha \longleftrightarrow \beta$;
- OOC: $a \longleftrightarrow \alpha$ or $b \longleftrightarrow \beta$.

Let $G'_{lo} = (Q', \Sigma, T' \cup \{\tau_\circ\}, \varrho, \mu, q'_\circ)$ be the Moore automaton which is obtained in this step where $\mu : Q' \longmapsto T' \cup \{\tau_\circ\}$ is properly redefined by the OCC and OOC algorithms. Let $\tilde{G}'_{lo}$ denote the observer automaton for G'_{lo}. By construction, $L(G'_{lo}) = L(G_{lo})$ and $L(\tilde{G}'_{lo}) = L(\tilde{G}_{lo})$.

Step 2: (Refining with respect to state estimates) Let $H_{lo} = G'_{lo} || \tilde{G}'_{lo}$. Recall that we had $L(\tilde{G}'_{lo}) = L(\tilde{G}_{lo})$; hence, it is easy to see that $L(H_{lo}) = L(G_{lo})$. We assume $H_{lo} = (F, \Sigma, T' \cup \{\tau_\circ\}, \vartheta, \lambda, f_\circ)$ is the refined Moore automaton we obtain where for each state $f \in F$ with $f = (q', \tilde{q})$, we assign $\lambda(f) = \mu(q')$.

Step 3: (Setting up output map ω for H_{lo}) First, recall that the output map $\omega : Q \longmapsto T \cup \{\tau_\circ\}$ is defined over the state set Q. Now, let $\hat{\omega} : L(G_{lo}) \longmapsto T \cup \{\tau_\circ\}$ be defined on $L(G_{lo})$ as follows:

$$\hat{\omega}(s) = \omega(q) \text{ where } q = \delta(q_\circ, s).$$

Then, since $\equiv_{H_{lo}} \leq \equiv_{G'_{lo}} \leq \equiv_{G_{lo}}$, we can define the output map $\hat{\omega} : L(G_{lo}) \longmapsto T \cup \{\tau_\circ\}$ on the state set F of H_{lo} and hence redefine the output map $\omega : F \longmapsto T \cup \{\tau_\circ\}$ as follows:

$$\omega(f) = \hat{\omega}(s) \text{ where } f = \vartheta(f_\circ, s).$$

Note that the redefinition of ω is well-defined since $\equiv_{H_{lo}} \leq \equiv_{G_{lo}}$. Therefore, we consider $H_{lo} = (F, \Sigma, T \cup \{\tau_\circ\}, \varsigma, \omega, f_\circ)$ as the new Moore automaton with the output map ω.

Step 4: (Characteristics and notation) Let $\tilde{H}_{lo} = (\tilde{F}, \Sigma_o, \tilde{\vartheta}, \tilde{f}_o)$ denote the observer automaton for H_{lo}. The automata H_{lo} and $\tilde{H}_{lo}$ have the property that first, states $\tilde{f} \in \tilde{F}$ are the cell blocks of the state set F of H_{lo} and second, if a potentially P_{lo}-silent path ξ reaches a state $\tilde{f} \in \tilde{F}$, then for every sequence

$r' \in A'_{\tilde{f}}(\xi)$ which reaches a state $f \in in(\tilde{f})$, we have

$$\Sigma^c(r') \neq \emptyset \Longleftrightarrow \lambda(f) \in T'_c. \tag{3.30}$$

Equation (3.30) implies that the controllability of the silent paths which correspond to ξ can be examined by checking the outputs of H_{lo} with respect to the map λ. We write $(H_{lo}, \lambda) = INIT(G_{lo})$ to mean that $H_{lo} = (F, \Sigma, T \cup \{\tau_\circ\}, \varsigma, \omega, f_\circ)$ is the final Moore automaton we obtain in Step 3 with λ as the virtual output map which we obtain in Step 2. Note that both ω and λ are defined on the state set F of H_{lo}. However, we refer to λ as a virtual output map only. □

Some useful definitions follow.

Definition 3.37. *We say the output map ω is well-defined if for every $\tilde{q} \in \tilde{Q}_{pvoc}$, $in(\tilde{q})$ can be assigned a unique output $\boldsymbol{\tau} \in T_\circ \cup \{\tau_\circ\}$; i.e. for every $q \in in(\tilde{q})$, there exists a unique $\boldsymbol{\tau} \in T_\circ \cup \{\tau_\circ\}$ such that $\omega(q) = \boldsymbol{\tau}$.*

Now, assume ω is not well-defined. This means that there exists a state $\tilde{q} \in \tilde{Q}$ whose incoming states set cannot be assigned a unique output. In this regard, a state $\tilde{q} \in \tilde{Q}$ is either silent or *potentially* P-vocal as defined in the following:

Definition 3.38. *Let $\boldsymbol{S} \in L(\tilde{G}_{lo})$ be a sequence in $\tilde{G}_{lo}$ and $\tilde{q} = \tilde{\delta}(\tilde{q}_o, \boldsymbol{S})$. $\tilde{q}$ is called a* potentially *P-vocal state if there exists a state $q \in \tilde{q}$ for which we have $\omega(q) \neq \tau_\circ$. An observable sequence $\boldsymbol{R} \in \Sigma_o^*$ which connects the root state to a* potentially *P-vocal state or connects two immediate* potentially *P-vocal states to each other, is called a* potentially *P_{lo}-silent path.*

A *potentially* P-vocal state is a generalization of the P-vocal state in the sense that the uniqueness of the output for $in(\tilde{q})$ is relaxed. Satisfying (3.1) guarantees each *potentially* P-vocal state (resp. *potentially* P_{lo}-silent path) is in fact a P-vocal state (resp. P_{lo}-silent path). Let $\tilde{Q}_{pvoc}$ now denote the set of Potentially P-vocal states in $\tilde{G}_{lo}$. For a potentially P_{lo}-silent path $\xi \in \Sigma_o^+$ which enters a state $\tilde{q} \in \tilde{Q}_{pvoc}$ define a set function $A'_{\tilde{q}} : \Sigma_o^+ \longmapsto \Sigma^*$,

$$A'_{\tilde{q}}(\xi) = \{r' \in P_{lo}^{-1}(\xi) \mid \exists \tilde{p} \in \tilde{Q}_{pvoc} \text{ and } \exists p \in out(\tilde{p}) : \delta(p, r') \in in(\tilde{q})\} \quad (3.31)$$

to return the sequences which start from an outgoing state p of a potentially P-vocal state $\tilde{p}$, whose projections are ξ and end in an incoming state q of the potentially P-vocal state $\tilde{q}$. Note that if $\tilde{\omega}$ is well-defined, $\tilde{q}$ is a P-vocal state and ξ is a P_{lo}-silent path entering $\tilde{q}$ and thus, for each sequence $r' \in A'_{\tilde{q}}(\xi)$ there will exist a sequence $r \in A_{\tilde{q}}(\xi)$ such that $r = \eta(r)r'$ where $\eta(.)$ is given in (3.11) and thus $|A_{\tilde{q}}(\xi)| = |A'_{\tilde{q}}(\xi|$. Furthermore, if UUPO property holds in the system, $r = \eta(r)r'$ implies

$$\Sigma^c(r) \neq \emptyset \iff \Sigma^c(r') \neq \emptyset. \quad (3.32)$$

Therefore, as long as we intend to ensure UUPO property in the system, for a given potentially P_{lo}-silent path entering a state $\tilde{q}$, sequences $r' \in A'_{\tilde{q}}(\xi)$ can be examined instead of $r \in A_{\tilde{q}}(\xi)$. Note that in case $\tilde{\omega}$ is not well-defined $|A_{\tilde{q}}(\xi)| \leq |A'_{\tilde{q}}(\xi|$ and thus there may not exist a one-to-one map between $A_{\tilde{q}}(\xi)$ and $A'_{\tilde{q}}(\xi)$. Therefore, exploiting $A'_{\tilde{q}}(.)$ instead of $A_{\tilde{q}}(.)$ is inevitable in cases $\tilde{\omega}$ is not well-defined and the controllability of the silent paths is of interest.

3.7.3 FP Test: ES Algorithm

We present a test, based on Proposition 3.36, to check if the Factorization Property holds in the system.

Factorization Property (FP) Test: G_{lo} satisfies Factorization Property if

$\omega(q) = \omega(q') \in T_\circ$ for every $q, q' \in in(\tilde{q})$ where $\tilde{q} \in \tilde{Q}_{pvoc}$. □

Consider a transition $\tilde{p} \xrightarrow{\xi} \tilde{q}$ where $\tilde{p}$ and $\tilde{q}$ are potentially P-vocal states and ξ is a potentially P_{lo}-silent path (i.e. we have $\tilde{q} = \tilde{\delta}(\tilde{p}, \xi)$). If the FP Test fails (at the state $\tilde{q}$), it means $\tilde{\omega}$ is not well-defined. Three cases are distinguishable; we modify the reporting map in each case.

Remark 3.39. *For the sake of convenience at this step, we assume the primary model G_{lo} is such that we have $(G_{lo}, \lambda) = INIT(G_{lo})$.*

FP Test, Case (1): For some $q, q' \in in(\tilde{q})$, $\omega(q) \neq \omega(q')$ where for every $r' \in A'_{\tilde{q}}(\xi)$ we have $\Sigma^c(r') = \emptyset$ or equivalently for all $q \in in(\tilde{q})$ we have $\lambda(q) \in T'_{uc}$. This case represents the silent paths whose natural projection is the same, all are uncontrollable but the output they generate are not the same. We propose the following modification.

Algorithm 3.40. *[Modification in Case (1)]*

Input: Automaton G_{lo} which fails FP Test in case (1).

```
i. Initial step: Let $W = \tilde{Q}_{pvoc}$

ii. Choose any $\tilde{q} \in W$ and let $W \longleftarrow W - \{\tilde{q}\}$ (Suppose it is desired that
    $\varpi(\tilde{q}) = \boldsymbol{\tau}$ where $\boldsymbol{\tau} \in T_\circ$).

iii. If for all $q \in in(\tilde{q})$ we have $\lambda(q) \in T'_{uc}$, let $V = in(\tilde{q})$ and do the
     following:

    1. Choose any $q \in V$. Let $V \longleftarrow V - \{q\}$ and $\tau = \omega(q)$ and

    2. If $\tau = \tau_\circ$, let $\omega(q) = \boldsymbol{\tau}$.

    3. Otherwise, If $\tau_\circ \neq \tau \neq \boldsymbol{\tau}$, let $Q \longleftarrow Q \cup \{q'\}$ where $q'$ is a new state
       of the system and do the following:

        a. Let $E = \{q'' \xrightarrow{\sigma} q \mid \exists q'' \in Q, \exists \sigma \in \Sigma_\circ : q = \delta(q'', \sigma)\}$ be the set
           of transitions which reach $q$ by some observable event.
```

b. Define a new set of transitions $F = \{q'' \xrightarrow{\sigma} q' \mid q'' \xrightarrow{\sigma} q \in E\}$ *to consist of transitions which reach the new state* q' *and where there is a one-to-one map between the transitions of* E *and* F.

c. Let $(\Sigma_{uo} \cap \Sigma_{uc}) \longleftarrow (\Sigma_{uo} \cap \Sigma_{uc}) \cup \{\beta\}$ *where* β *is a new unobservable-uncontrollable event of the system.*

d. Remove E *from the system transitions.*

e. Add $F \cup \{q' \overset{\beta}{\dashrightarrow} q\}$ *to the system.*

f. Let $\omega(q') = \boldsymbol{\tau}$ *and* $\omega(q) = \tau$.

4. If $V \neq \emptyset$, *go to Step 1.*

iv. If $W \neq \emptyset$, *go to Step ii.*

Output: Automaton G_{lo} *which passes the FP Test in Case (1).*

Algorithm 3.40 ensures that in each potentially P-vocal state $\tilde{q}$, if for all $q \in in(\tilde{q})$ we have $\lambda(q) \in T'_{uc}$, then the outputs $\omega(q)$ which are generated by the system are the same.

Case (2): For some $q, q' \in in(\tilde{q})$, $\omega(q) \neq \omega(q')$ and for every $r' \in A'_{\tilde{q}}(\xi)$ we have $\Sigma^c(r') \neq \emptyset$, or equivalently for all $q \in in(\tilde{q})$, we have $\lambda(q) \in T'_c$. This case deals with the silent paths whose natural projection is the same, all are controllable but the output they generate are not the same.

The modification required in this case is the same as that given for Case (1) in Algorithm 3.40 except that here line (iii). in Algorithm 3.40 should consider $\lambda(q) \in T'_c$. Without loss of generality, we modify line (iii). in Algorithm 3.40 to cover both Cases (1) and (2). We refer to this extended algorithm as Event Split (ES) Algorithm.

Algorithm 3.41. ***Event Split (ES)***

Input: Automaton G_{lo} *which fails the FP Test in Cases (1) or (2).*

i. *Initial step: Let* $W = \tilde{Q}_{pvoc}$

ii. *Choose any* $\tilde{q} \in W$ *and let* $W \longleftarrow W - \{\tilde{q}\}$ *(Suppose it is desired that* $\varpi(\tilde{q}) = \boldsymbol{\tau}$ *where* $\boldsymbol{\tau} \in T_\circ$*).*

iii. *If for all* $q \in in(\tilde{q})$ *we have* $\lambda(q) \in T'_{uc}$ *or* $\lambda(q) \in T'_c$*, then let* $V = in(\tilde{q})$ *and do the following:*

1. *Choose any* $q \in V$. *Let* $V \longleftarrow V - \{q\}$ *and* $\tau = \omega(q)$ *and*
2. *If* $\tau = \tau_\circ$*, then let* $\omega(q) = \boldsymbol{\tau}$.
3. *Otherwise, If* $\tau_\circ \neq \tau \neq \boldsymbol{\tau}$*, let* $Q \longleftarrow Q \cup \{q'\}$ *where* q' *is a new state of the system and do the following:*

   a. *Let* $E = \{q'' \xrightarrow{\sigma} q \mid \exists q'' \in Q, \exists \sigma \in \Sigma_\circ : q = \delta(q'', \sigma)\}$ *be the set of transitions which reach* q *by some observable event.*

   b. *Define a new set of transitions* $F = \{q'' \xrightarrow{\sigma} q' \mid q'' \xrightarrow{\sigma} q \in E\}$ *to consist of transitions which reach the new state* q' *and where there is a one-to-one map between the transitions of* E *and* F.

   c. *Let* $(\Sigma_{uo} \cap \Sigma_{uc}) \longleftarrow (\Sigma_{uo} \cap \Sigma_{uc}) \cup \{\beta\}$ *where* β *is a new unobservable-uncontrollable event of the system.*

   d. *Remove* E *from the system transitions.*

   e. *Add* $F \cup \{q' \overset{\beta}{\dashrightarrow} q\}$ *to the system.*

   f. *Let* $\omega(q') = \boldsymbol{\tau}$ *and* $\omega(q) = \tau$.

4. *If* $V \neq \emptyset$*, go to Step 1.*

iv. *If* $W \neq \emptyset$*, go to Step ii.*

Output: Automaton G_{lo} *which passes the FP Test in Cases (1) and (2).*

The third case requires a different algorithm which is explained in the following section.

3.7.4 FP Test: MSPSS Algorithm

Case (3): For some $q, q' \in in(\tilde{q})$, $\omega(q) \neq \omega(q')$ where for some $r, r' \in A'_{\tilde{q}}(\xi)$ we have $\Sigma^c(r) = \emptyset \neq \Sigma^c(r')$, or equivalently for some $q, q' \in in(\tilde{q})$ we have $\lambda(q) \in T'_c$ and $\lambda(q') \in T'_{uc}$. This case deals with the silent paths whose natural projection is the same, but some of them are controllable and some of them are uncontrollable. Of course, by construction, the outputs of states $q \in in(\tilde{q})$ cannot be the same. The following example demonstrates our solution in this case.

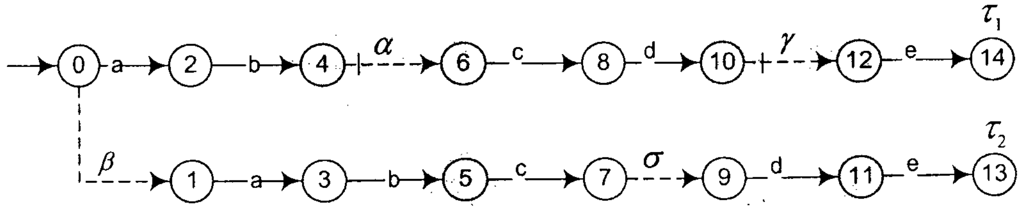


Figure 3.17: Model G_{lo} which violates the Factorization Property in Case (3): The shaded areas are vocalized such that unobservable-controllable transitions are isolated.

Example 3.42. *Fig. 3.17 shows a model G_{lo} where we have $\Sigma_c = \{\alpha, \gamma\}$ and $\Sigma_o = \{a, b, c, d, e\}$ and two silent paths $s_1 = a.b.\alpha.c.d.\gamma.e$ and $s_2 = \beta.a.b.c.\sigma.d.e$ have the same projection $P_{lo}(s_1) = P_{lo}(s_2) = \boldsymbol{a.b.c.d.e}$. Therefore, $\tau_1 = \omega(s_1)$*

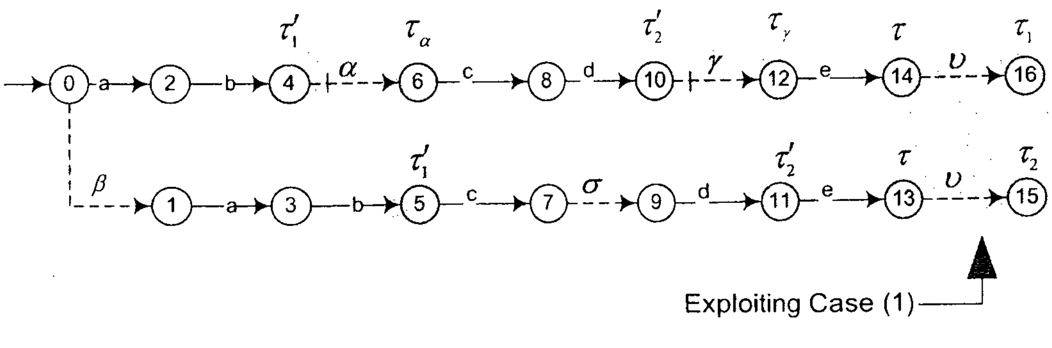


Figure 3.18: The shaded area in Fig. 3.17 has been assigned proper outputs.

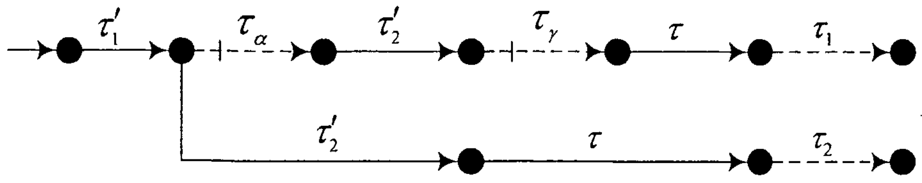


Figure 3.19: High-level model G_{hi} which is obtained from Fig. 3.18.

is controllable and $\tau_2 = \omega(s_2)$ is uncontrollable while their corresponding silent paths have the same natural projection. It is easy to see that here, and in general in Case (3) that FP Test fails, all observable events are uncontrollable and those silent paths which are controllable include unobservable-controllable events $\alpha, \gamma \in \Sigma_{uo} \cap \Sigma_c$. The procedure which we follow for modifying the vocalization of the system isolates the unobservable-controllable events as individual silent paths. Two shaded areas in Fig. 3.17 encompass two unobservable-controllable events α and γ whose exiting and entering states are to be vocalized; i.e. states 4, 6, 10 *and* 12 *are vocalized. On the other hand, the Factorization Property requires that if state* 4 *(resp. state* 10*) is vocalized, then state* 5 *(resp. state* 11*) should be vocalized too. Fig. 3.18 shows the outputs which have been assigned to the newly vocalized states in Fig. 3.17. The following assignments are notable: $\omega(4) = \omega(5) = \tau_1'$, $\omega(6) = \tau_\alpha$, $\omega(10) = \omega(11) = \tau_2'$ and $\omega(12) = \tau_\gamma$ where we have $\tau_1', \tau_2' \in T_{uc}$ and $\tau_\alpha, \tau_\gamma \in T_c^{uo}$. Furthermore, states* 13 *and* 14 *now satisfy the conditions in Case (1). Therefore, a new unobservable-uncontrollable transition ν is added to states* 13 *and* 14 *to have $\omega(13) = \omega(14) = \tau \in T_{uc}$. Note how the initial vocalization τ_1 and τ_2 are assumed uncontrollable thereafter. Fig. 3.19 shows the final high-level model G_{hi} where $\theta(s_1) = \tau_1'\tau_\alpha\tau_2'\tau_\gamma\tau\tau_1$, $\theta(s_2) = \tau_1'\tau_2'\tau\tau_2$ with $P_{hi}(\theta(s_1)) = P_{hi}(\theta(s_2)) = \tau_1'\tau_2'\tau \in T_o^+$.*

We now give a general algorithm for dealing with the failure of FP Test in

Case (3). Define a set

$$\mathfrak{T} = \{(p, \sigma, p') \mid \exists p, p' \in Q,\ \exists \tilde{p} \in \tilde{Q} - \tilde{Q}_{pvoc},\ \exists \sigma \in \Sigma_c \cap \Sigma_{uo} : \\ p, p' \in \tilde{p},\ \ p' = \delta(p, \sigma)\}. \qquad (3.33)$$

to consist of the transitions which are unobservable and controllable and whose beginning and ending states are in a state $\tilde{p}$ which is not potentially P-vocal (i.e. $p, p' \in \tilde{p} \in \tilde{Q} - \tilde{Q}_{pvoc}$). The set $\mathfrak{T}$ includes the transitions which might be selected to be vocalized such that unobservable-controllable transitions are isolated in Case (3). More specifically, for a state estimate $\tilde{q}$ define the set

$$\mathfrak{T}_{|\tilde{q}} = \{(p, \alpha, p') \in \mathfrak{T} \mid \exists r \in \Sigma^* : P_{lo}(r) \neq \epsilon \ \text{ and } \ \delta(p', r) \in in(\tilde{q})\} \qquad (3.34)$$

to include a subset of $\mathfrak{T}$ which can reach the incoming states of $\tilde{q}$. We refer to $\mathfrak{T}_{|\tilde{q}}$ as the restriction of $\mathfrak{T}$ to $\tilde{q}$.

Lemma 3.43. *The set $\mathfrak{T}$ in (3.33) is finite.*

Proof: Let $Trns = \{(p, \sigma, q) \mid \exists p, q \in Q,\ \sigma \in \Sigma :\ q = \delta(p, \sigma)\}$ be the set of transitions of the system. Clearly $Trns$ is finite and we have $\mathfrak{T} \subseteq Trns$, hence, $\mathfrak{T}$ is finite. $\square$

Consider a potentially P-vocal state $\tilde{q}$ and a potentially P_{lo}-silent path ξ which enters $\tilde{q}$. Define a property $\Pi_{\tilde{q}}$ as

$$\Pi_{\tilde{q}}(\xi) \text{ holds} \iff \forall r' \in A'_{\tilde{q}}(\xi) : \Sigma^c(r') = \emptyset \ \text{ or } \ \forall r' \in A'_{\tilde{q}}(\xi) : \Sigma^c(r') \neq \emptyset (3.35)$$

The property $\Pi_{\tilde{q}}$ holds if and only if all of the silent paths which enter $\tilde{q}$ are controllable or uncontrollable. As the first step in any remedy to Case (3), it should be ensured that for every ξ which enters $\tilde{q}$, we have $\Pi_{\tilde{q}}(\xi)$. In order to do so, some transitions from $\mathfrak{T}$ are selected and vocalized such that for every P_{lo}-silent path ξ which enters $\tilde{q}$, $\Pi_{\tilde{q}}(\xi)$ holds. We first show how a single potentially P-vocal state is treated in this case.

Algorithm 3.44. ***Single State Projected String Split (SSPSS):***
Input: A potentially P-vocal state $\tilde{q}$ consisting of incoming states with outputs under the map λ that are both controllable and uncontrollable.

i. Find $\mathfrak{T}$ and $\mathfrak{T}_{|\tilde{q}}$ and let $\mathfrak{T} \longleftarrow \mathfrak{T} - \mathfrak{T}_{|\tilde{q}}$.

ii. Choose any transition $(p, \alpha, p') \in \mathfrak{T}_{|\tilde{q}}$. Let $\mathfrak{T}_{|\tilde{q}} \longleftarrow \mathfrak{T}_{|\tilde{q}} - \{(p, \alpha, p')\}$ and do the followings:

1. If state p is silent, vocalize it such that $\omega(p) \neq \tau_o$.

2. Vocalize state p' such that $\omega(p') = \tau_\alpha \in T_c^{uo}$.

iii. If $|\mathfrak{T}_{|\tilde{q}}| > 0$, then go to Step ii.

Output: A potentially P-vocal state $\tilde{q}$ that for every potentially P_{lo}-silent path ξ which enters it, $\Pi_{\tilde{q}}(\xi)$ holds.

Proposition 3.45. *Algorithm SSPSS terminates after a finite number of iterations.*

Proof: By Lemma 3.43, $\mathfrak{T}$ is finite; hence, $\mathfrak{T}_{|\tilde{q}}$ is finite. Therefore, after a finite number of iterations Algorithm SSPSS stops. □

Let $SSPSS(\tilde{q})$ denote the application of the SSPSS algorithm to a potentially P-vocal state $\tilde{q}$. If the states $p, p' \in \tilde{p}$ in a transition $(p, \alpha, p') \in \mathfrak{T}$ are vocalized, then $\tilde{p}$ will be considered a potentially P-vocal state. Let $\tilde{Y}$ denote all the new potentially P-vocal states $\tilde{p}$ which are formed by the SSPSS algorithm.

Notation: We write $(\tilde{Y}, \omega_{new}) = SSPSS(\tilde{q})$ to mean that $\tilde{Y}$ is the set of new potentially P-vocal states and ω_{new} is the updated output map when SSPSS Algorithm is applied to $\tilde{q}$.

In Proposition 3.46 we study an important characteristics of the new P-vocal states $\tilde{p} \in \tilde{Y}$ which are formed by SSPSS Algorithm.

Proposition 3.46. *Let $(\tilde{Y}, \omega_{new}) = SSPSS(\tilde{q})$. Then, (i) for every potentially P_{lo}-silent path ξ which enters a state $\tilde{p} \in \tilde{Y}$ and for every sequence $r' \in A'_{\tilde{p}}(\xi)$, we have $\Sigma^c(r') = \emptyset$ and (ii) for every newly formed potentially P_{lo}-silent path ς which enters $\tilde{q}$ and for every sequence $r' \in A'_{\tilde{q}}(\varsigma)$, we have $\Sigma^c(r') = \emptyset$.*

Proof: (i): Suppose in the application of SSPSS algorithm to a potentially P-vocal state $\tilde{q}$, the states $p, p' \in \tilde{p} \in \tilde{Y}$ in a transition $(p, \alpha, p') \in \mathfrak{T}_{\tilde{q}}$ have been vocalized. Then assume ξ is a potentially P_{lo}-silent path which enters $\tilde{p}$ from a potentially P-vocal state, say $\tilde{x}$; i.e. $\tilde{p} = \delta(\tilde{x}, \xi)$. Consider any sequence $r' \in A'_{\tilde{p}}(\xi)$ such that for some $x \in out(\tilde{x})$, we have $\delta(x, r') \in in(\tilde{p})$. Then note that by construction, $\tilde{q}$ is reachable from $\tilde{p}$. This further implies that $\tilde{q}$ is reachable from $\tilde{x}$; i.e., for some $s \in \Sigma^*$ we have

$$\delta(x, r's) \in in(\tilde{q}). \tag{3.36}$$

Now, by contradiction, if $\Sigma^c(r') \neq \emptyset$, then it should be the case that for some $r'_1, r'_2 \in \Sigma^*$ and $\gamma \in \Sigma_c \cap \Sigma_{uo}$, we have $r' = r'_1 \gamma r'_2$. Therefore from (3.36)

$$\delta(x, r'_1 \gamma r'_2 s) \in in(\tilde{q}). \tag{3.37}$$

Let $y_1 = \delta(x, r'_1)$ and $y_2 = \delta(x, r'_1 \gamma)$. Therefore, (3.37) implies $(y_1, \gamma, y_2) \in \mathfrak{T}_{\tilde{q}}$. Then, since the SSPSS algorithm has been applied to $\tilde{q}$, we should have $\omega(y_1), \omega(y_2) \neq \tau_\circ$ which is a contradiction since ξ is a potentially P_{lo}-silent path and r' should be silent. Therefore, γ does not exist and we have $\Sigma^c(r') = \emptyset$. This completes the proof for part (i). Part (ii) is proved similarly. □

Proposition 3.46 implies that $\Pi_{\tilde{p}}(\xi)$ is true for every state $\tilde{p} \in \tilde{Y}\dot{\cup}\{\tilde{q}\}$ and every Potentially P_{lo}-silent path ξ which enters $\tilde{p}$. Specifically, the condition in Case (1) of the FP Test is recovered at the incoming states of $\tilde{p} \in \tilde{Y}\dot{\cup}\{\tilde{q}\}$. However, by applying algorithm SSPSS to a potentially P-vocal state $\tilde{q}$, other potentially P-vocal states might be affected. Thus, we need to check other potentially P-vocal states $\tilde{q} \in \tilde{Q}_{pvoc}$ and possibly apply algorithm SSPSS to them until no other potentially P-vocal state violates the property $\Pi_{\tilde{q}}$ given in (3.35) and a fixed point is reached. We show a fixed point is reached in a finite number of iterations.

Algorithm 3.47. ***Multi State Projected String Split (MSPSS)***:

Input: A model $G_{lo} = (Q_G, \Sigma, T \cup \{\tau_\circ\}, \sigma_G, \omega, q_{\circ,G})$ which fails the FP Test in Case (3).

i. Let $i = 1$ and $\tilde{W}_i = \tilde{Q}_{pvoc}$.

ii. Choose any $\tilde{q} \in \tilde{W}_i$ and let $\tilde{W}_i \longleftarrow \tilde{W}_i - \{\tilde{q}\}$.

iii. If there exist states $x_1, x_2 \in in(\tilde{q})$ such that $\lambda(x_1) \in T'_c$ and $\lambda(x_2) \in T'_{uc}$, then do the following:

1. Let $(\tilde{Y}, \omega_{new}) = SSPSS(\tilde{q})$.

2. Redefine $G_{lo} = (Q_G, \Sigma, T_{new} \cup \{\tau_\circ\}, \sigma_G, \omega_{new}, q_{\circ,G})$ where T_{new} has been updated according to ω_{new}.

3. Let $(J_{lo}, \lambda) = INIT(G_{lo})$.

4. Let $\tilde{K}_{lo} = \tilde{J}_{lo} \times \tilde{G}_{lo}$ where $\tilde{J}_{lo}$ and $\tilde{G}_{lo}$ are the observer automata of J_{lo} and G_{lo}. Also, let the states of $\tilde{K}_{lo}$ be of the form $\tilde{q}_K = (\tilde{q}_J, \tilde{q}_G)$ where $\tilde{q}_J$ and $\tilde{q}_G$ are the states of $\tilde{J}_{lo}$ and $\tilde{G}_{lo}$.

5. Let $\tilde{W}_{i+1} = \{\tilde{q}_K \mid \exists \tilde{q}_J, \tilde{q}_G : \tilde{q}_K = (\tilde{q}_J, \tilde{q}_G)$ and $\tilde{q}_G \in \tilde{W}_i\}$.

6. Let $G_{lo} \longleftarrow J_{lo}$ and $i \longleftarrow i + 1$.

iv. If $\mathfrak{T} = \emptyset$ or $\tilde{W}_i = \emptyset$, then exit; otherwise go to Step ii.

Output: A model G_{lo} whose outputs satisfy property $\Pi_{\tilde{q}}$ given in (3.35).

3.7.5 Convergence of MSPSS Algorithm

To show the convergence of the MSPSS algorithm, we first show how the set $\mathfrak{T}$ is updated. Let $(p, \alpha, p') \in \mathfrak{T}$ be an unobservable-controllable transition. Suppose in line iii.3 of the MSPSS algorithm, the transition (p, α, p') which is within the behavior of G_{lo} splits as a result of using OCC and OOC algorithms which are exploited in INIT Procedure. We denote these splits as a set $split((p, \alpha, p'))$. Let δ_G and δ_J be the partial transition maps of G_{lo} and J_{lo} respectively in lines iii.2 and iii.3 of MSPSS Algorithm. Also let Q_J and $q_{o,J}$ respectively denote the state set and the initial state of J_{lo}. The set $split((p, \alpha, p'))$ is characterized by

$$\begin{aligned} split((p,\alpha,p')) = \{(p_i,\alpha,p_i') \mid \exists p_i, p_i' \in Q_J, \exists \alpha \in \Sigma_{uo} \cap \Sigma_c, \\ \forall r, s \in \Sigma^*, \forall m \in \mathbb{N}, \exists i, j \in \mathbb{N} : 1 \le i, j \le m \\ (\delta_J(q_{o,J}, s) = p_i \implies \delta_G(q_{o,G}, s) = p) \text{ and} \\ (\delta_G(p', r)! \iff \delta_J(p_i', r)!) \quad \text{and} \quad (\delta_J(p_i', r) = \delta_J(p_j', r))\}. \end{aligned} \tag{3.38}$$

Next, we show how the membership of a transition (p, α, p') of G_{lo} in $\mathfrak{T}_{\tilde{q}}$ with $\tilde{q} \in \tilde{W}_i$ extends to the membership of the transitions $split(p, \alpha, p')$ of J_{lo} in $\mathfrak{T}_{\tilde{q}'}$ with $\tilde{q}' \in \tilde{W}_{i+1}$.

Lemma 3.48. *Let $split((p, \alpha, p'))$ be the set of unobservable-controllable transitions in J_{lo} which is characterized by (3.38) where $(p, \alpha, p') \in \mathfrak{T}_{|\tilde{q}}$ is an unobservable-controllable in G_{lo}. Then, for every potentially P-vocal state $\tilde{q}' \in \tilde{W}_{i+1}$ in J_{lo} either $split((p, \alpha, p')) \cap \mathfrak{T}_{|\tilde{q}'} = \emptyset$ or $split((p, \alpha, p')) \subseteq \mathfrak{T}_{|\tilde{q}'}$.*

Proof: Consider any P-vocal state $\tilde{q}' \in \tilde{W}_{i+1} \subseteq \tilde{Q}_J$ where $\tilde{Q}_J$ is the state set of J_{lo}. We show that if $split((p, \alpha, p')) \cap \mathfrak{T}_{|\tilde{q}'} \neq \emptyset$, then $split((p, \alpha, p')) \subseteq \mathfrak{T}_{|\tilde{q}'}$. Consider a transition $(c, \alpha, c') \in split((p, \alpha, p'))$. If $(c, \alpha, c') \in \mathfrak{T}_{|\tilde{q}'}$, then by the definition of $\mathfrak{T}_{|\tilde{q}'}$ there exists $r \in \Sigma^*$ for which we have $\delta(c', r) \in in(\tilde{q}')$. On the other hand, definition of $split((p, \alpha, p'))$ implies that all the transitions $(p_i, \alpha, p_i') \in split((p, \alpha, p'))$ extend the same in $L(G_{lo})$. Specifically, for every $r \in \Sigma^*$ that $\delta(c', r) \in in(\tilde{q}')$ we will have $\delta(p_i', r) \in in(\tilde{q}')$ for $1 \leq i \leq |split((p, \alpha, p'))|$ which implies $(p_i, \alpha, p_i') \in \mathfrak{T}_{|\tilde{q}'}$. This proves $split((p, \alpha, p')) \subseteq \mathfrak{T}_{|\tilde{q}'}$. □

Lemma 3.48 implies that in finding $\mathfrak{T}_{\tilde{q}}$, the set $split((p, \alpha, p'))$ can be treated as a single transition in G_{lo} since a P-vocal state $\tilde{q} \in \tilde{W}_{i+1}$ is either reachable from all of the transitions $(c, \alpha, c') \in split((p, \alpha, p'))$ or it is not reachable from any of them. We use Lemma 3.48 to show that the convergence of the MSPSS algorithm is independent from the split action that might occur in *INIT* Procedure in line iii.3 of the MSPSS algorithm. We now show the MSPSS algorithm stops in a finite number of iterations.

Proposition 3.49. *The MSPSS algorithm terminates in a finite number of iterations and specifically, after at most N times of calling SSPSS Algorithm where N is the number of unobservable-controllable transitions in* $\mathfrak{T}$ *at the initial step* $i = 0$.

Proof: Let $N = |\mathfrak{T}|$ at the initial step $i = 0$. In each iteration of the MSPSS algorithm a state $\tilde{q} \in \tilde{W}_i$ is chosen and removed from $\tilde{W}_i$ in line ii. If the statement in line iii. is true, then the SSPSS algorithm is called by which at least one transition (p, α, p') or its subsequent splits, i.e. $split((p, \alpha, p'))$ by Lemma 3.48, are removed from $\mathfrak{T}$. On the other hand, if the statement in line iii. is not true, then the inner loop in line iii. is skipped and the system remains the same while the cardinality of $\tilde{W}_i$ has decreased by one in line ii.

Therefore, either $\mathfrak{T} = \emptyset$ after at most N times of executing the inner loop or $|\tilde{W}_i| = 0$ after a finite number of times the outer loop is performed. $\square$

Proposition 3.46 implies that in treatment of Case (3), the new potentially P-vocal states which are formed by the MSPSS algorithm, in the worst case, are such that conditions in Case (1) are recovered for them. This implies that the output of a potentially P-vocal state $\tilde{q}$ will be well-defined after at most two treatments, first by the MSPSS algorithm and second by the ES algorithm. Therefore, by distinguishing three cases in the FP Test, the reporting map of the system can be examined and modified in the order given below.

Procedure 3.50. ***Factorization Property (FP)***

(i) Perform FP Test.

(ii) Apply MSPSS Algorithm.

(iii) Apply ES Algorithm.

We next, show how the other properties which we require for hierarchical consistency can be satisfied.

3.7.6 SCE and UET Algorithms

While the Factorization Property serves as a fundamental property for the HSC problem under partial observation, there aret other properties which we intend to satisfy in the system for the purpose of exercising control. Recall that to achieve hierarchical consistency it is sufficient for the system to satisfy the UUC, UUPO and PO-SOCC properties. In the following we show how these control properties can be ensured in the system.

Satisfying UUC Property: Define a new set of labels

$$T_c^{uo} = \{\tau_\sigma \mid \sigma \in \Sigma_{uo} \cap \Sigma_c\} \tag{3.39}$$

and add them to the set of high-level events. Also by definition we let $T_c^{uo} \subseteq T^{uo} \cap T_c$. To satisfy the UUC property, we assign outputs from the set T_c^{uo} to the destination node of every unobservable-controllable transition of every unobservable-controllable silent path.

Algorithm 3.51. ***Single Controllable Event (SCE)***

Input: A model $G_{lo} = (Q, \Sigma, T \cup \{\tau_\circ\}, \sigma, \omega, q_\circ)$ whose P-vocal states $\tilde{Q}_{pvoc}$ might not satisfy the UUC property.

i. *Let $W = \tilde{Q}_{pvoc}$.*

ii. *Choose any state $\tilde{q} \in W$ and let $W \longleftarrow W - \{\tilde{q}\}$.*

iii. *For every incoming state $q \in in(\tilde{q})$ and for every sequence $r \in \Sigma_{uo}^*$ that $\omega(\delta(q, r)) \neq \tau_\circ$ if $|\Sigma^c(r)| > 1$, then let $\hat{\omega}(sr) = \tau$ and do the following.*

1. *Choose $r_j \in \Sigma^*\Sigma_c \cap \Sigma_{uo}^*$ for $j = 1, \cdots, m$ and $u \in \Sigma_{uc}^* \cap \Sigma_{uo}^*$ such that $|\Sigma^c(r_j)| = 1$ for $j = 1, \cdots, m$ and $r = r_1 \cdots r_m u$.*

2. *Let $\{\sigma_j\} = \Sigma^c(r_j)$ for $j = 1, \cdots, m$. Clearly we have $\sigma_j \in \Sigma_c \cap \Sigma_{uo}$.*

3. *Let $\hat{\omega}(sr_1 \cdots r_j) = \tau_{\sigma_j} \in T_c^{uo}$ for $j = 1, \cdots, m-1$.*

4. *If $u \neq \epsilon$ let $\hat{\omega}(sr_1 \cdots r_m) = \tau_{\sigma_m} \in T_c^{uo}$. Otherwise if $u = \epsilon$ and $\tau \neq \tau_{\sigma_m}$, do the followings.*

   a. *Let $(\Sigma_{uo} \cap \Sigma_{uc}) \longleftarrow (\Sigma_{uo} \cap \Sigma_{uc}) \cup \{\beta\}$ and $Q \longleftarrow Q \cup \{q'\}$ where β and q' are new event and state of the system.*

   b. *For $r' < r_m$ and $\sigma_m \in \Sigma_c \cap \Sigma_{uo}$ that $r_m = r'\sigma_m$, let $q_1 = \delta(q_\circ, sr')$ and $q_2 = \delta(q_\circ, sr)$.*

   c. *Remove the transition $q_1 \overset{\sigma_m}{\dashrightarrow} q_2$ and add a new transition $q_1 \overset{\sigma_m}{\dashrightarrow} q' \overset{\beta}{\dashrightarrow} q_2$ instead to the system.*

   d. *Let $\omega(q') = \hat{\omega}(sr) = \tau_{\sigma_m}$ and $\omega(q_2) = \hat{\omega}(sr\beta) = \tau \in T_{uo} \cap T_{uc}$.*

5. If $W \neq \emptyset$, then go to ii.

Output: A model G_{lo} that satisfies the UUC property.

Therefore, in the SCE algorithm if the label of an unobservable-controllable transition is σ, the assigned label will be $\tau_\sigma \in T_c^{uo}$. In this way, by choosing labels from T_c^{uo}, we include the information about the unobservable-controllable events in the high-level model G_{hi}. The UUC property and the modification performed in SCE Algorithm ensure that if an unobservable silent path is controllable, then its controllability originates from only a single controllable event, say $\sigma \in \Sigma_c$, and furthermore the occurrence of σ is shown in G_{hi} with a unique symbol τ_σ.

Satisfying The UUPO Property: The UUPO property restricts the silent paths whose output are observable and controllable. Specifically, if those silent paths start with an unobservable segment, the segment should be uncontrollable. Assume initially $(G_{lo}, \lambda) = INIT(G_{lo})$.

Algorithm 3.52. ***Uncontrollable Exiting Transitions (UET)***

Input: A model G_{lo} whose P-vocal states $\tilde{Q}_{pvoc}$ may not satisfy the UUPO property.

i. For every P-vocal state $\tilde{q} \in \tilde{Q}_{pvoc}$ and every outgoing state $q \in out(\tilde{q})$ if $\lambda(q) \in T_c'$, then let $\omega(q) = \tau$ where $\tau \in T_c \cap T_{uo}$

Output: A model G_{lo} whose P-vocal states $\tilde{Q}_{pvoc}$ satisfy the UUPO property.

3.7.7 PO-SOCC Algorithm

Suppose two nodes $\tilde{q}_1$ and $\tilde{q}_2$ are P-partners in the reachability tree (RT) of $\tilde{G}_{lo}$ such that for some $\tilde{p} \in \tilde{Q}_{pvoc}\dot{\cup}\{\tau_o\}$ and some $\mathbf{S}_1, \mathbf{S}_2 \in \Sigma_o^+$, we have $\tilde{q}_1 = \tilde{\delta}(\tilde{p}, \mathbf{S}_1)$ and $\tilde{q}_2 = \tilde{\delta}(\tilde{p}, \mathbf{S}_2)$ and by the P-partnership status there exist sequences $\mathbf{R}, \mathbf{R}_2, \mathbf{R}_3 \in \Sigma_o^+$ such that $\mathbf{S}_1 = \mathbf{R}\mathbf{R}_1$ and $\mathbf{S}_2 = \mathbf{R}\mathbf{R}_2$ and at least one of the sequences $\mathbf{R}_1$ or $\mathbf{R}_2$ is P_{lo}-uncontrollable from $\tilde{\delta}(\tilde{p}, \mathbf{R})$. It can be seen that the largest common prefix of $\mathbf{S}_1$ and $\mathbf{S}_2$ always includes $\mathbf{R}$. Therefore, without loss of generality, assume $\mathbf{R}$ is the largest prefix common to $\mathbf{S}_1$ and $\mathbf{S}_2$.

In order to modify the system to satisfy the PO-SOCC property, the node which is reached from $\tilde{p}$ by $\mathbf{R}$ in the RT of $\tilde{G}_{lo}$ should be vocalized. This is very similar to the SOCC algorithm where the common controllable path should be vocalized. In fact, satisfying the PO-SOCC property for G_{lo}, can be seen as satisfying SOCC property for $\tilde{G}_{lo}$ where the terms controllable and uncontrollable in G_{lo} are respectively replaced with the terms P_{lo}-controllable and P_{lo}-uncontrollable in $\tilde{G}_{lo}$.

Notation: We use the subroutine **SOCC**, as the counterpart of SOCC algorithm in the partial observation case. Specifically, $\tilde{H}_{lo} = \mathbf{SOCC}(\tilde{G}_{lo})$ is obtained as follows.

(i) Assign outputs $\tau \in T_\circ$ to suitable nodes of the RT of $\tilde{G}_{lo}$ such that no two nodes are partners there and

(ii) split the states of $\tilde{G}_{lo}$ as needed such that the **OCC** property is satisfied.

Here, the terms controllability and uncontrollability in the application of OCC and SOCC to G_{lo} are respectively replaced with P_{lo}-controllability and P_{lo}-uncontrollability in the application of **OCC** and **SOCC** to $\tilde{G}_{lo}$. □

Therefore, let G_{lo} be a Moore automaton with an output map ω which satisfies the Factorization Property, the UUC and UUPO properties. Also, let $\tilde{G}_{lo}$ be the observer automaton of G_{lo} with the output map ϖ.

Algorithm 3.53. ***PO-SOCC Algorithm:***

Input: Model G_{lo} which is not PO-SOCC.

i. Let $\tilde{H}_{lo} = \boldsymbol{SOCC}(\tilde{G}_{lo})$.

ii. Let $K_{lo} = \tilde{H}_{lo} || G_{lo}$.

iii. Let $q_k = (\tilde{q}_h, q_g)$ *be any state of* K_{lo} *where* $\tilde{q}_h$ *and* q_g *respectively denote the states of* $\tilde{H}_{lo}$ *and* G_{lo}. *Furthermore, let* $q_g \in \tilde{q}_g$ *where* $\tilde{q}_g$ *is a state of* $\tilde{G}_{lo}$.

iv. Let $\varphi : Q_h \longmapsto T_\circ \cup \{\tau_\circ\}$ *be the output map of* $\tilde{H}_{lo}$. *Then, define the output map* $\vartheta : Q_k \longmapsto T \cup \{\tau_\circ\}$ *for* K_{lo} *as follows:*

$$\vartheta(q_k) = \begin{cases} \omega(q_g), & \textit{if } \varpi(\tilde{q}_g) \neq \tau_\circ; \\ \varphi(\tilde{q}_h), & \textit{if } \varpi(\tilde{q}_g) = \tau_\circ \textit{ and } q_g \in in(\tilde{q}_g); \\ \tau_\circ, & \textit{otherwise.} \end{cases}$$

v. Apply the OCC and OOC algorithms to K_{lo} *and call the result* K_{lo} *again.*

vi. Let $G_{lo} \longleftarrow K_{lo}$.

Output: A model G_{lo} *which is PO-SOCC (but may not satisfy the Factorization Property).*

The output assignment in line iv. of the PO-SOCC algorithm implies that if a state $\tilde{q}_g$ is P-vocal, then q_k inherits the outputs of $q_g \in \tilde{q}_g$; however, if $\tilde{q}_g$ is silent, which does not limit $\tilde{q}_h$ from being P-vocal, then q_k inherits the outputs of $q_g \in \tilde{q}_g$ only if $q_g \in in(\tilde{q}_g)$ is an incoming state. q_k remains silent otherwise. It should be noted that the procedure given in the PO-SOCC algorithm does not necessarily give a Moore automaton that satisfies the Factorization Property. In fact, after applying the PO-SOCC algorithm to G_{lo} we need to apply the Factorization Property Procedure to G_{lo} one more time so that we are guaranteed the Factorization Property and the other properties of UUC, UUPO and PO-SOCC are satisfied. Fig. 3.20 shows the order in which G_{lo} should be treated by the FP Procedure and PO-SOCC algorithm. We show that the final system which is obtained from the procedures in Fig. 3.20 is still PO-SOCC

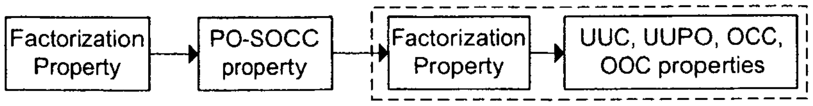


Figure 3.20: Logical ordering for satisfying Factorization Property and PO-SOCC property

and, of course by construction, satisfies the Factorization Property, UUC and UUPO properties.

Notation: Let $H = SCE \circ UET \circ FP(G)$ denote the function $H = SCE(UET(FP(G)))$ in Proposition 3.54 where $SCE(.)$, $UET(.)$ and $FP(.)$ denote the application of the SCE algorithm, UET algorithm and FP procedure to an automaton.

Proposition 3.54. *Let G be a model which satisfies the Factorization Property and M be computed from G as follows:*

(i) H=PO-SOCC(G)

(ii) $M = SCE \circ UET \circ FP(H)$.

Then M is PO-SOCC.

Proof: Let $\tilde{G}$, $\tilde{H}$ and $\tilde{M}$ be the observer automata for G, H and M. We define the output maps ω_G, ω_H, ω_M and $\tilde{\omega}_G$, $\tilde{\omega}_H$ and $\tilde{\omega}_M$ as usual for these models. Then, by contradiction, suppose M is not PO-SOCC. Therefore, there should exist sequences $\mathbf{S} \in \Sigma_o^+ \cup \{\epsilon\}$ and $\mathbf{R}, \mathbf{R}_1, \mathbf{R}_2 \in \Sigma_o^+$ such that the nodes $\mathbf{SRR}_1$ and $\mathbf{SRR}_2$ are P-partners in the RT of $\tilde{M}$ where $\mathbf{S} \in L(\tilde{M})$ is either P-vocal or the null sequence (ϵ) and $\mathbf{RR}_1$ and $\mathbf{RR}_2$ are P_{lo}-silent paths with $\tilde{\omega}_M(\mathbf{SRR}_1), \tilde{\omega}_M(\mathbf{SRR}_2) \in T_c$ (i.e. $\mathbf{RR}_1$ and $\mathbf{RR}_2$ are P_{lo}-controllable P_{lo}-silent paths).

Next, we investigate how different algorithms including the SCE and UET algorithms and FP Procedure modify the system and how they affect the P_{lo}-silent paths in the systems. First, note that the SCE and UET algorithms do not affect the controllability of observable silent paths. Furthermore, the ES algorithm which is exploited in FP Procedure, does not generate new P-vocal states. Thus, it can be seen that P_{lo}-controllability and P_{lo}-uncontrollability do not change as a result of the application of the SCE, UET and ES algorithms. Thus, only the MSPSS algorithm, with the SSPSS algorithm as its core, might introduce new P-vocal states to the system. Now, note that Proposition 3.46 states that all P_{lo}-silent paths which are formed by the SSPSS algorithm are P_{lo}-uncontrollable and hence would generate uncontrollable outputs $\tau \in T_{uc}$. This implies P-vocal nodes $\mathbf{SRR}_1$ and $\mathbf{SRR}_2$ could not have been formed by the SSPSS algorithm. Specifically, the the nodes $\mathbf{SRR}_1$ and $\mathbf{SRR}_2$ have not been vocalized by $SCE \circ OG \circ FP$ in the RT of $\tilde{M}$. This implies they have been P-vocal in the RT of $\tilde{H}$. However, the P-vocal node $\mathbf{S}$ in the RT of $\tilde{M}$ may not be a P-vocal node in the RT of $\tilde{H}$.

Now suppose, the node $\mathbf{S}$ is not P-vocal in the RT of $\tilde{H}$. Let $\mathbf{Y} < \mathbf{S}$ be the largest prefix of $\mathbf{S}$ which is either P-vocal in the RT of $\tilde{H}$ or is the null sequence ϵ and $\mathbf{V} \in \Sigma_o^*$ be a sequence such that $\mathbf{S} = \mathbf{YV}$. First, note that this arrangement, implies $\mathbf{VRR}_1$ and $\mathbf{VRR}_2$ are P_{lo}-silent paths in the RT of $\tilde{H}$. Then, note that $\mathbf{VRR}_1$ and $\mathbf{VRR}_2$ which start from the node $\mathbf{Y}$ are P_{lo}-controllable and share a segment $\mathbf{VR}$ which is in turn followed by $\mathbf{R}_1$ and $\mathbf{R}_2$ where at least one of $\mathbf{R}_1$ or $\mathbf{R}_2$ is P_{lo}-uncontrollable. In other words, nodes $\mathbf{YVRR}_1$ and $\mathbf{YVRR}_2$ are P-partners in the RT of $\tilde{H}$ which is a contradiction since H is PO-SOCC. Therefore, no two nodes with controllable outputs are P-partners in $\tilde{M}$ and thus M is PO-SOCC. □

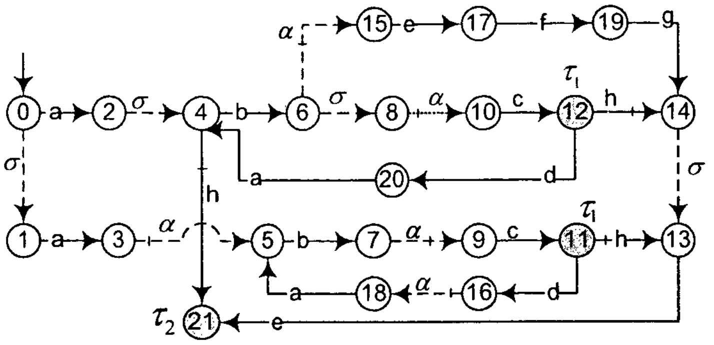


Figure 3.21: Low-level model G_{lo}: Shaded states are vocal.

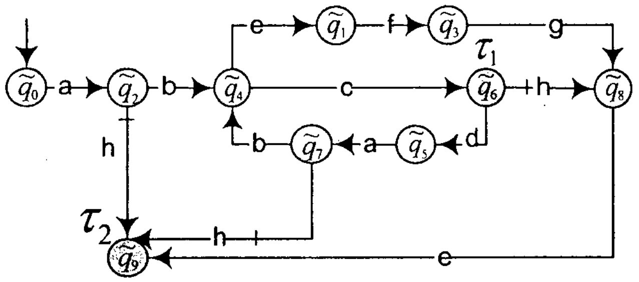


Figure 3.22: Observer automaton $\tilde{G}_{lo}$: Shaded states are P-vocal.

$$\begin{tikzcd}
\arrow[r] & \bullet \arrow[r, "\tau_1"] \arrow[d, "\tau_2"'] & \bullet \arrow[loop right, "\tau_1"] \arrow[dl, bend left, "\tau_2"] \\
& \bullet &
\end{tikzcd}$$

Figure 3.23: High-level model G_{hi}.

3.7.8 An Illustrative Example

Example 3.55. *Fig. 3.21 shows a Moore automaton G_{lo} where $\Sigma = \{a, b, c, d, e, f, g, h, \alpha, \sigma\}$, $\Sigma_c = \{h, \alpha\}$ and $\Sigma_{uo} = \{\alpha, \sigma\}$. The unobservable events are shown with the dashed lines. $T = T_c = T_\circ = \{\tau_1, \tau_2\}$ is the high-level alphabet. Fig. 3.22 shows the observer automaton $\tilde{G}_{lo}$. Let the output maps $\omega, \hat{\omega}, \varpi, \tilde{\omega}$ be defined as before. In Fig. 3.22 $\tilde{Q}_{poc} = \{\tilde{q}_6, \tilde{q}_9\}$, $\varpi(\tilde{q}_6) = \tau_1 \in T_c$ and $\varpi(\tilde{q}_9) = \tau_2 \in T_c$. It can be verified that G_{lo} satisfies the Factorization Property (3.1). Fig. 3.23 shows the high-level model G_{hi}.*

A: Satisfying PO-SOCC Property

It can be verified that the node ***a.b.e.f.g.e*** *is P-partner with the node* ***a.b.c*** *in the RT of $\tilde{G}_{lo}$ in Fig. 3.22.*

PO-SOCC Algorithm

(i) State $\tilde{q}_4$ in Fig. 3.22 is vocalized by ***SOCC*** *subroutine so that $\varpi(\tilde{q}_4) = \tau_3$.*

(ii) Fig. 3.24 shows the resulting model G_{lo} with the newly-induced output map ω (See the PO-SOCC algorithm for details).

(iii) In Fig. 3.24, the incoming states of $\tilde{q}_4$ are vocalized so that $\omega(6) = \omega(7) = \tau_3$.

(iv) State $\tilde{q}_9$ in Fig. 3.22 is split to satisfy the ***OCC*** *property. Thus in Fig. 3.24 we have $\omega(21) = \tau_c$ and $\tau_{2,uc} = \omega(26) = \tau_{2,uc}$.*

B: Satisfying Factorization Property

The resulted model in Fig. 3.24 fails the FP test in case (3); in particular, $\lambda(6) \in T'_c$ and $\lambda(7) \in T'_{uc}$ where λ is defined by $(H_{lo}, \lambda) = INIT(G_{lo})$ in Section 3.7.2. Thus, the FP procedure is applied to modify the system.

(1): MSPSS Algorithm

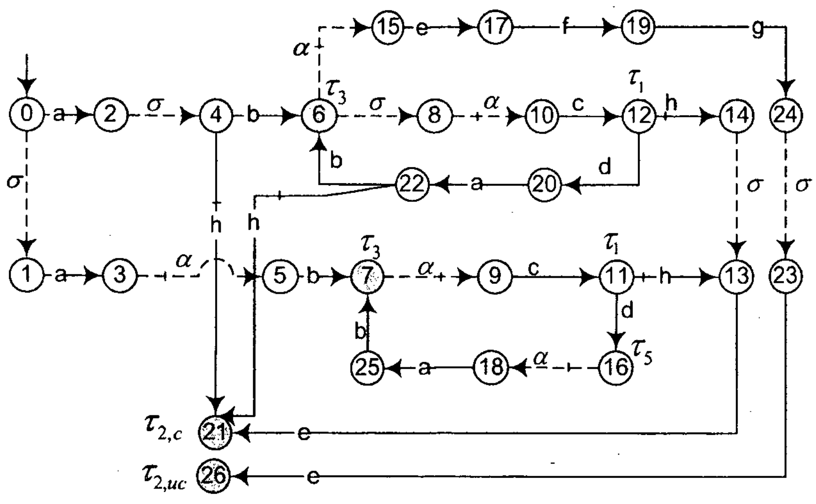


Figure 3.24: PO-SOCC(G_{lo}) satisfies the PO-SOCC property. Shaded states have been vocalized by the PO-SOCC algorithm

(i) *Transitions* $3 \overset{\alpha}{\dashrightarrow} 5$ *and* $16 \overset{\alpha}{\dashrightarrow} 18$ *are vocalized in Fig. 3.25.*

(ii) *MSPSS Algorithm stops after one iteration. The resulted automaton is given in Fig. 3.25 where* $\omega(3) = \tau_4$, $\omega(16) = \tau_5$ *and* $\omega(5) = \omega(18) = \tau_\alpha$.

(2): ES Algorithm

(i) *States* 2 *and* 20 *are vocalized so that* $\omega(2) = \omega(3) = \tau_4$ *and* $\omega(20) = \omega(16) = \tau_5$.

This ensures all incoming states of the new P-vocal states generate the same output.

C: Satisfying UUC and UUPO Properties

SCE and UET Algorithms

(i) *States* 9, 10 *and* 15 *in Fig. 3.25 are vocalized so that* $\omega(9) = \omega(10) = \omega(15) = \tau_\alpha$ *where* $\tau_\alpha \in T_c^{uo}$.

D: Satisfying OCC and OOC Properties

OCC and OOC Algorithms*: The final model in Fig. 3.26 is OCC and*

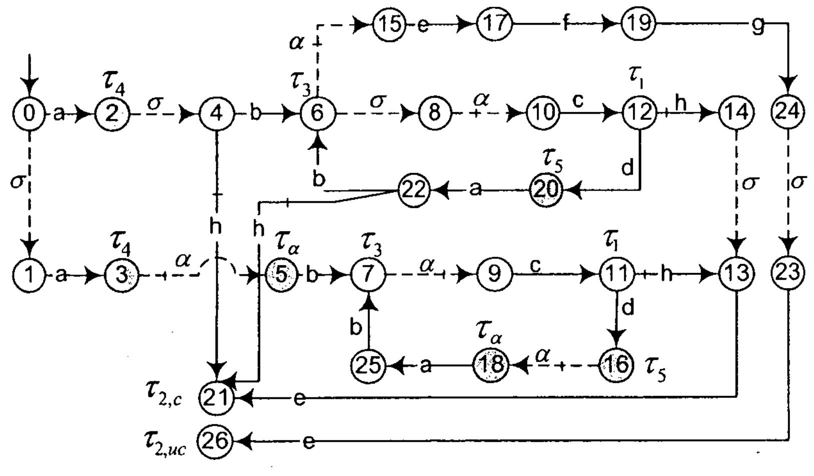


Figure 3.25: FP∘PO-SOCC(G_{lo}) satisfies the PO-SOCC property and Factorization Property. Shaded states have been vocalized by the FP Procedure.

OOC.

Note that $\tau_{2,c} \in T_c$ and $\tau_{2,uc} \in T_{uc}$ are respectively a controllable and uncontrollable copy of previously assigned label τ_2. Therefore, the new high-level event set is $T = \{\tau_1, \tau_{2,c}, \tau_{2,uc}, \tau_3, \tau_4, \tau_5, \tau_\alpha\}$ where $T_c = \{\tau_{2,c}, \tau_\alpha\}$ and $T_{uo} = \{\tau_\alpha\}$. It can be verified that the Factorization property and UUC, UUPO and PO-SOCC properties hold in the system in Fig. 3.26. The final high-level model has been given in Fig. 3.27. □

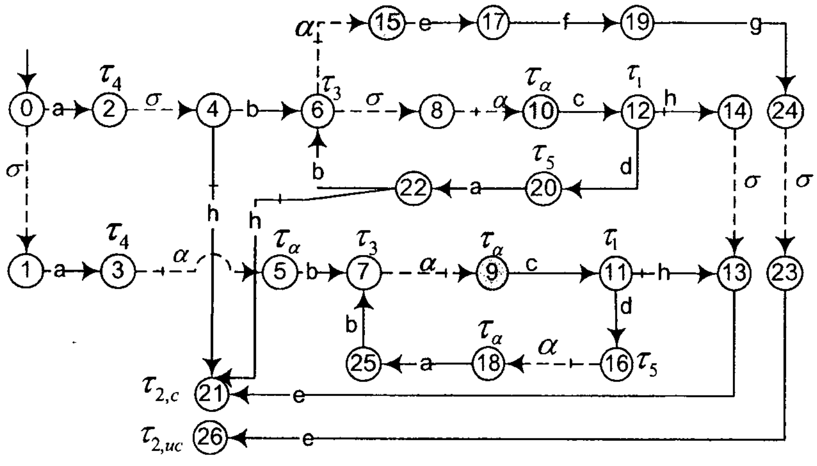


Figure 3.26: SCE∘UET∘FP∘PO-SOCC(G_{lo}) satisfies the PO-SOCC property, Factorization Property, UUC and UUPO properties. Shaded states have been vocalized by the SCE and UET algorithms.

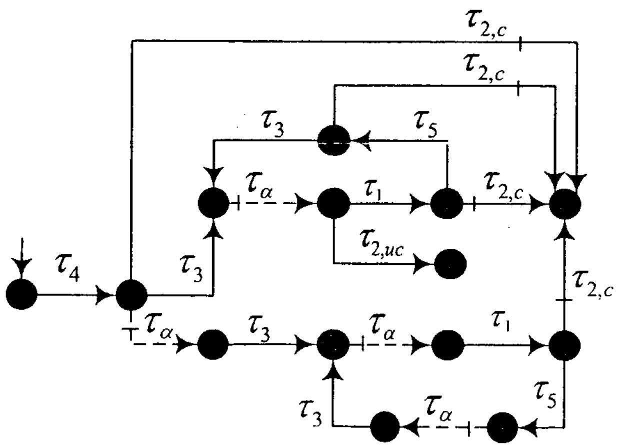


Figure 3.27: Final high-level model G_{hi}. Unobservable events are shown by dashed lines.

Chapter 4

Hierarchical Robust Supervisory Control

In this chapter we extend the robust supervisory control (RSC) problem of [4] and [45] (reviewed in Section 2.4) to the hierarchical framework of [59] (reviewed in Section 2.5) for the case of full observation and closed languages. Our main results for Hierarchical Robust Supervisory Control (HRSC) problem include a hierarchical solution for the RSC problem of [4] and [45] for a finite set of DES plants in which each plant has a separate closed specification language. It is shown that suitable extensions of Output-Control-Consistency (OCC) and Strict-Output-Control-Consistency (SOCC) properties can be used to establish a high-level control structure and to ensure hierarchical consistency in the system.

4.1 Proposed Control Structure

We aim to solve a robust control problem for a finite number of models, representing different modes of operation of a plant, in a hierarchical framework. Fig. 4.1 shows a HRSC configuration for a system having two models. A

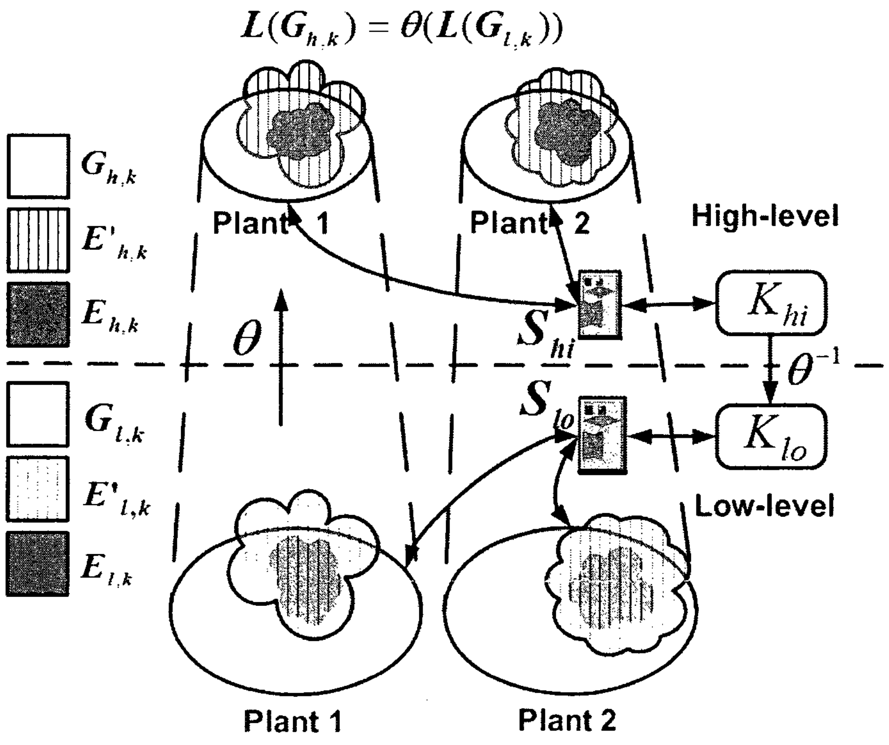


Figure 4.1: HRSC problem configuration with two models

detailed definition of HRSC problem is given below.

Consider a set of low-level Moore automata $G_{l,k} = (Q_k, \Sigma_k, T \cup \{\tau_o\}, \delta_k, \omega_k, q_{o,k})$, $(k = 1, \cdots, n)$. We assume all models agree on the controllability of the events. In RSC problem (Section 2.4), for a set of plant models $G_{l,k}$ and the corresponding legal behaviors $E_{l,k}$, it is desired to find a robust supervisor S_{lo} such that

$$L(S_{lo}/G_{l,k}) \subseteq E_{l,k} \qquad k = 1, \cdots, n.$$

This means for every k, $G_{l,k}$ under the supervision of robust supervisor S_{lo} must satisfy its design specification. Here, in HRSC problem our objective is to design a robust supervisor for the above plant models in a hierarchical fashion; in other words, we want to develop a hierarchical solution for the RSC problem. So we assume there exists a map $\theta : \bigcup_k L(G_{l,k}) \longmapsto T^*$ that reports the important sequences from the low-level to the high-level. We will discuss how θ can be obtained. Following the approach of Zhong-Wonham in [59], $G_{h,k}$ $(k = 1, \cdots, n)$ denote the abstract model at the high-level that generates $\theta(L(G_{l,k}))$. We also assume the desired legal behavior at the high-level for $G_{h,k}$ is given in the form of closed language $E'_{h,k}$. With the legal behavior $E'_{h,k}$ given at the high-level, we build the equivalent low-level legal languages $E'_{l,k} = \theta^{-1}(E'_{h,k})$. With the previous notations, a road map of the solution follows.

(i) The RSC problem is solved at the high-level for the models $G_{h,k}$ and the specifications $E'_{h,k}$, yielding a robust supervisor S_{hi}.

(ii) The systems under the supervision at the high-level would be $L(S_{hi}/G_{h,k})$ $= E_{h,k} \subseteq E'_{h,k}$.

(iii) Let S_{lo} be the implementation of S_{hi} at the low-level which will be discussed in details later.

(iv) The relation between the system under the supervision at the high-level, i.e. $E_{h,k} = L(S_{hi}/G_{h,k})$ and that under the supervision at the low-level, i.e. $L(S_{lo}/G_{l,k})$ needs to be investigated.

4.2 Information Mapping

In order to transmit the information about the important low-level event sequences to the high-level and also to translate the high-level commands to the low-level, we need to have a map capable of uniquely transmitting the low-level strings to the high-level.

Consider a set of low-level Moore automata $G_{l,k} = (Q_k, \Sigma_k, T \cup \{\tau_\circ\}, \delta_k, \omega_k, q_{\circ,k})$, $(k = 1, \cdots, n)$ where $\omega_k : Q_k \longmapsto T \cup \{\tau_\circ\}$ are the individual output maps. Similar to Section 2.5 consider the map $\hat{\omega}_k : L(G_{l,k}) \longmapsto T \cup \{\tau_\circ\}$ defined as follows: for $s \in L(G_{l,k})$, $\hat{\omega}_k(s) = \omega_k(q)$ where $q = \delta(q_{0,k}, s)$. Each output map $\hat{\omega}_k$ in fact corresponds to a map $\theta_k : L(G_{l,k}) \longmapsto T^*$ as elaborated previously in Section 2.5 for HSC. The image of the language $L(G_{l,k})$ under the map θ_k would be a language $L_{h,k} \subsetneq T^*$, i.e. $L_{h,k} = \theta_k(L(G_{l,k}))$. Let $G_{h,k}$ be any automaton whose closed behavior is $L_{h,k}$; i.e we have:

$$L(G_{h,k}) = \theta_k(L(G_{l,k})). \tag{4.1}$$

The high-level languages $L(G_{h,k})$ $(k = 1, \cdots, n)$, are defined over the same event set T and hence it would be necessary to expect any low-level sequence in different models $G_{l,k}$ to be mapped to the same high-level sequence. In other words, information maps θ_k are well-defined if and only if for any i, j, $1 \leq i, j \leq n$ we have:

$$\theta_i(s) = \theta_j(s) \text{ for } s \in L(G_{l,i}) \cap L(G_{l,j}) \tag{4.2}$$

or equivalently in terms of the maps $\hat{\omega}_k$:

$$\hat{\omega}_i(s) = \hat{\omega}_j(s) \text{ for } s \in L(G_{l,i}) \cap L(G_{l,j}). \tag{4.3}$$

The condition in (4.2) (or (4.3)) requires any string s which is common among several models, to be transmitted the same in all those models. If the condition in (4.2) is satisfied for all $1 \leq i, j \leq n$, it would be possible to define an information (reporting) map $\theta : \bigcup_k L(G_{l,k}) \longmapsto T^*$ as follows:

$$\theta(s) = \theta_k(s) \text{ for any k such that } s \in L(G_{l,k}). \tag{4.4}$$

Similarly we can define $\hat{\omega} : \bigcup_k L(G_{l,k}) \longmapsto T \cup \{\tau_\circ\}$ with

$$\hat{\omega}(s) = \hat{\omega}_k(s) \text{ for any k such that } \quad s \in L(G_{l,k}). \tag{4.5}$$

Obtaining a well-defined information map θ for a set of Moore automata $G_{l,k}$ is guaranteed if it can be ensured that the condition of (4.2) (or (4.3)) holds. To verify (4.2) for only two Moore automata $G_{l,i}$ and $G_{l,j}$ for $1 \leq i, j \leq n$, let $\hat{G}_{i,j} = G_{l,i} \times G_{l,j}$ be the meet product of them. Then condition (4.2) holds if and only if in every state (q_i, q_j) of $\hat{G}_{i,j}$, we have $\omega_i(q_i) = \omega_j(q_j)$ where ω_i and ω_j are the output maps of the models $G_{l,i}$ and $G_{l,j}$.

Remark 4.1. *The procedure to obtain the information map θ can be done in reverse direction, guaranteeing conditions (4.2) or (4.3) automatically. Consider low-level models $G_{l,k}$ (which are not yet equipped with an output map). Now one can take advantage of the fact that with a given information map $\theta : \bigcup_k L(G_{l,k}) \longmapsto T^*$, the restrictions of θ to the individual languages $L(G_{l,k})$, say θ_k, would automatically satisfy the condition in (4.2). Specifically, the following steps can be taken to obtain feasible maps $\theta_k : L(G_{l,k}) \longmapsto T^*$.*

(i) Let a map $\theta : \bigcup_k L(G_{l,k}) \longmapsto T^*$ be given or defined.

(ii) Let the restriction of θ to $L(G_{l,k})$ $(k = 1, \cdots, n)$ be called θ_k and call the corresponding output map ω_k.

(iii) Combine the model $G_{l,k}$ with the map ω_k to get the Moore automata $\hat{G}_{l,k}$.

The output maps ω_k in the Moore automata $\hat{G}_{l,k}$ satisfy the condition (4.3) by construction, and hence θ is the information map for the system.

Remark 4.2. *In the rest of this chapter, we assume that the Moore automata $G_{l,k}$'s satisfy condition (4.2) or (4.3) and hence we only refer to an information* *map θ defined in (4.4) for the purpose of exchanging the information between the low-level and high-level layers. Therefore, we can rewrite (4.1) as (4.6):*

$$L(G_{h,k}) = \theta(L(G_{l,k})). \tag{4.6}$$

4.3 Joint-OCC Property

Consider the low-level models $G_{l,k}$ $(k = 1, \cdots, n)$. Suppose $s \in L(G_{l,i})$ for some $1 \leq i \leq n$ leads to a vocal state generating $\hat{\omega}(s) = \tau$. If s can also be generated in a another model, say $G_{l,j}$, that is $s \in L(G_{l,j})$, then by (4.3), $\hat{\omega}_j(s) = \hat{\omega}_i(s) = \tau$. In the hierarchical framework proposed in this chapter, we intend to design a high-level robust supervision for high-level models $G_{h,k}$. As with robust control problems discussed in Section 2.4, we expect the high-level models agree on the controllability of high-level events. Therefore, τ must be controllable (resp. uncontrollable) for $G_{l,i}$ if and only if it is controllable (resp. uncontrollable) for $G_{l,j}$. Therefore, we need a *joint output-control-consistency* property as described in the following.

Let $T_{c,k}$ and $T_{uc,k}$ denote the controllable and uncontrollable event sets of $G_{l,k}$, and define $T_c = \bigcup\limits_k T_{c,k}$ and $T_{uc} = \bigcup\limits_k T_{uc,k}$ and $T = T_c \cup T_{uc}$. Furthermore, define L_{voc} over $\bigcup\limits_k L(G_{l,k})$ as $L_{voc} = \{s \in \bigcup\limits_k L(G_{l,k}) | s = \epsilon \text{ or } \hat{\omega}(s) \neq \tau_o\}$. Now, similar to Section 2.5, the maps χ^c and $color^c$ can be defined on $L_{voc} - \{\epsilon\}$ as follows. Let $s \in L_{voc} - \{\epsilon\}$. Therefore, there exists $s' \in L_{voc}$ and $r \in L_{voc}(s')$ such that $s = s'r$. We define

$$\chi^c(s) = \begin{cases} red, & \text{if } \hat{\omega}(s) \in T_c \\ green, & \text{if } \hat{\omega}(s) \in T_{uc} \end{cases}$$

and

$$color^c(s) = \begin{cases} red, & \text{if } \Sigma^c(r) \neq \emptyset; \\ green, & \text{if } \Sigma^c(r) = \emptyset. \end{cases}$$

Note in particular that *color*$^c(.)$ assigns a color to s based on whether the last silent segment of s contains a controllable event or not. Since, by assumption, all low-level models $G_{l,k}$ agree on the controllability of events, *color*$^c(.)$ is well-defined. Next we define the Joint-OCC property.

Definition 4.3. *Consider a set of Moore automata $G_{l,k} = (Q_k, \Sigma_k, T \cup \{\tau_o\}, \delta_k, \omega_k, q_{o,k})$ $(k = 1, \cdots, n)$. We say $G_{l,k}$ $(k = 1, \cdots, n)$ satisfy the Joint-Output-Control-Consistency (Joint-OCC) property if for all $s \in L_{voc} - \{\epsilon\}$ $\chi^c(s) = color^c(s)$.*

One way of constructing a Moore automaton G_{lo} that generates $\bigcup\limits_k L(G_{l,k})$ and captures the outputs of $G_{l,k}$'s is described in the following.

Remark 4.4. *Consider a set of Moore automata $G_{l,k}$ $(k = 1, \cdots, n)$ with the output maps $\hat{\omega}_k : L(G_{l,k}) \longmapsto T \cup \{\tau_o\}$. Assuming that all the states in $G_{l,k}$'s are marked, we construct G_{lo} as follows:*

$$G_{lo} = trim[(G_{l,1}^{co} \times \cdots \times G_{l,n}^{co})^{co}] \tag{4.7}$$

where trim(.) denotes the trim function and G^{co} stands for the complement of automaton G w.r.t the set $\Sigma = \bigcup\limits_k \Sigma_k$. Each state of G_{lo} is of the form $q = (q_1, \cdots, q_n)$. For state $q = (q_1, \cdots, q_n)$, at least one of the elements, say q_i, belongs to the state set Q_i of $G_{l,i}$. That means any sequence leading to state q in G_{lo}, leads to q_i in $G_{l,i}$. The closed behavior of G_{lo} with above structure is $L(G_{lo}) = \bigcup\limits_k L(G_{l,k})$ and besides, contains the outputs of $G_{l,k}$'s. Using the map $\hat{\omega}$, to any state q of G_{lo} we assign an output $\omega(q) = \hat{\omega}(s)$ where s is any sequence such that $q_o \xrightarrow{s} q$ is a path in G_{lo}.

Proposition 4.5. *Consider the Moore automata $G_{l,k}$ $(k = 1, \cdots, n)$ and let $G_{lo} = G_{l,1} \cup \cdots \cup G_{l,n}$ be defined as in Remark 4.4. Then $G_{l,k}$'s satisfy the Joint-OCC property if and only if G_{lo} satisfies the OCC property.*

Proof: Follows from Definition 4.3 and that G_{lo} generates $\bigcup_k L(G_{l,k})$. $\square$

Proposition 4.5 states that satisfying the Joint-OCC property which is defined on $\bigcup_k L(G_{l,k})$ is equivalent to OCC property being satisfied by G_{lo}. In Proposition 4.6 and the following corollary we show how OCC property is inherited by G_{lo} where all $G_{l,k}$'s are individually OCC.

Proposition 4.6. *Consider the Moore automata $G_{l,k}$ $(k = 1, \cdots, n)$. Then $G_{l,k}$'s satisfy the Joint-OCC property if and only if each $G_{l,k}$ satisfies the OCC property.*

Proof: An automaton G is OCC if every $s \in L(G)$ leading to a vocal state q satisfies the following:

(i) If the output at q is controllable, then the last silent segment of s includes at least one controllable event;

(ii) If the output at q is uncontrollable, then the last silent segment of s contains only uncontrollable events.

Now, if $G_{l,k}$'s are Jointly-OCC, then every string s in $L(G_{lo}) = \bigcup_k L(G_{l,k})$, (and thus every string in $L(G_{l,k})$) for which $\hat{\omega}(s) \neq \tau_o$ satisfies conditions (i) and (ii). Therefore, $G_{l,k}$ will be OCC (for every k). Conversely, if all $G_{l,k}$'s are OCC, then all strings in $L(G_{l,k})$ (for every k), and thus all strings in $L(G_{lo}) = \bigcup_k L(G_{l,k})$, will satisfy the above property. $\square$

Finally, we have the following useful corollary.

Corollary 4.7. *$G_{l,k}$ $(k = 1, \cdots, n)$ are individually OCC if and only if $\bigcup_k G_{l,k}$, is OCC.*

Proof: Follows from Proposition 4.5 and Proposition 4.6 $\square$

4.4 Supervision Implementation at the Low-level

The ability to synthesize the Moore low-level models that satisfy the Joint-OCC property, equips the high-level with the control structure. In this section, we discuss supervisor design and implementation, and the issue of hierarchical consistency.

4.4.1 Supervision Implementation

Let $G_{h,k}$ be the models at the high-level whose closed languages are $L(G_{h,k}) = \theta(L(G_{l,k}))$ $(k = 1, \cdots, n)$. We assume the desired behavior is given by $E'_{h,k}$. Let S_{hi} be the solution to the robust control problem at the high-level and $E_{h,k} = L(S_{hi}/G_{h,k})$ the closed behavior of the system under the supervision at the high-level for each model. Also, let $E_{hi} = \bigcup_k E_{h,k}$, $E_{lo} = \theta^{-1}(E_{hi})$ and finally $E'_{l,k} = \theta^{-1}(E'_{h,k})$ as the corresponding legal language for each model $G_{l,k}$ at the low-level. Let G_{hi} be any automaton at the high-level whose closed behavior is $L(G_{hi}) = \bigcup_k L(G_{h,k})$. Theorem 2.27 implies $L(S_{hi}/G_{hi}) = E_{hi}$ and E_{hi} is controllable w.r.t $L(G_{hi})$. Also, let G_{lo} be the Moore automaton given by (4.7) in Remark 4.4. Then, $L(G_{hi}) = \bigcup_k \theta(L(G_{l,k})) = \theta(\bigcup_k L(G_{l,k})) = \theta(L(G_{lo}))$. Now, from Theorem 2.33, we conclude that if the high-level supervisor S_{hi} is implemented by a disabled-event map Δ_{hi}, then there exists a low-level supervisor S_{lo} which is implemented by a disabled-event map Δ_{lo} given in (2.13). We show S_{lo} solves the robust supervisory control problem at the low-level for the set of models $G_{l,k}$ and the legal languages $E'_{l,k}$.

Theorem 4.8. *If the set of models $G_{l,k}$ are Jointly-OCC, then*

(i) $L(S_{lo}/G_{lo}) = E_{lo}^{\uparrow}$

(ii) $L(S_{lo}/G_{l,k}) = E_{lo}^{\uparrow} \cap L(G_{l,k}) \subseteq E'_{l,k}$

Proof Part (1): Since $G_{l,k}$'s are Jointly-OCC, it follows from Proposition 4.5 that G_{lo} is OCC. Therefore, by Theorem 2.33 we have

$$L(S_{lo}/G_{lo}) = (\theta^{-1}(E_{hi}))^{\uparrow} = E_{lo}^{\uparrow} \tag{4.8}$$

Part (2): S_{lo} solves the supervisory control problem for the model G_{lo} and the specification E_{lo}. Therefore for $G_{l,k}$ with $L(G_{l,k}) \subseteq L(G_{lo})$ we have:

$$\begin{aligned} L(S_{lo}/G_{l,k}) &= L(S_{lo}/G_{lo}) \cap L(G_{l,k}) \text{ (By Proposition 2.28)} \\ &= E_{lo}^{\uparrow} \cap L(G_{l,k}) \end{aligned} \tag{4.9}$$

Therefore

$$\begin{aligned} \theta[L(S_{lo}/G_{l,k})] &= \theta[E_{lo}^{\uparrow} \cap L(G_{l,k})] \\ &= \theta[(\theta^{-1}(E_{hi}))^{\uparrow} \cap L(G_{l,k})] \\ &\subseteq \theta[\theta^{-1}(E_{hi}) \cap L(G_{l,k})] \\ &\subseteq \theta(\theta^{-1}(E_{hi})) \cap \theta(L(G_{l,k})) \\ &= E_{hi} \cap L(G_{h,k}) \\ &= L(S_{hi}/G_{hi}) \cap L(G_{h,k}) \\ &= L(S_{hi}/G_{h,k}) \text{ (By Proposition 2.28)} \\ &= E_{h,k}. \end{aligned} \tag{4.10}$$

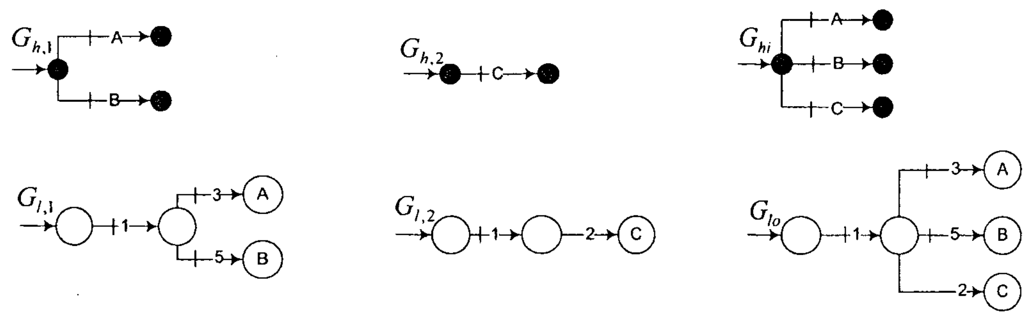


Figure 4.2: $G_{l,1}$ and $G_{l,2}$ satisfy the SOCC condition individually but the model G_{lo}, which describes the union of them, does not.

Thus (4.10) implies

$$\begin{aligned} L(S_{lo}/G_{l,k}) &\subseteq \theta^{-1}(E_{h,k}) \\ &\subseteq \theta^{-1}(E'_{h,k}) \\ &= E'_{l,k} \end{aligned} \tag{4.11}$$

□

Therefore, we have shown S_{lo} solves a robust control problem at the low-level for models $G_{l,k}$ and the legal behaviors $E'_{l,k} = \theta^{-1}(E'_{h,k})$. We illustrate our conclusions in Theorem 4.8 in the following example.

Example 4.9. *Fig. 4.2 shows two low-level models $G_{l,k}$ $(k = 1, 2)$ and the corresponding high-level models $G_{h,k}$. In Fig. 4.2, the odd numbers denote the controllable events and the even numbers, the uncontrollable events. Each $G_{l,k}$ is OCC, hence, they will be Jointly-OCC; let the high-level specifications $E'_{h,1} = \overline{A+B}$ and $E'_{h,2} = \{\epsilon\}$ be given respectively for $G_{h,1}$ and $G_{h,2}$. The automaton $G_{lo} = G_{l,1} \cup G_{l,2}$ and $G_{hi} = G_{h,1} \cup G_{h,2}$ are also shown in Fig. 4.2. $E_{hi} = E'_{h,1} \cup E'_{h,2} = \overline{A+B}$ is controllable and S_{hi} can be constructed so that $L(S_{hi}/G_{hi}) = E_{hi} = \overline{A+B}$ with $L(S_{hi}/G_{h,1}) = \overline{A+B}$ and $L(S_{hi}/G_{h,2}) = \epsilon$. A low-level supervisor S_{lo} would implement the high-level commands at the low-level. Let S_{lo}/G_{lo} and $E_{lo} = \theta^{-1}(E_{hi})$ be the system under supervision and* *the image of E_{hi} at the low-level. We will have $E_{lo} = \{\epsilon, \text{“1”}, \text{“1.3”}, \text{“1.5”}\}$. The supremal controllable sublanguage of E_{lo} is $E_{lo}^{\uparrow} = \{\epsilon\}$ and therefore, by Theorem 4.8 we have $L(S_{lo}/G_{lo}) = E_{lo}^{\uparrow} = \{\epsilon\}$. In this example, S_{hi} intends to disable C. To implement this at the low-level, S_{lo} disables event 1, leading to $L(S_{lo}/G_{l,1}) = L(S_{lo}/G_{l,2}) = \{\epsilon\}$. Note that $\epsilon \subseteq E'_{l,1}$ and $\epsilon \subseteq E'_{l,2}$ where $E'_{l,1} = \theta^{-1}(E'_{h,1}) = G_{l,1}$ and $E'_{l,2} = \theta^{-1}(E'_{h,2}) = \{\epsilon, \text{“1”}\}$.*

4.4.2 Robust Hierarchical Consistency

So far we have shown how the commands of a robust supervisor S_{hi} at the high-level can be translated to a robust supervisor S_{lo} at the low-level. The relation between the behavior of system under supervision at the low-level $(L(S_{lo}/G_{l,k}))$ and that of the high-level $(L(S_{hi}/G_{h,k}))$ is not clear yet. We would like the behavior of the system under supervision at the high-level for each model, i.e. $L(S_{hi}/G_{h,k})$, be recovered from $L(S_{lo}/G_{l,k})$ as explained below. We refer to this one-to-one hierarchical consistency as *robust hierarchical consistency.*

Definition 4.10. *We say the robust hierarchical consistency property holds in the system if $\theta(L(S_{lo}/G_{l,k})) = L(S_{hi}/G_{h,k})$ for $k = 1, \cdots, n$.*

Of course, the *robust hierarchical consistency* property is an extension of hierarchical consistency in [59] to the robust supervisory control problem. In hierarchical control of [59], to have hierarchical consistency, it must be ensured that low-level implementation of a high-level command to disable a high-level event does not produce unintended consequences in the form of disablement of other high-level events. If the plant is SOCC, then hierarchical consistency can be guaranteed. In HRSC problem discussed here, we have to make sure that the implementation of a high-level command to disable an event does not produce unintended consequences in *any possible plant.* To ensure this, we bring in a Joint-SOCC property.

Definition 4.11. *The Moore automata $G_{l,k}$ $(k = 1, \cdots, n)$ are Jointly-SOCC if (i) they are Jointly-OCC and (ii) in the reachability tree of $\bigcup_k G_{l,k}$, no two vocal nodes with controllable outputs are partners.*

In the following proposition, we show if $G_{l,k}$ $(k = 1, \cdots, n)$ are Jointly-SOCC, then $G_{l,k}$ are individually SOCC. The reverse is not necessarily true as shown in an example following the proposition.

Proposition 4.12. *Consider the set of Moore automata $G_{l,k}$ $(k = 1, \cdots, n)$. If $G_{l,k}$'s $(k = 1, \cdots, n)$ are Jointly-SOCC then $G_{l,k}$ $(k = 1, \cdots, n)$ are individually SOCC.*

Proof: Let $G_{lo} = \bigcup_k G_{l,k}$. We use the fact that the reachability trees (RT) of $G_{l,k}$ $(k = 1, \cdots, n)$ are subtrees of the RT of G_{lo}. Therefore, if two nodes with controllable outputs are partners in the RT of $G_{l,k}$, they will be partners in the RT of G_{lo}. Hence, if $G_{l,k}$ $(k = 1, \cdots, n)$ are Jointly-SOCC, each $G_{l,k}$ would be individually SOCC.
Next by contradiction, let B and C be two partners in the RT of $G_{l,k}$ for some $1 \leq k \leq n$, say $G_{l,1}$, which are reached respectively through the silent paths $s_1\sigma s_2 s_3$ and $s_1\sigma s_2 s_4$ originated from the initial state or a vocal node A. Let A be reached through a sequence s in the RT of $G_{l,1}$. The arrangement above implies we have $\hat{\omega}(ss_1\sigma s_2 s_3) = B$ and $\hat{\omega}(ss_1\sigma s_2 s_4) = C$. Note that

$$s \in L(G_{l,1}) \subseteq L(G_{lo}) \Longrightarrow s \in L(G_{lo}).$$

Similarly we have $ss_1\sigma s_2 s_3$, $ss_1\sigma s_2 s_4 \in L(G_{lo})$. It follows from the feasibility condition that in the RT of G_{lo} we have $\hat{\omega}(s) = A$, $\hat{\omega}(ss_1\sigma s_2 s_3) = B$ and $\hat{\omega}(ss_1\sigma s_2 s_4) = C$. If in the RT of G_{lo} we have $\hat{\omega}(ss_1\sigma) = \tau_o$ then B and C are also partners in it which is a contradiction since, by the hypothesis, in the RT of G_{lo}, no two vocal nodes with controllable outputs are partners. So

$\hat{\omega}(ss_1\sigma) \neq \tau_o$ which again from the feasibility condition we conclude B and C cannot be partners in $G_{l,1}$. $\square$

Example 4.13. *(Example 4.9 continued) In Example 4.9, $G_{l,1}$ and $G_{l,2}$ are individually SOCC but not Jointly-SOCC since in the RT of G_{lo} the node corresponding to sequence "1.2" with output C is partner with the node corresponding to sequence "1.3" generating A (also with "1.5" generating B).*

Joint-SOCC property of $G_{l,k}$'s is equivalent to SOCC property of the union model G_{lo} (defined in (4.7)).

Proposition 4.14. *The models $G_{l,k}$ $(k = 1, \cdots, n)$ are Jointly-SOCC if and only if G_{lo} is SOCC.*

Proof: Follows from Corollary 4.7 and Definition 4.11 $\square$

In the following we show that the Joint-SOCC property is sufficient for *robust hierarchical consistency.*

Theorem 4.15. *If the low-level Moore automata $G_{l,k}$ $(k = 1, \cdots, n)$ are Jointly-SOCC, then*

(i) $\theta(L(S_{lo}/G_{lo})) = E_{hi}$;

(ii) $\theta(L(S_{lo}/G_{l,k})) = E_{h,k}$.

Proof: **(i)** Solving the robust supervisory control at the high-level amounts to solving supervisory control for plant G_{hi} and legal behavior E_{hi}. Thus $E_{hi} = L(S_{hi}/G_{hi})$. Since $G_{l,k}$'s are Jointly-SOCC, it follows from Proposition 4.14 that G_{lo} is SOCC and therefore by Theorem 2.37, $\theta(L(S_{lo}/G_{lo})) = E_{hi}$.
(ii) From (4.10) we have:

$$\theta(L(S_{lo}/G_{l,k})) \subseteq E_{h,k} \tag{4.12}$$

For the reverse inclusion we have:

$$\begin{aligned} E_{h,k} &= E_{hi} \cap L(G_{h,k}) \qquad \text{(since } S_{hi} \text{ solves RSC)} \\ &= \theta(L(S_{lo}/G_{lo})) \cap L(G_{h,k}) \qquad \text{(by part (i))} \\ &= \theta(L(S_{lo}/G_{lo})) \cap \theta(L(G_{l,k}). \end{aligned} \tag{4.13}$$

Now let T be the event set at the high-level. For any $t \in T^*$ that

$$t \in \theta(L(S_{lo}/G_{lo})) \cap \theta(L(G_{l,k})) \tag{4.14}$$

let

$$\Sigma'_t = \{s \in L(G_{lo}) | \omega(s) \neq \tau_o \ \text{ and } \ \theta(s) = t\}.$$

In other words, Σ'_t is the strings whose images are $t \in T^*$ and lead to vocal states in G_{lo}. Then, Lemma 4.18 in the Appendix states

$$\Sigma'_t \subseteq L(S_{lo}/G_{lo}). \tag{4.15}$$

Also, $t \in \theta(L(G_{l,k}))$ implies

$$\Sigma'_t \cap L(G_{l,k}) \neq \emptyset \tag{4.16}$$

Equations (4.15) and (4.16) together imply for some $s' \in \Sigma'_t$ we have:

$$s' \in L(S_{lo}/G_{lo}) \cap L(G_{l,k}) \quad \text{and} \quad \theta(s') = t \tag{4.17}$$

It is concluded from (4.17) that

$$t \in \theta(L(S_{lo}/G_{lo}) \cap L(G_{l,k})) \tag{4.18}$$

and hence from (4.14) and (4.18) we have:

$$\begin{aligned}\theta(L(S_{lo}/G_{lo})) \cap \theta(L(G_{l,k})) &\subseteq \theta(L(S_{lo}/G_{lo}) \cap L(G_{l,k})) \\ &= \theta(L(S_{lo}/G_{l,k})). \text{ (Proposition 2.28)} \quad (4.19)\end{aligned}$$

Finally, equations (4.13) and (4.19) yield

$$E_{h,k} \subseteq \theta(L(S_{lo}/G_{l,k})). \tag{4.20}$$

Therefore, (4.12) and (4.20) together prove that $\theta(L(S_{lo}/G_{l,k})) = E_{h,k}$ which completes the proof. □

Theorem 4.15, specifically part (ii), show that the Joint-SOCC property guarantees robust hierarchical consistency.

Example 4.16. *(Example 4.9 continued) The system in Example 4.13 (Fig. 4.2) does not satisfy the robust hierarchical consistency. Specifically, we have* $\theta(L(S_{lo}/G_{l,1})) = \{\epsilon\} \neq E_{h,1} = E'_{h,1} = \overline{A+B}$. *It was shown in Example 4.13 that* $G_{l,i}$ $(i = 1, 2)$ *do not satisfy the Joint-SOCC property and hence the violation of robust hierarchical consistency is not unexpected. Again note that in general the Joint-SOCC property is a sufficient and not a necessary condition for robust hierarchical consistency.*

We conclude this chapter with an example.

Example 4.17. *The theory developed in this chapter will be illustrated here. Fig 4.3 shows two Moore automata* G_1 *and* G_2 *with individually assigned output maps* ω_i *for* $i = 1, 2$. *There* $\Sigma = \{a, b, c, d, m, n, p\}$ *of which* $\Sigma_c = \{a, b, d\}$ *is the set of controllable events.* $T = T_c = \{A, B, C\}$ *is also the set of high-level events. It can be checked that the condition (4.3) holds for them; however, they are neither Jointly-SOCC nor Jointly-OCC. In fact,* G_2 *is neither SOCC* *nor OCC and hence G_1 and G_2 would not be Jointly-OCC or Jointly-SOCC. Specifically, the OCC property is violated at state $5'$ in G_2 since the silent path $0' \xrightarrow{a.n.a.m} 5'$ contains controllable events and the silent path $5' \xrightarrow{p.n} 5'$ is uncontrollable. We follow a procedure, based on Proposition 4.5 and Proposition 4.14 to modify G_1 and G_2 such that they become Jointly-SOCC.*

***Step 1**: Fig. 4.4 shows $G_{lo} = G_1 \cup G_2$ which is not SOCC or OCC.*

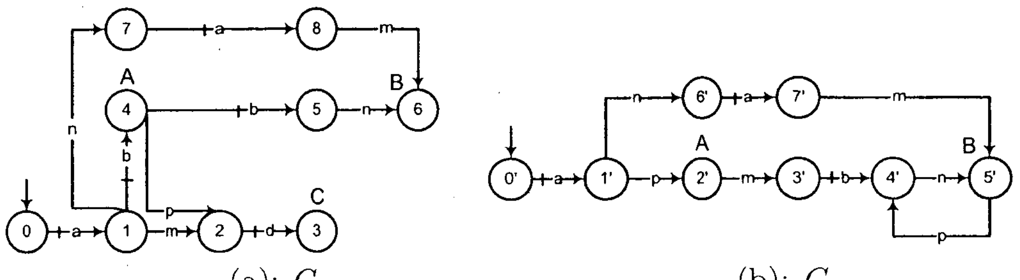


Figure 4.3: Moore automata models (a): G_1 and (b): G_2 which are neither Jointly-SOCC nor Jointly-OCC

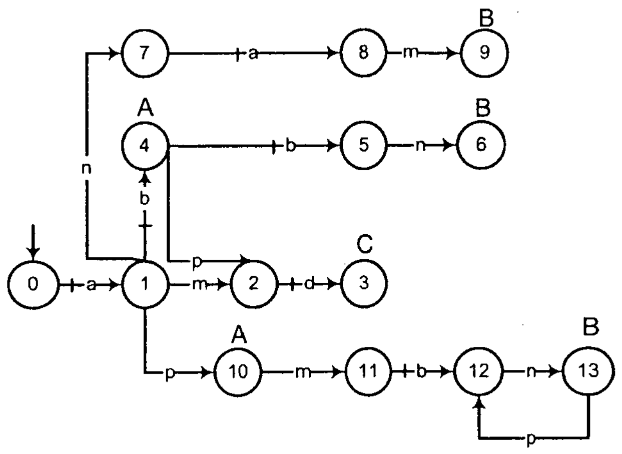


Figure 4.4: $G_{lo} = G_1 \cup G_2$ which is neither SOCC nor OCC.

***Step 2**: Modify G_{lo} so that it becomes SOCC. (Fig. 4.5). In Fig. 4.4 OCC fails at state 13. Therefore, it has been split as shown in Fig. 4.5 into states 14 and 15. This split requires us to have two controllable and uncontrollable copies of the event B. We denote this output relabeling by $B \longrightarrow \{B_c, B_{uc}\}$ in Fig. 4.5. Moreover, G_{lo} in Fig. 4.4 is not SOCC since in the reachability*

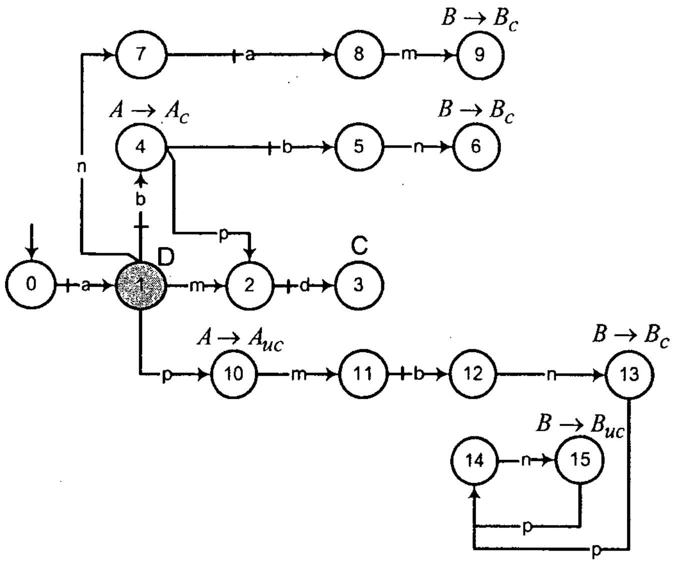


Figure 4.5: $G_{lo} = G_1 \cup G_2$ has been modified to be SOCC.

tree of G_{lo}, the nodes a.p and a.m.d are partners. As a remedy, state 1 *is vocalized such that in Fig. 4.5 we have $\omega(1) = D$ (ω is the output map of G_{lo}). Following this vocalization, two controllable and uncontrollable copies of event A need to be generated, i.e. we have $A \longrightarrow \{A_c, A_{uc}\}$. The final result is shown in Fig. 4.5. We call the automaton in Fig. 4.5 G_{lo} again.*

***Step 3**: Let $\hat{G}_i = G_{lo} \times G_i$ for $i = 1, 2$ (Fig. 4.6). The states of $\hat{G}_i$ are of the form $\hat{q} = (q, x)$ where q is the state of G_{lo} and x is the state of G_i. Vocalize $\hat{G}_i$ such that the output of $\hat{q}$ be equal to the output of q in G_{lo}; i.e. $\bar{\omega}_i(\hat{q}) = \omega(q)$ where $\bar{\omega}_i$ is the output map for $\hat{G}_i$. Therefore, the set of high-level events would*

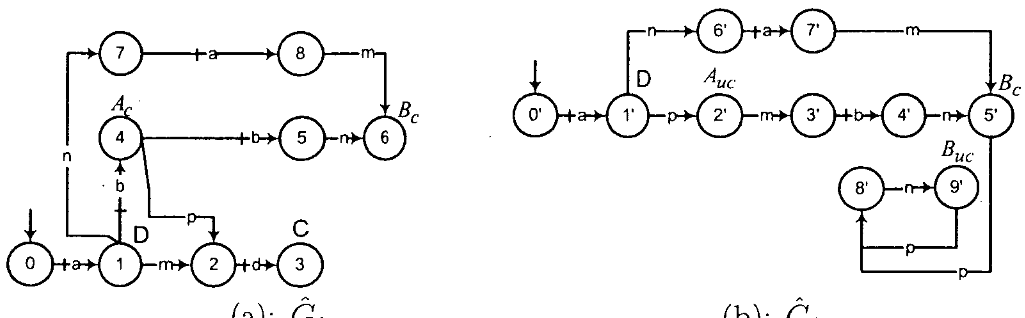


Figure 4.6: Modified Moore automata $\hat{G}_1 = G_{lo} \times G_1$ (a) and $\hat{G}_2 = G_{lo} \times G_2$ (b) are Jointly-SOCC.

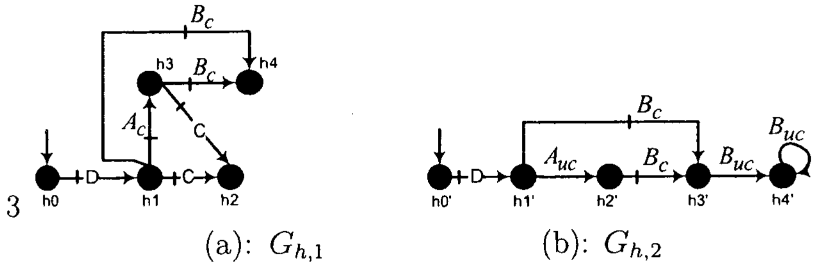


Figure 4.7: High-level models (a) $G_{h,1}$ and (b) $G_{h,2}$

be $T = \{A_c, A_{uc}, B_c, B_{uc}, C, D\}$ *of which* $T_c = \{A_c, B_c, C, D\}$ *is the set of controllable events. By construction and based on Proposition 4.5 and Proposition 4.14,* $\hat{G}_1$ *and* $\hat{G}_2$ *are Jointly-SOCC. Two high-level models* $G_{h,1}$ *and* $G_{h,2}$ *for which we have* $L(G_{h,1}) = \theta(L(G_1))$ *and* $L(G_{h,2}) = \theta(L(G_2))$ *are given in Fig. 4.7. Now, suppose two specifications* $E'_{h,1}$ *and* $E'_{h,2}$ *are given as "every thing is permitted" and "system should be loop-free" or equivalently* $E'_{h,1} = L(G_{h,1})$ *and* $E'_{h,2} = \overline{D.(B_c + A_{uc}.B_c).B_{uc}}$. *The solution of RSC problem at the high-level is* $E_{h,1} = L(S_{hi}/G_{h,1}) = \overline{D.(A_c.(C + B_c)}$ $\overline{+C)}$ *and* $E_{h,2} = L(S_{hi}/G_{h,2}) = \overline{D.A_{uc}}$. *The high-level systems under supervision is shown in Fig. 4.8. Let* S_{lo} *be the supervisor which implements the*

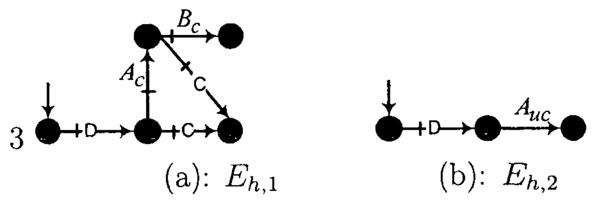


Figure 4.8: High-level system under supervision (a) $E_{h,1}$ and (b)$E_{h,2}$

commands of S_{hi} *at the low-level. The systems under supervision* $S_{lo}/\hat{G}_1$ *and* $S_{lo}/\hat{G}_2$ *are shown in Fig. 4.9. It can easily be verified that robust hierarchical consistency holds in the system and we have* $\theta(L(S_{lo}/\hat{G}_1)) = E_{h,1}$ *and* $\theta(L(S_{lo}/\hat{G}_2)) = E_{h,2}$. *The operation of the hierarchical robust supervisors can be explained as follows. At the high-level (Fig. 4.7), the* B_c *transition from states* $h1'$ *and* $h2'$ *to* $h3'$ *have to be disabled to avoid the uncontrollable ille-*

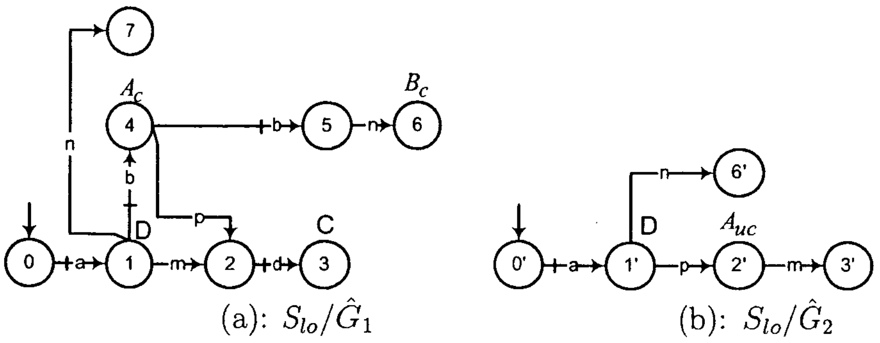


Figure 4.9: The systems under supervision at the low-level (a) $S_{lo}/\hat{G}_1$, (b) $S_{lo}/\hat{G}_2$

gal self-loop in state $h3'$. *This in particular disables the* B_c *event following a D event (state* $h1'$*). As a result, the robust supervisor* S_{hi} *disables the* B_c *transition from state* $h1$ *to* $h4$ *in* $G_{h,1}$. *To implement this disablement at the low-level, the controllable transition a from state 7 to 8 in* $\hat{G}_1$ *and from* $6'$ *to* $7'$ *in* $\hat{G}_2$ *are disabled. Similarly, to implement the disablement of* B_c *from* $h2'$ *to* $h3'$, *at the low-level in* $\hat{G}_2$ *the controllable transition b from* $3'$ *to* $4'$ *is disabled.*

4.5 Conclusion

In this chapter a hierarchical solution for the problem of robust supervisory control of a finite family of DES plants was derived based on Zhong-Wonham approach. The Joint-OCC and Joint-SOCC properties, as the extensions of the OCC and SOCC properties, were developed to guarantee robust hierarchical consistency. We showed that a maximally permissive robust supervisor which is designed for the high-level yields the maximally permissive controllable behavior at the low-level while assuring robust hierarchical consistency.

4.6 Appendix

Consider the robust supervisor S_{lo} defined in Section 2.4. Let $t \in T^*$ be a high-level sequence and define $\Sigma'_t = \{s \in L(G_{lo}) |\ \omega(s) \neq \tau_o \text{ and } \theta(s) = t\}$ to be the strings whose images are $t \in T^*$ and lead to vocal states in G_{lo}. If $t \in \theta(L(S_{lo}/G_{lo})))$, the membership of Σ'_t in $L(S_{lo}/G_{lo})$ is investigated. Note that by Theorem 2.37 which states $\theta(L(S_{lo}/G_{lo})) = L(S_{hi}/G_{hi})$, for every $t \in L(S_{hi}/G_{hi})$ it is only concluded that for some sequence $s \in \Sigma^*$ that $\theta(s) = t$ we have $s \in L(S_{lo}/G_{lo})$.

Lemma 4.18. *If G_{lo} is SOCC, then for every $t \in L(S_{hi}/G_{hi})$ we have $\Sigma'_t \subseteq L(S_{lo}/G_{lo})$.*

Proof: Although not directly addressed, the proof is originally given for Theorem 2.1 in [59]. This is elaborated in the following.

By contradiction suppose for some $t \in L(S_{hi}/G_{hi})$ and some $s \in \Sigma'_t$ we have $s \in \Sigma'_t - L(S_{lo}/G_{lo})$. Since $s \in L_{voc}$ is vocal it can be split as $s = s'r$ where $s' \in L_{voc}$ and $r \in L_{voc}(s')$. Let $s'' \leq s$ be the largest vocal prefix of s for which we have $s'' \in \Sigma'_t \cap L(S_{lo}/G_{lo})$. If $s'' \neq s'$ then we choose $s = s''r$ where $r \in L_{voc}(s'')$ and we let $t = \theta(s) = \theta(s''r)$. Therefore without loss of generality we assume it is the case that $s' \leq s$ is the largest prefix of s such that $s' \in \Sigma'_t \cap L(S_{lo}/G_{lo})$ and that for which we have $s = s'r$ where $r \in L_{voc}(s')$. Now let $\theta(s') = t'$ and $\hat{\omega}(s) = \tau$ so that $t = t'\tau$. If $s \notin L(S_{lo}/G_{lo})$ by the fact that $L(S_{lo}/G_{lo})$ is controllable with respect to G_{lo} and G_{lo} is OCC, we should have $\Sigma^c(r) \neq \emptyset$. Furthermore for some $\sigma' \in \Sigma^c(r)$ and appropriate $s_1, s_2 \in \Sigma^*$ that $r = s_1\sigma's_2$ we have

$$\sigma' \in \Delta_{lo}(s's_1, \theta(s's_1)) = \Delta_{lo}(s's_1, t').$$

By definition of Δ_{lo} there should exist $s_3 \in \Sigma^*_{uc}$ such that $\hat{\omega}(s's_1\sigma's_3) \in \Delta_{hi}(t')$.

Let $\hat{\omega}(s's_1\sigma's_3) = \tau'$. Since by assumption $t'\tau = t \in E_{hi}$ it follows that $\tau \notin \Delta_{hi}(t')$ and therefore

$$\tau \neq \tau'.$$

This implies nodes corresponding to sequences $s's_1\sigma's_2$ and $s's_1\sigma's_3$ in reachability tree of G_{lo} are partners which is a contradiction since G_{lo} is SOCC and this cannot be the case. Therefore we should have $s \in L(S_{lo}/G_{lo})$. This completes the proof that $\Sigma_t' \subseteq L(S_{lo}/G_{lo})$. $\square$

Chapter 5

Hierarchical Robust Supervisory Control under Partial Observation

This chapter extends our results on hierarchical robust supervisory control (HRSC) in Chapter 4 to the case of control under partial observation, that is to say combines our results on hierarchical supervisory control under partial observation (USCPO) in Chapter 3 and our result on HRSC problem in Chapter 4. Robust supervisory control under partial observation has previously been studied in [45]. However, in this thesis, we study HRSC under partial observation which implies robust supervisory control is performed in a hierarchical fashion while partial observation is assumed in the modeling. Our main results for HRSC under partial observation include a hierarchical solution for the RSC problem under partial observation [45] for a finite set of DES plants in which each plant has a separate closed specification language. It is shown that suitable extensions of UUC, UUPO and PO-SOCC properties can be used to establish a high-level control structure and to ensure hierarchical consistency in the system.

5.1 Proposed Control Structure

We aim to solve a robust control problem under partial observation for a finite number of models, representing different modes of operation of a plant, in a hierarchical framework. Consider a set of low-level Moore automata $G_{l,k} = (Q_k, \Sigma_k, T \cup \{\tau_o\}, \delta_k, \omega_k, q_{o,k})$, $(k = 1, \cdots, n)$. We assume all the models agree on the controllability and observability of the events. Let $\Sigma = \bigcup_k \Sigma_k$. Assume $\Sigma_o \subseteq \Sigma$ and $\Sigma_c \subseteq \Sigma$ are the set of observable and controllable events. Next, for each model $G_{l,k}$, $\Sigma_{k,o} = \Sigma_k \cap \Sigma_o$ and $\Sigma_{k,c} = \Sigma_k \cap \Sigma_c$ denote the sets of observable and controllable events and $\Sigma_{k,uo} = \Sigma_k - \Sigma_{k,o}$ and $\Sigma_{k,uc} = \Sigma_k - \Sigma_{k,c}$ denote the sets of unobservable and uncontrollable events.
In RSC problem under partial observation (Section 2.4), for a set of plant models $G_{l,k}$ and the corresponding legal behaviors $E_{l,k}$, it is desired to find a robust feasible supervisor S_{lo} such that

$$L(S_{lo}/G_{l,k}) \subseteq E_{l,k} \qquad k = 1, \cdots, n.$$

This means for every k, $G_{l,k}$ under the supervision of feasible robust supervisor S_{lo} must satisfy its design specification and furthermore, the sequences with the natural projection are treated the same by supervisor. Here, in HRSC problem under partial observation our objective is to design a robust supervisor for the above plant models in a hierarchical fashion; in other words, we want to develop a hierarchical solution for the RSC problem under partial observation. So we assume there exists a map $\theta : \bigcup_k L(G_{l,k}) \longmapsto T^*$ that reports the important sequences from the low-level to the high-level. We will discuss how θ can be obtained. Following the approach of Zhong-Wonham in [59],

$G_{h,k}$ $(k = 1, \cdots, n)$ denotes the abstract model at the high-level that generates $\theta(L(G_{l,k}))$. We also assume the desired legal behavior at the high-level for $G_{h,k}$ is given in the form of closed language $E'_{h,k}$. With the legal behavior $E'_{h,k}$ given at the high-level, we build the equivalent low-level legal languages $E'_{l,k} = \theta^{-1}(E'_{h,k})$. With the previous notations, a road map of the solution follows.

(i) The RSC problem under partial observation is solved at the high-level for the models $G_{h,k}$ and the specifications $E'_{h,k}$, yielding a robust feasible supervisor S_{hi}. The systems under supervision at the high-level would be $L(S_{hi}/G_{h,k}) = E_{h,k} \subseteq E'_{h,k}$.

(ii) Let $\tilde{S}_{lo}$ be the feasible implementation of S_{hi} at the low-level which will be discussed in details later.

(iii) The relation between the system under supervision at the high-level, $E_{h,k} = L(S_{hi}/G_{h,k})$, and that under the supervision at the low-level, $L(\tilde{S}_{lo}/G_{l,k})$, needs to be investigated.

5.2 Information Mapping

In order to transmit the information about the important low-level event sequences to the high-level and also to translate the high-level commands to the low-level, we need to have the capability of uniquely mapping the sequences to the high-level. This is very similar to Chapter 4 where full observation was assumed. Therefore, consider a set of low-level Moore automata $G_{l,k} = (Q_k, \Sigma_k, T \cup \{\tau_o\}, \delta_k, \omega_k, q_{o,k})$, $(k = 1, \cdots, n)$ where $\omega_k : Q_k \longmapsto T \cup \{\tau_o\}$ are the individual output maps. Similar to Section 2.5 consider the map $\hat{\omega}_k : L(G_{l,k}) \longmapsto T \cup \{\tau_o\}$ defined as follows: for $s \in L(G_{l,k})$, $\hat{\omega}_k(s) = \omega_k(q)$

where $q = \delta(q_{0,k}, s)$. Each output map $\hat{\omega}_k$ in fact corresponds to a map $\theta_k : L(G_{l,k}) \longmapsto T^*$ as elaborated previously in Section 2.5 for HSC. The information maps θ_k are well-defined if and only if for any i, j, $1 \leq i, j \leq n$ we have:

$$\theta_i(s) = \theta_j(s) \;\text{ for }\; s \in L(G_{l,i}) \cap L(G_{l,j}) \tag{5.1}$$

or equivalently in terms of the maps $\hat{\omega}_k$:

$$\hat{\omega}_i(s) = \hat{\omega}_j(s) \;\text{ for }\; s \in L(G_{l,i}) \cap L(G_{l,j}). \tag{5.2}$$

If the condition in (5.1) holds for all $1 \leq i, j \leq n$, it would be possible to define an information (reporting) map $\theta : \bigcup_k L(G_{l,k}) \longmapsto T^*$ as follows:

$$\theta(s) = \theta_k(s) \;\text{ for any k such that }\; s \in L(G_{l,k}). \tag{5.3}$$

Similarly, we can define $\hat{\omega} : \bigcup_k L(G_{l,k}) \longmapsto T \cup \{\tau_o\}$ as

$$\hat{\omega}(s) = \hat{\omega}_k(s) \;\text{ for any k such that }\; s \in L(G_{l,k}). \tag{5.4}$$

Therefore, if $G_{h,k}$ is the abstract model at the high-level which generates $\theta_k(L(G_{l,k}))$ $(k = 1, \cdots, n)$ we can write

$$L(G_{h,k}) = \theta(L(G_{l,k})) \tag{5.5}$$

where θ is given by (5.3). *In the rest of this chapter, we assume that Moore automata $G_{l,k}$'s satisfy the condition (5.1) or (5.2) and hence we only refer to an information map θ defined in (5.3) for the purpose of exchanging the information between the low-level and high-level layers.*

5.3 Joint-OCC and Joint-OOC Properties

Consider the low-level models $G_{l,k}$ $(k = 1, \cdots, n)$. Suppose $s \in L(G_{l,i})$ for some $1 \leq i \leq n$ leads to a vocal state generating $\hat{\omega}(s) = \tau$. If s can also be generated in another model, say $G_{l,j}$, that is $s \in L(G_{l,j})$, then by (5.2), $\hat{\omega}_j(s) = \hat{\omega}_i(s) = \tau$. In the hierarchical framework proposed in this chapter, we intend to design a high-level robust supervision for high-level models $G_{h,k}$. As with robust control problems discussed in Section 2.4, we expect the high-level models agree on the controllability of high-level events. Therefore, τ must be controllable (resp. uncontrollable) for $G_{l,i}$ if and only if it is controllable (resp. uncontrollable) for $G_{l,j}$. Joint-OCC which was derived in Chapter 4 ensures this property and remains valid through this chapter. Hence we assume T_c and T_{uc} denote the subsets of controllable and uncontrollable events at the high-level and the models $G_{l,k}$'s are Joint-OCC. Nevertheless, the system should still be examined for observability of the events. Similar to Joint-OCC property, we argue that the output τ must be observable (resp. unobservable) for $G_{l,i}$ if and only if it is observable (resp. unobservable) for $G_{l,j}$. Therefore, we need a joint-output-observation-consistency property as described in the following. Assume $T_o \subseteq T$ and $T_{uo} = T - T_o$ denote the subsets of observable and unobservable events at the high-level. Furthermore, define L_{voc} over $\bigcup\limits_k L(G_{l,k})$ as $L_{voc} = \{s \in \bigcup\limits_k L(G_{l,k}) | s = \epsilon \text{ or } \hat{\omega}(s) \neq \tau_o\}$. Now, the maps χ^o and $color^o$ can be defined on $L_{voc} - \{\epsilon\}$ as follows. Let $s \in L_{voc} - \{\epsilon\}$. Therefore, there exist $s' \in L_{voc}$ and $r \in L_{voc}(s')$ such that $s = s'r$. We define

$$\chi^o(s) = \begin{cases} red, & \text{if } \hat{\omega}(s) \in T_o \\ green, & \text{if } \hat{\omega}(s) \in T_{uo} \end{cases}$$

and

$$color^{\circ}(s) = \begin{cases} red, & \text{if } P_{lo}(r) \neq \epsilon \\ green, & \text{if } P_{lo}(r) = \epsilon \end{cases}$$

The map χ° shows if the assigned label to s belongs to the observable set $T_{\circ}$ or the unobservable set T_{uo}. On the other hand, the map $color^{o}$ assigns a color to s based on whether the last silent segment of s is observable or not. Since, by assumption, all low-level models $G_{l,k}$ agree on the observability of events, $color^{\circ}(.)$ is well-defined. Next we define the Joint-OOC property.

Definition 5.1. *Consider a set of Moore automata $G_{l,k} = (Q_k, \Sigma_k, T \cup \{\tau_{\circ}\}, \delta_k, \omega_k, q_{\circ,k})$ $(k = 1, \cdots, n)$. We say $G_{l,k}$ $(k = 1, \cdots, n)$ satisfy the Joint-Output-Observation-Consistency (Joint-OOC) property if for all $s \in L_{voc} - \{\epsilon\}$, $\chi^{\circ}(s) = color^{\circ}(s)$.*

A paraphrasing of the OOC property (Definition 3.2) follows.

Remark 5.2. *Consider a Moore automaton $H = (Q, \Sigma, T \cup \{\tau_{\circ}\}, \delta, \omega, q_{\circ})$. Let $q = \delta(q_{\circ}, sr)$ where r is a silent path which extends $s \in L(H)$. Then, H is OOC if the following holds: r contains at least one observable event if the output at q is observable and r contains only unobservable events if the output at q is unobservable.*

Proposition 5.3. *Consider the Moore automata $G_{l,k}$ $(k = 1, \cdots, n)$ and let $G_{lo} = G_{l,1} \cup \cdots \cup G_{l,n}$ be defined as in Remark 4.4. Then $G_{l,k}$'s satisfy the Joint-OOC property if and only if G_{lo} satisfies the OOC property.*

Proof: Follows from Definition 5.1 and that G_{lo} generates $\bigcup_{k} L(G_{l,k})$. □
Proposition states that satisfying the Joint-OOC property which is defined on $\bigcup_{k} L(G_{l,k})$ is equivalent to OOC property being satisfied by G_{lo}. In Proposition 5.4 and the following corollary we show how OOC property is inherited by G_{lo} where all $G_{l,k}$'s are individually OOC.

Proposition 5.4. *Consider the Moore automata $G_{l,k}$ $(k = 1, \cdots, n)$. Then $G_{l,k}$'s satisfy the Joint-OOC property if and only if each $G_{l,k}$ satisfies the OOC property.*

Proof: Remark 5.2 implies, an automaton G is OOC if every $s \in L(G)$ leading to a vocal state q satisfies the following:

(i) If the output at q is observable, then the last silent segment of s includes at least one observable event;

(ii) If the output at q is unobservable, then the last silent segment of s contains only unobservable events.

Now, if $G_{l,k}$'s are Jointly-OOC, then every string s in $L(G_{lo}) = \bigcup_k L(G_{l,k})$, (and thus every string in $L(G_{l,k})$) for which $\hat{\omega}(s) \neq \tau_o$ satisfies conditions (i) and (ii). Therefore, $G_{l,k}$ will be OOC (for every k). Conversely, if all $G_{l,k}$'s are OOC, then all strings in $L(G_{l,k})$ (for every k), and thus all strings in $L(G_{lo}) = \bigcup_k L(G_{l,k})$, will satisfy the above property. □

Finally, we have the following useful corollary.

Corollary 5.5. *$G_{l,k}$ $(k = 1, \cdots, n)$ are individually OOC if and only if $\bigcup_k G_{l,k}$, is OOC.*

Proof: Follows from Proposition and Proposition 5.4. □

Therefore, we assume the plant models $G_{l,k}$'s are Jointly-OCC and Jointly-OOC. If the models $G_{l,k}$'s are Joint-OCC and Joint-OOC then a control structure can be defined for them at the high-level. Let $P_{hi} : T^* \longmapsto T_o^*$ be a natural projection at the high-level. Similar to the Factorization Property (3.1) for a single model in Chapter 3, we argue that if two sequences $s \in L(G_{l,i})$ and $s' \in L(G_{l,j})$ $(1 \leq i, j \leq n)$ have the same observation at the low-level, i.e. $P_{lo}(s) = P_{lo}(s')$, then the images of s and s' should have the same observation at the high-level, i.e. $P_{hi}(\theta(s)) = P_{hi}(\theta(s'))$ where θ is given by (5.3); that is to say

$$P_{lo}(s) = P_{lo}(s') \Longrightarrow P_{hi}(\theta(s)) = P_{hi}(\theta(s')).$$

Equivalently, we should have

$$ker(P_{lo}) \leq ker(P_{hi} \circ \theta). \tag{5.6}$$

Condition (5.6) is, in fact, a generalization of the Factorization Property (3.1) to a model G_{lo} which generates $\bigcup_k L(G_{l,k})$ where G_{lo} is synthesized in Remark 4.4 by (4.7). Note that, by construction, if (5.6) holds, then (3.1) holds for each model $G_{l,k}$ and we can define the reporting maps $\tilde{\theta}_k$ and the output maps $\tilde{\omega}_k$ for observer automata $\tilde{G}_{l,k}$ $(k = 1, \cdots, n)$ as explained in Chapter 3. Therefore, (5.6) implies that for every $s \in L(G_{l,i})$ and $s' \in L(G_{l,j})$ $(i \neq j)$ if $\mathbf{S} = P_{lo}(s) = P_{lo}(s')$, then $P_{hi}(\theta_i(s)) = P_{hi}(\theta_j(s'))$. This implies we can define a reporting map $\tilde{\theta} : P_{lo}(\bigcup_k L(G_{l,k})) \longmapsto T_\circ^*$. For every $\mathbf{S} \in P_{lo}(\bigcup_k L(G_{l,k}))$

$$\tilde{\theta}(\mathbf{S}) = \tilde{\theta}_k(\mathbf{S}) \text{ for any k such that } \mathbf{S} \in L(\tilde{G}_{l,k}). \tag{5.7}$$

Similarly, an output map $\tilde{\omega} : P_{lo}(\bigcup_k L(G_{l,k})) \longmapsto T_\circ \cup \{\tau_\circ\}$ can be defined as follows. For every $\mathbf{S} \in P_{lo}(\bigcup_k L(G_{l,k}))$

$$\tilde{\omega}(\mathbf{S}) = \tilde{\omega}_k(\mathbf{S}) \text{ for any k such that } \mathbf{S} \in L(\tilde{G}_{l,k}). \tag{5.8}$$

Remark 5.6. *In the rest of this chapter, in addition to conditions (5.1) or (5.2), we assume that the maps P_{lo}, P_{hi} and θ satisfy the Factorization Property (5.6).*

We propose the following algorithm for satisfying conditions (5.1) (or (5.2)) and the Factorization Property (5.6):

Proposed algorithm for satisfying conditions (5.1) (or (5.2)) and the Factorization Property (5.6): Consider the low-level models $G_{l,k}$ (which are not yet equipped with an output map). We take advantage of the fact that with a given information map $\theta : \bigcup_k L(G_{l,k}) \longmapsto T^*$ which satisfies Factorization Property (3.1), the restrictions of θ to the individual languages $L(G_{l,k})$, say θ_k, would automatically satisfy condition (5.1) (or (5.2)) and (5.6). Specifically, the following steps can be taken to obtain a feasible set of reporting maps $\Theta = \{\theta_1, \cdots, \theta_n\}$.

(i) Let a map $\theta : \bigcup_k L(G_{l,k}) \longmapsto T^*$ be given or defined such that it satisfies the Factorization Property (5.6). If θ does not satisfy the Factorization Property, then θ can be modified by the algorithms given in Section 3.7, such that it satisfies the Factorization Property.

(ii) Let the restriction of θ to $L(G_{l,k})$ $(k = 1, \cdots, n)$ be called θ_k and call the corresponding output map ω_k.

(iii) Combine the model $G_{l,k}$ with the map ω_k (which corresponds to θ_k) to get the Moore automaton $\hat{G}_{l,k} = (Q_k, \Sigma_k, T \cup \{\tau_o\}, \delta_k, \omega_k, q_{o,k})$.

The reporting maps θ_k which correspond to output maps ω_k satisfy the conditions (5.1) (or (5.2)) and the Factorization Property (5.6) and hence θ is the information map (5.3) for the system.

5.4 Supervision Implementation At The Low-Level

The ability to synthesize the Moore low-level models that satisfy the Joint-OCC and Joint-OOC properties, equips the high-level with the control structure. In this section, we discuss supervisor design and implementation, and the issue of hierarchical consistency.

Let $G_{h,k}$ be the models at the high-level whose closed languages are $L(G_{h,k}) = \theta(L(G_{l,k}))$ $(k = 1, \cdots, n)$. We assume the desired behavior is given by $E'_{h,k}$. Let S_{hi} be the feasible solution to the robust control problem at the high-level and $E_{h,k} = L(S_{hi}/G_{h,k})$ the closed behavior of the system under supervision at the high-level for each model. Also, let $E_{hi} = \bigcup_k E_{h,k}$, $E_{lo} = \theta^{-1}(E_{hi})$ and finally $E'_{l,k} = \theta^{-1}(E'_{h,k})$ as the corresponding legal language for each model $G_{l,k}$ at the low-level. Let G_{hi} be any automaton at the high-level whose closed behavior is $L(G_{hi}) = \bigcup_k L(G_{h,k})$. Theorem 2.27 implies $L(S_{hi}/G_{hi}) = E_{hi}$ and E_{hi} is controllable and observable w.r.t $L(G_{hi})$. Also, let G_{lo} be the Moore automaton given by (4.7) in Remark 4.4. Then, $L(G_{hi}) = \bigcup_k \theta(L(G_{l,k})) = \theta(\bigcup_k L(G_{l,k})) = \theta(L(G_{lo}))$. Now, from Theorem 3.31, we conclude that if the high-level feasible supervisor S_{hi} is implemented by a disabled-event map Δ_{hi}, then there exists a low-level supervisor $\tilde{S}_{lo}$ which is implemented by a disabled-event map $\tilde{\Delta}_{lo}$ given in (3.14). We show $\tilde{S}_{lo}$ solves the robust supervisory control problem at the low-level for the set of models $G_{l,k}$ and the legal languages $E'_{l,k}$ and we investigate the relation between the behavior of system under supervision at the low-level $(L(S_{lo}/G_{l,k}))$ and that of the high-level $(L(S_{hi}/G_{h,k}))$. In Chapter 4 we showed that the closed behavior of the system under supervision for each model $G_{l,k}$ matches the supremal controllable sublanguage of the specification $E_{l,k} = \theta^{-1}(E_{h,k})$; however, under partial observation that would not necessarily be the case. In fact, reaching an supremal observable controllable sublanguage is not necessarily possible. Nevertheless, it was shown in Chapter 3 how hierarchical consistency under partial observation is ensured by the UUC, UUPO and PO-SOCC properties.

Definition 5.7 (repeated). *We say the property of robust hierarchical consistency holds in the system if* $\theta(L(\tilde{S}_{lo}/G_{l,k})) = E_{h,k}$ *for* $k = 1, \cdots, n$.

In hierarchical supervisory control of [59], to have hierarchical consistency, it must be ensured that low-level implementation of a high-level command to disable an high-level event does not produce unintended consequence in the form of disablement of other high-level events. In the case of control under partial observation, as shown in Chapter 3, if the plant is UUC, UUPO and PO-SOCC, then hierarchical consistency can be guaranteed. In HRSC under partial observation problem discussed here, we have to make sure that the implementation of a high-level command to disable an event does not produce unintended consequences in *any possible plant.* To ensure this, we bring in the Joint-UUC, Joint-UUPO and Joint-SOCC properties.

5.4.1 Joint-UUC Property

Recall that the UUC property (Property 3.19) requires for every $s \in L_{voc}$ and $s' \in L_{voc}(s)$ that $\hat{\omega}(ss') \in T_{uo} \cap T_c$, we have $|\Sigma^c(s')| = 1$. The UUC property is a property of silent paths, regardless of how they are connected to each other in the model. It is expected that if the models $G_{l,k}$ $(k = 1, \cdots, n)$ individually meet the UUC property, the union of them satisfies UUC property and vice versa.

Definition 5.8. *Consider a set of Moore automata* $G_{l,k} = (Q_k, \Sigma_k, T \cup \{\tau_o\}, \delta_k, \omega_k, q_{o,k})$ $(k = 1, \cdots, n)$. *Let* $L_{voc} = \{s \in \bigcup_k L(G_{l,k}) | s = \epsilon \ \text{ or } \ \hat{\omega}(s) = \tau_o\}$. *We say* $G_{l,k}$*'s satisfy the Joint-Unobservability-and-Unique-Controllability (Joint-UUC) property if for every* $s \in L_{voc}$ *property UUC holds.*

Proposition 5.9. *Consider a set of Moore automata* $G_{l,k}$ $(k = 1, \cdots, n)$. *Then* $G_{l,k}$*'s satisfy the Joint-UUC property if and only if each* $G_{l,k}$ *individually satisfies the UUC property.*

Proof (if): We use the fact that $L(G_{l,k}) \subseteq \bigcup_k L(G_{l,k})$ $(k = 1, \cdots, n)$.

Assume $G_{l,k}$'s satisfy the Joint-UUC property. Suppose for some $1 \leq j \leq n$ there exist a sequence $s \in L_{voc}$ and a silent path $r \in L_{voc}(s)$ such that $sr \in L(G_{l,j})$ and

$$\hat{\omega}_j(sr) \in T_c \cap T_{uo}. \tag{5.9}$$

We have $sr \in L(G_{l,j}) \subseteq L(G_{lo})$. Therefore, by (5.4),

$$\hat{\omega}(s) = \hat{\omega}_j(s) \neq \tau_\circ \qquad \text{and} \qquad \hat{\omega}(sr) = \hat{\omega}_j(sr) \in T_c \cap T_{uo}.$$

Next, note that r which is a silent path in $L(G_{l,j})$ will be a silent path in $\bigcup_k L(G_{l,k})$ since otherwise for some $r' \leq r$ by (5.4) we have $\hat{\omega}_j(sr') = \hat{\omega}(sr') \neq \tau_\circ$ which is a contradiction. Therefore r is a silent path in $\bigcup_k L(G_{l,k})$ and since $G_{l,k}$'s satisfy the Joint-UUC property, for $s \in L_{voc}$ and $r \in L_{voc}(s)$ we have

$$|\Sigma^c(r)| = 1. \tag{5.10}$$

Equations (5.9) and (5.10) imply that $G_{l,j}$ satisfies the UUC property for $1 \leq j \leq n$. This means each $G_{l,k}$ $(k = 1, \cdots, n)$ individually satisfies the UUC property.

(Only if): Assume each $G_{l,k}$ $(k = 1, \cdots, n)$ individually satisfies UUC property. Then, for every sequence $s \in L_{voc}$ and $r \in L_{voc}(s)$ that $sr \in \bigcup_k L(G_{l,k})$ and

$$\hat{\omega}(sr) \in T_c \cap T_{uo} \tag{5.11}$$

we have $sr \in L(G_{l,j})$ for some index $1 \leq j \leq n$. Then, again (5.4) implies that $s \in L_{voc} \cap L(G_{l,j})$, a silent path $r \in L_{voc}(s)$ extends s in $L(G_{l,j})$ and that

$$\hat{\omega}_j(sr) = \hat{\omega}(sr) \in T_c \cap T_{uo}.$$

Then, since by assumption $G_{l,j}$ satisfies the UUC property, it is concluded that

$$|\Sigma^c(r)| = 1. \tag{5.12}$$

Equation (5.12) implies that for every $s \in L_{voc}$ and $r \in L_{voc}(s)$ that (5.11) holds, $|\Sigma^c(r)| = 1$. This completes the proof that $G_{l,k}$'s satisfy the Joint-UUC property. □

Corollary 5.10. *$G_{l,k}$'s ($k = 1, \cdots, n$) individually satisfy the UUC property if and only if $G_{lo} = \bigcup_k G_{l,k}$ satisfies the UUC property.*

Proof: Follows from Proposition 5.9 and the definition of G_{lo}. □

5.4.2 Joint-UUPO Property

Recall that the UUPO property (Property 3.20) requires that every sequences $s \in L_{voc}$ and $r \in L_{voc}(s)$ with $P_{lo}(r) \neq \epsilon$ and $\hat{\omega}(sr) \in T_c$ be such that $\Sigma^c(\eta(r)) = \emptyset$ where for $r \in \Sigma^*$, $\eta(r) = \{u \mid u \in \Sigma_{uo}^*, \exists w \in \Sigma_o^+\Sigma^* \cup \{\epsilon\} : r = uw\}$ is the largest unobservable prefix of r.

It can be shown that individual models $G_{l,k}$ ($k = 1, \cdots, n$) inherit the UUPO property from G_{lo} and vice versa.

Definition 5.11. *Consider a set of Moore automata $G_{l,k} = (Q_k, \Sigma_k, T \cup \{\tau_\circ\}, \delta_k, \omega_k, q_{\circ,k})$ ($k = 1, \cdots, n$). Let $L_{voc} = \{s \in \bigcup_k L(G_{l,k}) | s = \epsilon$ or $\hat{\omega}(s) \neq \tau_\circ\}$. We say $G_{l,k}$'s satisfy the Joint-UUPO property if for every sequence $s \in L_{voc}$, the UUPO property holds.*

Proposition 5.12. *Consider a set of Moore automata $G_{l,k}$ ($k = 1, \cdots, n$). Then $G_{l,k}$'s satisfy the Joint-UUPO property if and only if each $G_{l,k}$ individually satisfies the UUPO property.*

Proof: The proof is similar to that for Proposition 5.9. We use the fact that $L(G_{l,k}) \subseteq \bigcup_k L(G_{l,k})$ $(k = 1, \cdots, n)$.

(if): Assume $G_{l,k}$'s satisfy the Joint-UUPO property and suppose for some sequences $s \in L_{voc}$ and $r \in L_{voc}(s)$ and some $1 \le j \le n$ that $sr \in L(G_{l,j})$

$$P_{lo}(r) \neq \epsilon \text{ and } \hat{\omega}_j(sr) \in T_c. \tag{5.13}$$

We have $sr \in L(G_{l,j}) \subseteq \bigcup_k L(G_{l,k})$. Therefore, similar to Proposition 5.9, (5.4) implies that r is a silent path in $\bigcup_k L(G_{l,k})$ with a controllable output $\hat{\omega}(sr) = \hat{\omega}_j(sr) \in T_c$. Then, since $G_{l,k}$'s satisfy the Joint-UUPO property,

$$\Sigma^c(\eta(r)) = \emptyset. \tag{5.14}$$

Equations (5.13) and (5.14) imply that $G_{l,j}$ satisfies the UUPO property where $1 \le j \le n$. Therefore, each automaton $G_{l,k}$ $(k = 1, \cdots, n)$ individually satisfies the UUPO property.

(Only if): Assume each $G_{l,k}$ $(k = 1, \cdots, n)$ individually satisfies the UUPO property. Then for every sequence $s \in L_{voc}$ and $r \in L_{voc}(s)$ that $sr \in \bigcup_k L(G_{l,k})$,

$$P_{lo}(r) \neq \epsilon \text{ and } \hat{\omega}(sr) \in T_c, \tag{5.15}$$

there exists some index $1 \le j \le n$ that $sr \in L(G_{l,j})$. Again (5.4) implies that r is a silent path in $G_{l,j}$, $s \in L_{voc} \cap L(G_{l,j})$, and that

$$P_{lo}(r) \neq \epsilon \text{ and } \hat{\omega}_j(sr) \in T_c.$$

Then, since by assumption $G_{l,j}$ satisfies the UUPO property,

$$\Sigma^c(\eta(r)) = \emptyset. \tag{5.16}$$

Equations (5.15) and (5.16) imply that $G_{l,k}$'s satisfy the Joint-UUPO property.

□

Corollary 5.13. *Consider a set of Moore automata $G_{l,k}$ $(k = 1, \cdots, n)$. Then, each $G_{l,k}$ $(k = 1, \cdots, n)$ individually satisfies the UUPO property if and only if G_{lo} satisfies the UUPO property.*

Proof: Follows from Proposition 5.12 and the definition of G_{lo}. □

In the rest of this chapter we assume the models $G_{l,k}$ satisfy the Joint-UUC and Joint-UUPO properties

5.4.3 Joint-PO-SOCC Property

The Joint-UUC and Joint-UUPO properties, in a sense, prevent unintended disablement of high-level unobservable events in the system. On the other hand, the PO-SOCC property, in Chapter 3, prevented unintended disablement of high-level observable events. We show how the PO-SOCC property is extended in HRSC problem under partial observation; some references are also made to P-partners (Definition 3.22) and the PO-SOCC property (Definition 3.23).

Definition 5.14. *Consider a set of Moore automata $G_{l,k} = (Q_k, \Sigma_k, T \cup \{\tau_o\}, \delta_k, \omega_k, q_{o,k})$ $(k = 1, \cdots, n)$ and let $\tilde{G}_{l,k}$ be the corresponding Moore observer automaton for $G_{l,k}$. We say $G_{l,k}$'s are Jointly-PO-SOCC if (i) $G_{l,k}$'s are Jointly-OCC and Jointly-OOC and (ii) no two P-vocal nodes with controllable outputs (nodes whose associated outputs are members of T_c), are P-partners in the reachability tree of $\bigcup\limits_k \tilde{G}_{l,k}$.*

We show how ensuring the Joint-PO-SOCC property guarantees each model $G_{l,k}$ $(k = 1, \cdots, n)$ meets the PO-SOCC property. The reverse is not true as shown in an example following Proposition 5.15.

Proposition 5.15. *Consider a set of Moore automata $G_{l,k}$ $(k = 1, \cdots, n)$. If $G_{l,k}$'s are Jointly-PO-SOCC, then each $G_{l,k}$ is PO-SOCC.*

Proof: Assume $G_{l,k}$'s $(k = 1, \cdots, n)$ are Jointly-PO-SOCC. Let $\tilde{G}_{l,k}$ be the observer automaton which corresponds to $G_{l,k}$ and $\tilde{\omega}_k$ be the output map for $\tilde{G}_{l,k}$. Then, let

$$\tilde{G}_{lo} = trim[(\tilde{G}^{co}_{l,1} \times \cdots \times \tilde{G}^{co}_{l,n})^{co}]$$

so that $L(\tilde{G}_{lo}) = \bigcup_k L(\tilde{G}_{l,k})$ and the output map $\tilde{\omega} : L(\tilde{G}_{lo}) \longmapsto T_\circ \cup \{\tau_\circ\}$ is given by (5.8). In fact, it can be shown that $\tilde{G}_{lo}$ is an observer automaton for G_{lo} given by (4.7). If $G_{l,k}$'s are Jointly-PO-SOCC, no two P-partners are found in the RT of $\tilde{G}_{lo}$. Now suppose, by contradiction, for some $1 \leq j \leq n$, $G_{l,j}$ is not PO-SOCC. In other words, there exist a sequence $\mathbf{S} \in L(\tilde{G}_{l,j})$ with $\mathbf{S} = \epsilon$ or $\tilde{\omega}_j(\mathbf{S}) \neq \tau_\circ$ and sequences $\mathbf{S}_1, \mathbf{S}_2, \mathbf{S}_3 \in \Sigma_\circ^+$ such that $\mathbf{S}_1\mathbf{S}_2$ and $\mathbf{S}_1\mathbf{S}_3$ are two P_{lo}-controllable P_{lo}-silent paths which extend $\mathbf{S}$ in the RT of $\tilde{G}_{l,j}$, $\tilde{\omega}(\mathbf{S}\mathbf{S}_1\mathbf{S}_2) \neq \tilde{\omega}(\mathbf{S}\mathbf{S}_1\mathbf{S}_3)$ and at least one of the sequences $\mathbf{S}_2$ or $\mathbf{S}_3$ is P_{lo}-uncontrollable. Without loss of generality, assume $\mathbf{S}_2$ is P_{lo}-uncontrollable. Because the RT of $\tilde{G}_{l,j}$ is a subtree of the RT of $\tilde{G}_{lo}$, we have $\mathbf{S} \in L(\tilde{G}_{lo})$. Next, (5.8) implies either $\mathbf{S} = \epsilon$ or $\tilde{\omega}(\mathbf{S}) \neq \tau_\circ$ and furthermore, $\mathbf{S}_1\mathbf{S}_2$ and $\mathbf{S}_1\mathbf{S}_3$ are P_{lo}-controllable P_{lo}-silent paths which extend $\mathbf{S}$ in the RT of $\tilde{G}_{lo}$. Therefore we have

$$\begin{aligned}
\tilde{\omega}(\mathbf{S}\mathbf{S}_1\mathbf{S}_2) &= \tilde{\omega}_j(\mathbf{S}\mathbf{S}_1\mathbf{S}_2) \in T_c; \\
\tilde{\omega}(\mathbf{S}\mathbf{S}_1\mathbf{S}_3) &= \tilde{\omega}_j(\mathbf{S}\mathbf{S}_1\mathbf{S}_3) \in T_c; \\
\tilde{\omega}(\mathbf{S}\mathbf{S}_1\mathbf{S}_2) &\neq \tilde{\omega}(\mathbf{S}\mathbf{S}_1\mathbf{S}_3). \qquad (5.17)
\end{aligned}$$

Nevertheless, the controllability of $\mathbf{S}_2$ is yet to be determined. Since $\mathbf{S}_2$ is P_{lo}-uncontrollable (from the node corresponding to $\mathbf{S}\mathbf{S}_1$) in $\tilde{G}_{l,j}$, there exist sequences $s \in L(G_{l,j})$, $s_1 \in \Sigma^*$ and $s_2 \in \Sigma_\circ^+\Sigma^*\Sigma_\circ^+$ with $P_{lo}(s) = \mathbf{S}$, $P_{lo}(s_1) = \mathbf{S}_1$

and $P_{lo}(s_2) = \mathbf{S}_2$ such that $s_1s_2 \in L_{voc}(s)$ is a silent path in $G_{l,j}$, and

$$s_2 \in \Sigma_{uc}^{+}. \tag{5.18}$$

Then, since $ss_1s_2 \in L(G_{l,j}) \subseteq L(G_{lo})$, we have

$$ss_1s_2 \in P_{lo}^{-1}(\mathbf{S}\mathbf{S}_1\mathbf{S}_2) \cap L(G_{lo}). \tag{5.19}$$

Equations (5.18) and (5.19) imply that $\mathbf{S}_2$ is P_{lo}-uncontrollable from the node corresponding to $\mathbf{S}\mathbf{S}_1 = P_{lo}(ss_1)$ in $\tilde{G}_{lo}$. Therefore, (5.17) and the fact that $\mathbf{S}_2$ is P_{lo}-uncontrollable imply that the nodes $\mathbf{S}\mathbf{S}_1\mathbf{S}_2$ and $\mathbf{S}\mathbf{S}_1\mathbf{S}_3$ are P-partners in the RT of $\tilde{G}_{lo} = \bigcup\limits_k \tilde{G}_{l,k}$. This is a contradiction since $G_{l,k}$'s are Jointly-PO-SOCC. Therefore, $\mathbf{S}_2$ cannot be P_{lo}-uncontrollable in $\tilde{G}_{l,j}$ and no two nodes with controllable outputs are P-partners in the RT of $\tilde{G}_{l,j}$. Hence, $G_{l,j}$ is PO-SOCC. This completes the proof that each $G_{l,k}$ $(k = 1, \cdots, n)$ is PO-SOCC. □

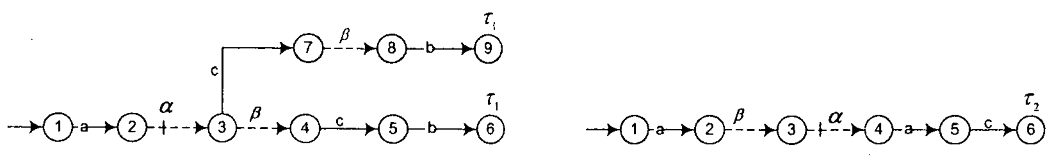
Proposition 5.16. *The models $G_{l,k}$'s $(k = 1, \cdots, n)$ are Jointly-PO-SOCC if and only if $G_{lo} = \bigcup\limits_k G_{l,k}$ is PO-SOCC.*

Proof: Follows from Definition 5.14 and that $\tilde{G}_{lo}$ generates $\bigcup\limits_k L(\tilde{G}_{l,k})$. □

Remark 5.17. *If the models $G_{l,k}$'s are not Joinly-UUC, Jointly-UUPO or Joinly-PO-SOCC, then it follows that the model $G_{lo} = \bigcup\limits_k G_{l,k}$, given by (4.7), is not UUC, UUPO or PO-SOCC. Therefore, by ensuring $G_{lo} = \bigcup\limits_k G_{l,k}$ satisfies above properties, as discussed in the Appendix of Chapter 3, we can ensure $G_{l,k}$'s are Joinly-UUC, Jointly-UUPO and Joinly-PO-SOCC.*

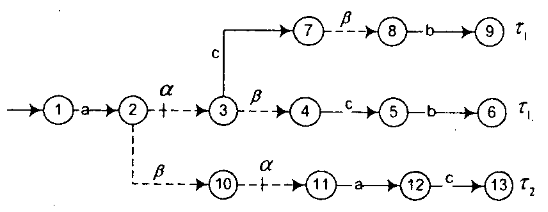
The following shows that if the models are individually PO-SOCC, they are not necessarily Jointly-PO-SOCC.

Example 5.18. *Figs. 5.1.(a) and 5.1.(b) show two models $G_{l,1}$ and $G_{l,2}$ which are PO-SOCC. There, $\Sigma = \{a, b, c, \alpha, \beta\}$ in which $\Sigma_o = \{a, b, c\}$ and $\Sigma_c = \{\alpha\}$. Also $T = T_c = T_o = \{\tau_1, \tau_2\}$. Fig. 5.2 shows the union model $G_{lo} = G_{l,1} \cup G_{l,2}$ which is not PO-SOCC. In other words, $G_{l,1}$ and $G_{l,2}$ are individually PO-SOCC but they are not Jointly-PO-SOCC. Let $\tilde{G}_{lo}$, in Fig. 5.3, be the observer automaton of G_{lo}. In Fig. 5.3, in the RT of $\tilde{G}_{lo}$, the node **a.a.c** is P-partner with the node **a.c.b** since the outputs $\tau_1 = \tilde{\omega}(\boldsymbol{a.c.b})$ and $\tau_2 = \tilde{\omega}(\boldsymbol{a.a.c})$ are controllable, $\tau_1 \neq \tau_2$ and furthermore, the segments **a.c** and **c.b** are P_{lo}-uncontrollable. Therefore, disabling τ_1 and τ_2 might not be independent from each other as is the case here too. In fact, if a feasible robust supervisor disables τ_1 in Fig. 5.2, the event α after state 2 should be disabled as a result of which event α after state 10 is disabled. This yields the unintended disablement of event τ_2 at the high-level. Note that G_{lo} in Fig. 5.2 is SOCC and this unintended disablement would not occur if full observation was assumed.*

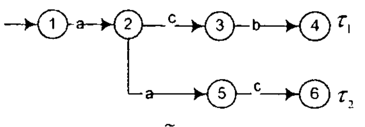


(a): $G_{l,1}$ (b): $G_{l,2}$

Figure 5.1: Two models $G_{l,1}$ (a) and $G_{l,2}$ (b) which are individually PO-SOCC.



G_{lo}



$\tilde{G}_{lo}$

Figure 5.2: Union model $G_{lo} = G_{l,1} \cup G_{l,2}$ which is not PO-SOCC.

Figure 5.3: Observer automaton $\tilde{G}_{lo}$ which corresponds to G_{lo}

5.4.4 Robust Hierarchical Consistency under Partial Observation

Next, we show how robust hierarchical consistency property can be ensured in an HRSC problem under partial observation. Let S_{hi} be the feasible robust supervisor which solves the RSC problem at the high-level and $\tilde{S}_{lo}$ the induced feasible supervisor at the low-level which corresponds to S_{hi} and is given by (3.15).

Theorem 5.19. *If the low-level Moore automata $G_{l,k}$ $(k = 1, \cdots, n)$ are Joinly-UUC, Jointly-UUPO and Jointly-PO-SOCC, then $\tilde{S}_{lo}$ is a feasible robust supervisor for models $G_{l,k}$'s and legal languages $E'_{l,k}$'s and we have:*

(i) $\theta(L(\tilde{S}_{lo}/G_{lo})) = E_{hi}$;

(ii) $\theta(L(\tilde{S}_{lo}/G_{l,k})) = E_{h,k}$.

Proof: **(i)**: Let $G_{hi} = trim[(G_{h,1}^{co} \times \cdots \times G_{h,n}^{co})^{co}]$. Union operation ($\cup$) and information map θ commute with each other and it can easily be shown that $L(G_{hi}) = \theta(L(G_{lo}))$ where G_{lo} is given by (4.7). On the other hand, solving the robust supervisory control at the high-level amounts to solving supervisory control for the plant G_{hi} and the legal behavior $E_{hi} = \bigcup_k E_{h,k}$. Thus $L(S_{hi}/G_{hi}) = E_{hi}$. Then, since $G_{l,k}$'s satisfy the Joint-UUC, Joint-UUPO and Joint-PO-SOCC properties (G_{lo} satisfies the UUC, UUPO and PO-SOCC properties), it follows from Theorem 3.31 that for the feasible supervisor $\tilde{S}_{lo}$

$$\theta(L(\tilde{S}_{lo}/G_{lo})) = L(S_{hi}/G_{hi}) = E_{hi}.$$

(ii): First, note that by Lemma 2.28 we have:

$$L(\tilde{S}_{lo}/G_{l,k}) = L(\tilde{S}_{lo}/G_{lo}) \cap L(G_{l,k}). \qquad (5.20)$$

Therefore we have:

$$\begin{aligned}
\theta(L(\tilde{S}_{lo}/G_{l,k})) &= \theta(L(\tilde{S}_{lo}/G_{lo}) \cap L(G_{l,k})) \\
&\subseteq \theta(L(\tilde{S}_{lo}/G_{lo})) \cap \theta(L(G_{l,k})) \\
&= E_{hi} \cap \theta(L(G_{l,k})) \quad \text{(from part (i))} \\
&= E_{hi} \cap L(G_{h,k}) \quad \text{(by definition of } G_{h,k}\text{)} \\
&= E_{h,k} \quad \text{(since } S_{hi} \text{ solves RSC)}
\end{aligned}$$

which means

$$\theta(L(\tilde{S}_{lo}/G_{l,k})) \subseteq E_{h,k}. \tag{5.21}$$

For the reverse inclusion we have:

$$\begin{aligned}
E_{h,k} &= E_{hi} \cap L(G_{h,k}) \quad \text{(since } S_{hi} \text{ solves RSC)} \\
&= \theta(L(\tilde{S}_{lo}/G_{lo})) \cap L(G_{h,k}) \quad \text{(from part (i))} \\
&= \theta(L(\tilde{S}_{lo}/G_{lo})) \cap \theta(L(G_{l,k}))
\end{aligned} \tag{5.22}$$

Now let T be the event set at the high-level. For any $t \in T^*$ that

$$t \in \theta(L(\tilde{S}_{lo}/G_{lo})) \cap \theta(L(G_{l,k})) \tag{5.23}$$

let $\Sigma'_t = \{s \in L(G_{lo})| \; \omega(s) \neq \tau_o \text{ and } \theta(s) = t\}$ be the strings which lead to vocal states in $L(G_{lo})$ and $\theta(s) = t$. By Lemma 5.20 in the Appendix we have:

$$\Sigma'_t \subseteq L(\tilde{S}_{lo}/G_{lo}). \tag{5.24}$$

Also, $t \in \theta(L(G_{l,k}))$ implies

$$\Sigma'_t \cap L(G_{l,k}) \neq \emptyset \tag{5.25}$$

Equations (5.24) and (5.25) together imply for some $s' \in \Sigma'_t$ we have:

$$s' \in L(\tilde{S}_{lo}/G_{lo}) \cap L(G_{l,k}) \quad \text{and} \quad \theta(s') = t, \tag{5.26}$$

which implies

$$t \in \theta(L(\tilde{S}_{lo}/G_{lo}) \cap L(G_{l,k})). \tag{5.27}$$

Therefore, equations (5.22), (5.23) and (5.27) imply that

$$\begin{aligned} E_{h,k} &= \theta(L(\tilde{S}_{lo}/G_{lo})) \cap \theta(L(G_{l,k}) \\ &\subseteq \theta(L(\tilde{S}_{lo}/G_{lo}) \cap L(G_{l,k})) \\ &= \theta(L(\tilde{S}_{lo}/G_{l,k})) \text{ (by Lemma 2.28).} \end{aligned} \tag{5.28}$$

Equations (5.21) and (5.28) complete the proof that $\theta(L(\tilde{S}_{lo}/G_{l,k})) = E_{h,k}$. Furthermore, it can be concluded from (5.21) that

$$\begin{aligned} L(\tilde{S}_{lo}/G_{l,k}) &\subseteq \theta^{-1}(E_{h,k}) \\ &\subseteq \theta^{-1}(E'_{h,k}) \\ &\subseteq E'_{l,k} \end{aligned} \tag{5.29}$$

Equation (5.29) implies $\tilde{S}_{lo}$ is a robust supervisor for the models $G_{l,k}$ ($k = 1, \cdots, n$) and the corresponding legal behavior $E'_{l,k}$ where $\tilde{S}_{lo}$ is feasible by construction. □

5.5 Conclusion

In this chapter a hierarchical solution for the problem of robust supervisory control under partial observation for a finite family of DES plants was derived, based on Zhong-Wonham approach. Conditions were derived so that a unique reporting map can be defined and constructed for the system. The Joint-OCC and Joint-OOC properties, as the extensions of OCC and OOC properties, were shown to enure a control structure at the high-level. We showed that the Joint-UUC, Joint-UUPO and Joint-PO-SOCC properties, as the extensions of UUC, UUPO and PO-SOCC properties, suffice to ensure robust hierarchical consistency in the system under partial observation. It was also shown how a feasible robust supervisor can be constructed at the low-level.

5.6 Appendix

Consider the supervisors $\tilde{S}_{lo}$ given by (3.15) and S_{lo} given by (3.16). Now, let $t \in L(S_{hi}/G_{hi})$ be a sequence in the high-level system under supervision and define $\Sigma'_t = \{s \in L(G_{lo})|\ \omega(s) \neq \tau_o \text{ and } \theta(s) = t\}$ to be the low-level strings whose images are $t \in T^*$ and lead to vocal states in G_{lo}. Theorem 3.31 which states $\theta(L(\tilde{S}_{lo}/G_{lo})) = L(S_{hi}/G_{hi}) = E_{hi}$ implies that for every $t \in L(S_{hi}/G_{hi})$ there exists some sequence $s \in \Sigma'_t$ that $s \in L(\tilde{S}_{lo}/G_{lo})$. Lemma 4.18 states that $\Sigma'_t \subseteq L(S_{lo}/G_{lo})$, however, the membership of the set Σ'_t in $L(\tilde{S}_{lo}/G_{lo})$ should still be investigated. Lemma 5.20 gives an extension of Lemma 4.18 to the case of control under partial observation.

Lemma 5.20. *For every $t \in L(S_{hi}/G_{hi})$, let $\Sigma'_t = \{s \in L(G_{lo})|\ \omega(s) \neq \tau_o$ and $\theta(s) = t\}$. Then, if G_{lo} is PO-SOCC, $\Sigma'_t \subseteq L(\tilde{S}_{lo}/G_{lo})$.*

Proof: Theorem 3.31 states $\theta(L(\tilde{S}_{lo}/G_{lo})) = L(S_{hi}/G_{hi}) = E_{hi}$, hence we have $t \in \theta(L(\tilde{S}_{lo}/G_{lo})))$. Now, suppose, by contradiction, that for some

$s \in \Sigma_t'$, $s \notin L(\tilde{S}_{lo}/G_{lo})$). Let $s_2 < s$ be the largest prefix of s in $L(\tilde{S}_{lo}/G_{lo})$. The definition of $\tilde{S}_{lo}$ implies that there exists $\sigma \in \Sigma_c$ and $s_3 \in \Sigma^*$ such that $s = s_2\sigma s_3$ with $s_2 \in L(\tilde{S}_{lo}/G_{lo})$, $\theta(s_2) \in \theta(L(\tilde{S}_{lo}/G_{lo}))$ and $\sigma \in \tilde{\Delta}_{lo}(s_2, \theta(s_2))$. Since $s_2 < s$ and s is vocal, $\theta(s_2) \neq \theta(s)$ and hence the sequence σs_3 goes through at least one vocal state. Thus there exist $s_4 \leq s_3$ and $\tau \in T$ such that $\sigma s_4 \in L_{voc}(s_2)$ and $\hat{\omega}(s_2\sigma s_4) = \tau$. $\sigma \in \tilde{\Delta}_{lo}(s_2, \theta(s_2))$ is disabled by $\tilde{S}_{lo}$ but since by Lemma 4.18, $\Sigma_t' \subseteq L(S_{lo}/G_{lo})$ and also $L(S_{lo}/G_{lo})$ is closed, $\sigma \notin \Delta_{lo}(s_2, \theta(s_2))$ and $s_2\sigma s_4 \in L(S_{lo}/G_{lo})$ or equivalently

$$\theta(s_2\sigma s_4) = \theta(s_2)\tau \in \theta(L(S_{lo}/G_{lo})). \tag{5.30}$$

Recall from Theorem 2.33 that $L(S_{lo}/G_{lo}) = E_{lo}^{\uparrow}$. Therefore, $\sigma \notin \Delta_{lo}(s_2, \theta(s_2))$ implies

$$s_2\sigma \in E_{lo}^{\uparrow}. \tag{5.31}$$

Next, note that since $\sigma \in \tilde{\Delta}_{lo}(s_2, \theta(s_2))$, by (3.14), there should exist $s' \in P_{lo}^{-1}P_{lo}(s_2) \cap L(G_{lo})$ and $t' \in P_{hi}^{-1}P_{hi}(\theta(s_2)) \cap L(G_{hi})$ such that $\sigma \in \Delta_{lo}(s', t')$. The Factorization Property (3.1) for s_2 and s' with $P_{lo}(s') = P_{lo}(s_2)$ implies $P_{hi}(\theta(s')) = P_{hi}(\theta(s_2))$ and since $P_{hi}(t') = P_{hi}(\theta(s_2))$, we have $P_{hi}(t') = P_{hi}(\theta(s'))$. Then, since S_{hi} is feasible it is concluded that $\Delta_{hi}(t') = \Delta_{hi}(\theta(s'))$. This implies $\sigma \in \Delta_{lo}(s', \theta(s'))$ and hence

$$s'\sigma \notin E_{lo}^{\uparrow}. \tag{5.32}$$

Let $u' \in \Sigma_{uc}^*$ be the corresponding sequence in definition of Δ_{lo} for which we have $\sigma u' \in L_{voc}(s')$ and $\hat{\omega}(s'\sigma u') \in \Delta_{hi}(\theta(s'))$. Let $\hat{\omega}(s'\sigma u') = \tau'$ for some $\tau' \in T$. Therefore, we should have

$$\tau' \in \Delta_{hi}(\theta(s')) \tag{5.33}$$

and $\theta(s')\tau' \notin \theta(L(S_{lo}/G_{lo}))$. Recall that we assumed $\hat{\omega}(s_2\sigma s_4) = \tau$. We have $P_{lo}(s_2) = P_{lo}(s')$ and that

$$s_2\sigma \in E_{lo}^{\uparrow} \text{ and } s'\sigma \in L(G_{lo}) \text{ and } s'\sigma \notin E_{lo}^{\uparrow} \tag{5.34}$$

where the first and the last term were given respectively in (5.31) and (5.32). Note that by Proposition 3.26, the system is SOCC, hence we have $act_\theta(s'\sigma) = \{\tau'\}$. Now if $\tau \neq \tau'$, then $act_\theta(s_2\sigma) - act_\theta(s'\sigma) \neq \emptyset$ which together with (5.34) imply $E_{lo}^{\uparrow}$ is not (G, P_{lo}, θ)-observable. This cannot be the case since G_{lo} is PO-SOCC, hence by Proposition 3.29, $E_{lo}^{\uparrow}$ is (G, P_{lo}, θ)-observable. This implies $\tau = \tau'$. Besides, S_{hi} is feasible which implies $\Delta_{hi}(\theta(s_2)) = \Delta_{hi}(\theta(s'))$. Therefore, (5.33) implies $\tau \in \Delta_{hi}(\theta(s_2))$ or equivalently $\theta(s_2)\tau \notin E_{hi}$. Then, since by Theorem 2.37 $\theta(L(S_{lo}/G_{lo})) = E_{hi}$, we have

$$\theta(s_2)\tau \notin \theta(L(S_{lo}/G_{lo})) \tag{5.35}$$

which contradicts (5.30). Therefore, $s_2 = s$ and thus for every $s \in \Sigma_t'$, $s \in L(\tilde{S}_{lo}/G_{lo})$. This completes the proof that $\Sigma_t' \subseteq L(\tilde{S}_{lo}/G_{lo})$. □

Chapter 6

Case Study

A flexible manufacturing system (FMS) consisting of two machines and two buffers is studied in this chapter. This case study, in part, has been inspired by the example given in [42]; however, our framework, modeling and results differ from those in [42]. In this case study, we use the hierarchical robust supervisory control (HRSC) framework to model an FMS. First, the problem statement, which includes an overview of the system and the interactions among the components of the system, is given. Next, different components and interactions within the system are modeled. This includes obtaining the low-level model. Following this, the outputs of system are chosen properly so that enough information is gathered from the system and hence the specifications at the high-level can be recovered through the low-level implementation. This includes choosing the vocal states and obtaining the high-level models. Then, the high-level specifications are modeled. Finally, the HRSC problem is solved. It is shown that the Joint-SOCC property holds in the system and hence the system satisfies the robust hierarchical consistency property. Finally, possible extensions of this case study are discussed.

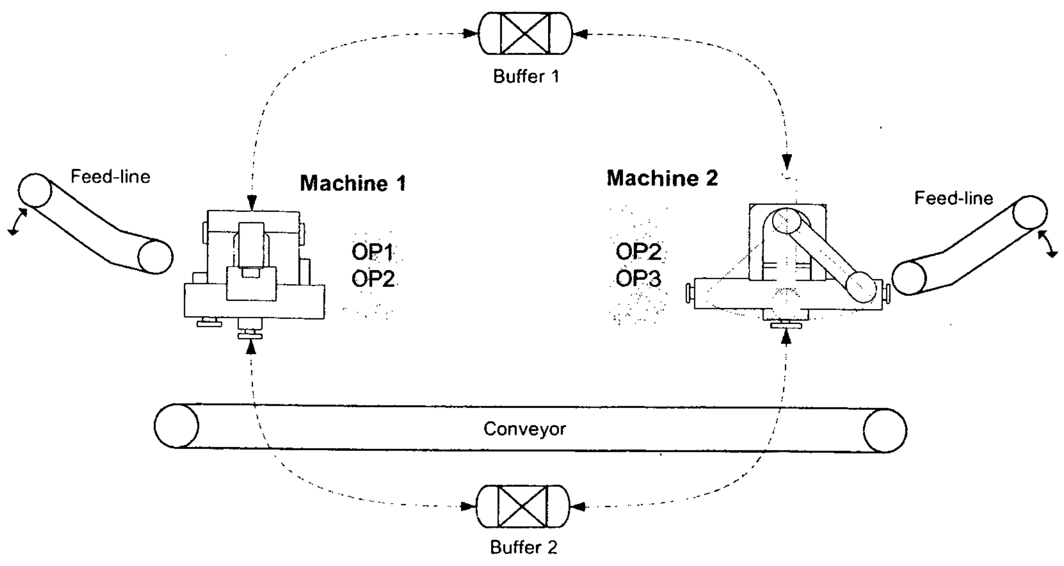


Figure 6.1: A flexible manufacturing system, consisting of two machines and two buffers

6.1 Problem Statement

Fig. 6.1 shows a flexible manufacturing system (FMS) which consists of two machines, two buffers and one conveyor. The machines work independently and can be run in parallel. Machine 1 can perform Operations 1 and 2 (OP1 and OP2) and machine 2 can perform Operations 2 and 3 (OP2 and OP3). To start a new product, new workpieces are fed into the machines through two *Feed-lines.* When a machine completes its task, it deposits the workpiece into either a buffer or the conveyor depending on the product recipes. Each buffer has a capacity of one and each buffer has access to both machines. Therefore, a transfer from one machine to another is possible via buffers. Two products recipes **A** = OP1+OP3+OP2 and **B** = OP3+OP2 are available. Safety regulation requires that after two consecutive production of **A**, one **B** is produced. Furthermore, the system is prone to fault. For brevity, we only consider failure in Machine 1. Machine 1 might fail to perform OP2 at any time. In order to start the process, the FMS should receive the start signal from the master controller of the plant. The system might also be shut down

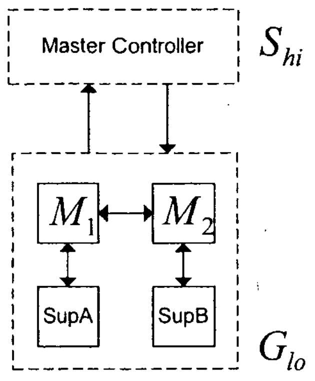


Figure 6.2: Master controller vs. recipe controllers

at any time (stop signal). It is required that the buffers do not overflow or underflow, the system satisfies the safety regulation at both the normal and faulty modes (i.e. after every two **A**'s, one **B** is produced) and the maximum capability of the machines be employed at any time (i.e. the machines can be used simultaneously). Finally a repair action is possible only when the system has stopped.

Remark 6.1. *As explained in the following sections, two types of controllers are designed for the FMS we already described. One type is the recipe controller which is designed for each product, regardless of the presence of the other product in the system (SupA and SupB in Fig. 6.2). Each recipe controller works independently. However, there are other interactions in the system in which we are more interested. Handling the shared resources of the two recipe controllers and managing the order in which the products are produced are issues which cannot be taken into consideration by the individual recipe controllers. We use the term* master controller *to simply refer to the controller for such inter-recipe interactions (S_{hi} in Fig. 6.2). Moreover, as we will see later, different levels of authority exist in the system where, for example, the command* STOP *which is uncontrollable for the recipe controller is considered controllable for the master controller. The machines under the supervision of recipe*

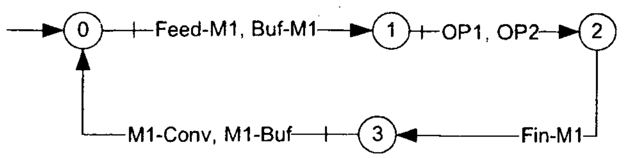


Figure 6.3: General performance of Machine 1 without considering the interactions within the FMS

controllers form the low-level model G_{lo} used in a hierarchical approach to design the master controller (based on the HRSC problem). The details are given in the following sections.

6.2 Modeling

6.2.1 Normal Mode

Machine 1 and 2: The general performance of Machine 1, without considering the product recipes, is given in Fig. 6.3. Machine 1 starts a task by either taking a new workpiece from the Feed-line (Feed-M1) or by taking a previously processed workpiece from a buffer (Buf-M1). It then performs an operation (OP1 or OP2) on the workpiece, signals the end of the task (Fin-M1) and deposits the workpiece in either the buffer (M1-Buf) or the conveyor (M1-Conv).

The model described in Fig. 6.3, although generally correct for modeling, does not represent the real interactions in our setup. First, the resource sharing and second, the maximum flexibility that we want to have from the machines in the system, require more specific and distinct events in the system so that a certain task, say OP2, can independently be enabled or disabled for different product recipes; we assume an operation which is performed by either of the machines, does not necessarily have the same effect for the workpieces of **A** and B. In other words, OP1 for workpieces **A** and **B** is not necessarily the same. As another example, consider OP2 which is common between **A** and **B**. If it is desired that **A** is done before **B**, at some point in the run-time of the system, OP2 should be enabled for **A** and disabled for **B**. This implies disabling OP2 for **A** and disabling OP2 for **B** should be distinguished. Machine 1 and Machine 2 can perform OP2. This common capability should also be addressed and included in the modeling. Finally, START and STOP commands should be incorporated into the component models as appropriate. These facts suggest more elaborated models are needed here. Next, we present the complete models of the components.

Fig. 6.4 shows the complete dynamics of Machine 1. Several improvements

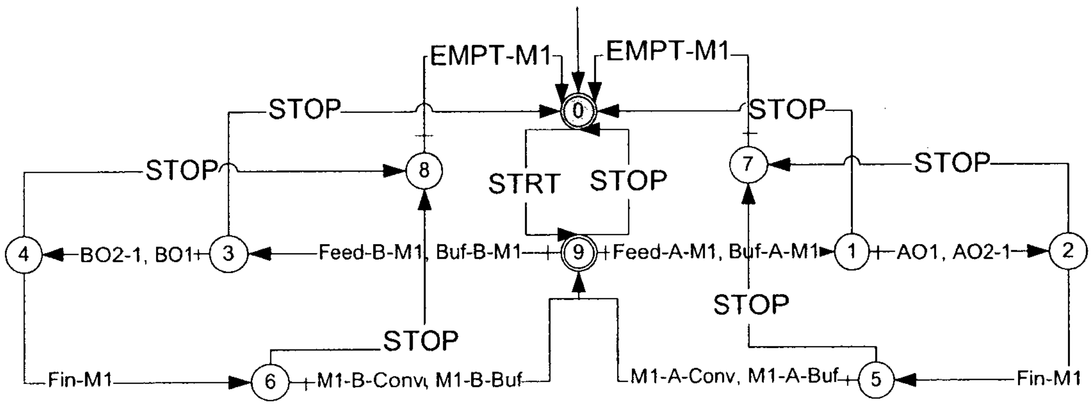


Figure 6.4: M1: Machine 1 behavior in Normal mode

can be seen in Fig. 6.4 over the model in Fig. 6.3. The model in Fig. 6.4 consists of two symmetric parts; the right side corresponds to product **A** and the left side corresponds to product **B**. The events which denote a task for **A**, hold a tag A in their names. Similarly, the events which denote a task for **B**, hold a tag B in their names. Thus, event *Buf-A-M1* represents a transfer of the workpiece of product **A** from the buffer to Machine 1 and event *M1-B-Conv* indicates the completion of the product B by Machine 1 (depositing **B** into the conveyor). It should be noted that, from Machine 1 point of view, two events *M1-A-Conv* and *M1-B-Conv* are essentially the same and denote the same task, namely, depositing a workpiece into the conveyor. However, one flags a product **A** and one flags a product **B** completion and hence, supervisor considers them as different events.

The models of Machine 1 in Fig. 6.4 can be explained as follows. Machine 1 starts from state 0 with the command *STRT* which is uncontrollable and is issued by the master controller. It evolves through the states 1-6 and rests at state 9 unless a STOP command occurs. If a *STOP* command is issued, the system switches off and goes to the initial state 0. An *EMPT-M1* action might then be employed to empty Machine 1 if it is not at the rest state 9 when the *STOP* command is issued. Machine 2 works similarly except that it can perform operations OP2 and OP3.

Then, a task schedule for processing product *A* follows: Machine 1 takes a workpiece **A** either from the Feed-line (*Feed-A-M1*) or from the buffer (*Buf-A-M1*), processes it (*AO1* or *AO2-1*), signals the completion of the job (*Fin-M1*) and deposits the workpiece into either the conveyor (*M1-A-Conv*) or the buffer (*M1-A-Buf*). Product **A** is processed similarly in Machine 2 (see Fig.

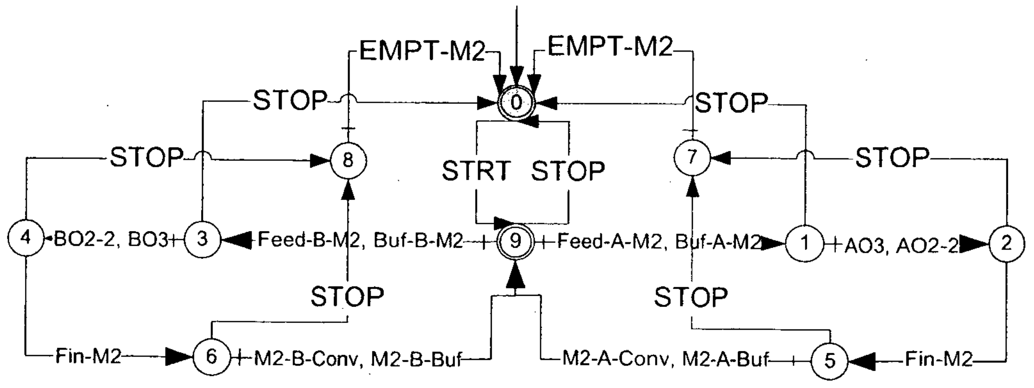


Figure 6.5: M2: Machine 2 behavior in Normal mode

6.5); Machine 2 initially starts with the *STRT* command, then takes the workpiece **A** either from the Feed-line (*Feed-A-M2*) or from the buffer (*Buf-A-M2*), processes it (*AO2-2* or *AO3*). signals the completion of the job (*Fin-M2*) and finally deposits the workpiece into either the conveyor (*M2-A-Conv*) or the buffer (*M2-A-Buf*).

The same tasks are allowed for product **B** with tag *A* replaced with *B* in above actions (events). Note that both of the events *AO2-1* and *AO2-2* denote operation OP2 where the former is performed in Machine 1 and the latter is performed in Machine 2. Table 6.1 gives the list of actions and their task definitions in the system. In the next sections, we will refer to Machines 1 and Machine 2 in Fig. 6.4 and 6.5 as M1 and M2.

Buffer Specifications: Buffers represent the robots in our setup. They can be considered as the medium for transferring workpieces between the machines. Buffers should not underflow or overflow. Therefore, a general specification for a buffer, without considering the interactions within the FMS, could be given by Fig. 6.6. Fig. 6.6 shows a workpiece can be deposited into the buffer by

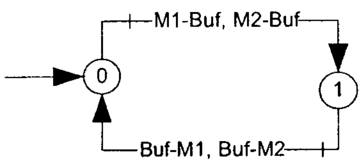


Figure 6.6: General performance of a Buffer without considering the interactions within the FMS

either of the machines (*M1-Buf* or *M2-Buf*) and, in return, buffer deposits the workpiece into one of the machines (*Buf-M1*, *Buf-M2*) for further processing. This prevents the buffer from under- or overflowing. However, the model in Fig. 6.6 is not complete and ignores the resource sharing and the *STOP* command occurrence in the system; it does not differentiate between a transfer of workpiece **A** and a transfer of workpiece **B**, and furthermore, if buffer is full and a *STOP* command is issued, the buffer takes no action to be emptied. We further assume that each buffer is in charge of one product. This assumption means one buffer handles product **A** transfers and one buffer handles product

Event	c/uc	Definition
AO1	c	Machine 1 performs OP1 for making **A**
AO2-1	c	Machine 1 performs OP2 for making **A**
AO2-2	c	Machine 2 performs OP2 for making **A**
AO3	c	Machine 2 performs OP2 for making **A**
BO1	c	Machine 1 performs OP1 for making **B**
BO2-1	c	Machine 1 performs OP2 for making **B**
BO2-2	c	Machine 2 performs OP2 for making **B**
BO3	c	Machine 2 performs OP3 for making **B**
Buf-A-M1	c	Buffer transfers a workpiece to Machine 1 for making **A**
Buf-A-M2	c	Buffer transfers a workpiece to Machine 2 for making **A**
Buf-B-M1	c	Buffer transfers a workpiece to Machine 1 for making **B**
Buf-B-M2	c	Buffer transfers a workpiece to Machine 2 for making **B**
EMPT-M1	c	Machine 1 is emptied
EMPT-M2	c	Machine 2 is emptied
EMPT-bufA	c	Buffer **A** is emptied
EMPT-bufB	c	Buffer **B** is emptied
Fin-M1	c	Machine 1 finished a process
Fin-M2	c	Machine 2 finishes a process
Feed-A-M1	c	New workpiece is loaded into Machine 1 for making **A**
Feed-B-M1	c	New workpiece is loaded into Machine 1 for making **B**
Feed-A-M2	c	New workpiece is loaded into Machine 2 for making **A**
Feed-B-M2	c	New workpiece is loaded into Machine 2 for making **B**
Fault	uc	Fault: Machine 1 cannot perform OP2
M1-A-Buf	c	Machine 1 deposits a workpiece of **A** into the buffer
M1-B-Buf	c	Machine 1 deposits a workpiece of **B** into the buffer
M2-A-Buf	c	Machine 2 deposits a workpiece of **A** into the buffer
M2-B-Buf	c	Machine 2 deposits a workpiece of **B** into the buffer
M1-A-Conv	c	Machine 1 deposits product **A** into the conveyor
M1-B-Conv	c	Machine 1 deposits product **B** into the conveyor
M2-A-Conv	c	Machine 2 deposits product **A** into the conveyor
M2-B-Conv	c	Machine 2 deposits product **B** into the conveyor
Reset-A	c	Machine 1 is rest in making **A**
Reset-B	c	Machine 1 is rest in making **B**
REPAIR	c	Machine 1 is repaired
STRT	uc	System starts
STOP	uc	System stops

Table 6.1: Event Definition and Tasks of the system

B transfers. Therefore, the buffer specifications can be as given by Fig. 6.7 and Fig. 6.8 for respectively product **A** and product **B**.

In Fig. 6.7, a workpiece is deposited in BufA by either Machine 1 (*M1-A-Buf*) or Machine 2 (*M2-A-Buf*). The buffer, then, is emptied by depositing the workpiece into either Machine 1 (*Buf-A-M1*) or Machine 2 (*Buf-A-M2*). If the uncontrollable *STOP* command occurs, the buffer takes no action if it is resting at the initial state 0 and it is emptied (*EMPT-A*) otherwise. Note that a self-loop at the state 2 of the buffer is necessary because the models of Machine 1 and Machine 2 have a *STRT-STOP* loop from their state 0 to 9. Thus, as the *STOP* command is considered uncontrollable, a *STOP* loop at the state 2 of the buffer prevents an initial dead-lock in the system. Fig. 6.8 shows a similar task schedule for BufB.

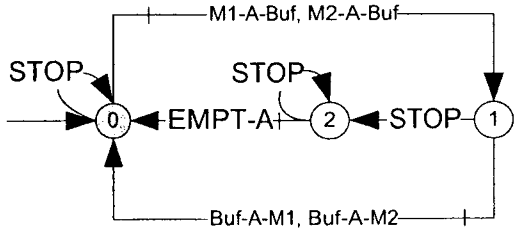


Figure 6.7: BufA handles product **A** transfers.

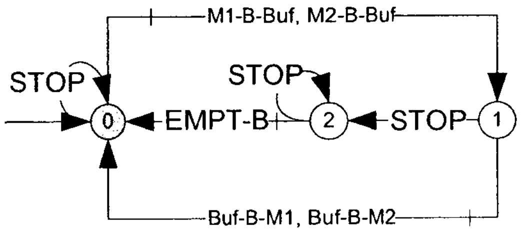


Figure 6.8: BufB handles product **B** transfers.

Recipes: Fig. 6.9 and 6.10 show the detailed recipes of products **A** (SpecA) and **B** (SpecB). In SpecA in Fig. 6.9, first, a new workpiece is allowed to be fed into Machine 1 (*Feed-A-M1*), then, operations OP1 (*AO1*), OP3 (*AO3*) and OP2 (*AO2-1*/*AO2-2*) are successively applied to the workpiece and finally the workpiece is deposited into the conveyor from Machine 1 (*M1-A-Conv*) or from Machine 2 (*M2-A-Conv*) depending on which one has performed the final operation on the workpiece. In dealing with the *STOP* command, the controller takes no action if it is at the initial state 0 and resets otherwise. In Fig. 6.9, the box, which surrounds the states 1-5, with an event *STOP* exiting

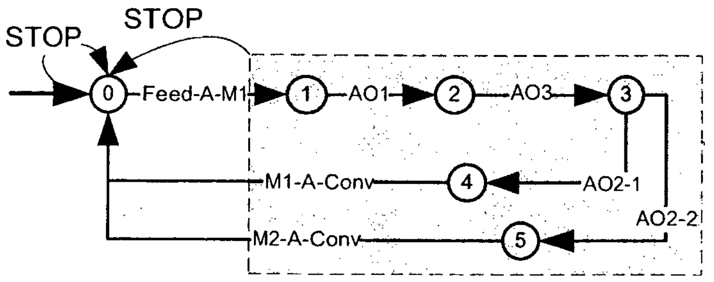


Figure 6.9: SpecA: product **A** recipe in Normal mode

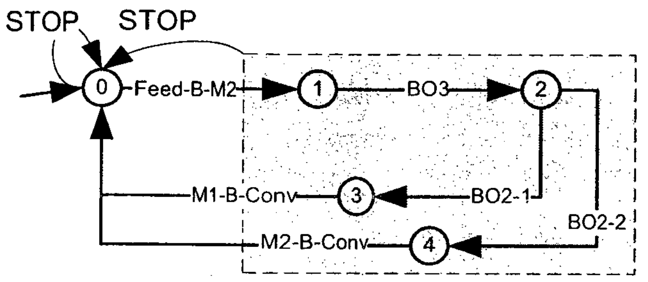


Figure 6.10: SpecB: product **B** recipe in Normal mode

from it denotes a reset action from every state inside the box to the initial state. A similar task description is given for SpecB in Fig. 6.10.

Some of the (controllable) events in the system are not used in producing the

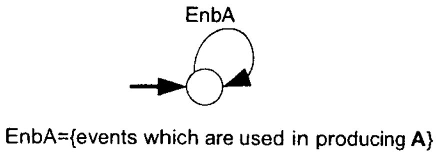


Figure 6.11: EnbA: Permitted events for product **A** in Normal mode; EnbA = Σ − {Feed-A-M2}

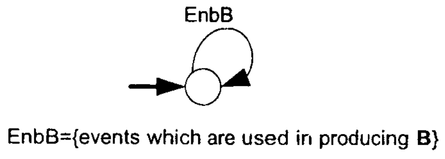


Figure 6.12: EnbB: Permitted events for product **B** in Normal mode; EnbB = Σ − {Feed-B-M1, BO1}

products **A** or **B**. We wish to have these events disabled in the system so that only those events are allowed to occur in the system that, at some point, are executed and used. Let Σ denote the set of low-level events. Then we define EnbA = Σ − {Feed-A-M2} in Fig. 6.11 and EnbB = Σ − {Feed-B-M1, BO1} in 6.12 to be the sets of enabled events in product recipes **A** and **B**.

Low-level model: We assume only states 0 and 9 in M1 and M2 are marked.

That is, only the states at which the machines rest are marked. However, all of the states of the specifications SpecA, SpecB, BufA, BufB, EnbA and EnbB are marked. Thus, the specification do not contribute to the marking of the states in the system under supervision. Now, let

$$LeA = (M1||M2||BufA||SpecA) \times EnbA$$

and

$$LeB = (M1||M2||BufB||SpecB) \times EnbB$$

where || and $\times$ denote the *sync* and the *meet* operations for automata. LeA and LeB denote the individual legal behaviors that would produce respectively products **A** and **B**. We also need to ensure that they are controllable and non-blocking. Let $(LeA)_N^{\uparrow} = SupC_{Nb}(LeA)$ and $(LeB)_N^{\uparrow} = SupC_{Nb}(LeB)$ denote the supremal controllable and nonblocking sublanguage of respectively LeA and LeB. Therefore, $(LeA)_N^{\uparrow}$ and $(LeB)_N^{\uparrow}$ represent the systems under supervision which individually would produce **A** and **B** in Normal mode. However, it should be noted that $(LeA)_N^{\uparrow}$ and $(LeB)_N^{\uparrow}$ are generated by the supervisors which in parallel act on the same resources (M1||M2). In other words, $(LeA)_N^{\uparrow}$ and $(LeB)_N^{\uparrow}$ are generated by modular supervisors. We obtain the common behavior of the systems under supervision $(LeA)_N^{\uparrow}$ and $(LeB)_N^{\uparrow}$ as

$$ProdAB = (LeA)_N^{\uparrow} \times (LeB)_N^{\uparrow}.$$

ProdAB was checked by the software TCT [1] and was verified to be nonblocking. It is also controllable with respect to the plant model M1||M2. ProdAB represents the joint behavior of the supervisors which would independently produce **A** and **B**. It has 253 states and 706 transitions. We let

$$G_{lo,normal} = ProdAB = (LeA)_N^{\uparrow} \times (LeB)_N^{\uparrow}$$

to represent the low-level model in Normal mode.

6.2.2 Faulty Mode

We consider a single-fault scenario. We assume Machine 1 may fail to perform operation OP2 at any time. The fault is considered to be permanent, that is, if the fault occurs, Machine 1 will not be able to perform OP2 unless it is repaired manually (*REPAIR*).

Machine 1 and 2: The model M1Fault in Fig. 6.13 denotes Machine 1 in Faulty mode. The dashed lines in Fig. 6.13 indicate the occurrence of the fault. As with other events, the fault is considered to be observable in this case study. If a fault occurs, operation OP2 in Machine 1 (AO2-1, BO2-1) is disabled. However, if a workpiece has previously been deposited into the machine, the machine job has not started yet and the fault occurs (i.e. the fault at the states 1 or 3 in Fig. 6.13) the machine could be emptied (*Reset-A* or *Reset-B*) and the workpiece should be transferred to Machine 2 for processing. The states 0, 7, 18 and 19 will be marked in M1Fault. However, Machine 2 is not affected by the fault and hence we model it as M2 again.

Buffers Specifications: Buffers specifications are not affected by the fault.

Recipes: If the fault occurs, the resources of the system change and thus an action (*Reset-A* or *Reset-B*) might be needed to empty Machine 1 and furthermore the workpieces should be guided to Machine 2 thereafter for an OP2 operation (if needed). These facts imply the product recipes will experience changes in their models. SpecAFault in Fig. 6.14 and SpecBFault

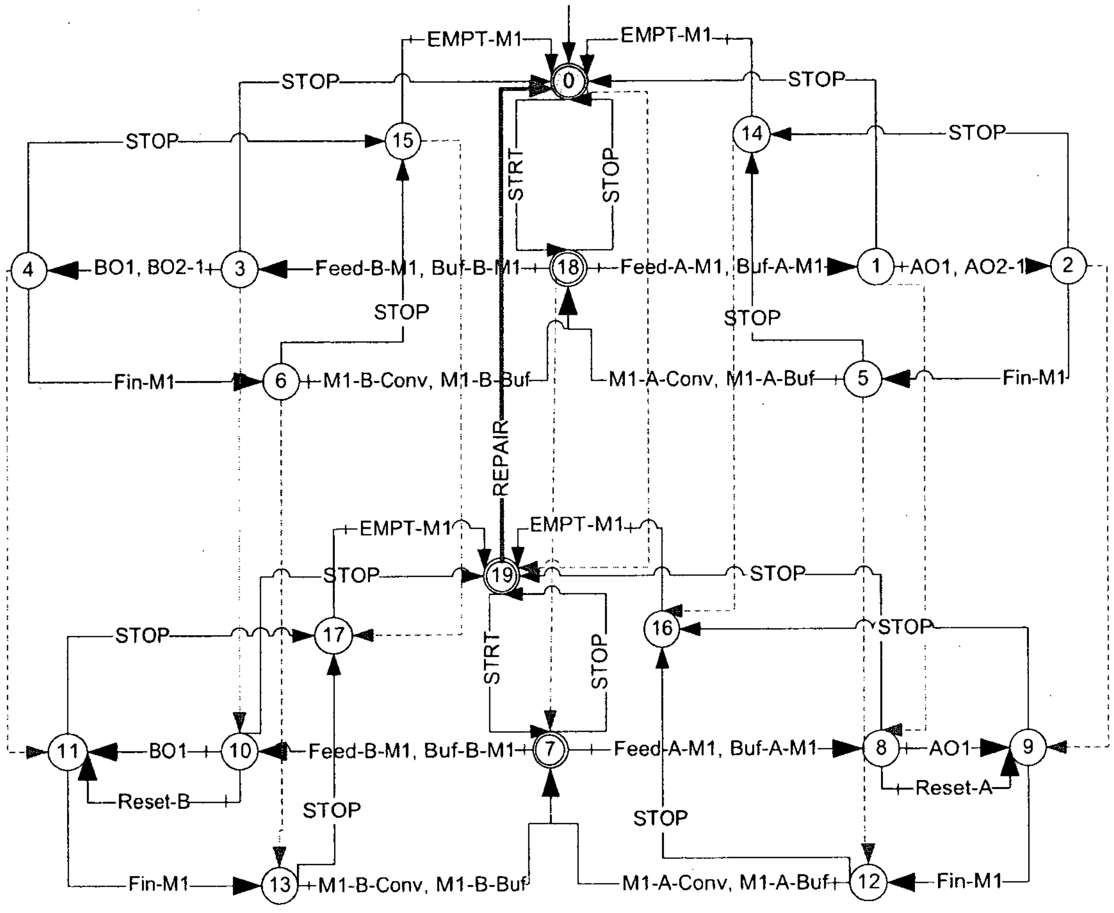


Figure 6.13: M1Fault: Machine 1 behavior in Faulty mode; dashed lines indicate the occurrence of the fault

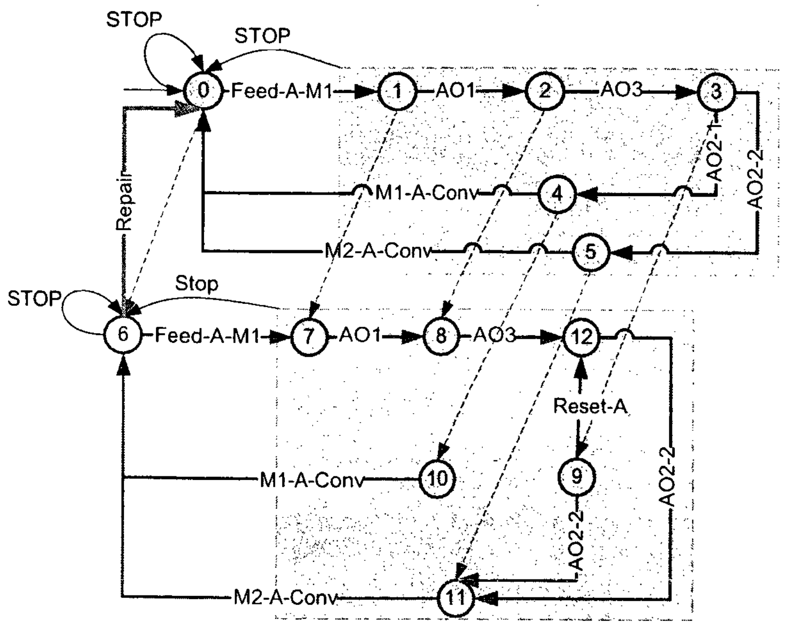


Figure 6.14: SpecAFault: product **A** recipe in Faulty mode. Dashed lines indicate the occurrence of the fault.

in 6.15 show respectively the recipes of products **A** and **B** in Faulty mode.

In SpecAFault in Fig. 6.14, if the system is to execute operation OP2 at state 3 (*AO2-1* or *AO2-2*) and the *Fault* occurs (Fault takes the system to state 9), two options will be available. At state 9 if the workpiece has previously been deposited in Machine 1, the Machine 1 is first reset (*ResetA*) and then operation OP2 from Machine 2 (*AO2-2*) is executed. However, if at state 9 previously Machine 2 has been in charge of processing the workpiece and *Fault* occurs, operation OP2 from Machine 2 (*AO2-2*) is directly chosen and the system keeps on processing the workpiece as if the fault has not occurred. Note that, ResetA will be used at most once after the occurrence of *Fault*. In each case if *Fault* happens, the specification has been designed not to choose Machine 1 for operation OP2 again. A similar description can be given for SpecBFault in Fig. 6.15. The models of the enabled-events need to be updated although they might not change. Let Σ denote the set of events in Faulty mode. Then, EnbAFault=$\Sigma - \{$Feed-A-M2$\}$ in Fig. 6.16 and

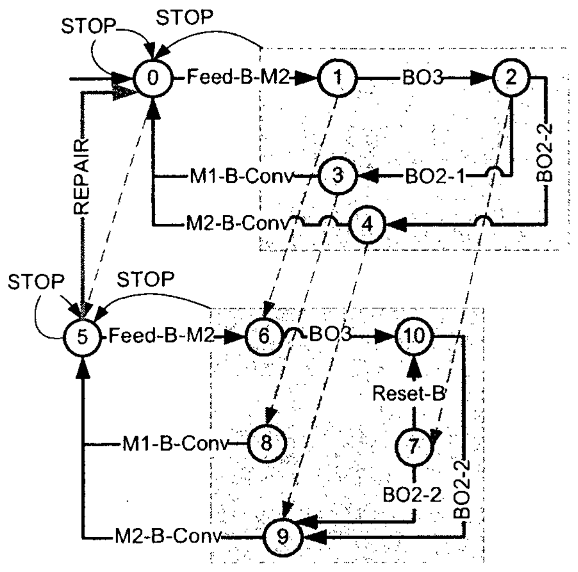


Figure 6.15: SpecBFault: product **B** recipe in Faulty mode. Dashed lines indicate the occurrence of the fault.

EnbBFault=Σ – {Feed-B-M1, BO1} in Fig. 6.17 denote the set of enabled events in products recipes **A** and **B** in Faulty mode.

Low-level model: Low-level models in Faulty mode are obtained similar to

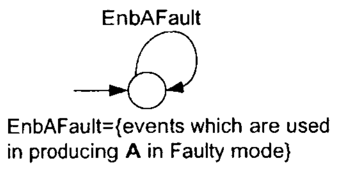

EnbAFault
EnbAFault={events which are used in producing A in Faulty mode}

Figure 6.16: EnbAFault: Permitted events for product **A** in Faulty mode; EnbAFault=Σ – {Feed-A-M2}

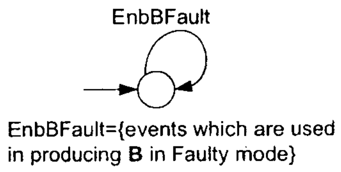

EnbBFault
EnbBFault={events which are used in producing B in Faulty mode}

Figure 6.17: EnbBFault: Permitted events for product **B** in Faulty mode; EnbBFault=Σ – {Feed-B-M1, BO1}

their counterparts in Normal mode. We have

$$LeAFault = (M1Fault||M2||BufA||SpecAFault) \times EnbAFault$$

and

$$LeBFault = (M1Fault||M2||BufB||SpecBFault) \times EnbBFault.$$

Also let $(LeAFault)_N^{\uparrow} = SupC_{Nb}(LeAFault)$ and $(LeBFault)_N^{\uparrow} = SupC_{Nb}(Le$ $BFault)$ denote the supremal controllable and nonblocking sublanguage of respectively $LeAFault$ and $LeBFault$. Therefore, $(LeAFault)_N^{\uparrow}$ and $(LeBFault$ $)_N^{\uparrow}$ are the systems under supervision which individually would produce **A** and **B** in Faulty mode. Again, we obtain the common behavior of the above controllers by

$$ProdABFault = (LeAFault)_N^{\uparrow} \times (LeBFault)_N^{\uparrow}.$$

ProdABFault was checked by TCT and it was shown to be nonblocking and controllable with respect to the plant model $M1Fault||M2$. ProdABFault represents the joint behavior of the supervisors which independently would produce **A** and **B** when Machine 1 is subject to failure. It has 1187 states and 3716 transitions. We let

$$G_{lo,Fault} = ProdABFault = (LeAFault)_N^{\uparrow} \times (LeBFault)_N^{\uparrow}.$$

Therefore, $G_{lo,Fault}$ would represent the low-level model in Faulty mode.

Remark 6.2. *Recall the recipe controllers SupA and SupB in Fig. 6.2. Since the specifications of Normal mode are a subset of those of Faulty mode, automata generating and marking* $(LeAFault)_N^{\uparrow}$ *and* $(LeBFault)_N^{\uparrow}$ *can be considered as implementations of the supervisors SupA and SupB for both Normal and Faulty modes. Therefore, SupA and SupB are the modular (recipe) supervisors that in effect mark the behaviors* $G_{lo,normal}$ *and* $G_{lo,Fault}$ *in Normal and Faulty modes.*

Finally and in agreement with the HRSC problem notations in Chapter 4, we let

$$G_{lo,1} = G_{lo,normal} \qquad \text{and} \qquad G_{lo,2} = G_{lo,fault}. \tag{6.1}$$

We will refer to $G_{lo,1}$ and $G_{lo,2}$ in the next sections as the low-level models in Normal and Faulty modes.

6.2.3 High-level Models

To obtain the high-level model we should specify the states of $G_{lo,1}$ and $G_{lo,2}$ at which the outputs should be generated (a Moore automaton should be synthesized). First, the aspects of the system which are important for the high-level (the aspects in which the high-level is interested), are derived. Here, we are interested in the order in which the products **A** and **B** are generated. Furthermore, the commands *STRT* and *STOP* are issued by the master controller and should be taken into account for their significant roles in managing the process. Finally, the occurrence of *Fault* as well as *REPAIR* are important since *Fault* changes the model of the system and *REPAIR* resets it to its initial status. We choose the related transitions in the system that refer to the above actions. Note that this transition selection yields a Mealy automaton finally. Therefore, we label the events *M1-A-Conv* and *M2-A-Conv* as "accomplishing A" or simply as $\mathcal{A}$ and the events *M1-B-Conv* and *M2-B-Conv* as $\mathcal{B}$. Furthermore, the events *STRT*, *STOP*, *Fault* and *REPAIR* are respectively labeled as $\mathcal{STRT}$, $\mathcal{STOP}$, $\mathcal{FAULT}$ and $\mathcal{REPAIR}$ at the high-level.

Remark 6.3. *In designing the hierarchical supervisory control, the controllability status of the events may differ from their status in designing the low-level supervisors SupA and SupB which would implement* $LeAFault_N^{\uparrow}$ $(LeA_N^{\uparrow})$ *and* $LeBFault_N^{\uparrow}$ $(LeB_N^{\uparrow})$. *For instance, the events* STRT *and* STOP *will be considered controllable hereafter since they are issued by the master controller at*

Event	c/uc in Primary Modeling	c/uc in HRSC Problem
AO1	c	uc
AO2-1	c	uc
AO2-2	c	uc
AO3	c	uc
BO1	c	uc
BO2-1	c	uc
BO2-2	c	uc
BO3	c	uc
Buf-A-M1	c	uc
Buf-A-M2	c	uc
Buf-B-M1	c	uc
Buf-B-M2	c	uc
EMPT-M1	c	uc
EMPT-M2	c	uc
EMPT-bufA	c	uc
EMPT-bufB	c	uc
Fin-M1	c	uc
Fin-M2	c	uc
Feed-A-M1	c	uc
Feed-B-M1	c	uc
Feed-A-M2	c	uc
Feed-B-M2	c	uc
Fault	uc	uc
M1-A-Buf	c	uc
M1-B-Buf	c	uc
M2-A-Buf	c	uc
M2-B-Buf	c	uc
M1-A-Conv	c	c
M1-B-Conv	c	c
M2-A-Conv	c	c
M2-B-Conv	c	c
Reset-A	c	uc
Reset-B	c	uc
REPAIR	c	c
STRT	uc	c
STOP	uc	c

Table 6.2: Change of controllability of events in HRSC problem

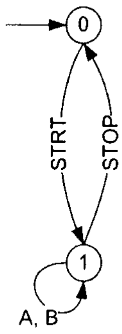


Figure 6.18: $G_{hi,normal}$: High-level model at Normal mode

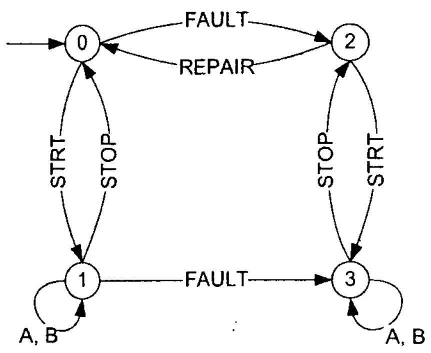


Figure 6.19: $G_{hi,fault}$: High-level model at Faulty mode

the high-level. Table 6.2 gives the changes of the controllability for all events.

Therefore, we synthesize the high-level event set T as

$$T = \{\mathcal{A}, \mathcal{B}, \mathcal{STRT}, \mathcal{STOP}, \mathcal{FAULT}, \mathcal{REPAIR}\}$$

in which all events are observable and the subset of uncontrollable events is $T_{uc} = \{\mathcal{FAULT}\}$. Then, the high-level events assignments which we gave earlier and are in Mealy automata form now should be expressed as Moore automata which are OCC. The conversion is a routine procedure in the literature and it is not given here for brevity. Let $G_{lo,normal}$ and $G_{lo,fault}$ denote the Moore automata which we obtain for Normal and Faulty modes in this step. Fig. 6.18 and 6.19 give the final high-level models $G_{hi,normal} = Higen(G_{lo,normal})$ and $G_{hi,fault} = Higen(G_{lo,fault})$ where $Higen$ is the command in TCT which generates the high-level model from a Moore automaton. To follow the HRSC notation we further let

$$G_{hi,1} = G_{hi,normal} \quad \text{and} \quad G_{hi,2} = G_{hi,fault}. \tag{6.2}$$

Note that blocking is not considered at the high-level and hence it is assumed all the states of $G_{hi,1}$ and $G_{hi,2}$ are marked. In the following sections, we also make a reference to the reporting map θ that relates the low-level and high-level languages:

$$L(G_{hi,1}) = \theta(L(G_{lo,1})) \qquad \text{and} \qquad L(G_{hi,2}) = \theta(L(G_{lo,2})).$$

6.2.4 High-level Specifications

As the problem statement demands, the high-level specifications are:

- Safety regulations are satisfied at both Normal and Faulty modes; i.e. one product $\mathcal{B}$ is generated after two consecutive $\mathcal{A}$.
- Process starts with the $\mathcal{STRT}$ command and may $STOP$ at any time.
- $\mathcal{RAPAIR}$ is possible if the process has stopped.

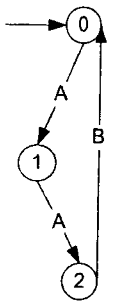


Figure 6.20: $E_{hi,normal}$: High-level specification at Normal mode

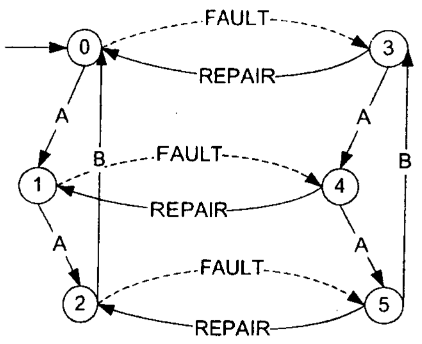


Figure 6.21: $E_{hi,fault}$: High-level specification at Faulty mode

We propose $E_{hi,normal}$ and $E_{hi,fault}$, respectively given in Fig. 6.20 and 6.21, to denote the high-level specifications. $E_{hi,normal}$ and $E_{hi,fault}$ simply satisfy the safety regulation in Normal and Faulty modes; note that the production sequence (i.e. safety regulation) is the same for both Normal and Faulty modes. For instance, in Fig. 6.21 states 1 and 4 are reachable from each other through the occurrence of $\mathcal{FAULT}$ or the execution of $\mathcal{REPAIR}$ where the set of enabled events after both states are the same. On the other hand, $E_{hi,normal}$ and $E_{hi,fault}$ do not include or constrain $\mathcal{STRT}$ and $\mathcal{STOP}$ commands. In fact, $\mathcal{STRT}$ and $\mathcal{STOP}$ commands will be appear in legal behaviors when we consider the joint behavior of $E_{hi,normal}$ and $E_{hi,fault}$ and their corresponding plant models as follows:

$$E_{hi,1} = E_{hi,normal}||G_{hi,normal} \qquad \text{and} \qquad E_{hi,2} = E_{hi,fault}||G_{hi,fault}. \tag{6.3}$$

We consider $E_{hi,1}$ and $E_{hi,2}$ as the final high-level legal behaviors at the high-level. This means, we intend to implement and recover the supremal controllable sublanguages of $E_{hi,1}$ and $E_{hi,2}$ in the system.

6.3 Hierarchical Robust Supervisory Control

Equations (6.1), (6.2) and (6.3) show respectively the low-level models, high-level models and high-level legal behaviors. In order to define an HRSC problem, we still need to show that the models above satisfy condition (4.2) and that the low-level models are Jointly-SOCC. Then, note that we have $L(G_{lo,1}) \subseteq L(G_{lo,2})$. This implies the condition (4.2) holds by construction and furthermore, if $G_{lo,2}$ is SOCC, then $G_{lo,j}$ $(j = 1, 2)$ are Jointly-SOCC. $G_{lo,2}$ was checked by TCT and it was verified to be SOCC. Therefore, $G_{lo,1}$ and $G_{lo,2}$ are Jointly-SOCC and an HRSC problem can be solved for them such that robust hierarchical consistency holds in the system.

We first solve an RSC problem [4, 45] for the pairs $(G_{hi,1}, E_{hi,1})$ and $(G_{hi,2}, E_{hi,2})$. Note that $E_{hi,1} \subseteq E_{hi,2}$ and furthermore, $E_{hi,2}$ is controllable with respect to $G_{hi,2}$. Therefore

$$SupC(E_{hi,2}) = E_{hi,2}.$$

It can be verified that $E_{hi,1}$ and $E_{hi,2}$ are a solution to the RSC problem which is solved at the high-level (see Section 2.4 for details). In other words, for a high-level robust supervisor S_{hi} which can be implemented by $E_{hi,2}$ we will have

(i) $L(S_{hi}/G_{hi,1}) = E_{hi,1}$,

(ii) $L(S_{hi}/G_{hi,2}) = E_{hi,2}$.

$E_{hi,2}$ has 12 states and 27 transitions. Next, by Theorem 4.15, the low-level supervisor S_{lo} which is built based on S_{hi} will be a robust supervisor at the low-level and since $G_{lo,j}$ $(j = 1, 2)$ are Jointly-SOCC, robust hierarchical consistency property holds in the system and we will have:

(i) $\theta(L(S_{lo}/G_{lo,1})) = E_{hi,1}$;

(ii) $\theta(L(S_{lo}/G_{lo,2})) = E_{hi,2}$.

A (low-level) non-hierarchical robust supervisory control S_{nh} following the approach in [4] can be implemented by $SupC(\theta^{-1}(E_{hi})) = E_{lo}^{\uparrow}$. Using TCT, it

Supervisor	Number of States	Number of Transitions
S_{hi}	12	27
S_{nh}	2047	6646

Table 6.3: The number of the states and transitions for the implementation of low-level and high-level supervisors

was shown that $E_{lo}^{\uparrow}$ has 2047 states and 6646 transitions. Table 6.3 shows a comparison between the high-level supervisor which we have designed here and a (low-level) implementation design which could otherwise be obtained with the approach given in [4].

The investigation of nonblocking properties has not been done in this case study; the low-level models $G_{lo,1}$ and $G_{lo,2}$ were synthesized to be nonblocking in the first step. However, whether or not nonblocking is inherited from the high-level to the low-level is an issue that has not been studied in this case study. One could manually check nonblocking in the system by calculating the supremal controllable languages $E_{lo,1}^{\uparrow}$ and $E_{lo,2}^{\uparrow}$ and see if they are nonconflicting.

The case study presented here, has a natural hierarchy which would make it

Model	Number of States	Number of Transitions
$G_{lo,1}$	253	706
$G_{lo,2}$	1187	3716
$G_{hi,1}$	2	4
$G_{hi,2}$	4	11

Table 6.4: The number of the states and transitions in low-level and high-level models

a good example for a hierarchical solution resulting in clarity in the designed supervisors. Table 6.4 provides some information on the size of the models involved. It indicates a considerable reduction in the state sizes of the abstract models. The state sizes of $G_{l,1}$ has decreased to 2 from 253 and that of $G_{l,2}$ has decreased to 4 from 1187. This suggests a considerable decrease in the computational complexity of supervisor synthesis as it is polynomial in terms of the number of the system states.

6.4 Conclusion and Extensions

A flexible manufacturing system which consists of two machines, two buffers and one conveyor was studied in this chapter. It was shown how an HRSC problem can be used to design a master controller for this system. It was shown the low-level system is Jointly-SOCC and hence the high-level specifications could fully be recovered through the implementation of the low-level robust supervisor. Significant clarity at the high-level modeling and master controller design as well as a reduction of an order of three in computational complexity of supervisor synthesis was gained, showing the advantages of using the HRSC framework.

The case study presented in this chapter can be extended further by including more faults and restrictions in it. New faults in machines or buffers or restriction on the number of the available buffers can add extra complexity to the problem. Recipes as well as the number of the products can also change. When the number of the buffers is less than the number of the products, then the buffers should have a different model. The strategy that the system chooses in dealing with the fault is another extension to this case study. Moreover, if the software capability is available to take the unobservability of events into account, then we would be able to solve the more realistic problem of HRSC under Partial Observation. That will be an interesting improvement to this case study and it would make it a good example for the applications of HRSC under Partial Observation.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

In this thesis, hierarchical robust supervisory control problem under partial observation (HRSCPO) was introduced and studied. Motivated by fault recovery problems in discrete-event systems (DES) which recently have been addressed in the literature and are computationally demanding, we introduced HRSCPO setup to obtain a more transparent model and design, and to reduce supervisor synthesis computational complexity. HRSCPO is a useful setup in cases where the model of the system changes and different layers of control or authority exist in the system. We refer to such layers of control or authority as hierarchy in this thesis. Recognizing hierarchy in a system makes the control design in the system more transparent and likely decreases the computational complexity of supervisor synthesis. We arrived at our results by extending the hierarchical control setup of [59] to the case of control under partial observation, introducing the HRSC framework, and finally extending the HRSC framework to include partial observation.

We first extended the hierarchical control setup of [59] to the case of control under partial observation. The Factorization Property was derived as the condition which ensured the existence of a map that translated low-level observations to high-level observations. To avoid unintentional disablement and to ensure hierarchical consistency, we derived three sufficient conditions for the system; the UUC and UUPO properties were introduced to ensure unintentional disablement does not occur along unobservable silent paths. Furthermore, the PO-SOCC property was derived to prevent unintentional disablement along observable silent paths. The PO-SOCC property can be regarded as the natural extension of the SOCC property [59] since it reduces to SOCC if full observation is assumed. We introduced a type of langauge observability which is weaker than the standard observability property and we showed that the supremum controllable sublanguage of the low-level specification $E_{lo}^{\uparrow}$ would be observable in this weaker sense if UUC, UUPO and PO-SOCC hold in the system. Next, we developed a map for implementing the commands of the high-level supervisor based on low-level and high-level observations, and showed that assuming UUC, UUPO and PO-SOCC, hierarchical consistency could be achieved. This ensures that the low-implementation of the high-level commands matches the expectations of the high-level supervisor. In cases when any of the Factorization Property, UUC, UUPO or PO-SOCC is not satisfied, we developed an algorithm to modify the reporting map θ to ensure the satisfaction of the above properties. In the next step, we considered a hierarchical robust supervisory control (HRSC) framework. HRSC framework is built based on HSC and RSC frameworks. We showed how hierarchical framework can be employed to design a robust supervisor for a system in which the nominal model is not precisely known but it is known to be among a finite set of models G_k. It was shown that a hierarchical control structure can be assigned to the system if either each model is individually OCC or the union of the models is OCC (Joint-OCC property). To achieve robust hierarchical consistency (i.e. to achieve hierarchical consistency regardless of which model is active at the time), it was shown that the low-level models should satisfy a Joint-SOCC property.

Next, we showed how our HRSC setup can be extended to the case of control under partial observation. We showed a combination of our previous results on partial observation and HRSC framework can ensure robust hierarchical consistency under partial observation in the system. We showed, to achieve robust hierarchical consistency, the system models should satisfy Joint-OCC, Joint-OOC, Joint-UUC, Joint-UUPO and Joint-PO-SOCC properties. We showed that Joint-OCC, Joint-OOC, Joint-UUC and Joint-UUPO properties are equivalent to each model satisfying the properties individually but Joint-PO-SOCC is not equivalent to individual PO-SOCC for all models.

We concluded the thesis with a case study. Through a supervisory control design for a flexible manufacturing system (FMS), we showed how the HRSC framework (in the full observation case) could be used to design recovery procedures. Specifically, we showed how using HRSC setup could add more clarity to design and implementation while significantly reducing complexity of the models involved.

7.2 Future Work

An important extension of our results would be to include the nonblocking properties. Including nonblocking properties in HSC problem has previously been studied in [52]. The important *observer* property which has been given in [52] is a sufficient condition for ensuring nonblocking in a hierarchical supervisory control problem. An automaton-based algorithm has been given in

[54] to modify the reporting map such that it becomes observer. However, to include nonblocking properties in HRSC problem, we need to show how a reporting map can be modified so that it is observer with respect to different models. Thus an algorithm should be given that can satisfy the Factorization Property, UUC, UUPO and PO-SOCC properties and furthermore, ensures that the reporting map is observer with respect to all the models.

Extensions to timed systems can also be mentioned as another line of research for the HRSC framework that we have presented in this thesis. Hierarchical timed systems have previously been studied, in a more general framework, in [53].

Other direction for future work includes the development of software for implementing the algorithms of hierarchical control under partial observation, and the study of modularity in hierarchical control under partial observation for further reduction of computational complexity.

Bibliography

[1] Software XPTCT. *http://www.control.utoronto.ca/DES/*, 2009.

[2] F. Baccelli, B. Gaujal, and D. Simon. Analysis of preemptive periodic real-time systems using the (max, plus) algebra with applications in robotics. *IEEE Transactions on Control Systems Technology*, 10(3):368–380, May 2002.

[3] G. Birkhoff. Lattice theory. *American Mathematical Society*, 1992.

[4] S. Bourdon, M. Lawford, and W. Wonham. Robust nonblocking supervisory control of discrete-event systems. *IEEE Transactions on Automatic Control*, 50(12):2015–2020, 2005.

[5] Y. Brave and M. Heymann. Control of discrete-event systems modeled as hierarchical state machines. *IEEE Transactions on Automatic Control*, 38(12):1803–1819, 1993.

[6] P. Caines and Y.J. Wei. The hierarchical lattices of a finite machine. *Systems and Control Letters*, 25(4):257–263, 1995.

[7] P. Caines and Y.J. Wei. Hierarchical hybrid control systems: a lattice theoretic formulation. *IEEE Transactions on Automatic Control*, 43(4):501–508, 1998.

[8] P.E. Caines, V. Gupta, and G. Shen. The hierarchical control of ST-finite state machines. *Proceedings of the 36th Conference on Decision and Control*, pages 3584–3589, San Diego, California, USA, December 1997.

[9] C.G. Cassandras and S. lafortune. Introduction to discrete event systems. *Springer*, Second Edition, 2008.

[10] S.F. Chew and M.A. Lawley. Robust supervisory control for production systems with multiple resource failures. *IEEE Transactions on Automation Science and Engineering*, 3(3):309–323, 2006.

[11] K.H. Cho and J.T. Lim. Stability and robustness of discrete-event dynamic systems. *International Journal of Systems Science*, 28(7):691–703, 1997.

[12] J.E.R Cury and B.H. Krogh. Robustness of supervisors for discrete-event systems. *IEEE Transactions on Automatic Control*, 44(2):376–379, 1999.

[13] J.E.R. Cury and B.H. Krogh. Design of robust supervisors for discrete event systems with infinite behaviors. *Proceedings of the IEEE Conference on Decision and Control*, 2:2219–2224, Kobe, Japan, 1996.

[14] I. Elmahi, O. Grunder, and A. Elmoudni. A max plus algebra approach for modelling and control a supply chain. *Proceedings of 2003 IEEE Control Conference on Applications*, 2:1425– 1430, Istanbul, Turkey, June 2003.

[15] L. Feng and W.M. Wonham. Supervisory control architecture for discrete-event systems. *IEEE Transactions on Automatic Control*, 53(6):1449–1461, 2008.

[16] B. Gaudin and H. Marchand. Supervisory control of product and hierarchical discrete event systems. *European Journal of Control*, 10(2), 2004.

[17] B. Gaudin and H. Marchand. Supervisory control of product and hierarchical discrete-event systems. *European Journal of Control*, 10(2):131–145, 2004.

[18] P. Gohari. and W. M. Wonham. On the complexity of supervisory control design in the RW framework. *IEEE Transactions on Systems, Man and Cybernetics, Special Issue on DES*, 30(5):643–652, 2000.

[19] Y. Guan. Implementation of hierarchical observer theory. *Master's thesis*, Department of Electrical and Computer Engineeirng, University of Toronto, 1997.

[20] W. Han and M.A. Jafari. Component and agent-based FMS modeling and controller synthesis. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 33(2):193–206, 2003.

[21] D. Harel. Statecharts: a visual formalism for complex systems. *Science of Computer Programming*, 8:231–274, 1987.

[22] M. Heymann. Concurrency and discrete-event control. *IEEE Control System Magazine*, 10:103–112, April 1990.

[23] R.C. Hill, D.M. Tilbury, and S. Lafortune. Modular supervisory control with equivalence-based conflict resolution. *Proceedings of American Control Conference*, pages 491–498, Seattle, June 2008.

[24] L.E. Holloway, B.H. Krogh, and A. Giua. A survey of Petri net methods for controlled discrete-event systems. *Discrete-Event Dynamic Systems: Theory and Applications*, 7(2):151–190, April 1997.

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

[26] R. Kumar and M. Heymann. Masked prioritized synchronization for interaction and control of discrete-event systems. *IEEE Transactions on Automatic Control*, 45(11):1970–1982, Nov. 2000.

[27] R.J. Leduc, M. Lawford, and W.M. Wonham. Hierarchical interface-based supervisory control-part I: serial case. *IEEE Transactions Automatic Control*, 50(9):1322–1335, Sept. 2005.

[28] R.J. Leduc, M. Lawford, and W.M. Wonham. Hierarchical interface-based supervisory control-part II: parallel case. *IEEE Transactions Automatic Control*, 50(9):1336–1348, Sept. 2005.

[29] R.J. Leduc, D. Pengcheng, and S. Raoguang. Synthesis method for hierarchical interface-based supervisory control. *IEEE Transactions on Automatic Control*, 54(7):1548–1560, 2009.

[30] F. Lin. Robust and adaptive supervisory control of discrete event systems. *IEEE Transactions on Automatic Control*, 38(12):1848–1852, 1993.

[31] C. Ma and W.M. Wonham. Nonblocking supervisory control of state tree structures. *IEEE Transactions on Automatic Control*, 51(5):782–793, 2006.

[32] C. Ma and W.M. Wonham. Control of state tree structures. *Proceedings of the 11th Mediterranean Conference on Control and Automation*, pages T4-005,p. 6, Rhodes, Greece, June 2003.

[33] H. Marchand and B. Gaudin. Supervisory control problems of hierarchical finite state machines. *Proceedings of 41th IEEE Conference on Decision and Control*, pages 1199–1204, Las Vegas, Nevada, USA, December 2002.

[34] J.O. Moody and P.J. Antsaklis. Supervisory control of discrete-event systems using Petri nets. *Kluwer Academic publishers*, Boston, 1998.

[35] C. Ozveren and A. Willsky. Output stabilizability of discrete-event dynamic systems. *IEEE Transactions on Automatic Control*, 36(8):925–935, 1991.

[36] C. Ozveren and A. Willsky. Stability and stabilizability of discrete event dynamic systems. *Journal of the Association for Computing Machinery*, 38(3):730–752, 1991.

[37] S.J. Park. Robust supervisory control of uncertain timed discrete event systems based on activity models and eligible time bounds. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E88-A(3), March 2005.

[38] S.J. Park and J.T. Lim. Hierarchical supervisory control of discrete event systems with model uncertainty. *International Journal of Systems Science*, 32(6):739–744, 2001.

[39] S.J. Park and J.T. Lim. Robust and nonblocking supervisory control of nondeterministic discrete event systems using trajectory models. *IEEE Transactions on Automatic Control*, 47(4):655–658, 2002.

[40] S. Perk, T. Moor, and K. Schmidt. Controller synthesis for an i/o-based hierarchical system architecture. *Proceedings of the 9th International Workshop on Discrete Event Systems*, pages 474–479, Goteborg, Sweden, May 2008.

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

[42] A. Ramirez-Serrano, S.C. Zhu, and B. Benhabib. Moore automata for the superviosry control of robotic manufacturing workcells. *Autonomous Robots*, 9(1):59–69, 2000.

[43] K. Rudie and W. M. Wonham. Think globally, act locally: decentralized supervisory control. *IEEE Transactions on Automatic Control*, 37(11):1692–1708, 1992.

[44] A. Saboori and S. Hashtrudi Zad. Fault recovery in discrete event systems. *Proc. Computational Intelligence: Methods and Applications, CIMA'05 (ICSC/IEEE)*, Istanbul, Turkey, Dec 2005.

[45] A. Saboori and S. Hashtrudi Zad. Robust nonblocking supervisory control of discrete-event systems under partial observation. *Systems and Control Letters*, 55(10):839–848, Oct 2006.

[46] K. Schmidt, T. Moor, and S. Perk. Nonblocking hierarchical control of decentralized discrete event systems. *IEEE Transactions on Automatic Control*, 53(10):2252–2265, 2008.

[47] J.T. Lim S.J. Park. Robust and nonblocking supervisor for discrete-event systems with model uncertainty under partial observation. *IEEE Transactions on Automatic Control*, 45(12):2393–2396, December 2000.

[48] S. Takai. Maximally permissive robust supervisors for a class of specification languages. *Proceedings of the IFAC Conference on System Structure and Control*, 2:429–434, 1998.

[49] J.P. Thomas, N. Nissanke, and K.D. Baker. A hierarchical petri net framework for the representation and analysis of assembly. *IEEE Transactions on Robotics and Automation*, 12(2):268–279, 1996.

[50] B. Wang. Top-down design for RW supervisory control. *Master's thesis*, Department of Electrical and Computer Engineeirng, University of Toronto, 1995.

[51] K.C. Wong. On the complexity of projections of discrete-event systems. *In IEE Workshop on Discrete Event Systems*, pages 201–208, 1998.

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

[53] K.C. Wong and W. M.Wonham. Hierarchical control of timed discrete-event systems. *Discrete Event Dynamic Systems:Theory and Applications*, 6(3):275–306, 1996.

[54] K.C. Wong and W. M.Wonham. On the computation of observers in discrete-event systems. *Discrete Event Dynamic Systems:Theory and Applications*, 14(1):55–107, 2004.

[55] K.C. Wong, W. M.Wonham, and W. Murray. Modular control and coordination of discrete-event systems. *Discrete Event Dynamic Systems*, 8(3):247–297, 1998.

[56] W.M. Wonham. Lecture notes: Supervisory control of discrete-event systems. *Department of Electrical and Computer Engineering*, University of Toronto, 2008.

[57] W.M. Wonham and P.J. Ramadge. On the supremal controllable sublanguage of a given language. *SIAM J. Control Optim*, 25(3):637–659, 1987.

[58] W.M. Wonham and P.J. Ramadge. Modular supervisory control of discrete-event systems. *Journal of Mathematics of Control, Signals, and Systems*, 1(1):13–30, 1988.

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