Supervisory control of switching control systems

Mani Mesgarpour Tousi

A Thesis

in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of Master of Applied Science at

Concordia University

Montréal, Québec, Canada

March 2006

Your file Votre référence
ISBN: 0-494-14271-5
Our file Notre référence
ISBN: 0-494-14271-5

ABSTRACT

Supervisory control of switching control systems

Mani Mesgarpour Tousi

In this thesis, we show that the problem of designing a switching policy for an adaptive switching control system can be formulated as a problem of Supervisory Control of a Discrete-Event System (DES). Two important problems in switching control are then addressed using the DES formulation and the theory of supervisory control under partial observation. First we examine whether for a given set of controllers, a switching policy satisfying a given set of constraints (on the transitions among controllers) exists. If so, then we design a minimally restrictive switching policy. Next, we introduce an iterative algorithm for finding a minimal set of controllers for which a switching policy satisfying the switching constraints exists. In our study we show that in the supervisory control problem considered in this thesis, limitation on event observation is the factor that essentially restricts supervisory control. In other words, once observation limitations are respected, limitation on control will be automatically satisfied. We use the above result to simplify our iterative algorithm for finding minimal controller set.

"I have looked at life for the first time through the eyes of my soul, and what I discovered about life astounded me. You see, for many years I was in search of myself. I was capable of avoiding myself, thus avoiding the truth about myself. Then the day came when I no longer wanted to avoid the truth. I wanted to confront it and when I did, what I found was what I was in search of all those restless years... my soul." -Peter Caine

I dedicate this work to my great family, dad (Esi), mom (Shahla), brother (Nima), and his wife (Ladan), for their keen and warm comfort throughout my life.

ACKNOWLEDGEMENTS

I would like to thank my supervisor Dr. Hashtrudi Zad for his strong support on this work.

I would also appreciate the passionate encourage of my best friend, Yashar Nikouseresht, my aunt, Zari Niknejadpour, and my classmate Idin Karuei throughout these years.

I truly appreciate those people who have enriched me as a professional and as a human being. I sincerely thank them because I would not be who I am without them.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

Introduction

1.1 Switching Control

Recently switching control schemes have been developed to accomplish a wide variety of tasks for which the traditional adaptive control is not applicable, [15], [19], [16], [18], [20], [21], [2], [4], [6], [5], [25], [1]. In fact, when variation of system parameters is large, the traditional adaptive control is not effective, however, the switching control can be used to achieve the adaptive control objectives. Switching control is applied to a variety of applications such as fault recovery with finite number of failures, [12].

A switching control scheme for a family of plant models is studied by Miller and Davison in [16], in which it is assumed that the plant model always belongs to a finite set of known models called the family of plant models. A sudden change from one plant model to another in the set represents variation of system parameters. There, it is also assumed a high-performance controller is designed for each member of family of plant models. Controller design is such that each controller stabilizes only its corresponding plant model and destabilizes all the others. Suppose initially, the current controller stabilizes the plant. If the plant model changes, after a bounded time, instability is detected and as a result, the system switches to another controller. If the resulting plant/controller pair is unstable, the system will switch to another controller. This is repeated until a stabilizing controller is found.

Throughout this thesis, switching to a controller that stabilizes the system is referred to as a "**stable switching**" and switching to a controller that destabilizes the system is called an "**unstable switching**".

Switching through a number of destabilizing controllers before locking onto the correct controller makes the transient response poor which is one of the main problems in switching control methods. Having p number of plant models, it can be easily verified that in the worst case, $p-2$ unstable switchings may occur before the correct controller is found. The switching control structure introduced in [16] is referred to as a single-layer structure, since it

is composed of only one set of controllers. A multi-layer structure proposed in [11] can potentially reduce the number of unstable switchings. In this setup, the first layer controllers are the high-performance controllers similar to those in the single-layer structure. The n^{th} layer $(1 \leq n \leq p)$ consists of a set of simultaneous stabilizing controllers, such as $\mathrm{C}_{i_1 i_2 \ldots i_n}$ (with $i_1, i_2, \ldots, i_n$ being n distinct integers) that stabilizes plants $\mathrm{P}_{i_1}, \mathrm{P}_{i_2}, \ldots, \mathrm{P}_{i_n}$ and destabilizes all other ones. Although there are at most p layers of controllers for a family of p plant models, it can be shown that only $p-2$ layer of controllers are required in a multi-layer structure to minimize the number of unstable switchings [11].

In [23], [24] and then in [25] and [10], a high level controller called "supervisor" for a family of controllers is introduced to solve the reference tracking problem of a plant with unmodeled dynamics. In this work the plant models are assumed to be Single Input Single Output (SISO) and linear. The supervisor's role is choosing one controller at each time instant to stabilize the system. The controllers are compared based on a norm-squared output estimation error used as performance signal [11]. This switching control method can deal with the cases that the uncertainties are not sufficiently small for linear feedback systems to stabilize. Potential examples include parametric uncertainties, unmodeled dynamics, and exogenous disturbances.

Switching control can be used for both discrete-time and continuous-time systems. The application of computers and sampled-data systems has grown in practical control systems.

This is due to the fact that computers and digital circuits can be easily programmed and used in different environments. Problems arise in the presence of unmodeled or changing dynamics. When classical adaptive control methods fail to stabilize such discrete-time systems, discrete-time switching control can potentially stabilize the system. In [3] a discrete-time supervisory control is given for tracking the reference input of a family of plants. The switching method of [25] is modified for discrete-time models in this paper. This method guarantees globally bounded states and zero-offset tracking.

Switching control is an alternative to classical adaptive control methods with the following pros and cons. First, it is effective for stabilizing non-minimum phase systems. Secondly, switching control design is generally simpler than the conventional adaptive control counterparts because the design stage has only two steps: designing a set of candidate controllers solving the robust servomechanism problem [5], [7] for all possible plant models, and then applying a proper switching mechanism. Furthermore, switching control can be applied to the systems with unknown high-frequency gain sign and also unknown relative degree [26], [27]. More importantly, switching methods are very useful for highly uncertain systems that are known to be difficult to control by applying traditional adaptive techniques. The overall performance of switching controllers depends on each individual controller and the switching mechanism. Moreover, in the presence of any error in polarity of the input or output signals as well as any type of fault, switching control methods are proven efficient [12].

However, bad transient response is the main disadvantage of switching control in general. This is due to the fact that the system may switch through several non-matching controllers that fail to stabilize the system and cause undesirable overshoot in the output signal before locking onto the correct, matching controller. Several methods have been proposed to tackle this shortcoming of switching control [28], [17], [11].

The proposed algorithm in [11] for switching between the controllers of different layers guarantees that no more than one unstable switching occurs before the correct controller in the first layer is found. This method utilizes different layers of controllers with different properties. That is an extension of the switching method introduced in [16], which assumes that the set of plant models $\{P_i : i = 1, 2, \ldots, p\}$ is given and upper-bounds on the disturbance and reference input magnitudes are available. It is also assumed that each plant is controllable and observable. The plant may change slowly and the changes of the plant are unknown. When a change in the dynamics of plant or any other parameter like polarity occurs, the switching system intelligently chooses an ordered subset of the set of all controllers, switches through them one by one and waits for a finite time on each controller until stability or instability is sensed. It is to be noted that the proposed multi-layer architecture can be applied to any switching mechanism that does not switch more than once to each controller. In other words, one can use switching control methods other than the one given in [16] as long as it has the above mentioned property.

The issues and challenges in switching control could be summarized as follows.

i Designing a family of different controllers to stabilize a subset of the plant models.

ii Choosing the appropriate switching time according to the plant models and the controllers.

iii Selecting the switching method and the order of controllers by a top level supervisor.

iv Reducing the large transient response that occurs mainly due to switching through destabilizing controllers.

1.2 Thesis Motivations and Contributions

In this thesis, we examine the multi-layered switching control of [11]. In the switching control problem, it would be desirable to impose restrictions on transitions. For instance, to maintain the quality of the transient response, one may want to limit the total number of unstable transitions before the system locks onto the correct controller in the first layer. We refer to a switching policy that can find the correct controller without violating the desired restrictions on the transitions a **proper switching policy**.

The number of controllers in a multi-layer structure grows exponentially with the number of plants p. In practice, designing all of these controllers can be cumbersome and sometimes impossible. Furthermore, only a subset of all possible controllers may be sufficient to achieve the objectives, i.e. to meet the design specifications for transitions.

Motivated by the above observations, in this thesis we address the following two problems:

1 Designing a top level supervisor (switching policy) to ensure that the design specifications (restrictions on transitions) can be met.

2 Finding a minimal set of controllers for which a switching policy satisfying the design specifications exists.

The solution on the first problem solves issue (iii) and also helps with solving issue (iv) discussed in the previous section. The solution to the second problem simplifies the task of controller design (issue (i) in the previous section).

In this thesis, we show that the problem of designing a proper switching policy can be formulated and solved as a problem of Supervisory Control of Discrete Event Systems (DES) under partial observation in the Ramadge-Wonham framework [29].

The DES control is a research area applied to a wide range of application domains such as information systems, manufacturing systems, traffic management, communication protocols, and logistic (service) systems. In the Supervisory Control framework proposed by Ramadge and Wonham [29], the DES is modeled as a generator of a formal language, which can be controlled by an external supervisor by the enablement or disablement of certain events (transitions). The supervisor enables or disables events to restrict the system behavior in order to satisfy a variety of criteria. Safety specifications like the avoidance of prohibited regions of state space, or the observation of service priorities are among the criteria.

After translating the problem of designing switching policy to a problem of supervisory control of a DES, the problems of designing switching policy and finding a minimal set of controllers are solved. One of the main advantages of solving the problem in the framework of supervisory control of DES is that this framework provides a systematic method to represent various design specifications and to solve the corresponding problems. Note that for instance, the switching policy provided in [11] is only applicable to the cases where we have the single specification of having at most one unstable switching. The approach in this thesis is suitable for dealing for various types of specifications.

In this thesis, we propose an iterative procedure to find a minimal set of controllers for which a proper switching policy exists. In our study of supervisory control of switching control systems we show that limitations on event observation are the factors that essentially

restrict supervisory control. In other words, once observation limitations are respected, limitations on control will be automatically satisfied. We use the above result to simplify our iterative algorithm for finding a minimal controller set.

1.3 Thesis Outline

This thesis is organized as follows. In Chapter 2 the fundamentals of supervisory control of DES are presented. In this chapter, we cover the definitions and results used throughout the thesis. This chapter also briefly discusses the software package TTCT used in designing DES supervisors.

In Chapter 3, switching control is reviewed. This chapter contains a brief review of single-layer and multi-layer switching control which helps us in subsequent chapters.

In Chapter 4, we show how the problem of designing switching policies can be converted to a problem of supervisory control of a DES.

After the conversion procedure, in Chapter 5, we provide solutions to the problems of designing switching policies and finding a minimal controller set.

In Chapter 6, we summarize the contributions of the thesis and discuss future research.

Chapter 2

Overview of Supervisory Control

Ramadge and Wonham (RW)[29]proposed a framework to model and construct supervisory controllers for Discrete Event Systems (DES). The objective in this framework is to design a supervisor (controller) for a given plant so that the plant under supervision satisfies given design specifications. This solution treats the controller and the open-loop plant separately, while other previous approaches only deal with the closed-loop system models.

Computational complexity for large-scale systems is the main challenge in this framework. However, by using techniques such as decomposition and distributed modeling and control, this problem can be mitigated. In this chapter, a brief review of the RW supervisory

theory is provided.

2.1 Languages

Let Σ represent an alphabet, and $L \subseteq \Sigma^*$ a language. If $s = tu$, where $s, t, u \in \Sigma^*$, t, u are called a *prefix* and a *suffix* of s, respectively. Denote the *prefix-closure* of L with $\overline{L}$. In other words,

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

L is called *prefix-closed* (or simply *closed*) if $L = \overline{L}$. For two languages L, $M \subseteq \Sigma^*$, L is called *M-closed* if $L = \overline{L} \cap M$ [29].

Two languages, L_1 and L_2 are *nonconflicting* ([29]) if

$$\overline{L_1 \cap L_2} = \overline{L_1} \cap \overline{L_2}$$

Example 2.1 *Let* $\Sigma = \{p_2, c_1, c_2, c_{12}, \sigma_s\}$ *be an alphabet. Then the set of finite sequences*

$L_{E1a}, L_{E1b}, L_{E1c}$ *and* $L_{E1c} \subseteq \Sigma^*$

$$L_{E1a} = \{p_2 c_{12} \sigma_s c_2 \sigma_s\}$$

$$L_{E1b} = \{\varepsilon, p_2, p_2 c_{12}, p_2 c_{12} \sigma_s, p_2 c_{12} \sigma_s c_2, p_2 c_{12} \sigma_s c_2 \sigma_s\}$$

$$L_{E1c} = \{p_2 c_{12} \sigma_s c_2 \sigma_s, \sigma_s\}$$

are languages over alphabet Σ. p_2 *and* $c_2\sigma_s$, *for instance, are a prefix and a suffix of* $p_2 c_{12} \sigma_s c_2 \sigma_s$, *respectively. Moreover,* L_{E1b} *is the prefix-closure of* L_{E1a}, *i.e.* $L_{E1b} = \overline{L_{E1a}}$. *Because* $\overline{L_{E1a}} \neq L_{E1a}$, L_{E1a} *is not a closed language. However* L_{E1a} *is* L_{E1c}*−closed, since* $L_{E1a} = \overline{L_{E1a}} \cap L_{E1c}$. *In addition,* L_{E1b} *is closed, since its prefix-closure is the same as itself.* L_{E1a} *and* L_{E1b} *are nonconflicting, since*

$$\overline{L_{E1a}} \cap \overline{L_{E1b}} = \{\varepsilon, p_2, p_2 c_{12}, p_2 c_{12} \sigma_s, p_2 c_{12} \sigma_s c_2, p_2 c_{12} \sigma_s c_2 \sigma_s\} = \overline{L_{E1a} \cap L_{E1b}}$$

2.2 Automata

In Ramadge-Wonham (RW) framework [29], it is assumed that the DES plant can be represented by a (deterministic) Finite State Automaton (FSA)

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

where Q and Σ are the finite state set and the event set, respectively. δ represents the transition partial function $\delta : Q \times \Sigma \to Q$. q_0 and $Q_m \subseteq Q$ are the initial state and the marked state set,

respectively.

The *closed behavior*, denoted by $L(G)$, represents all possible event sequences taking G from the initial state to some reachable state:

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

Here $\delta(q_0, s)!$ means that $\delta(q_0, s)$ is defined. The *marked behavior* defined as

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

represents the set of all possible event sequences taking G from the initial state to some marked state.

2.2.1 Nonblocking Property

One of the main issues in the study of DES is the blocking property. A system is called *blocking* if either it can reach an unmarked state from which there is no transition going out (which is called a deadlock) or the system can reach a set of strongly connected unmarked states and there is no transition going out of this set (which is called a livelock). Therefore a (deterministic) DES is called *nonblocking* if $\overline{L}_m(G) = L(G)$. In other words, G is nonblocking if from every state reachable from q_0 in G there is a path to a marked state.

Example 2.2 *Let the following DES* $G_{E2} = (Q, \Sigma, \delta, q_0, Q_m)$ *be shown as in Figure 2.1. Here* $Q_m = \{6\}$, $q_0 = 0$, $\Sigma = \{p_2, c_1, c_2, c_{12}, \sigma_s, \sigma_u\}$. *The marked state 6 is identified by an outgoing arrow.*

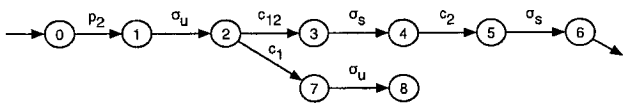

Figure 2.1: G_{E2} DES model on Example 2.2

We have $L_m(G_{E2}) = \{p_2\sigma_u c_{12} c_2 \sigma_s\}$. G_{E2} *is blocking since state 8 is a deadlock.*

2.2.2 Operations on Automata

In this section, we review two operations on automata that will be used in this thesis.

Synchronous and Parallel Products We need to build models by using individual small models of the system components. To be able to do so two DES operations, *parallel product* and *synchronous product* are defined.

The parallel product of two DES models, G_1 and G_2 denoted by $meet(G_1, G_2)$, contains

the synchronized occurrence of the common events in the two models. Consider

$$G_1 = (Q_1, \Sigma_1, \delta_1, q_{0_1}, Q_{m_1})$$

$$G_2 = (Q_2, \Sigma_2, \delta_2, q_{0_2}, Q_{m_2})$$

$meet(G_1, G_2)$ is the reachable subgenerator of $(Q_1 \times Q_2, \Sigma_1 \cap \Sigma_2, \delta, (q_{0_1}, q_{0_2}), Q_{m_1} \times Q_{m_2})$. The transition partial function of $\delta : (Q_1 \times Q_2) \times \Sigma \rightarrow Q_1 \times Q_2$ is

$$\delta((x_1, x_2), \sigma) = \begin{cases} (\delta_1(x_1, \sigma), \delta_2(x_2, \sigma)) & \text{if } \delta_1(x_1, \sigma)! \text{ and } \delta_2(x_2, \sigma)! \\ \text{not defined} & \text{otherwise.} \end{cases}$$

Thus, it can be easily verified that

$$L(meet(G_1, G_2)) = L(G_1) \cap L(G_2)$$

$$L_m(meet(G_1, G_2)) = L_m(G_1) \cap L_m(G_2)$$

The synchronous product of two DES models G_1 and G_2, denoted by $sync(G_1, G_2)$ or $G_1 || G_2$ is the reachable subgenerator of $(Q_1 \times Q_2, \Sigma_1 \cup \Sigma_2, \delta, (q_{0_1}, q_{0_2}), Q_{m_1} \times Q_{m_2})$ where $\delta : (Q_1 \times Q_2) \times \Sigma \rightarrow Q_1 \times Q_2$ is defined according to:

$$\delta((x_1, x_2), \sigma) = \begin{cases} (\delta_1(x_1, \sigma), \delta_2(x_2, \sigma)) & \text{if } \delta_1(x_1, \sigma)! \text{ and } \delta_2(x_2, \sigma)! \\ (\delta_1(x_1, \sigma), x_2) & \text{if } \delta_1(x_1, \sigma)! \text{ and } \sigma \in \Sigma_1 - \Sigma_2 \\ (x_1, \delta_2(x_2, \sigma)) & \text{if } \delta_2(x_2, \sigma)! \text{ and } \sigma \in \Sigma_2 - \Sigma_1 \\ \text{not defined} & \text{otherwise.} \end{cases}$$

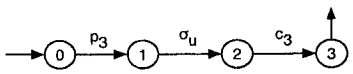

Figure 2.2: G_{E3a} in Example 2.3

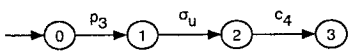

Figure 2.3: G_{E3b} in Example 2.3

Example 2.3 *Let G_{E3a} and G_{E3b} be as shown in Figures 2.2,2.3, respectively.*

The product of G_{E3a} and G_{E3b}, $meet(G_{E3a}, G_{E3b})$, is as shown in Figure 2.4 since only the events that are enabled in both machines can occur in $meet(G_{E3a}, G_{E3b})$.

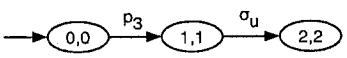

Figure 2.4: *meet* product of G_{E3a} and G_{E3b} in Example 2.3

The synchronous product of G_{E3a} and G_{E3b} is as displayed in Figure 2.5. Since the G_{E3a} does not contain c_4 in its alphabet, occurrence of c_4 is determined by the transition function of G_{E3b} only. Similarly, the occurrence of c_3 is determined by G_{E3a}.

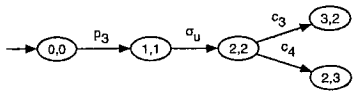

Figure 2.5: *sync* of G_{E3a} and G_{E3b} in Example 2.3

2.3 Supervisory Control

Consider a DES $G = (Q, \Sigma, \delta, q_0, Q_m)$. We assume that the event set consists of *controllable* and *uncontrollable* events, denoted Σ_c and Σ_{uc}, respectively, with $\Sigma_c \cap \Sigma_{uc} = \emptyset$. The controllable events are those that can be enabled or disabled at any time by some means of control. The events that are not controllable, are uncontrollable. Furthermore, some events are assumed *observable* Σ_o and the rest unobservable $\Sigma_{uo} = \Sigma - \Sigma_o$.

Suppose the desired behavior (specification) of the DES can be represented in the form of a finite state automaton *Spec*. Then $E = L_m(G) \cap L_m(Spec)$ represents the legal behavior of the system and is called the *legal (marked) behavior*. Obviously, $E \subseteq L_m(G)$.

Example 2.4 *As an example of plant and specification in supervisory control problem, let the DES plant (G_{E4}) and specification ($Spec_{E4}$) be defined over $\Sigma = \{p_2, c_2, c_1, \sigma_s, \sigma_u\}$ and be as shown in Figures 2.6 and 2.7, respectively. The observable event set is $\Sigma_o = \{c_2, c_1, \sigma_s, \sigma_u\}$ and the controllable event set is $\Sigma_c = \{c_2, c_1\}$.*

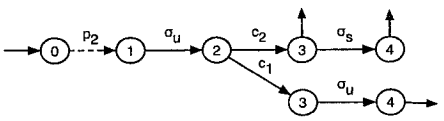

Figure 2.6: G_{E4} the DES plant model on Example 2.4

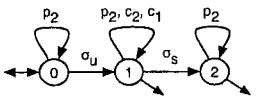

Figure 2.7: $Spec_{E4}$ the DES specification on Example 2.4

To satisfy the specification some means of control is essential. Supervisor is an agent that by enabling and disabling the controllable events prevents G from generating undesirable event sequences. For our purpose in this thesis a supervisor denoted by S can be modeled as a finite state automaton

$$S = (X, \Sigma, \eta, x_0, X_m)$$

where X, x_0, $X_m \subseteq X$, and $\eta : X \times \Sigma \rightarrow X$ are the state set, initial state, marked state set, and transition function, respectively. A supervisor should never disable uncontrollable events. A supervisor that respects this control restriction is called "admissible".

Plant Under Supervision In the RW theory [29], the supervisor and the plant are coupled. They form a closed loop system with the following mechanism of interaction between S and G.

Assume that at an instance, the plant is in state q_i and supervisor in state x_j. In the uncontrolled plant, a subset of events Σ can occur in q_i that is fed into the supervisor. The supervisor issues a control pattern in the form of the set of events that are permitted to occur in x_j.

Let us denote the system under supervision by S/G. The closed behavior of S/G defined by $L(S/G) = L(G) \cap L(S)$ consists of the event sequences of uncontrolled process that survive under supervision. The marked behavior of S/G defined by $L_m(S/G) = L_m(G) \cap L(S/G)$ consists of the event sequences marked by the uncontrolled process that are generated by S/G. Note that we assume all states of the supervisor are marked $X = X_m$ and as a result, S/G can be represented as $meet(S/G)$ (since $L_m(S/G) = L_m(G) \cap L(S/G) = L_m(G) \cap L(G) \cap L(S) = L_m(G) \cap L_m(S)$).

In this framework, the supervisor monitors the observable events generated by the plant and disables or enables the controllable events in G to assure that the system under supervision denoted by (S/G)

- satisfies the specification, i.e. $L_m(S/G) \subseteq E$, and
- is nonblocking, i.e. $\overline{L}_m(S/G) = L(S/G)$

If for a supervisor S, the closed-loop system S/G is nonblocking, then S is called a **nonblocking supervisor**.

The solutions of this problem can be described in terms of $L_m(G)$-closed, controllable, observable sublanguages of E [8, 14].

Definition 2.1 *([29]) A language $K \subseteq \Sigma^*$ is controllable with respect to G (or simply controllable) if $\overline{K}\Sigma_{uc} \cap L(G) \subseteq \overline{K}$.*

A supervisor S is admissible if and only if $L(S/G)$ is controllable.

Let $P : \Sigma^* \rightarrow \Sigma_o^*$ be the natural projection defined as follows:

$$P(\varepsilon) = \varepsilon$$

$$P(\sigma) = \begin{cases} \sigma & \text{if } \sigma \in \Sigma_o \\ \varepsilon & \text{otherwise} \end{cases}$$

$$P(s\sigma) = P(s)P(\sigma), \quad \textit{for } s \in \Sigma^*,\ \sigma \in \Sigma$$

Since by assumption, the supervisor can only monitor the observable events, for any s, s' $\in \Sigma^*$ with the same projection ($P(s) = P(s')$), the supervisor decision (i.e. the set of enabled events) should be the same. A supervisor satisfying this property is called **feasible**.

Definition 2.2 *([14]) A language $K \subseteq L(G)$ with a projection P is called $P-$observable (or simply* observable*) with respect to $L(G)$ if, for all $s, s' \in \Sigma^*$*

$$P(s) = P(s') \Rightarrow consis(s, s') = true$$

where $consis(s, s')$ is true if and only if

1 $(\forall \sigma \in \Sigma) s\sigma \in \overline{K} \wedge s' \in \overline{K} \wedge s'\sigma \in L(G) \Rightarrow s'\sigma \in \overline{K}$, and

2 $(\forall \sigma \in \Sigma) s'\sigma \in \overline{K} \wedge s \in \overline{K} \wedge s\sigma \in L(G) \Rightarrow s\sigma \in \overline{K}$

In other words, two strings satisfy $consis(s, s')$ if and only if, with respect to one-step continuations in K, they are consistent. Therefore, if two strings look the same to a supervisor, they should be consistent.

Theorem 2.1 *([14, 8]) Suppose $K \neq \emptyset$ and $K \subseteq E \subseteq L_m(G)$. There exists a nonblocking feasible supervisory control S for G such that $L_m(S/G) = K$ if and only if*

- *K is controllable with respect to G,*
- *K is observable with respect to $L(G)$,*
- *K is $L_m(G)$-closed.*

Since the union of observable languages is not necessarily observable, the class of $L_m(G)$-closed, controllable, observable sublanguages of a given language does not necessary have a supremal element. Therefore, an optimal solution for the supervisory control problem may not exist in general. An alternative solution to the above problem is to use the normal sublanguages instead of observable ones, because the normality property is closed under union, and therefore, the class of normal sublanguages of a given language has a supremal element.

Definition 2.3 *([13, 8]) A language $K \subseteq M$ is $(M,P)-$normal if*

$$K = M \cap P^{-1}(PK)$$

For $E \subseteq M$, $\mathscr{N}(E;M)$ denotes the $(M,P)-$normal sublanguages of E.

Let $E \subseteq L_m(G)$ denote the legal behavior. Consider the following classes of languages:

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

$$\overline{\mathscr{N}}(E;L(G)) := \{N \subseteq E | \overline{N} \text{ is } (L(G),P)-normal\}$$

$$R_G(E) := \{K \subseteq E | K \text{ is } L_m(G)-\text{closed.}\}$$

That the above classes are nonempty (They all include $\emptyset$). Also they are closed with respect to the union operation of languages. Therefore, they have *supremal* elements denoted by $Sup\mathscr{C}(E)$, $Sup\overline{\mathscr{N}}(E;L(G))$, and $SupR_G(E)$, respectively.

As mentioned previously, the controllable, normal , and $L_m(G)$−closed sublanguages defined by

$$\mathscr{C}\overline{\mathscr{N}}R_G(E) := \mathscr{C}(E) \cap \overline{\mathscr{N}}(E; L(G)) \cap R_G(E)$$

provide solutions to the supervisory control problem.

In the special case, when all controllable events are observable, the controllable, observable languages are normal. Therefore, in such cases, the control problem has an optimal solution given by the supremal controllable, normal, and $L_m(G)$−closed sublanguage of E.

Theorem 2.2 *([14]) Suppose all controllable events are observable, $K \neq \emptyset$ and $K \subseteq E \subseteq L_m(G)$. Then there exists a feasible nonblocking supervisor S such that $L_m(S/G) = K$ if and only if K is controllable, normal, and $L_m(G)$−closed.*

Therefore, $Sup\mathscr{C}\overline{\mathscr{N}}R_G(E)$ (if nonempty) characterizes the optimal (minimally restrictive) solution. Furthermore when the legal behavior is a regular language, then the supremal controllable, normal $L_m(G)$−closed sublanguage of E will also be regular. Thus, by using a nonblocking generator marking the supremal controllable, normal, and $L_m(G)$−closed sublanguage of E, the supervisor S can be realized. In this case, the product $meet(G, S)$ represents S/G (plant system under supervision). Note that if E is $L_m(G)$−closed, then so is $Sup\mathscr{C}\overline{\mathscr{N}}(E)$ and therefore, $Sup\mathscr{C}\overline{\mathscr{N}}R_G(E) = Sup\mathscr{C}\overline{\mathscr{N}}(SupR_G(E))$.

Example 2.5 *In this example the solution to the problem introduced in Example 2.4 is obtained using Theorem 2.2. The optimal supervisor to satisfy the given specification is shown in Figure 2.8.*

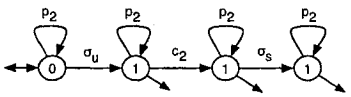

Figure 2.8: Example 2.5, S_{E4} the nonblocking supervisor.

Therefore the plant under supervision will be as displayed in Figure 2.9. It can be seen that the plant under supervision satisfies the specification and is nonblocking.

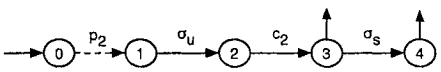

Figure 2.9: Example 2.5, S/G_{E4} the plant under supervision.

2.4 TTCT

This software is mostly used for analysis, synthesis, and verification of supervisory controls. Here we review the commands and procedures that are used in this thesis. More information about TTCT functionality and procedures can be found in [30].

Sync and Meet: The synchronous product of two automata in TTCT can be done by the **sync** procedure. This command produces the synchronous product, (G_3), of two DES models G_1 and G_2:

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

Parallel product of two DES models can be computed with the **meet** procedure. Having the same DES models described, this command produces G_3 with event set $\Sigma_3 = \Sigma_1 \cap \Sigma_2$:

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

Nonconflicting: The **nonconflicting** command in TTCT can be used to verify the nonconflicting property for two languages.

$$\mathbf{nonconflict}(G_1, G_2) = true? \tag{2.6}$$

A "true" answer means that every reachable state of the product automaton $\mathbf{meet}(G_1, G_2)$ is nonblocking, i.e. $\overline{L_m(G_1), L_m(G_2)} = L(G_1) \cap L(G_2)$. Thus, if G_1 and G_2 are non-blocking, **nonconflict** can be used to see if $L_m(G_1)$ and $L_m(G_2)$ are nonconflicting.

Condat The supervisor should be admissible, i.e. the closed behavior of the plant under supervision $L(S/G)$ should be controllable with respect to the closed behavior $L(G)$ of

the plant and uncontrollable events Σ_u.

In TTCT, **condat** is used to verify the admissibility of supervisor.

$$SDAT = \mathbf{condat}(G, S)$$

This procedure gives a list of all plant events disabled at each supervisor state. If this list only includes controllable events, the supervisor is admissible.

2.4.1 Computation of $Sup\mathscr{C}\overline{\mathscr{N}}(E)$

A procedure for the computation of supremal controllable, normal languages has been introduced in [30]. The procedure is as following.

Assume a DES plant model G is defined over the alphabet Σ. Bring in the following

languages:

$$N_o := P\Big(Sup\mathscr{N}\big(E;L_m(G)\big)\Big), \tag{2.7a}$$

$$K_o := Sup\mathscr{C}_o(N_o), \tag{2.7b}$$

$$J := P^{-1}(K_o), \tag{2.7c}$$

$$K := L_m(G) \cap J. \tag{2.7d}$$

Here $Sup\mathscr{C}_o(.)$ operation in 2.7b is performed with respect to the projection of G.

Theorem 2.3 *([9]) Assume G is nonblocking and $(L_m(G),J)$ are nonconflicting. Then we have:*

- $K = Sup\mathscr{C}\mathscr{N}\overline{\mathscr{N}}(E)$;
- *If E is $L_m(G)-$closed, then $K = Sup\mathscr{C}\overline{\mathscr{N}}(E)$.*

Hence the optimal supervisor can be obtained through the sequence of equations (2.7). If at the end K is empty, then $Sup\mathscr{C}\mathscr{N}\overline{\mathscr{N}}(E) = \emptyset$ and (if E is $L_m(G)-$closed) $Sup\mathscr{C}\overline{\mathscr{N}}(E) = \emptyset$.

The TTCT procedure to implement (2.7) is described in the following.

Computation of $Sup\mathscr{C}\overline{\mathscr{N}}(E)$ with TTCT

Based on Theorem 2.3 , one can construct a supervisor by TTCT with the following procedure [30]:

1 Given G, E, and the list $NULL$ of unobservable events

2 $N := \mathbf{supnorm}(E, G, NULL)$

3 $NO := \mathbf{project}(N, NULL)$

4 $GO := \mathbf{project}(G, NULL)$

5 $KO := \mathbf{supcon}(GO, NO)$

6 $KODAT := \mathbf{condat}(GO, KO)$

7 $PINVKO := \mathbf{selfloop}(KO, NULL)$

8 $\mathbf{nonconflict}(G, PINVKO) = true$?

9 $K := \mathbf{meet}(G, PINVKO)$

10 K *nonempty*?

If the answers of part 8, 10 are "yes", the procedure terminates successfully, and $PINVKO$ solves the supervisory control problem (i.e., $PINVKO$ is a DES representing the supervisor) and K is the corresponding controlled behavior (i.e., $L_m(S/G)$).

Chapter 3

Overview of Switching Control

The control of a partially known plant has received considerable attention in the adaptive control literature. One of the relatively new lines of research in this area is switching control which was motivated to weaken the classical *a priori* information required in classical adaptive control and can be traced back to [22]. During the past several years, switching control schemes have been developed to accomplish a wide variety of tasks which would not have been possible using traditional adaptive control methods [15], [19], [16], [18], [20], [21], [2], [4], [6], [5], [25], [1]. For instance, when traditional adaptive control methods are ineffective due to large variations of system parameters, switching control can be applied to achieve the desired objectives.

3.1 Single-Layer Switching

A switching control scheme for a family of plant models was studied by Miller and Davison in [16], where it was assumed that the LTI plant model at any time belongs to a finite set of known models called the family of plant models. A sudden change from one of the plant models to another one in the set represents variation of system parameters. It is assumed a high-performance controller is designed for each plant in the family. Each controller is designed such that it stabilizes its corresponding plant model and destabilizes all other ones.

Example 3.1 *Figure 3.1 shows a system modeled by a family of* 5 *plant models* $\{P_1,\ldots,P_5\}$. *A set of* 5 *controllers* $\{C_1,\ldots,C_5\}$ *forms the switching control structure. The pair* (P_i,C_j) *represents a stable closed-loop system if and only if* $i = j$.

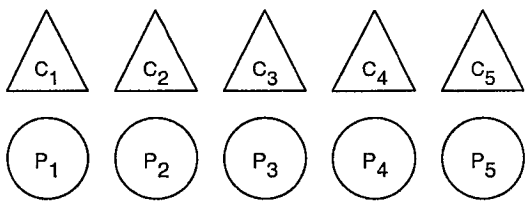

Figure 3.1: Single-layer switching control structure of Example 3.1

Suppose that the plant is initially P_1 *and is stabilized by* C_1. *Let the plant change to* P_4 *at some time instant. By assumption, the pair* (P_4,C_1) *is not stable. It is shown in [16] that*

the instability can be detected after a bounded time t_d by comparing an error signal with a proper "upper-bound" signal. At this point, it is known that the plant model has changed but the new model is unknown. The supervisor switches the controller to C_2 and stays there until instability is detected again which results in switching to C_3. Finally the system switches to C_4 which leads to closed-loop stability. The switchings for this example are shown in Figure 3.2.

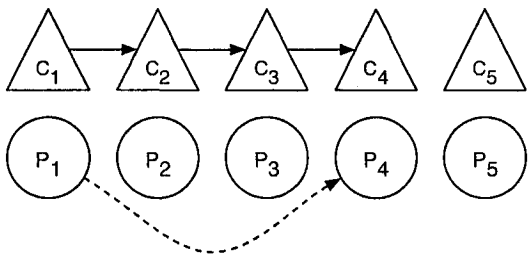

Figure 3.2: Single-layer switching mechanism of Example 3.1

Throughout this thesis, switching to a controller that stabilizes the system is called a "**stable switching**" and switching to a controller that destabilizes the system is called an "**unstable switching**". In Example 3.1, transitions to C_2 and C_3 represent unstable switchings and transition to C_4 represents a stable switching.

3.2 Multi-Layer Switching

One of the problems in switching control methods is the poor transient response which is mainly due to switching through a number of destabilizing controllers before the system locks onto the correct controller. When the plant dynamics change from one model to another, the system may switch to some destabilizing controllers before it finds the correct controller. Let p denote the number of plant models. It can be easily seen that a maximum of $p-2$ unstable switchings may occur before the correct controller is found. The switching control structure introduced in [16] is referred to as a single-layer structure as it is composed of only one set of controllers. A multi-layer structure is proposed in [11] that can potentially reduce the number of unstable switchings. The first layer consists of a set of high-performance controllers similar to those in the single-layer structure. The n^{th} layer consists of a set of simultaneous stabilizers, and it is assumed that the simultaneous stabilizer $C_{i_1 i_2 \ldots i_n}$, where $i_1, i_2, \ldots, i_n$ are n distinct integers, stabilizes plants $P_{i_1}, P_{i_2}, \ldots, P_{i_n}$ and destabilizes all other ones.

An algorithm is given in [11] for switching between the controllers of different layers, which guarantees that with at most one unstable switching the system finds the correct first layer controller. This is illustrated in the next example. It is to be noted that there are at most p layers of controllers for a family of p plant models. It can be shown that only the first $p-2$ layers are required in a multi-layer structure in order to minimize the number of unstable

switchings [11].

Example 3.2 *Consider the family of plant models given in Example 3.1. A multi-layer switching control structure is shown in Figure 3.3.*

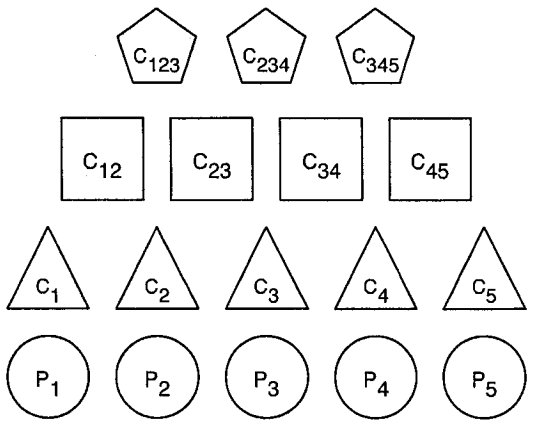

Figure 3.3: Multi-layer structure of Example 3.2

Assume that at some points the plant changes from P_1 to P_4 as in Example 3.1. Using the algorithm given in [11], the control system switches in the following sequence. As soon as instability is detected, the plant is known not to be P_1 and thus, the system switches to C_{234}. The reason for choosing C_{234} is as follows. After instability is detected, we conclude that the plant model is not P_1 and the candidate plant models will be P_2, P_3, P_4, and P_5. Since we intend to limit the number of unstable switches to one, after any unstable switching, we want to be sure about the current plant and switch to the corresponding first-layer stabilizing controller

(in this example, C_4). Therefore, the third layer controller C_{234} is chosen in such a way that it stabilizes all candidate plants except one (P_5). Therefore, if after switching instability is detected, the actual plant model will be immediately known. (The same logic is used for choosing controllers in subsequent steps.) After switching to C_{234} and once it is determined that the resultant closed-loop system is stable, then it can be concluded that the unknown plant must belong to the set $\{P_2, P_3, P_4\}$. The system then switches to C_{23} (which stabilizes P_2 and P_3 and destabilizes all the others) and remains in C_{23} until stability or instability is detected. In this example, the resultant closed-loop system is unstable. This also shows that the plant model cannot be either of P_2 or P_3; thus the system switches to C_4 which is, in fact the desired controller. The switchings are shown in Figure 3.4. Note that the sequence of transitions contains only one unstable switching.

In the switching control problem, it would be desirable to impose restrictions on transitions. For instance, to maintain the quality of the transient response, one may want to limit the total number of unstable transitions before the system locks onto the correct controller in the first layer. We refer to a switching policy that can find the correct controller without violating the desired restrictions on the transitions of a **proper switching policy**.

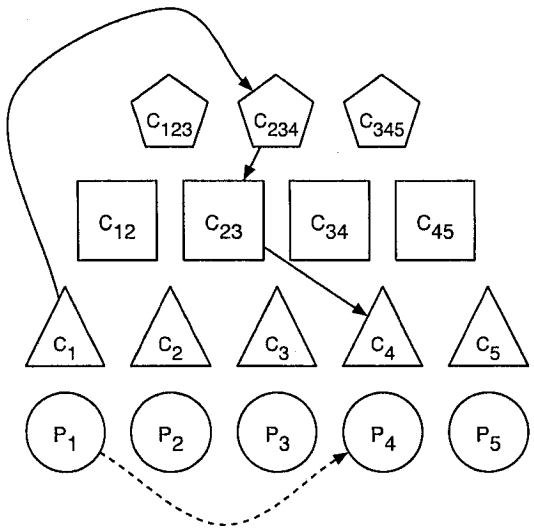

Figure 3.4: Multi-layer switching sequence of Example 3.2

The number of controllers in a multi-layer structure grows exponentially with the number of plants p. In practice, designing all of these controller can be cumbersome and sometimes impossible. Furthermore, only a subset of all possible controllers may be sufficient to achieve the objectives, i.e. to meet the design specifications for transitions.

In this thesis, the problem of designing a proper switching policy is formulated and solved as a problem of Supervisory Control of a Discrete-Event System (DES) under partial observation in the Ramadge-Wonham framework [29].

Chapter 4

Formulating Switching Control as a DES Supervisory Control Problem

In this chapter, we consider the multi-layer switching control scheme as introduced in [11] and show how the plant model changes and controller switchings can be represented using DES models and finally, how the synthesis of the switching control policy can be cast as a problem of supervisory control of DES.

4.1 Switching Control Formulation

Consider a given finite set of finite-dimensional, LTI plant models with bounded disturbances

$$\Pi = \{P_i : i \leq i \leq p\} \tag{4.1}$$

and assume that at any time t, the real plant model, $P(t)$, belongs to this set, i.e. $P(t) \in \Pi$. Let $\overline{p} = \{1, \ldots, p\}$. It is assumed that for each plant $P_i \in \Pi$ there exists a high performance controller C_i, which solves the robust servomechanism problem (RSP) for the plant P_i. These controllers build the first layer in the multi-layer structure and are represented by the following set

$$\Phi_1 = \{C_i : i \in \overline{p}\} \tag{4.2}$$

Similarly, the set of controllers of layer k $(2 \leq k \leq p)$ is denoted by Φ_k with

$$\Phi_k = \{C_{i_1 i_2 \ldots i_k} | i_1, i_2, \ldots, i_k \in \overline{p}\} \tag{4.3}$$

where $i_j, j = 1, \ldots, k$ are distinct integers and the indices of each controller represent the plants that are stabilized by that controller; e.g. $C_{i_1 i_2 \ldots i_k}$ "only" stabilizes plant models $P_{i_1}, P_{i_2}, \ldots, P_{i_k}$, and destabilizes the other plants in the set.

Suppose the plant model is P_j and the current controller is $C_{i_1 i_2 \ldots i_k}$. In order to determine

the stability of the closed-loop system, an upper-bound signal (introduced in [16]) is generated and then compared with a filtered error signal. If the closed-loop system is unstable, then within a bounded time (following the last controller or plant transition), the error signal meets (or crosses) the upper-bound signal. This time duration can be determined by considering the worst-case scenario associated with the initial conditions, reference inputs and disturbance signals. It may also be obtained experimentally. This time duration is referred to as the **maximum time for stability** and denoted by t_d. If the error signal does not meet the upper-bound signal in t_d seconds, then the closed-loop system is stable.

Remark 4.1 *It is to be noted that for the proposed switching method to stabilize the system, it is assumed that the time interval between two consecutive switchings is greater than t_d [3]. Thus, after every controller switching, we wait till the stability or instability of the plant/controller pair to be determined, before we perform another controller switching.*

4.2 Constructing a DES Model for the Switching Control Problem

We start by examining some assumptions that are usually made in the study of switching control.

4.2.1 Assumptions in Switching Control

As discussed in Chapter 3, the switching control system (single-layered or multi-layered) can switch to the correct first-layer stabilizing controller in a bounded time. Therefore, in the analysis of switching control system, typically it is assumed that initially, the plant and the controller match. Since as long as the plant does not change the the first layer stabilizing controller is the best matching controller, there is no need to switch to another controller and therefore, the first change in the coupled plant and controller system will be a change of plant to another (unknown) plant. Thus the following assumption is used in our study.

Assumption 4.1 *The initial plant is known and the controller is the corresponding first-layer stabilizing controller. The plant will then change to an unknown plant in the given family of models.*

In switching control, it is assumed that between two plant changes, there is a sufficient time for the switching policy to find the correct controller.

Assumption 4.2 *The plant model remains unchanged for a sufficiently long time.*

The time in Assumption 4.2 is long enough so that after every plant model change, the switching control system can find the appropriate first-layer stabilizing controller.

4.2.2 Event Set

Now we develop a finite-state automaton to model changes in plant model and controller switchings. The alphabet of this DES, Σ, can be partitioned according to $\Sigma = \Sigma_\Pi \cap \Sigma_\Phi \cup \Sigma_r$:

Plant Transition Events (Σ_Π): $\Sigma_\Pi = \{p_1, \ldots, p_p\}$ where events $p_1, \ldots, p_p$ represent changes in plant model. Specifically, p_i signifies "transition to plant model P_i". Events in Σ_Φ are obviously uncontrollable and by assumption, unobservable.

Controller Switching Commands (Σ_Φ): Switching between controllers is modeled

with controllable, observable events, named based on the controller after switching. For instance, the switching command to the controller C_i in the first layer is called c_i and the switching command to the controller $C_{i_1 i_2}(i_1 \neq i_2)$ in the second layer is denoted by $c_{i_1 i_2}$. Σ_Φ is the set of all switching commands.

Stability and Instability (Response) Events (Σ_r): If after a plant transition or controller switching the closed-loop system becomes unstable, as mentioned previously the error signal meets the upper-bound signal before t_d time units. This is represented in our work by an uncontrollable, observable event σ_u. When as a result of controller switching, the closed-loop system becomes stable, then after time t_d, we will know the closed-loop system is stable. We model the announcement of this conclusion by the monitoring system with an uncontrollable, observable event σ_s . Therefore, $\Sigma_r = \{\sigma_s, \sigma_u\}$ is the set of response events.

4.2.3 DES Model

In order to build a DES model for switching control system, we first construct two finite-state automata P_{DES} and C_{DES} to represent plant transitions and controller switchings, respectively. Each state of P_{DES} corresponds to a unique plant model. Similarly each state of C_{DES} corresponds to a unique controller in the set $\bigcup_{k=1}^{p} \Phi_k$. Without lost of generality, we assume initially the plant model is P_1 and hence, by Assumption 4.1, the controller is C_1. Let

$PC_{DES} = P_{DES}||C_{DES}$.

Each state of PC_{DES} is represented by a pair of plant model and controller (P,C). For each state (P,C), if the corresponding closed-loop system is stable (i.e. C stabilizes P), we add a σ_s selfloop, otherwise we add a σ_u selfloop. The resulting automaton will be called PCN_{DES}. We use a running example to illustrate the modeling procedure.

Example 4.1 *In this simple example, we build a DES model for a multi-layer switching problem. Consider a family of two plant models and a set of three controllers as shown in Figure 4.1.*

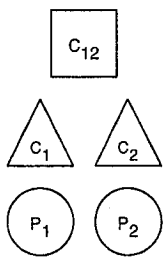

Figure 4.1: Multi-layer controller for two plants in Example 4.1

The automata P_{DES} and C_{DES} representing plant transition and controller switchings are shown in Figures 4.2 and 4.3, respectively.

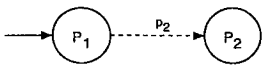

Figure 4.2: P_{DES}, DES model for two plants in Example 4.1

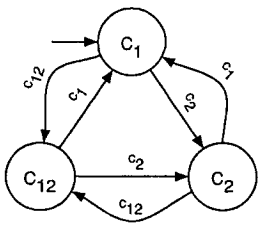

Figure 4.3: C_{DES}, DES model for the three controllers of Example 4.1

PCN_{DES} is constructed by forming the synchronous product of C_{DES} and P_{DES} and adding appropriate σ_s and σ_u selfloops at each state as shown in Figure 4.4.

As mentioned before, initially, the plant and the controller match. In this case, controller switching is not done unless the plant model changes and an instability event σ_u is generated. By Remark 4.1, after every controller switching, we wait for the corresponding response event (σ_s or σ_u) to be generated before a controller switching is ordered (if necessary). The above behavior (and restriction) is modeled by the automaton SU_{DES} in Figure 4.5. Finally, the complete plant model, represented by G, is built as

$$G = PCN_{DES} || SU_{DES}.$$

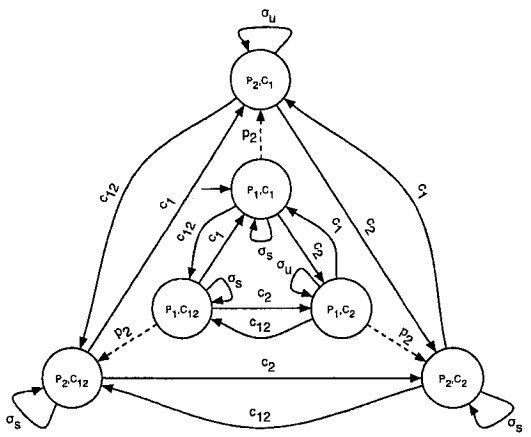

Figure 4.4: PCN_{DES}, *sync* product of P_{DES} and C_{DES} with selfloops of Example 4.1

In G, the initial state $(P_1, C_1, 0)$ and all states $(P_i, C_i, 2)$ $(2 \leq i \leq p)$ which correspond the cases in which the plant and controller match (and the response σ_s has been generated in the case of $(P_i, C_i, 2)$, $2 \leq i \leq p)$), are marked.

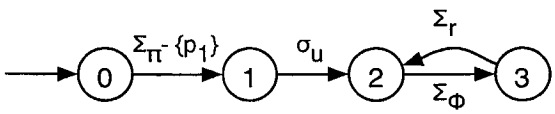

Figure 4.5: SU_{DES}, DES representing plant change/controller switching and response order

Example 4.2 *The automaton SU_{DES} modeling the order of plant transition and controller switchings, and the corresponding responses (σ_s, σ_u) is shown in Figure 4.6. The complete plant model G is given in Figure 4.7.*

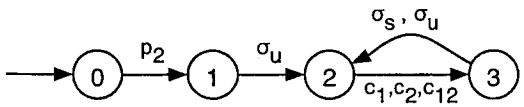

Figure 4.6: SU_{DES}, DES model of representing the order of plant change, controller switching and response in Example 4.1

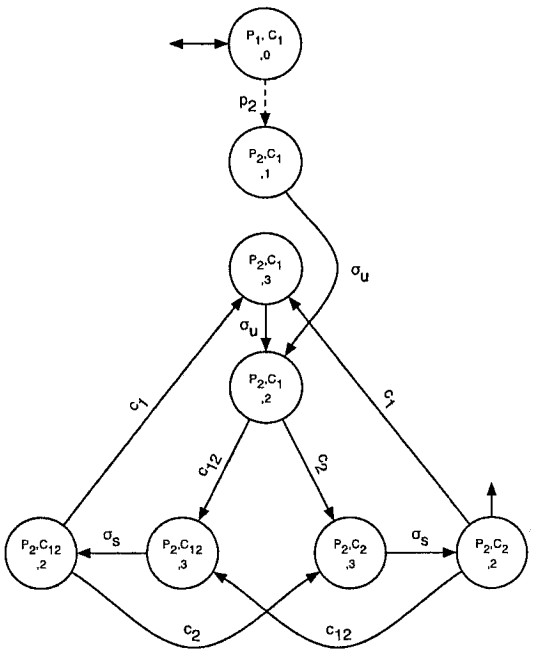

Figure 4.7: G, DES plant model of Example 4.1

Remark 4.2 *It follows from the transition structure of SU_{DES} that always a p_i event $(i \neq 1)$ is followed by σ_u. After this, G can only generate sequences of controller switching followed by the corresponding response (σ_s or σ_u). Thus for the closed behavior of G we have*

$$L(G) \subseteq (\Sigma_\Pi - \{p_1\})\sigma_u(\Sigma_\Phi \Sigma_r)^*$$

or in other words, $L(G)$ consists of sequences of the form $p_i\sigma_u c^{(1)}\sigma^{(1)}\ldots c^{(l)}\sigma^{(l)}$ with $p_i \neq p_1$, $c^{(j)} \in \Sigma_\Phi$, and $\sigma^{(j)} \in \Sigma_r$, and prefixes of all such sequences.

Remark 4.3 *In G those states are marked in which the controller is the stabilizing first layer controller for the current plant: $(P_1, C_1, 0)$ and all $(P_i, C_i, 2)$ for all $2 \leq i \leq p$. As a result, $L_m(G)$ consists of ε and all sequences of the form $p_i\sigma_u c^{(1)}\sigma^{(1)}\ldots c^{(l)}\sigma^{(l)}c_i\sigma_s$ with $c^{(i)} \in \Sigma_\Phi$, $\sigma^{(i)} \in \Sigma_r$, and $l \geq 0$.*

In summary, the final DES model will be an automaton $G = (Q, \Sigma, \delta, q_0, Q_m)$. Here Q is the state set. For a family of p plants and c controllers, the size of Q will be $(p-1) \times c \times 2 + 2$. Σ is the event set consisting of plant transition, controller switching, and response (stability/instability) events: $\Sigma = \Sigma_\Pi \cup \Sigma_\Phi \cup \Sigma_r$. δ is the transition function. q_0 is the initial state which is assumed to be known. Q_m is the marked state set consisting of the initial state $(P_1, C_1, 0)$ and states of the form $(P_i, C_i, 2)$ $(i \neq 1)$, which represent the cases where the appropriate first-layer controller is in the feedback loop, stabilizing the current plant.

One of the main advantages of transforming the switching control problem to a DES problem is the systematic procedures available for representing various design specifications and designing supervisors (switching policies) to meet the specifications. After building the DES plant model for the multi-layer switching problem, each specification can be represented by an automaton and the *meet* product of all the specifications will give us the combined design specification. Following this, a supervisor can be designed to satisfy the specifications.

For instance, suppose that it is desired to have at most one unstable switching before switching to the stabilizing first-layer controller. Figure 4.8 shows this design specification modeled as a DES. Note that the specification model counts two unstable events because the first unstable event (σ_u) happens after the plant model changes and is not considered as an unstable switching.

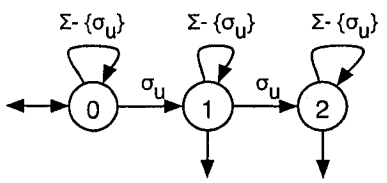

Figure 4.8: Specification for a maximum of one unstable switching.

Chapter 5

Designing Switching Policies

In this chapter, we address two main problems in switching control. First, given a set of controllers and specifications, we would like to determine if it is possible to design a switching policy to satisfy the specifications and if so, find a minimally restrictive switching policy. Secondly, assuming the answer to the first question is affirmative, we would like to find a minimal subset of controllers enforcing the specifications.

5.1 Minimally Restrictive Switching Policy

In the previous chapter, we showed how the problem of finding a multi-layered switching policy can be converted to a problem of supervisory control of DES under partial observation. The solution obtained for the supervisory control problem would be a supervisor that monitors the observable events unfolding in the plant and enables or disables "controller switching" commands to make sure the design specifications (restrictions on transitions) are met and the DES system under supervision is nonblocking (i.e., the appropriate first-layer stabilizing controller can be brought into feedback loop). Note that it may be possible that the supervisor enables transitions to more than one controller which means that all enabled transitions are acceptable and none of them will violate design specifications.

One of the main challenges in multi-layer switching control is to verify whether, for a given set of controllers, a switching policy satisfying the design specifications exists.

5.1.1 Problem Solvability

Problem 1: *Given the plant set* Π, *the controller set* Φ, *and an initial plant* P_{ini} *and controller* C_{ini},*and the legal behavior E, find a switching policy (supervisor) meeting the design*

specifications.

In our supervisory control problem all controllable events ($\Sigma_c = \Sigma_\Phi$) are observable. Therefore, all controllable, observable languages are normal and by Theorem 2.2, the supervisory control problem (Problem 1), is solvable if and only if

$$Sup\mathscr{C}\overline{\mathscr{N}}R_G(E) \neq \emptyset. \tag{5.1}$$

If (5.1) holds, then the supremal controllable, normal, $L_m(G)-$closed sublanguage of E provides a minimally restrictive supervisor (switching policy) for the problem. Note that if the legal behavior E is $L_m(G)-$closed, then (5.1) becomes

$$Sup\mathscr{C}\overline{\mathscr{N}}(E) \neq \emptyset. \tag{5.2}$$

The following example explores problem solvability and design of a minimally restrictive switching policy.

Example 5.1 *Consider a set of five plant models* $\{P_1, P_2, P_3, P_4, P_5\}$ *and the following controller set* $\{C_1, C_2, C_3, C_4, C_5, C_{12}, C_{23}, C_{34}, C_{45}, C_{123}, C_{234}, C_{345}\}$ *as shown in Figure 5.1.*

Let $G = (Q, \Sigma, \delta, q_0, Q_m)$ *be the DES plant of the supervisory control problem constructed as described in Chapter 4, with* $\Sigma = \{c_1, c_2, c_3, c_4, c_5, c_{12}, c_{23}, c_{34}, c_{45}, c_{123},$

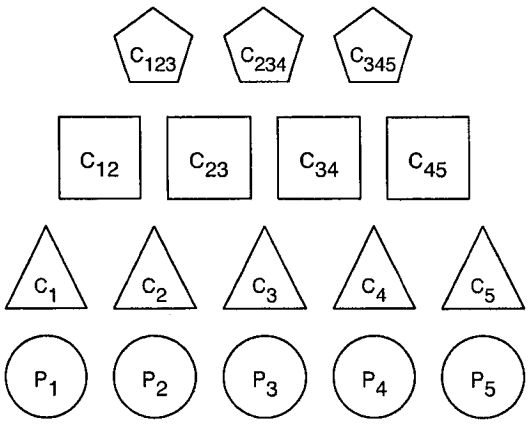

Figure 5.1: Multi-layer structure of Example 5.1

c_{234}, c_{345}, p_1, p_2 ,p_3, p_4, p_5, σ_s, $\sigma_u\}$. *The unobservable and the uncontrollable event sets are* $\Sigma_{uo} = \{p_1, p_2, p_3, p_4, p_5\}$ *and* $\Sigma_{uc} = \{p_1, p_2, p_3, p_4, p_5, \sigma_s, \sigma_u\}$ *respectively. Assume that initially the plant is* P_1 *and the active controller is* C_1, *i.e.* q_0 *is* $(P_1, C_1, 0)$. *The design specification is to have a maximum of one unstable switching, which is captured in the DES in Figure 5.2.*

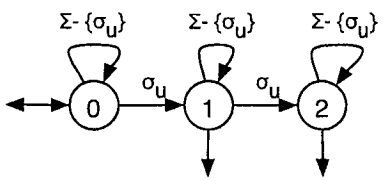

Figure 5.2: Specification for maximum one unstable switching

To verify whether the given controller set meets the specification, we use Theorem 2.2. Let us call the DES in Figure 5.2 SPECU. Then the legal behavior E will be $E = L_m(G) \cap L_m(SPECU)$. *E is* $L_m(G)-$*closed since* $L_m(SPECU)$ *is closed. Therefore, the supervisory control is solvable if and only if (5.2) holds. We compute* $Sup\mathscr{C}\overline{\mathscr{N}}(E)$ *using the procedure in [9] and find that* $Sup\mathscr{C}\overline{\mathscr{N}}(E) \neq \emptyset$. *This implies that a switching policy exists to lead the system to the correct first-layer controller with no more than one unstable switching.*

It can be easily verified that in Example 5.1 (assuming initial pair of (P_1, C_1)), after removing C_{234} and C_{345} from the set of controllers in layer 3, condition (5.2) will no longer hold, which implies that the design specification cannot be met using the given set of controllers. However, if C_{123} and C_{345} are removed instead, (5.2) holds and we can find a supervisor (proper switching policy) to arrive at the correct first-layer controller with no more than one unstable switching.

In the above example, we tested problem solvability assuming initial plant/controller pair of (P_1, C_1). In general, we have to perform the procedure for all initial pairs (P_i, C_i) $(i \in \overline{p})$.

Example 5.2 *Consider the set of plants and controllers as shown in Figure 5.3. We would like to deign a switching policy that can assure that, starting from the initial plant/controller*

(P_1, C_1), *if plant model changes, the system switches to the first layer stabilizing controller with at most one unstable switching. The plant G in this example has 34 and 81 transitions. Similar to the previous example, we design a minimally restrictive supervisor which has 37 states and 169 transitions.*

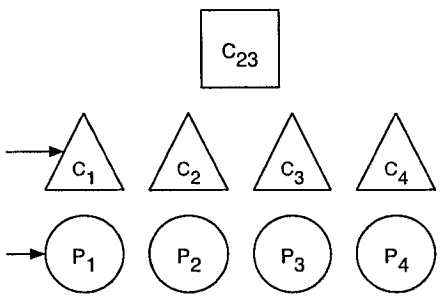

Figure 5.3: Part of plant model G for Example 5.2

Part of the plant and supervisor models are shown in Figures 5.4 and 5.5, respectively. Let us follow the switching procedure and see how the supervisor, by enabling and disabling the controllable events, satisfies the specification. At first both plant and supervisor are in their initial states 0 and 0. Then assume the plant model changes to P_3. *Accordingly the DES G moves to state 2. Note that the plant changing event* P_3 *is unobservable and appears as a selfloop in the supervisor (Figure 5.5). Therefore, the supervisor's state remains unchanged. At the next step, instability event* σ_u *occurs. Thus, G and S go to states 5 and 1, respectively. At state 5 of G all switching commands* c_2, c_3, c_4, c_{23} *can happen. On the other hand, the supervisor in its state 1 disables all the switching controller commands except* c_{23}. *Therefore,*

the plant G will go to state 10 and the supervisor goes to state 2 by switching the controller to C_{23}. At the next step the stability event σ_s occurs. Hence, the supervisor goes to state 4 and G goes to 15. At this state, the supervisor only permits c_2 and c_3 to happen, and it does not make any difference which one occurs. Let us assume that the controller switches to C_3, and accordingly the DES G goes to state 18 and the supervisor to 11. Again σ_s happens and the DES G and the supervisor go to states 21 and 13. Note that state 21 is marked and the controller C_3 matches the plant model P_3.

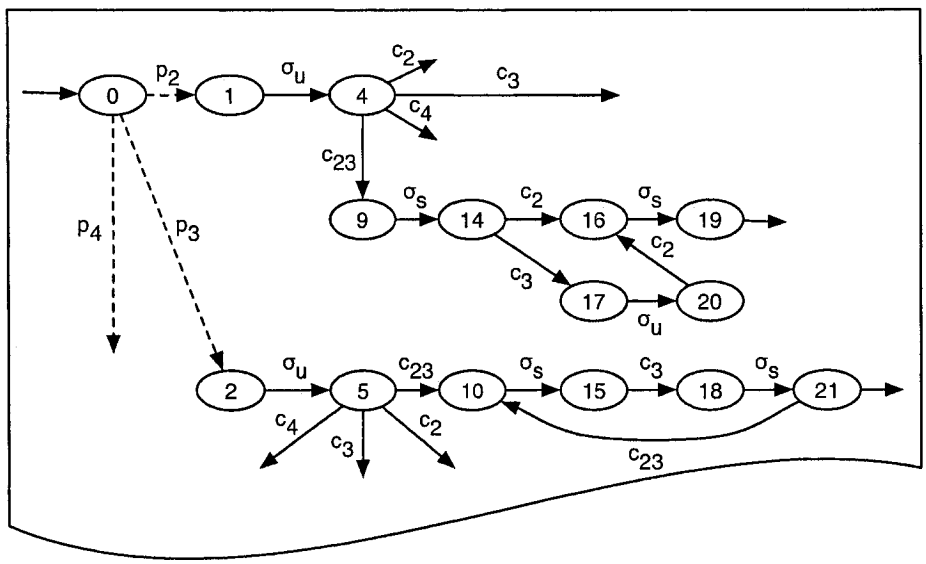

Figure 5.4: Part of plant model *G* for Example 5.2

It can be seen on Figures 5.4 and 5.5 that the DES G and the supervisor can continue to move on cycles 10 → 15 → 18 → 21 → 10 and 10 → 12 → 11 → 13 → 10, respectively. This happens because there is no limit on the number of stable switchings. Consequently, it

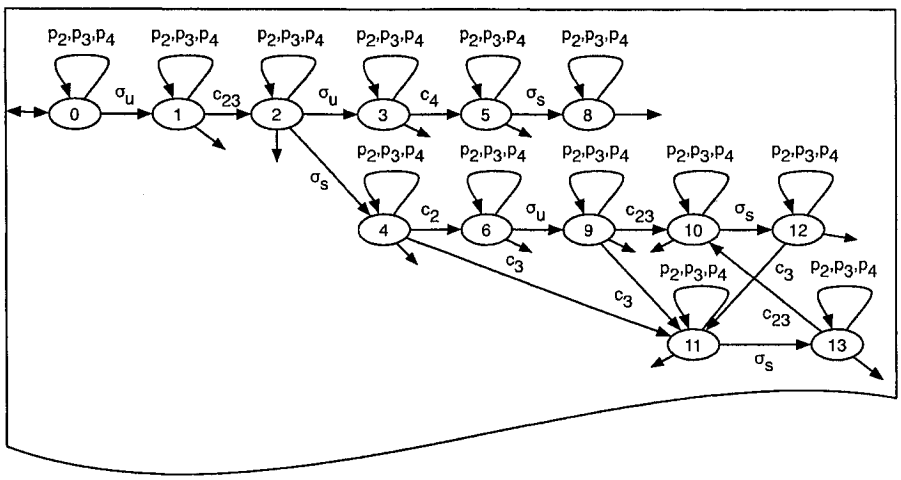

Figure 5.5: Part of supervisor controller model S for Example 5.2

is better to have an upper limit on the number of stable switchings to prevent unnecessary switching among stabilizing controllers.

In addition to limiting the number of unstable switchings, we may wish to limit the total number of stable and unstable switchings to prevent large transient oscillations.

Example 5.3 *Consider the same set of plants and controllers as in Example 5.1. It is desired to find a switching policy in order to lock onto the stabilizing first-layer controller with at most one unstable switching (similar to Example 5.1) and two stable switchings (i.e. not more than three switchings). The corresponding DES model of specifications is formed by the product of two models shown in Figure 5.2 and Figure 5.6. In this case, condition (5.2) is not satisfied*

meaning that the design specifications cannot be satisfied using the given controller set.

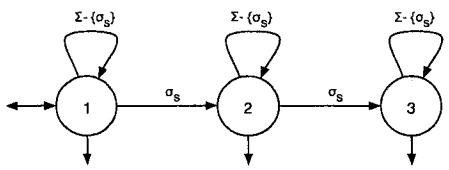

Figure 5.6: Specification for maximum of two stable switching

5.2 Finding a minimal controller set

Now we would like to address the problem of finding a minimal set of controllers for which there exists a switching policy to satisfy the design specifications.

5.2.1 Problem of Finding a Minimal Controller Set

Problem 2: *Given the plant set* Π, *the controller set* Φ, *the initial plant and controller pair of* P_{ini}, C_{ini}, *and the legal behavior* E, *find a nonempty subset of* Φ, *say* Φ', *such that:*

1 For the set of controllers Φ', *there exist a switching policy to satisfy the legal behavior.*

2 *For all* $\Phi'' \subset \Phi'$ $(\Phi'' \neq \Phi', \Phi'' \neq \emptyset)$, *there is no switching policy to satisfy the legal behavior.*

Minimizing the number of controllers is equivalent to minimizing the controller switching events. To achieve this, we may remove controller switching events from the model G and then see if the resulting supervisory control problem is solvable. Note that a minimal set Φ' must include all first-layer controllers: $\Phi_1 \subseteq \Phi'$.

Consider the plant $G = (Q, \Sigma, \delta, q_0, Q_m)$ and let $\Delta \subset \Sigma_\Phi$, $\Delta \neq \emptyset$, be a set of controller switching events. Suppose $G' = (Q', \Sigma', \delta', q_0, Q'_m)$ is the reachable automaton obtained from G after removing all transitions corresponding to the events in Δ. Thus,

$$\Sigma' = \Sigma - \Delta$$

$$Q' = Q - Q_\Delta$$

where Q_Δ is the set of states that become unreachable after the removal of events in Δ. When there is no command available to switch to a controller, that controller is in fact deleted from the controller set. It is not difficult to see that the following relations exist:

$$L(G') = L(G) - \Sigma^* \Delta \Sigma^*$$

$$L_m(G') = L_m(G) - \Sigma^* \Delta \Sigma^*$$

Furthermore, if E and E' denote the legal behaviors then

$$E' = E - \Sigma^* \Delta \Sigma^*$$

5.2.2 First Solution to the Problem of Finding a Minimal Controller Set

The most straightforward procedure for finding a minimal set of controllers is described in the following algorithm.

Algorithm 5.1

- ***Given:*** *DES plant* $G = (Q, \Sigma, \delta, q_0, Q_m)$ *and legal behavior* E

- ***Start:*** *Let* $G_t = (Q_t, \Sigma_t, \delta_t, q_0, Q_{m_t}) = G = (Q, \Sigma, \delta, q_0, Q_m)$ *and*

  $E_t = SupR_G(E)$

  If $Sup\mathscr{C}\overline{\mathscr{N}}(E_t) = \emptyset$, *then*

  Problem has no solution; stop

  End(If)

  While $\Sigma_t \cap (\bigcup_{i=2}^{p} \Sigma_{\Phi_i}) \neq \emptyset$

  For all $c \in \Sigma_t \cap (\bigcup_{i=2}^{p} \Sigma_{\Phi_i})$

Let $\Sigma_t' = \Sigma_t - \{c\}$.

Construct $G_t' = (Q_t', \Sigma_t', \delta_t', q_0, Q_{m_t}')$ *and* $E_t' = E_t - \Sigma^* c \Sigma^*$

- ***LC:*** *If* E_t' *is not* $L_m(G')$*−closed, then*

  Replace E_t' *with* $SupR_{G_t'}(E_t')$

  End(If)

  If $Sup\overline{\mathscr{C}\mathscr{N}}(E_t') \neq \emptyset$, *then*

  Replace G_t *and* E_t *with* G_t' *and* E_t'*; go to* ***E2***

  End(If)

End(For)

- ***E1:*** $Sup\overline{\mathscr{C}\mathscr{N}}(E_t)$ *provides the solution and* $\Sigma_t \cap \Sigma_\Phi$ *identifies a minimal set of controllers; stop*

- ***E2:*** *continue*

End(While)

In the main loop (the While loop), a controller event 'c' is removed from the event set. If the supervisory control problem is solvable for the resulting plant G_t' and legal behavior E_t', then the set of controller events in Σ_t is not minimal and G_t and E_t are replaced with G_t' and E_t' and the main loop continues. If for all controllers 'c', the supervisory control problem is not solvable, then the set of controller events in Σ_t is minimal and the algorithm terminates.

To find a minimal set of controllers satisfying the specifications, one needs to calculate the supremal controllable, normal, and $L_m(G)$−closed sublanguage of the legal behavior at every step of the algorithm (after deleting each controller). This could be time consuming. Next, we will simplify these computations.

5.2.3 Second Solution to the Problem of Finding a Minimal Controller Set

First we characterize the set of all controllable sublanguages of $L_m(G)$.

Proposition 5.1 *The controllable sublanguages of* $L_m(G)$ *are* $\emptyset$ *and all* K *such that* $\{p_2, \ldots, p_n\} \subseteq \overline{K}$.

To prove Proposition 5.1, we need the following result.

Lemma 5.1 *Let* $K \subseteq L_m(G)$ *and suppose* $\{p_2, \ldots, p_n\} \subseteq \overline{K}$. *Then* K *is controllable with respect to* G.

Proof: It is sufficient to show that for all $s \in \overline{K}$, $s\Sigma_{uc} \cap L(G) \subseteq \overline{K}$.

Four cases are possible:

- $s = \varepsilon$: *Then by Remark 4.2,* $s\Sigma_{uc} \cap L(G) = \{p_2, \ldots, p_n\}$; *therefore* $s\Sigma_{uc} \cap L(G) \subseteq \overline{K}$.

- *s ends in* p_i $(i = 2, \ldots, n)$: *Thus* $s = s'p_i$ *for some* $s' \in \Sigma^*$. *By Remark 4.2,* p_i *can only occur as the first event in every sequence and it is followed by only* σ_u, *therefore* $s = p_i$ *(i.e.,* $s' = \varepsilon$*) and* $s\Sigma_{uc} \cap L(G) = \{p_i\sigma_u\}$. *Now since* $s = p_i \in \overline{K}$ *then there exist* $t \in \Sigma^+$ *such that* $p_i t \in K \subseteq L_m(G)$. *Then by Remark 4.3, there exist* $t' \in \Sigma^+$ *such that* $t = \sigma_u t'$. *Therefore* $s\Sigma_{uc} \cap L(G) = \{p_i\sigma_u\} \subseteq \overline{K}$.

- *s ends in* σ_u *or* σ_s : *By Remark 4.2, in plant G, the event following* σ_u *or* σ_s *must be some* $c \in \Sigma_\Phi$. *Thus* $s\Sigma_{uc} \cap L(G) = \emptyset \subseteq \overline{K}$.

- *s ends in* $c \in \Sigma_\Phi$: *By Remark 4.2, in G, in the state reached with s, either* σ_s *is defined or* σ_u *(not both). Thus* $s\Sigma_{uc} \cap L(G) = \{s\sigma_s\}$ *or* $s\Sigma_{uc} \cap L(G) = \{s\sigma_u\}$. *In the first case, since* $s \in \overline{K}$ *there exist* $t \in \Sigma^+$ *such that* $st \in K \subseteq L_m(G)$. *By assumption,* $s\sigma_u \notin L(G)$; *therefore, there exists* $t' \in \Sigma^*$ *such that* $t = \sigma_s t'$ *and hence* $s\sigma_s t' \in K \subseteq L_m(G)$. *Thus* $s\sigma_s \in \overline{K}$ *and* $s\Sigma_{uc} \cap L(G) \subseteq \overline{K}$. *Similarly, in the second case, there exist* $t' \in \Sigma^+$ *such that* $s\sigma_u t' \in K \subseteq L_m(G)$. *Thus* $s\sigma_u \in \overline{K}$ *and* $s\Sigma_{uc} \cap L(G) \subseteq \overline{K}$. □

Proof of Proposition 5.1. Obviously $\emptyset$ *is controllable. Furthermore, by Lemma 5.1, all* $K \subseteq L_m(G)$ *such that* $\{p_2, \ldots, p_n\} \subseteq \overline{K}$ *are controllable. Conversely, suppose* $K \subseteq L_m(G)$ *and*

$K \neq \emptyset$. If $p_i \notin \overline{K}$ (for some $i \in \{2,\ldots,n\}$), then $p_i = \varepsilon p_i \in \overline{K}\Sigma_{uc} \cap L(G)$ but $p_i \notin \overline{K}$. Therefore, K is not controllable. □

The following theorem shows that all normal sublanguages of $L_m(G)$ are controllable.

Theorem 5.1 *If $K \subseteq L_m(G)$ and $\overline{K}$ is $(L(G),P)-$normal, then K is controllable.*

Proof: If $K = \emptyset$ then obviously K is controllable. Suppose $K \neq \emptyset$. It follows from Remark 4.2 that there exists p_i $(2 \leq i \leq p)$ such that $p_i \in \overline{K}$. Furthermore, $P^{-1}P(p_i) \cap L(G) = \{p_2,\ldots,p_p\}$. It follows from normality of $\overline{K}$ that $\{p_2,\ldots,p_p\} \subseteq \overline{K}$. Hence, by Proposition 5.1, K is controllable. □

It follows from Theorem 5.1 in Algorithm 5.1, all $Sup\mathscr{C}\overline{\mathscr{N}}(.)$ operations can be substituted with $Sup\overline{\mathscr{N}}(.)$.

Next, we consider the property of $L_m(G)-$closedness.

Theorem 5.2 *If K is $L_m(G)-$closed, then $K' = K - \Sigma^*\Delta\Sigma^*$ is $L_m(G') - closed$.*

Proof: We have to show $K' = \overline{K'} \cap L_m(G')$. $K \subseteq L_m(G)$ implies $K' \subseteq L_m(G)$ and therefore

$K' \subseteq \overline{K'} \cap L_m(G')$. *Thus we only have to show* $\overline{K'} \cap L_m(G') \subseteq K'$. *We have*

$$\begin{aligned}
\overline{K'} \cap L_m(G') &= \overline{\overline{K - \Sigma^* \Delta \Sigma^*} \cap (L_m(G) - \Sigma^* \Delta \Sigma^*)} \\
&= \overline{K \cap \overline{(\Sigma^* \Delta \Sigma^*)^{co}}} \cap L_m(G) \cap (\Sigma^* \Delta \Sigma^*)^{co} \\
&\subseteq \overline{K} \cap \overline{\overline{(\Sigma^* \Delta \Sigma^*)^{co}}} \cap (\Sigma^* \Delta \Sigma^*)^{co} \cap L_m(G) \\
&= \overline{K} \cap L_m(G) \cap (\Sigma^* \Delta \Sigma^*)^{co} \\
&= K \cap (\Sigma^* \Delta \Sigma^*)^{co} \qquad \textit{(K is } L_m(G)\textit{-closed)} \\
&= K'
\end{aligned}$$

□

Theorem 5.2 implies that if legal behavior E is $L_m(G)-$closed, then the legal behavior E' after removing events Δ from the event set Σ, will be $L_m(G')-$closed

In Algorithm 5.1, E_t is $L_m(G_t)-$closed and therefore, by Theorem 5.2, E_t' constructed in the loop will be $L_m(G_t')-$closed and step *LC* can be eliminated. After combining the results of Theorems 5.1 and 5.2, we simplify Algorithm 5.1 and arrive at Algorithm 5.2.

Algorithm 5.2

- ***Given:*** *DES plant* $G = (Q, \Sigma, \delta, q_0, Q_m)$ *and*

  legal behavior E

- ***Start:*** *Let* $G_t = (Q_t, \Sigma_t, \delta_t, q_0, Q_{m_t}) = G = (Q, \Sigma, \delta, q_0, Q_m)$ *and*

$E_t = SupR_G(E)$

If $Sup\mathscr{C}\overline{\mathscr{N}}(E_t) = \emptyset$, *then*

Problem has no solution; stop

End(If)

While $\Sigma_t \cap (\bigcup_{i=2}^{p} \Sigma_{\Phi_i}) \neq \emptyset$

For all $c \in \Sigma_t \cap (\bigcup_{i=2}^{p} \Sigma_{\Phi_i})$

Let $\Sigma_t' = \Sigma_t - \{c\}$.

Construct $G_t' = (Q_t', \Sigma_t', \delta_t', q_0, Q_{m_t}')$ *and* $E_t' = E_t - \Sigma^* c \Sigma^*$

If $Sup\overline{\mathscr{N}}(E_t') \neq \emptyset$, *then*

Replace G_t *and* E_t *with* G_t' *and* E_t'; *go to* ***E2***

End(If)

End(For)

- ***E1:*** $Sup\overline{\mathscr{N}}(E_t)$ *provides the solution and* $\Sigma_t \cap \Sigma_\Phi$ *identifies a minimal set of controllers; stop*

- ***E2:*** *continue*

End(While)

In general case, unlike the $L_m(G)-$closedness, the normality property is not necessarily preserved after deleting some controller switching events. The following counter example explains that the normality of $\overline{K}$ does not imply the normality of $\overline{K'} = \overline{K - \Sigma^* \Delta \Sigma^*}$.

Example 5.4 *Consider a problem with three plants* $\{P_1, P_2, P_3\}$. *Assume K as shown in Figure 5.7 is the legal behavior. The reader can verify that* $\overline{K}$ *is normal. Now let* K' *be the new legal behavior obtained by removing* c_{32} *from the set of controller commands (Figure 5.7).* $\overline{K'}$ *is not normal since for instance,* $p_2\sigma_u c_2 \in \overline{K'}$, $p_3\sigma_u c_2 \in P^{-1}P(p_2\sigma_u c_2) \cap L(G')$ *but* $p_3\sigma_u c_2 \notin \overline{K'}$.

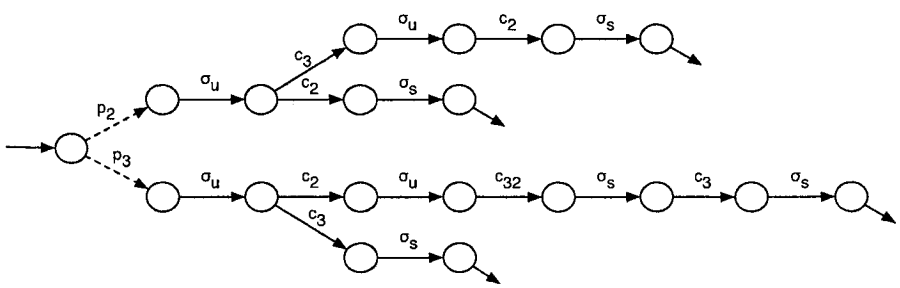

Figure 5.7: Specification K for Example 5.4

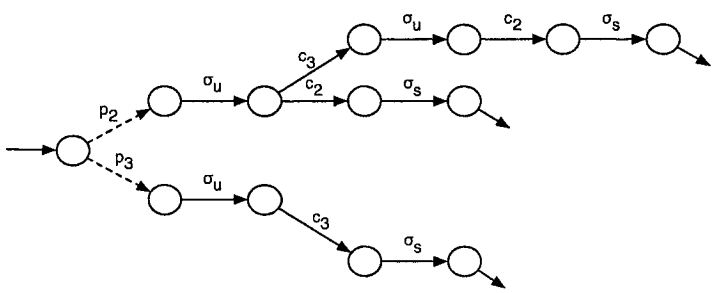

Figure 5.8: Specification K' for Example 5.4

Example 5.5 *Let us find a minimal set of controllers for the problem in Example 5.1. The design specification is to have a maximum of one unstable switching for the given set of plants*

and controllers. By using the same algorithm for finding the switching path from initial pair of plant and first layer stabilizing controller (P_1, C_1) *as used in Example 5.1, we arrive at controllers* C_1, C_2, C_3, C_4, C_5, C_{234}, C_{34}. *Next, for the initial pair* (P_2, C_2) *we obtain the set* C_1, C_2, C_3, C_4, C_5, C_{345}, C_{34} *following the same algorithm. To prevent adding redundant controllers, whenever possible, we have tried to use the controller from the list for* (P_1, C_1). *For the initial pair* (P_3, C_3), C_{512} *and* C_{12} *needed to be added to the controller set. It can be seen that for the initial pairs* (P_4, C_4) *and* (P_5, C_5) *the available controller set is sufficient and no more controller needs to be added.*

Thus, the set of controllers is C_1, C_2, C_3, C_4, C_5, C_{234}, C_{34}, C_{345}, C_{512}, *and* C_{12}. *Now let us use Algorithm 5.2 to minimize this set of controllers. Let us consider removing* C_{345}. *It can be seen that if we build DES G with the initial pair of* (P_1, C_1), *and run Algorithm 5.2,* $Sup\overline{\mathcal{N}}(E_t) \neq \emptyset$. *Therefore, starting from* (P_1, C_1), *in order to lock onto the matching controller (if the plant model changes), the controller* C_{345} *is not required. However, we should use this algorithm with all the possible initial pairs of plants and controllers. Building G with the initial pair of* (P_2, C_2) *and running the algorithm result in,* $Sup\overline{\mathcal{N}}(E_t') = \emptyset$ *and accordingly, the controller* C_{345} *cannot be removed from the controller set.*

If we follow the same procedure with different initial pairs, we will conclude that only C_{34} *can be removed from the controller set to come up with a minimal set of controllers. This can be justified in the following way. In the switching path, whenever it is needed to switch to*

C_{34} from the upper layer (layer 3), another controller in layer 3 can be used instead of C_{34}. For instance, suppose the initial pair is (P_1, C_1). The plant model changes and we first switch to C_{345}. If C_{345} results in instability, the plant model must be P_2. If C_{345} result in stability, then to see whether the plant model is P_5, the controller can switch to C_{34}. In this case, since we know P_2 is not the current plant model, the system can switch to C_{234} (instead of C_{34}). Here instability would imply P_5 and stability would point to P_3 and P_4 as the possible for the current plant models.

As shown in Example 5.5, switching to the lower layers is not always the best choice to keep the controller set minimal. In this example, considering switching between the controllers at the same layer as an alternative helps to reduce the number of controllers.

Remark 5.1 *As the previous example shows, in problems where we wish to limit the number of unstable switchings to one and controllers are chosen following the procedure suggested in [11], if there exists any controller, say C_{red}, at layer i that stabilizes plants that are stabilized by two controllers at layer $i+1$, then C_{red} can be removed from the controller set. In the example, C_{34} at layer 2, stabilizes P_3 and P_4. These plants are stabilized by both C_{234} and C_{345} at layer 3. Thus, C_{34} can be removed from the controller set.*

This chapter addresses two major problems in switching control. First, it provides a method to check whether there exists a switching policy for a given set of controllers and plants, and design specifications. Secondly, having a set of controllers that can satisfy the specifications, it finds a minimal set of controllers. The above tasks are accomplished using the computational procedure for supervisory control of DES.

Chapter 6

Conclusion

6.1 Summary

In this thesis, we show that the problem of designing a switching policy for a multi-layered switching control system can be converted into a supervisory control problem of a Discrete Event System (DES).

After briefly reviewing the theory of Supervisory Control of DES and Single-layered and Multi-layered switching control systems in Chapters 2 and 3, in Chapter 4, we introduce

a method to reformulate the switching control problems as a supervisory control problem of DES. In fact, this chapter contains one of the contributions of the thesis. To our knowledge, formulating the design of switching policy as a DES supervisory control problem has not been done previously in the literature. The described method is general enough to be applied to almost all single-layer and multi-layer switching problems. We demonstrate the transformation method with a running example.

Chapter 5 contains the main results of the thesis. Two main problems are addressed here. First a method is provided to verify whether a switching policy exists to satisfy the design specifications. If so, a minimally restrictive switching policy is designed. Next, an iterative algorithm is introduced to find a minimal controller set that can satisfy the design specifications. We show that in the supervisory control problem considered in this thesis, limitation on event observation is the factor that essentially restricts supervisory control. In other words, once observation limitations are respected, limitations on control will be automatically satisfied. We use the above result to simplify our iterative algorithm for finding a minimal controller set.

6.2 Future Research

In this section, we discuss directions for future research.

- In our study of designing switching policies in Chapter 4, we mentioned that typically it is assumed once a plant model changes, the plant model remains unchanged for a sufficiently long time (Assumption 4.2). In order to generalize our solution, we may attempt to relax this assumption. This assumption is also used in the study of switching systems at the continuous-time level. We expect that relaxing this assumption requires progress both the continuous-time as well as discrete-event levels of abstraction.

- One of the main results of this thesis is the proposed iterative algorithm for obtaining a minimal control set. It would be useful to develop a software to automate the implementation of the algorithm.

- In Chapter 5, we studied the issue of problem solvability for the design of switching policy. For a given set of controllers and design specifications, the result of this test may be negative. It would be interesting to develop an algorithm to find and add new controllers to the set of controllers to render the problem solvable. This basically corresponds to developing a bottom-up algorithm for finding a controller set for which a switching policy satisfying the specifications exists. Note that the algorithm proposed

in this thesis for obtaining a minimal controller set is a top-down algorithm.

- Finally, the minimal controller set in general is not unique. It would be useful to develop an algorithm that can find a minimal controller set that has the minimum number of controllers (i.e., the controller set with minimum cardinality).

Bibliography

[1] A. G. Aghdam and E. J. Davison. Pseudo-decentralized switching control. *Automatica*, 39(2):317–324, Feb. 2003.

[2] J. Balakrishnan and K. S. Narendra. Intelligent control using fixed and adaptive models. *Proceedings of the American Control Conference*, pages 597–601, Jun. 1995.

[3] D. Borrelli, A. S. Morse, and E. Mosca. Discrete-time supervisory control of families of two-degrees-of-freedom linear set-point controllers. *IEEE Transactions on Automatic Control*, 44:178–181, Jan. 1999.

[4] M. S. Branicky. Analyzing continuous switching systems: Theory and examples. *Proceedings of the American Control Conference*, 3:3110–3114, Jul. 1994.

[5] M. Chang and E. J. Davison. Control of unknown systems using switching controllers:

the self-tuning robust servomechanism problem. *Proceedings of the 33rd IEEE Conference on Decision and Control*, 3:2833–2838, Dec. 1994.

[6] M. Chang and E. J. Davison. Control of unknown systems using switching controllers: An experimental study. *Proceedings of the American Control Conference*, 3:2984–2989, Jul. 1994.

[7] M. Chang and E. J. Davison. Adaptive switching control of lti mimo systems using a family of controllers approach. *Automatica*, 35(3):453–465, Mar. 1999.

[8] R. Cieslak, C.Desclaux, A.S. Fawaz, and P. Varajya. Supervisory control of discrete-event processes with partial observation. *IEEE Transactions on Automatic Control*, 33:249–260, 1988.

[9] S. Hashtrudi-Zad, M. Moosaei, and W.M. Wonham. On computation of supremal controllable, normal sublanguages. *Systems & Control Letters*, 54(9):871–876, Sept. 2005.

[10] J. P. Hespanha, D. Liberzon, and A. S. Morse. Hysteresis-based switching algorithms for supervisory control of uncertain systems. *Automatica*, 39:263–272, Feb. 2003.

[11] I. Karuei, N. Meskin, and A.G. Aghdam. Multi-layer switching control. *Proceedings of the American Control Conference*, pages 4772–4777, 2005.

[12] E. B. Kosmatopoulos and P. A. Ioannou. A switching adaptive controller for feedback linearizable systems. *IEEE Transactions on Automatic Control*, 44(4):742–750, Apr. 1999.

[13] F. Lin and W.M. Wonham. Decentralized supervisory control of discrete-event systems. *Information Sciences*, 44:199–224, 1988.

[14] F. Lin and W.M. Wonham. On observability of discrete-event systems. *Information Sciences*, 44:173–198, 1988.

[15] B. Martensson. The order of any stabilizing regulator is sufficient a priori information for adaptive stabilization. *Systems & Control Letters*, 6(2):87–91, 1985.

[16] D. E. Miller and E. J. Davison. Adaptive control of a family of plants. *In D. Hinrichsen,& B. Martensson(Eds.), Control of uncertain systems: Proceedings of International Workshop, Berman , West Germany ,Progress in Systems and Control theory*, 6:197–219, 1989.

[17] D. E. Miller and E. J. Davison. An adaptive controller which provides an arbitrarily good transient and steady-state response. *IEEE Transactions on Automatic Control*, 36(1):68–81, Jan. 1991.

[18] D. E. Miller and E. J. Davison. The self-tuning robust servomechanism problem. *IEEE Transactions on Automatic Control*, 34(5):511–523, May. 1989.

[19] Daniel E. Miller and E. J. Davison. An adaptive controller which can stabilize any stabilizable and detectable lti system. *In C.I. Byrnes, C. F. Martin & R. E. Saek,(Eds.), Proceedings of the Eighth International Symposium on Mathematics of Network and Systems,AZ*, 1988.

[20] D.E. Miller and E.J. Davison. An adaptive tracking problem. *International Journal of Adaptive Control Signal Processing., vol.6*, pages 45–63, 1992.

[21] D.E. Miller and E.J. Davison. An adaptive tracking problem with a control input constraint. *Automatica*, 29:877–887, Jul. 1993.

[22] A. S. Morse. Recent problems in parameter adaptive control. *In I. D. Landau, editor, CNRS Colloquium on Development and Utilization of Mathematical Models in Automatic Control*, pages 733–740, 1983.

[23] A. S. Morse. Supervisory control of families of linear set-point controllers. *Proceedings of the 32nd IEEE Conference on Decision and Control*, 2:1055–1060, Dec. 1993.

[24] A. S. Morse. Supervisory control of families of linear set-point controllers–part 2: Robustness. *IEEE Transactions on Automatic Control*, 42(11):1500–1515, Nov. 1997.

[25] A. S. Morse. Supervisory control of families of linear set-point controllers-part 1: Exact matching. *IEEE Transactions on Automatic Control*, 41(10):1413–1431, Oct. 1996.

[26] A. S. Morse, D. Q. Mayne, and G. C. Goodwin. Applications of hysteresis switching in parameter adaptive control. *Proceedings of the 30th IEEE Conference on Decision and Control*, 1:767–772, December 1991.

[27] A. S. Morse, D. Q. Mayne, and G. C. Goodwin. Applications of hysteresis switching in parameter adaptive control. *IEEE Transactions on Automatic Control*, 37:1343–1354, Sep. 1992.

[28] K. S. Narendra and J. Balakrishnan. Improving transient response of adaptive control systems using multiple models and switching. *IEEE Transactions on Automatic Control*, 39(9):1861–1866, Sep. 1994.

[29] P.J. Ramadge and W.M. Wonham. Supervisory control of s class of discrete event processes. *SIAM Journal on Control and Optimization*, 25:206–230, 1987.

[30] W.M. Wonham. Supervisory control of discrete-event systems. *Systems Control Group, Edward S. Rogers Sr. Dept. of Electrical and Computer Engineering, University of Toronto, Canada; available at http://www.control.utoronto.ca/DES*, 2004.