

Structural Decentralised Control of Concurrent Discrete-Event Systems

Sang-Heon Lee

B.Sci Inha Univ. / M.EngSci Univ. of New South Wales

September 1998

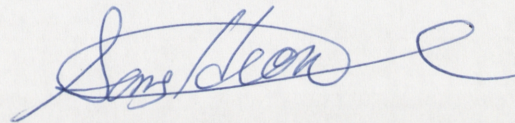
*A thesis submitted for the degree of Doctor of Philosophy
of The Australian National University*

Department of Systems Engineering
CRC for Robust and Adaptive Systems
Research School of Information Science and Engineering
Australian National University

Statement of Originality

These doctoral studies were conducted with Professor John. B. Moore as supervisor, and Dr Kai Wang and Robert E. Mahony as advisors.

The work presented in this thesis is the result of original research, and has not been submitted for a higher degree in any other university or educational institution.



Sang-Heon Lee

September 1998

Acknowledgements

I wish to thank firstly my supervisor Professor John Moore for his support and encouragement. I am specially grateful to Dr Kai Wong for his ideas, comments, many enjoyable discussions and his inspiration on the subject of this thesis. I also would like to thank Dr. Robert Mahony and Professor Iven Mareels for their ideas and comments on the areas in adaptive control. I am also grateful to Leonardo Kammer and Robert Orsi for their friendship and good advice on LaTeX and Matlab. With them, I feel more comfortable during my study.

I wish to thank Professor Murray Wonham from the University of Toronto for the TCT software which is used for synthesising supervisors throughout this thesis. I also would like to thank Dr. Jayantha Katupitiya from University of New South Wales, my master degree supervisor, for his encouragement and good advice. I would like to acknowledge the financial support from ANU graduate school and CRAsys.

I also would like to thank my family in Korea, especially my brother Sang-Tag Lee, for their spiritual support. I wish to thank my wife Eun-Joo Chong Lee for her persistent support and love. Finally, without smiles and endless implicit encouragement of my daughter, Nara Jee-Hyong, and my son, Joshua Min-Hyong, this thesis surely would not exist. Their understanding, support and especially patience are the most important stimuli during my study.

Abstract

In this thesis, within the framework of supervisory control theory, structural decentralised control of DES is investigated. Firstly, we have found two structural conditions, under which the distributivity of the control synthesis operator over the synchronous composition holds. That is, under these structural conditions, for a set of local requirements the concurrent actions of local decentralised controllers will achieve the global control objective and the control for one system does not cause blocking in the other subsystems. By achieving these two structural conditions, there may be an exponential saving of the computational efforts. Secondly, we have shown that under similar structural conditions, a coordination scheme can be used to solve some rescheduling problems among local plants. Thirdly, we have developed algorithms to modify the system structure, if that is possible, so that the resultant systems will possess the desired structural properties. Finally, we illustrate our result with a cleaning-in-place (CIP) process for a multipurpose, multiproduct batch plant. Using our results, a decentralised solution for the control of the CIP process is obtained.

Appendix contains the result of the first year of my degree, which is not related to the main subject of this thesis. In this appendix, a non-linear approach of model reference adaptive control is presented. Using a non-Euclidean gradient descent algorithm with respect to a Riemannian metric, we have shown that how a Riemannian metric can be chosen so that the modelled plant dynamics do in fact match the true plant dynamics. Simulations show that the proposed scheme can achieve faster asymptotic convergence of parameters when compared to a traditional model reference adaptive control scheme using the classical sensitivity derivatives (Euclidean gradients) for the descent algorithm.

Contents

1	Introduction	1
1.1	Discrete Event Systems(DES)	1
1.2	Literature Survey	3
1.3	Motivations	6
1.4	Contributions and Outline of the Thesis	8
2	Preliminaries	11
2.1	Formal Description of Discrete-Event Systems	11
2.1.1	Representation of DES	11
2.1.2	Closure and Nonblocking Properties	15
2.2	Projection and Products of DES	17
2.3	Controllability and Supervisory Control	21
2.3.1	Controlled DES and Nonblocking	21
2.3.2	Controllability	23
2.3.3	Specification Language	24
2.3.4	Supremal Controllable Sublanguage	24

2.4	Realisation of a Supervisory Control	25
2.5	Modular Control	27
2.5.1	Supervisor Conjunction and Nonconflicting	28
2.5.2	Modular Supervision	29
2.6	Decentralised Supervisory Control	30
2.6.1	Projection and Normality Condition	31
2.6.2	Local and Global Supervisors	32
2.6.3	Decentralised Supervision	33
2.7	Simple Facts	34
3	Structural Decentralised Control of Concurrent DES with Non-prefix-Closed Local Specifications	37
3.1	Introduction	37
3.2	Problem Formulation	39
3.3	Main Result	42
3.4	Example	53
3.5	Conclusions	63
4	Task Rescheduling using Coordinator	65
4.1	Introduction	65
4.2	A Coordination Scheme	68
4.3	Example	78
4.4	Conclusions	91

5	Structure Synthesis	93
5.1	Preliminaries	93
5.2	Arrangement for the Shared-Event-Marking Condition	94
5.3	Arrangement for the Mutual Controllability Condition	96
5.4	Example	113
5.5	Conclusions	117
6	Weakening the Shared-Event-Marking Condition	119
6.1	Preliminary	119
6.2	Problem Formulation	121
6.3	Using Bisimulation	123
6.4	Using H-Observers	126
6.5	Example	129
6.6	Conclusions	131
7	Application	133
7.1	Introduction	133
7.2	Decomposition of the CIP process	134
7.3	Water-Rinse Operation	140
7.3.1	Decomposition of the Components in the Water-Rinse Operation	141
7.3.2	Modelling and Synthesis for G_{pr}	142
7.3.3	Modelling and Synthesis for G_{pa}	144
7.3.4	Modelling and Synthesis for G_{fid} and G_{did}	148

7.3.5	Verification for Water-Rinse Operation	151
7.4	Detergent Preparation Operation	152
7.4.1	Decomposition of the Components in the Detergent Preparation	153
7.4.2	Modelling and Synthesis for G_{ap}	154
7.4.3	Modelling and Synthesis for G_{dnp}	160
7.4.4	Verification for Detergent Preparation Operation	161
7.5	Detergent-Rinse Operation	161
7.6	Conclusions	162
8	Conclusions and Future Works	163
	Appendix	179

List of Figures

2.1	A simple DES T_{30}	15
2.2	An example for closure and nonblocking	16
2.3	An example for product and synchronous composition	20
2.4	Supervisory control of DES	26
3.1	Decentralised control of concurrent DES	39
3.2	Generators for G_1 and G_2	53
3.3	A generator for $p_1 p_2^{-1}(L_2)$	54
3.4	Generators for E_1 and E_2	55
3.5	A simple batch reactor	56
3.6	DES models for the elementary components of the batch reactor	57
3.7	Physical constraints of the batch reactor	58
3.8	The specification for a centralised control of the batch reactor	58
3.9	Synchronisation flags for decentralised control of the batch reactor	59
3.10	Local specifications for decentralised control of the batch reactor	61
3.11	Modified local specification for G_2	63

4.1	An example of coordination	67
4.2	A coordination scheme	69
4.3	Modified batch reactor	79
4.4	DES models for the elementary components of the plant G_1	80
4.5	DES models for physical constraint and synchronisation flag for the plant G_1	81
4.6	Specification E_1 for the plant G_1	83
4.7	DES models for the elementary components of the plant G_2	84
4.8	Synchronisation flag (flag_2) for the plant G_2	85
4.9	Specification E_2 for the plant G_2	86
4.10	DES models for the elementary components of the plant G_3	87
4.11	DES models for physical constraint and synchronisation flag for the plant G_3	87
4.12	Specification E_3 for the plant G_3	88
4.13	Processes for the batch reaction	89
4.14	A coordination plant G_h	90
5.1	Example for controllability of G_a^s and G_b	98
5.2	Flow chart for Algorithm I: $G_f = F_1(G_a, G_b)$	100
5.3	Flow chart for Algorithm II: $(G'_1, G'_2) = F_2(G_1, G_2)$	103
5.4	Procedure of Algorithm II	105
5.5	Procedure of Algorithm III	107
5.6	Waste neutralisation plant and DES models for its components	109
5.7	Algorithm for arranging mutual controllability using κ operator	111

5.8	Algorithms for arranging mutual controllability	112
5.9	Example: arrange the system for the shared-event-marking condition	114
5.10	Example: arrange the system for the mutual controllability condition	118
6.1	Simplified batch reactor	129
6.2	DES models for W_1 and synchronisation flags	130
6.3	DES models for local specifications	130
6.4	Behaviours of the local supervisors	131
7.1	Schematic diagram of the batch process	135
7.2	DES models for the elementary components of CIP process I	137
7.3	DES models for the elementary components of CIP process II	138
7.4	DES models for the elementary components of CIP process III	139
7.5	Sequential control of the CIP process	140
7.6	Water-Rinse operation	141
7.7	Synchronisation flags for G_{pr}	143
7.8	Specifications for G_{pr}	144
7.9	Synchronisation flags for G_{pa}	145
7.10	Physical constraints of G_{pa}	146
7.11	Specification for G_{dpa} (E_{dpa})	147
7.12	Specification for G_{fpa} (E_{fpa})	149
7.13	Synchronisation flags for G_{fid} and G_{did}	150
7.14	Specifications for G_{fid} and G_{did}	150

7.15 Detergent Preparation operation.	152
7.16 synchronisation flags for G_{ap}	154
7.17 Physical constraints of G_{cr}	155
7.18 Physical constraints of G_{hc} (G_{cs} and G_{he})	156
7.19 Specification for G_{cr} (E_{cr})	157
7.20 Specification for G_{cs} (E_{cs})	158
7.21 Specification for G_{he} (E_{he})	159
7.22 Synchronisation flag for G_{dnp}	160
7.23 Specification for G_{dnp} (E_{dnp})	160
8.1 An example for building a better structure	166
A.1 System block diagram	182
A.2 Parameter estimates evolution plot	198
A.3 Error($\bar{y} - y$) plot	198
B.1 Subspace given by $\text{sp}\{E_1, g(k)\}$	201

Chapter 1

Introduction

1.1 Discrete Event Systems(DES)

Conventional control systems are usually described by differential equations, where the changes of underlying (continuous) state variables are continuously specified over time. However, over recent decades there is a growing need for dynamical models of systems whose behaviour is characterised by asynchronous, discrete and qualitative changes of state values with abrupt occurrences of events over time rather than by ticks of a clock. Unlike the conventional *time-driven* system, the micro-changes between event occurrences have no visible effects on the system, thus they are ignored. A useful example is a manufacturing plant with several machines to process different parts for a final product. This system may be adequately described by ‘start processing part A’, ‘transfer part A to an assembly line’ and ‘when machines A and B breakdown at the same time, then fix machine B first before fixing machine A’, etc. The main parts of such sentences are phrases: *start-processing-part*, *transfer-part*, *machine-brokedown*, *machine-fixed*, etc. These phrases indicate the occurrences of certain discrete events without mentioning micro-changes such as ‘the amount of metal cut’. So, the fundamental difference between these systems and the conventional control systems is that they only describe the orderly changes of behaviours but indicate no information about the real time at which the changes occur. This feature leads us to the new area of dynamic systems, called ‘discrete event

systems'. Abstractly, *Discrete Event System* (DES) is a dynamic system that evolves according to the abrupt occurrences of physical events at possibly unknown irregular time intervals. DES modelling is finding more applications on computer networks, automated manufacturing systems, intelligent communication networks, and air traffic control systems, etc.

In the past, DES has been sufficiently small so that intuitive or ad-hoc solutions to various problems have been considered adequate. But due to the rapid application of computer technology to these systems, the significant increase of complexity requires more formal methods for the analysis and design of DES. The developments of DES methods vary due to the many areas in which DES arises and the different aspects of behaviours in each area. The most basic methods are based on the assumption that ignores the time of occurrences of events and considers only the order in which they occur. This leads to 'logical DES models'. Some examples of this approach are finite state machine [RW87a, RW87b, WR87, CDFV88, RW89, WH91], or Petri-nets [Pet81, Den88, IH87, HK90]. However, in some applications the timing information is crucial, and must be included in the model. This leads to 'timed model.' This can be further classified as nonstochastic and stochastic models. The formalisms belong to this class are timed Petri-nets [DA94], the max-algebra [CDQV85, CMQV89, BCOQ92], timed finite state machine [OW90, BW94], queueing networks [Haj84, LK84, RVW82], and perturbation analysis [HC83, GG90, HC91], etc.

Each of the above approaches has its own problem of interests and limitations. So no method can satisfy all problems of the modelling and analysis of DES. In this work we focus on the approach using finite state machine initiated by Ramadge and Wonham [Ram83, RW87b, RW89] ¹. Their approach has one important feature. Since it treats the concept of open-loop plant and the feedback control separately, it allows us to evaluate and compare the effect of different control policies on the behaviour of the 'uncontrolled' system. Hence it permits the formulation and solution of a variety of control synthesis problems.

¹This approach is usually called the 'Supervisory Control Theory' of discrete event systems.

1.2 Literature Survey

The Supervisory Control Theory of DES was proposed by Ramadge and Wonham [RW87b, RW87a, WR87]. Briefly, in this approach, a DES to be controlled (a plant) is modelled as an automaton, called a *generator*. This generator is interpreted as a device that executes its state transitions spontaneously and outputs the corresponding transition or event label. In this way, the DES is said to generate a set of finite sequences of event labels (called *languages*). The behaviours of a DES are modelled by the languages that it generates. The control is implemented by disabling a subset of events and hence preventing them from occurring using an external controller (called a *supervisor*). A control requirement for this plant DES is given by the language generated by another automaton. This language is often called a *specification* language. The idea is to synthesise a controller which enables or disables events depending suitably on past behaviours of the generating plant DES so that the behaviour of the resulting closed-loop system (the language generated by the plant DES under supervision) could be made to satisfy the given requirement, i.e., the closed-loop behaviour is a subset of the specification language. It is often possible to construct a supervisor to meet the specification in an ‘optimal’, that is, minimally restrictive, fashion using the concept of the supremal controllable sublanguage. Conditions for the existence and method for the calculation of such a supervisor are given by Wonham and Ramadge [RW87b, WR87].

Computational problems in DES are usually complex. The computational complexity to synthesise supervisors is polynomial in terms of state sizes of the generators representing the plant behaviours and specifications (see [WR88]). However, since the system is usually composed of a number of components, the effort to compute a supervisor increases exponentially with the number of components [Ram88, RW89]. To overcome this problem, within this framework, modular control [RW87a, WR88], decentralised control [LW88a, CDFV88, WH91, RW92], and hierarchical control [ZW90] have been proposed and studied.

Modular approach in this framework provides a way to synthesise supervisors more efficiently. In [RW87b], a modular approach to the synthesis of feedback controls for maintaining logical invariants is investigated. A companion paper [WR88] examines the modular synthesis of supervisors when the control task is split into several subtasks. For each subtask, a subcon-

troller is designed using the existing theory, and the resultant subcontrollers are combined to form a solution for the original problem. To ensure that the supervisor that is synthesised modularly is nonblocking, it is necessary to check that subcontrollers are *nonconflicting* [WR88].

In decentralised supervisory control, instead of a single global supervisor, there are several local supervisors that operate concurrently. Each local supervisor observes and controls only a subset of the events of the global system. In [CDFV88], a necessary and sufficient condition was introduced to guarantee that the decentralised control scheme achieves the global optimal behaviour. Authors in [LW88a] consider the following situation in decentralised control: specifications are given on local models of a global process and supervisors are synthesised locally. The problem of interest is to find conditions under which the concurrent operations of these local supervisors are equivalent to a global supervisor synthesised for the overall requirements. Sufficient conditions for the decentralised supervisors to obtain the same optimal behaviour as in the centralised case are presented. Decentralised supervisory control is further investigated in [RW92] by considering problem formulations that model systems whose specifications are given as global constraints but whose solution is described by the local controllers. In [WH91] authors investigate decentralised control in a more structural situation, namely when the global system G is composed of a number of subsystems G_1, G_2, \dots, G_n , operating concurrently. Suppose that for each subsystem there is a local supervisor that observes and controls only the corresponding local events. For a global specification on G , a necessary and sufficient condition is given for the concurrent controls of a set of local supervisors on the G_i 's to obtain the optimal behaviour of a global supervisor. Some guidelines for the construction of decentralised solutions from centralised ones are proposed in [KW95]. Authors in [LW90, Lin91, WW96c] introduce a concept of coordination. After decentralised local supervision has been established, a second or 'higher' level of supervision is added to supervise the coordination among lower level subsystems. In [WvS96], authors investigate a problem of existence of communication channels between two decentralised supervisors for implementing a supervisory control.

Finally hierarchical aspects have been considered within the RW framework [ZW90]. The concept is that the process model is split into two hierarchical layers, with communicating supervisors applied to both. The top layer (the manager) views an abstracted process model derived from the bottom layer (the process) according to a set of *vocalised* states. High level

control instructions are passed down to an agent supervising the actual process. The concept of *hierarchical consistency* is introduced by which the information sent up from the process is timely and sufficiently detailed for various critical low level situations to be distinguished.

Control under partial observation is introduced in [LW88b, CDFV88]. This is the situation in which the supervisor does not see all the events of the plant but only a proper subset of *observable* events. Events in the system model would not be observable by the supervisor due to the absence of sensors or due to limitations on communication. For the generalisation of the controllability theorem with partial observation, the concept of the *observability* is introduced in [LW88b].

Recently, a control theory of timed discrete event systems (TDES) with its own control mechanism is developed in [Bra93, BW94]. The passage of time is modelled with a special event *tick*, representing the tick of a clock. In addition to the provision that certain events are controllable, another means of control is 'forcing'; a forcible event can be used (by the supervisor) to preempt a *tick* event and hence models the situation in which an action is forced to occur within certain time limits. Even though the control mechanism is somewhat different in the time setting, the class of controllable languages shares with the standard theory certain key algebraic properties such as closure under (arbitrary) union. Brandin and Wonham [BW94] generalised the concepts of controllability and maximally permissive supervisory controls to timed DESs. With time modelled explicitly, the computational complexity of TDES is inevitably increased. Thus the architectural approaches mentioned above could be used in the time setting to render the complexity problem more manageable. To this end, the modular, decentralised and hierarchical control were extended to the timed framework in [Bra93, WW96b].

More recently, other control problems have been investigated. In the papers [Ove94, YLA95, HL96, HL98], the authors extend supervisory control theory for the control of systems modelled as nondeterministic finite state machines. Nondeterministic finite state machines may result from the projection of the continuous parts of a hybrid process model onto a discrete model. Li and Wonham [LW93, LW94] consider the control of Vector Discrete Event Systems (VDES). VDES is the system in which states are represented by vectors with integer components and state transitions by integer vector addition. Also, limited lookahead policy

[CLL92] and adaptive and robust supervisory control [Lin93] are studied in this framework.

An application of supervisory control theory employing automated toy trains and cranes is reported by Leduc [Led96]. Model reduction techniques are developed to solve the state explosion problem so that controllability of modular supervisors can be established. Lauzon et al. [LMMB96] report another application to a laboratory scale robotic workcell. Importantly, in this work, the authors present an automatic generator of ladder logic code from finite state machines. Also other various applications of supervisory control theory have been addressed in the areas of database management [Laf88], communication protocol [RW90], workcell control [BBW91], robotics [KB94, RSR95], and manufacturing systems [BHG⁺90, WBS96].

1.3 Motivations

Our basic motivation is the same as that of some other works in modular and decentralised control (e.g., [WR88, LW88a, WH91]), namely, to investigate methods to overcome the complexity problem. However, our approach is different from these works. In all the existing works, the conditions, which guarantee that decentralised control achieves the same behaviour as centralised control would, are specification-dependent. That is, the conditions have to be verified for each given specification. In our approach, we are seeking conditions which are ‘not specification-dependent’. That is, once those conditions have been verified, then decentralised control is achieved for a set of specifications. We will call this approach ‘structural decentralised control’ to distinguish it from other works. A similar approach [Als96] is also taken in a slightly different control framework, called Procedural Control Theory [San96].

One of the advantages of this structural approach is the following. Assume that the effort of computing decentralised supervisors is less than that of synthesising a centralised supervisor. In a specification-dependent approach, the effort of verifying the conditions is incurred for each new specification. To justify this approach, one needs to ensure that the effort of designing decentralised supervisors and verifying the conditions is less than that of computing a centralised supervisor. For example in [WR88], such an analysis is carried out. But in other works in decentralised control, it is not clear whether one can generally achieve savings in

computational effort. However, in the structural approach the effort of verifying or establishing the conditions is incurred only once. Thus, if the established structure is used for a sufficiently large number of specifications, structural decentralised control will generally provide savings in computational effort.

In the paper [WR88], the authors consider the situation in which the legal specification can be decomposed into an intersection of partial specifications, and determine conditions under which it is possible to synthesise the appropriate control for the overall specification in a modular fashion. If we use an operator, denoted κ in this thesis, to represent the process of synthesising a least restrictive supervisor, then they show that under certain conditions the distributivity of the control synthesis operator κ over the language intersection holds. Here we are seeking conditions under which an equivalent property in decentralised control framework, i.e, the distributivity of κ over the synchronous composition of languages, holds. Consider a number of DES plants G_1, G_2, \dots, G_n . Let $\kappa(G_i, E_i)$ denote the function of synthesising the optimum supervisor for G_i and a given specification DES E_i . So the main question of this thesis is:

Under what structural conditions, i.e., the conditions on the G_i 's, is it true that for any specification E_i on G_i ($i = 1, 2, \dots, n$),

$$\kappa(G_1 \parallel G_2 \parallel \dots \parallel G_n, E_1 \parallel E_2 \parallel \dots \parallel E_n) = \kappa(G_1, E_1) \parallel \kappa(G_2, E_2) \parallel \dots \parallel \kappa(G_n, E_n),$$

where \parallel denotes synchronous composition ?

Recall that the complexity of computing a supervisor increases exponentially with respect to the number of components in the plant. By achieving the above equation, one could have exponential savings in computational complexity.

Having obtained the conditions for the above question, the next natural question is:

For given G_i 's, how can the conditions be arranged ?

This thesis mainly addresses these two questions.

1.4 Contributions and Outline of the Thesis

The main contributions of this thesis are : we have

1. obtained structural properties to guarantee that the distributivity of the control operator κ over synchronous composition holds(Chapter 3).
2. investigated a coordination and decentralised control architecture (Chapter 4).
3. developed procedures to arrange the structural properties for distributivity (Chapter 5).

Chapter 2

This chapter contains a brief review of some basic concepts in a theory of supervisory control of discrete event systems that are used throughout this thesis.

Chapter 3

This chapter contains the main conceptual idea of this thesis. In Section 3.2, we formulate a problem of decentralised control of concurrent discrete event systems. We consider the system as the synchronous composition of a number of subsystems as in [WH91]. The local specification languages are generally non-prefix-closed. The problem is to find conditions under which the concurrent actions of decentralised supervisors will achieve the global optimality, and control of one system never causes blocking in the other subsystems. As we will see, this problem can be seen as that of achieving the distributivity of the control synthesis operator κ over synchronous composition. In Section 3.3, we establish two sufficient conditions on the system structure and their shared events so that decentralised control in the above sense is achieved, for any local specification languages closed relative to the marked behaviours of the subsystems. The first condition, called the shared-event-marking condition, is mainly concerned with ‘marking’ in subsystems. This condition roughly says that the states before the shared events in each subsystem should be marked. Intuitively this may mean that the subsystems have completed their respective tasks and are ready for the synchronisation. The second condition, called the mutual controllability condition, says that a subsystem is able to track any uncontrollable shared event that could occur in the other subsystem. As we will see in a more detailed anal-

ysis, by achieving these two structural conditions there may be an exponential saving of the computational efforts involved.

Chapter 4

In this chapter, we show how coordination can be used to solve some rescheduling problems among local plants. The coordination plant model consists of only the shared events among the subplants. For a specification on the coordination plant, the corresponding supervisor, called a coordinator, can be designed. We show that under structural conditions similar to the ones in Chapter 3, the combined actions of the coordinator with the existing local supervisors will satisfy some rescheduling requirements. For different tasks, different coordinators can be designed. Again we note that the conditions are structural. So, once established, the coordination architecture can be used for a number of tasks.

Chapter 5

In this chapter, we have developed procedures to modify the system structure, if that is possible, so that the resultant systems will possess the desired structural properties. For the shared-event-marking condition, we mark the state before a shared event if that state is not already marked. For the mutual controllability condition, uncontrollable shared events are selflooped to the necessary states.

Chapter 6

In some applications, we found that the shared-event-marking condition may be too ‘strong’, namely that too many states have to be marked. So, in this chapter, we develop two ways to relax this condition. Here we consider a global system consisting of two subsystems. In Section 6.3, we show that if local supervisors are bisimilar with each other and with the marked behaviour of the other plant, then decentralised control can be achieved. In Section 6.4, we describe how an observer property is used to relax the condition. In both cases, the mutual controllability condition still remains as a structural condition. However, the conditions that replace the shared-event-marking condition now become specification-dependent. We still allow local specification languages to be non-prefix-closed.

Chapter 7

In this chapter, we apply our results to a cleaning-in-place (CIP) process for a multipurpose, multiproduct batch plant. In most chemical processes, the model for each component is usually simple. However, the overall system generally consists of many components and thus is very complex. For example, the plant model for the CIP process has 5.0×10^8 states. Using our results, decentralised solution for the control of the CIP process is obtained.

Chapter 8

This chapter contains conclusions and suggestions for future research.

Appendix

In this appendix, the result of the first year of my degree, which is not related to the main subject of this thesis, is presented. Here, we develop a new non-linear approach to the design of adaptive control scheme based around the use of a non-Euclidean gradient descent algorithm with respect to a Riemannian metric. It is shown that how a Riemannian metric can be chosen so that the modelled plant dynamics do in fact match the true plant dynamics. The approach is demonstrated by a case study of a simple single-pole and no zero, linear, discrete-time plant. The performance of the proposed scheme is compared to a traditional model reference adaptive control scheme using the classical sensitivity derivatives (Euclidean gradients) for the descent algorithm. Simulations show that the scheme described in this appendix offers faster asymptotic convergence of parameters and more flexibility in the transient response of the closed-loop system. The key contributions are: the formulation of the gradient descent algorithm in such a way as to incorporate the true plant dynamics and the development of a criterion and a method to determine suitable positive definite matrices for the adaptation mechanism. Further work is required to investigate stability issues as well as the generalisation to continuous-time plants and more general system models.

Chapter 2

Preliminaries

In this chapter we recall some basic concepts in supervisory control theory of discrete-event systems using automata and formal language models, pioneered by Ramadge and Wonham [RW87b, RW89]. In Section 2.1, the basic DES model is presented. In Section 2.2, we recall the ideas of product and synchronous composition for combining several DES into a single DES. In Section 2.3, the concept of controllable languages and some basic control problems are presented. In Section 2.4, a way of implementing the supervisory control using automata is discussed. In view of complexity in the control of DES, two solutions, modular and decentralised control syntheses, are presented in Section 2.5 and Section 2.6, respectively. Finally, in Section 2.7 we present some simple facts which will be used later in this thesis.

2.1 Formal Description of Discrete-Event Systems

2.1.1 Representation of DES

In the supervisory control approach [RW87b, Won96], an uncontrolled DES is modelled by an 5-tuple automaton, called a *generator*¹,

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

¹Also the term *finite state machine* or *deterministic finite automaton* is often used in the literature.

where Q is a set of states (we assume that Q is finite),

Σ is a finite set of *event* labels, often called an *alphabet*,

$\delta : Q \times \Sigma \longrightarrow Q$ is a (partial) *state transition function*,

$q_0 \in Q$ is the *initial* state,

$Q_m \subseteq Q$ is a set of *marker* states.

The transition function δ is generally a partial function, meaning that, for each $q \in Q$, $\delta(\sigma, q)$ is defined for a (possibly proper) subset $\Sigma_{\mathbf{G}}(q) \subseteq \Sigma$ that depends on q . We call $\Sigma_{\mathbf{G}}(q)$ the *active event set* at the state q of a given generator \mathbf{G} . Formally, a transition or event of \mathbf{G} is a triple of the form (q, σ, q') , where $\delta(q, \sigma) = q'$, and $q, q' \in Q$ are respectively the *exit state* and the *entrance state*, while $\sigma \in \Sigma$ is a event label. The event set of \mathbf{G} is the set of all such triples. A DES \mathbf{G} starts in the initial state q_0 , executes transitions according to δ , and produces a sequence of events. Events are considered to occur spontaneously, asynchronously (not clock-driven) and instantaneously. \mathbf{G} is thought to be nondeterministic in the sense that more than one event may be available for a given state. However, it is assumed to be deterministic in the sense that a distinct event exiting from a given state always have a distinct label. For brevity we shall often refer to ‘the event σ ’, meaning any or all events (transitions) that happen to be labelled by σ .

The behaviours of a process \mathbf{G} are modelled by sets of finite sequences of events generated by \mathbf{G} . Formally they are languages. Let Σ^+ be the set of all finite strings of event labels in Σ . Let $\Sigma^* = \Sigma^+ \cup \{\epsilon\}$, where $\epsilon \notin \Sigma$ represents the *empty string*, the string with length 0. Any subset of Σ^* is a *language* over Σ (see, e.g., p. 17 of [Lin96]). The transition function is inductively extended to strings as

$$\delta : Q \times \Sigma^* \longrightarrow Q$$

according to

$$\delta(q, \epsilon) = q$$

and

$$\delta(q, s\sigma) = \delta(\delta(q, s), \sigma)$$

whenever both $q' := \delta(q, s)$ and $\delta(q', \sigma)$ are defined.

The languages associated with \mathbf{G} are the *closed behaviour* of \mathbf{G}

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

and the *marked behaviour* of \mathbf{G}

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

The language $L(\mathbf{G})$ is the set of all possible finite sequences of events that \mathbf{G} can generate from the initial state q_0 . While $L_m(\mathbf{G}) \subseteq L(\mathbf{G})$ is a subset of these sequences that reach marker states, perhaps representing completed “tasks” carried out by the physical process that \mathbf{G} is intended to model. Note that there is no implication that the generating action by the generator halts after the completion of some marked sequences, i.e., it is not necessarily for the marker states of \mathbf{G} to be the “final” states.

Next, it is usually convenient to eliminate the states of \mathbf{G} which can never be reached from q_0 . The set of *reachable states*, denoted by $Q_r \subseteq Q$, is defined to be

$$Q_r = \{q \in Q \mid (\exists s \in \Sigma^*) \delta(q_0, s) = q\}.$$

DES \mathbf{G} is called *reachable* if $Q_r = Q$. A state q of \mathbf{G} is said to be *coreachable* if a marker state is reachable from q . Formally, the set of *coreachable states* $Q_{cr} \subseteq Q$ is defined by

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

The generator \mathbf{G} is *coreachable* if $Q_{cr} = Q$. If \mathbf{G} is both reachable and coreachable, it is called *trim*. Every transition structure has a unique trim structure [Eil74, Sec. 3.5].

To introduce control to DES \mathbf{G} , it is assumed that a subset of events can be disabled (prevented from occurring) and enabled (permitted to occur) by some control agent(s) whenever desired. These events $\Sigma_c \subseteq \Sigma$ are *controllable events*. The remaining events $\Sigma_u := \Sigma - \Sigma_c$ are *uncontrollable events*. It may be thought that since uncontrollable events cannot be prevented from occurring, they can be considered as always enabled. How a given event is chosen as an uncontrollable event usually depends on the designer’s point of view. However, some events in the following cases would be modelled as uncontrollable events: the event is inherently impossible to prevent from occurring (a failure event such as machine breakdown, accident),

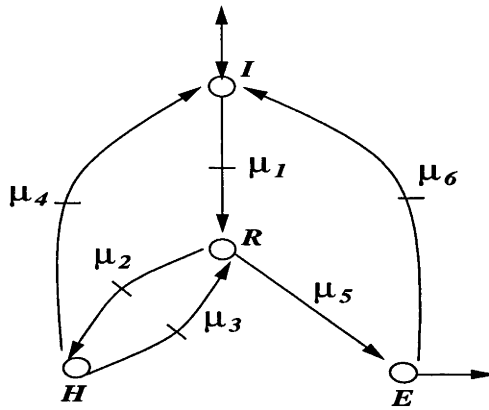
the event models some changes of sensor reading, or the event represents the expiration of a specified time period.

Remark 2.1 We explain how a generator can be used to represent a language. The starting point is that a generator is a device that generates a language according to a specified set of well-defined rules. The basic question is: Can we construct a finite state automaton (we call a generator) that generates a given language? The answer is ‘yes’ if the language is *regular* (for details see [Cas93, Chapter 2], [Lin96, Chapter 2]). That is if a language is regular, then it can be generated by a finite state automaton; and if a language is generated by a finite state automaton, then it is regular. Therefore, it is always possible to represent a regular language with a finite state automaton². \diamond

A DES can be represented by a transition graph in which the nodes of the graph represent the states and the directed branches (arrows) represent the events. In a transition graph, the initial state is labelled with an entering arrow ($\longrightarrow\circ$), and a marker state is labelled with an exiting arrow ($\circ\longrightarrow$). A double arrow ($\circ\longleftrightarrow$) represents that the initial state is also a marker state. An event is described by an arrow from an exit state to an entrance state. An arrow with a tick (\dashrightarrow) represents a controllable event, while an arrow without a tick (\longrightarrow) represents an uncontrollable event.

Example 2.1 Consider a simple DES representing a timer called **T30** as shown in Figure 2.1. The system is comprised of four states $Q = \{I, R, H, E\}$ for ‘idle’, ‘released’, ‘held’ and ‘expired’, respectively. The initial state is $q_0 = I$ and the marker states are $Q_m = \{I, E\}$. The controllable events are $\Sigma_c = \{\mu_1, \mu_2, \mu_3, \mu_4, \mu_6\}$, while the uncontrollable event is $\Sigma_u = \{\mu_5\}$. The operation of **T30** proceeds as follows. Starting from the initial state I , **T30** executes a sequence of events according to its transition graph. The event μ_1 is a controllable event. So if μ_1 is enabled by an external agent, **T30** can execute the transition (I, μ_1, R) which represents ‘release the timer’. In the state R , either the transition (R, μ_2, H) (representing ‘hold the timer’) or (R, μ_5, E) (representing ‘timer expired’) may occur. However, **T30** can select just

²Note that it is always possible to present any language by an automaton (maybe with a infinite state set) [CLO95]. However, if the language is regular (which is the case we consider in this thesis), it can be represented by a finite state automaton.

Figure 2.1: A simple DES **T30**

one of them. Note that since μ_5 is uncontrollable, this event is always possible. Since all the states are reachable and coreachable, **T30** is trim. The closed behaviour of **T30** is

$$L(\mathbf{T30}) = \{\epsilon, \mu_1, \mu_1\mu_2, \mu_1\mu_2\mu_3, \mu_1\mu_5, \mu_1\mu_5\mu_6, \mu_1\mu_2\mu_3\mu_5, \mu_1\mu_2\mu_4, \dots\}.$$

The marked behaviour is

$$L_m(\mathbf{T30}) = \{\epsilon, \mu_1\mu_5, \mu_1\mu_5\mu_6, \mu_1\mu_2\mu_3\mu_5, \mu_1\mu_2\mu_4, \dots\}. \quad \diamond$$

2.1.2 Closure and Nonblocking Properties

For two strings $s, u \in \Sigma^*$, s is a *prefix* of u if there is a string $v \in \Sigma^*$ such that $u = sv$. Let $H \subseteq \Sigma^*$ be an arbitrary language. The *prefix closure* of H , denoted \overline{H} , is the set of all prefixes of strings in H , formally defined by

$$\overline{H} = \{s \in \Sigma^* \mid \exists v \in \Sigma^* \text{ such that } sv \in H\}.$$

The language $H \subseteq \Sigma^*$ is *closed* if $\overline{H} = H$. Let $F \subseteq H \subseteq \Sigma^*$. Then, the language F is said to be *H-closed* if

$$F = \overline{F} \cap H.$$

Thus, a sublanguage F of H is *H-closed* if and only if any prefix of F which is also a string of H should be a string of F . Let \mathcal{F}_H be the set of all *H-closed* languages.

A system is said to be *blocked* if it can no longer complete its tasks. There are two typical types of blocking in this framework. One case is that if an automata G is reached a state

q where $\Sigma_{\mathbf{G}}(q) = \phi$ but $q \notin Q_m$, this is called *deadlock* because no further event can be executed. The other case is when there is a repeated cycle of unmarked states in \mathbf{G} without the events going out of the cycle. If the system falls into such a situation, this is called *livelock*. Even though the system can always execute an event, it can never complete its tasks. In both case, the system is not able to reach the marker states. The problem of avoiding blocking is an important issue in DES. The concept of *nonblocking* is introduced to capture this notion [RW87b]. Let $H \subseteq \Sigma^*$. Then a language F is *nonblocking* with respect to H if

$$\overline{F} = \overline{F \cap H}.$$

Namely, a language F is nonblocking with respect to H if any string in the closure of F can always be extended to a string in H within the closure of F . If $F \subseteq H$, then F is nonblocking with respect to H . It is also clear that if F is H -closed, then it is nonblocking with respect to H . We say that \mathbf{G} is *nonblocking* if $L(\mathbf{G})$ is nonblocking with respect to $L_m(\mathbf{G})$, i.e.,

$$L(\mathbf{G}) = \overline{L_m(\mathbf{G})}.$$

In other words, if \mathbf{G} is nonblocking then every reachable state of \mathbf{G} is coreachable.

The following example illustrates the concepts of closed languages and nonblocking.

Example 2.2 Let a generator \mathbf{G} be as displayed in Figure 2.2. Let H be the marked behaviour

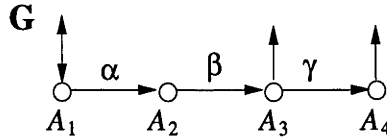


Figure 2.2: An example for closure and nonblocking

of \mathbf{G} , i.e. $H = \{\epsilon, \alpha\beta, \alpha\beta\gamma\}$. Then $\overline{H} = \{\epsilon, \alpha, \alpha\beta, \alpha\beta\gamma\}$. Since $H \neq \overline{H}$, the language H is not closed. To find the set of all H -closed languages, \mathcal{F}_H , let $F_1 := \{\epsilon, \alpha\beta\} \subseteq H$. Clearly $\overline{F_1} = \{\epsilon, \alpha, \alpha\beta\}$ and $\overline{F_1} \cap H = F_1$. So F_1 is a member of \mathcal{F}_H . Now let $F_2 := \{\alpha\beta\} \subseteq H$. Then $\overline{F_2} = \{\epsilon, \alpha, \alpha\beta\}$ and $\overline{F_2} \cap H = \{\epsilon, \alpha\beta\} \neq F_2$. So F_2 is not a member of \mathcal{F}_H . Similarly, the set of all H -closed languages can be found: $\mathcal{F}_H = \{\{\epsilon\}, \{\epsilon, \alpha\beta\}, \{\epsilon, \alpha\beta, \alpha\beta\gamma\}\}$.

To illustrate the concept of nonblocking, let $K_1 = \{\epsilon, \alpha\beta\}$. Hence $\overline{K_1} = \{\epsilon, \alpha, \alpha\beta\}$ and $\overline{K_1} \cap H = \{\epsilon, \alpha\beta\} = K_1$ (H -closed). Thus $\overline{K_1} \cap \overline{H} = \{\epsilon, \alpha, \alpha\beta\} = \overline{K_1}$ which shows that

K_1 is nonblocking with respect to H . To give a negative example, let $K_2 = \{\alpha\}$. Then $\overline{K_2} = \{\epsilon, \alpha\}$, $\overline{K_2} \cap H = \{\epsilon\}$, and $\overline{\overline{K_2} \cap H} = \{\epsilon\} \neq \overline{K_2}$. So K_2 is not nonblocking with respect to H . Since the string α in $\overline{K_2}$ cannot be extended to a string in H within $\overline{K_2}$, it is blocked at the state ' A_2 '. \diamond

2.2 Projection and Products of DES

It is often required to combine several DES into a single, more complex DES. Such cases arise naturally when the modelling of control problems involves the coordination or synchronisation of several DES. In this section, we describe two ways to combine DES. For this, firstly we introduce the union, intersection and concatenation of two languages $L(\mathbf{G}_1)$ and $L(\mathbf{G}_2)$ as follows, where both languages are subsets of Σ^* :

- Union: $L(\mathbf{G}_1) \cup L(\mathbf{G}_2) = \{s \in \Sigma^* \mid s \in L(\mathbf{G}_1) \text{ or } s \in L(\mathbf{G}_2)\}$,
- Intersection: $L(\mathbf{G}_1) \cap L(\mathbf{G}_2) = \{s \in \Sigma^* \mid s \in L(\mathbf{G}_1) \text{ and } s \in L(\mathbf{G}_2)\}$,
- Concatenation: $L(\mathbf{G}_1)L(\mathbf{G}_2) = \{s \in \Sigma^* \mid s = s_1s_2, s_1 \in L(\mathbf{G}_1) \text{ and } s_2 \in L(\mathbf{G}_2)\}$.

Then we define the natural projection. Let Σ_1 and Σ_2 be two event alphabets, not necessarily disjoint, i.e., $\Sigma_1 \cap \Sigma_2 \neq \phi$. Let $\Sigma = \Sigma_1 \cup \Sigma_2$. Define the *natural projection*

$$p_i : \Sigma^* \longrightarrow \Sigma_i^* \quad (\text{for } i = 1, 2),$$

according to

$$p_i(\epsilon) = \epsilon,$$

$$p_i(\sigma) = \begin{cases} \sigma & \text{if } \sigma \in \Sigma_i, \\ \epsilon & \text{otherwise,} \end{cases}$$

$$p_i(s\sigma) = p_i(s)p_i(\sigma),$$

for $s \in \Sigma^*$ and $\sigma \in \Sigma$. The action of p_i on a string $s \in \Sigma^*$ is just to erase all the elements σ of s which do not belong to Σ_i .

Example 2.3 Let $\Sigma_1 = \{\alpha, \beta\}$ and $\Sigma = \{\alpha, \beta, \gamma, \delta\}$. The natural projection $p_1 : \Sigma^* \longrightarrow \Sigma_1^*$ is given

$$\begin{aligned} p_1(\epsilon) &= \epsilon, & p_1(\alpha) &= \alpha, & p_1(\beta) &= \beta, \\ p_1(\gamma) &= \epsilon, & p_1(\delta) &= \epsilon, & p_1(s\sigma) &= p_1(s)p_1(\sigma), \end{aligned}$$

for $s \in \Sigma^*$ and $\sigma \in \Sigma$. For example, $p_1(\alpha\beta\gamma\alpha\delta) = p_1(\alpha)p_1(\beta)p_1(\gamma)p_1(\alpha)p_1(\delta) = \alpha\beta\alpha$. Notice that due to $p_1(\gamma) = p_1(\delta)$, we can no longer distinguish the events γ and δ when observing only the outputs of p_1 . \diamond

Now we introduce two ways of combining DES; one is the *product*, denoted \times , and the other is the *synchronous composition*, denoted $||$. The product is called *completely synchronous composition*, while the synchronous composition is often termed *parallel composition* [CLO95].

Consider two automata defined as

$$\begin{aligned} \mathbf{G}_1 &= (Q_1, \Sigma_1, \delta_1, q_{10}, Q_{1,m}), \\ \mathbf{G}_2 &= (Q_2, \Sigma_2, \delta_2, q_{20}, Q_{2,m}). \end{aligned}$$

Let the active event sets be, for $q_1 \in Q_1$ and $q_2 \in Q_2$,

$$\Sigma_{\mathbf{G}_1}(q_1) = \{\sigma \mid \delta_1(q_1, \sigma) \text{ is defined}\}, \text{ and } \Sigma_{\mathbf{G}_2}(q_2) = \{\sigma \mid \delta_2(q_2, \sigma) \text{ is defined}\}.$$

The *product* of \mathbf{G}_1 and \mathbf{G}_2 is given by

$$\mathbf{G}_1 \times \mathbf{G}_2 = \text{Rch}(Q_1 \times Q_2, \Sigma_1 \cap \Sigma_2, \delta_1 \times \delta_2, (q_{10}, q_{20}), Q_{1,m} \times Q_{2,m}),$$

according to

$$\delta_1 \times \delta_2((q_1, q_2), \sigma) = \begin{cases} (\delta_1(q_1, \sigma), \delta_2(q_2, \sigma)) & \text{if } \sigma \in \Sigma_{\mathbf{G}_1}(q_1) \cap \Sigma_{\mathbf{G}_2}(q_2), \\ \text{undefined} & \text{otherwise,} \end{cases}$$

where $\text{Rch}(\dots)$ means the reachable component of an automaton. In the product, only the transitions of two automata with common event labels are permitted to occur and they should occur simultaneously. It can be verified that the closed and marked behaviours of the product are

$$\begin{aligned} L(\mathbf{G}_1 \times \mathbf{G}_2) &= L(\mathbf{G}_1) \cap L(\mathbf{G}_2), \\ L_m(\mathbf{G}_1 \times \mathbf{G}_2) &= L_m(\mathbf{G}_1) \cap L_m(\mathbf{G}_2). \end{aligned}$$

The *synchronous composition* is defined by

$$\mathbf{G}_1 \parallel \mathbf{G}_2 = \text{Rch}(Q_1 \times Q_2, \Sigma_1 \cup \Sigma_2, \delta_{12}, (q_{10}, q_{20}), Q_{1,m} \times Q_{2,m}),$$

according to

$$\delta_{12}((q_1, q_2), \sigma) = \begin{cases} (\delta_1(q_1, \sigma), \delta_2(q_2, \sigma)) & \text{if } \sigma \in \Sigma_{\mathbf{G}_1}(q_1) \cap \Sigma_{\mathbf{G}_2}(q_2), \\ (\delta_1(q_1, \sigma), q_2) & \text{if } \sigma \in \Sigma_{\mathbf{G}_1}(q_1) \ \& \ \sigma \notin \Sigma_2, \\ (q_1, \delta_2(q_2, \sigma)) & \text{if } \sigma \in \Sigma_{\mathbf{G}_2}(q_2) \ \& \ \sigma \notin \Sigma_1, \\ \text{undefined} & \text{otherwise.} \end{cases}$$

In the synchronous composition, an event shared by both \mathbf{G}_1 and \mathbf{G}_2 occurs only when it is permitted by both automata simultaneously. The unshared events can occur whenever possible. In this sense, the two automata are ‘synchronised’ on the shared events. We can obtain the behaviours of $\mathbf{G}_1 \parallel \mathbf{G}_2$ from those of \mathbf{G}_1 and \mathbf{G}_2 as follows:

$$\begin{aligned} L(\mathbf{G}_1 \parallel \mathbf{G}_2) &= p_1^{-1}(L(\mathbf{G}_1)) \cap p_2^{-1}(L(\mathbf{G}_2)), \\ L_m(\mathbf{G}_1 \parallel \mathbf{G}_2) &= p_1^{-1}(L_m(\mathbf{G}_1)) \cap p_2^{-1}(L_m(\mathbf{G}_2)), \end{aligned}$$

where p_i^{-1} is the inverse image map of p_i ; formally, for $L_i \in \Sigma_i^*$ and $i = 1, 2$,

$$p_i^{-1}(L_i) := \{s \in (\Sigma_1 \cup \Sigma_2)^* \mid (\exists t \in L_i) p_i(s) = t\}.$$

Also, observe that we can define a corresponding synchronous composition of languages. With $L_i, L_{i,m} \subseteq \Sigma_i^*$ and p_i defined as above, let

$$\begin{aligned} L_1 \parallel L_2 &:= p_1^{-1}(L_1) \cap p_2^{-1}(L_2), \\ L_{1,m} \parallel L_{2,m} &:= p_1^{-1}(L_{1,m}) \cap p_2^{-1}(L_{2,m}), \end{aligned}$$

Finally, if $L(\mathbf{G}_i) = L_i \subseteq \Sigma_i^*$ for $i = 1, 2$, then one can think of \mathbf{G}_1 and \mathbf{G}_2 as generating $L_1 \parallel L_2$ ‘cooperatively’ by agreeing to synchronise the events with common labels. Also, events with different labels are assumed not to occur at the same time.

In the special case of $\Sigma_1 = \Sigma_2$, all transitions are forced to be synchronised and thus the synchronous composition can be reduced to the product. In the case of $\Sigma_1 \cap \Sigma_2 = \phi$, there are no synchronised transitions. Then the synchronous composition is called the *shuffle* of \mathbf{G}_1 and \mathbf{G}_2 . Thus the language of the shuffle of \mathbf{G}_1 and \mathbf{G}_2 over disjoint alphabets is the

language consisting of all possible interleaving of strings of $L(G_1)$ with the strings of $L(G_2)$. One can think of G_1 and G_2 as generating $L(G_1) \parallel L(G_2)$ by independent and asynchronous generation of $L(G_1)$ and $L(G_2)$ respectively.

Example 2.4 Let $\Sigma_1 = \{\alpha, \beta\}$ and $\Sigma_2 = \{\beta, \lambda\}$. Thus $\Sigma = \Sigma_1 \cup \Sigma_2 = \{\alpha, \beta, \lambda\}$. Consider $L(G_1)$ and $L(G_2)$ as given in Figure 2.3. The behaviour of product can be obtained easily by

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

For the synchronous composition, the language $P_i^{-1}L(G_i)$ for $i = 1, 2$ is computed. The language $P_i^{-1}L(G_i)$ models the behaviour of G_i with selflooping³ of the events in $\Sigma - \Sigma_i$ as displayed in Figure 2.3. The synchronous composition is obtained by

$$L_m(G_1 \parallel G_2) := p_1^{-1}(L_m(G_1)) \cap p_2^{-1}(L_m(G_2)).$$

Here, the two systems G_1 and G_2 generate event β synchronously while the other events are permitted to occur whenever possible. ◇

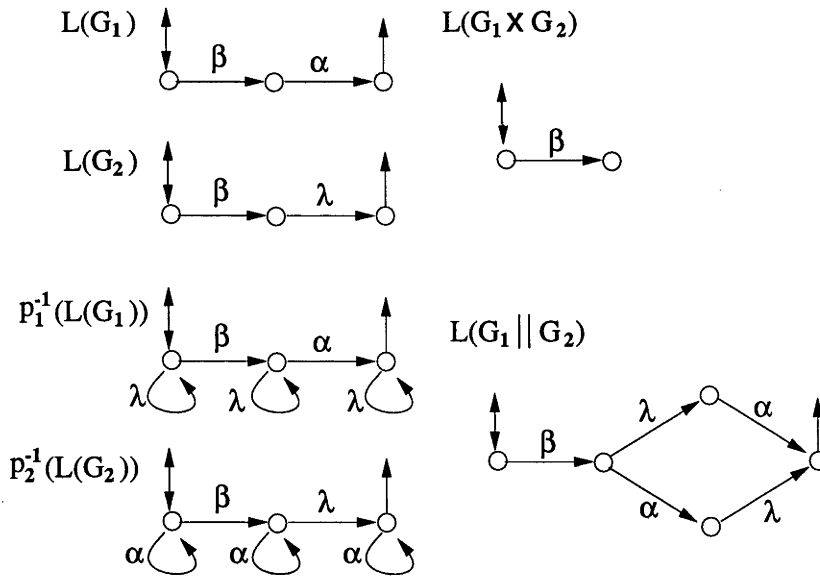


Figure 2.3: An example for product and synchronous composition

³Selfloop is a transition of a generator for which the exit and the entrance states are the same. Formally, for a DES $G = (Q, \Sigma, \delta, q_0, Q_m)$, we say that an event $\sigma \in \Sigma$ is a *selfloop* at the state $q \in Q$ if $\delta(q, \sigma) = q$.

2.3 Controllability and Supervisory Control

In supervisory control theory, the generator G plays the role of the ‘plant’ in the sense of classical control theory. The uncontrolled DES G described in the previous sections is simply a spontaneous generator of event strings without means of external control. To implement control to such a DES, a ‘controller’ (called *supervisor* in DES), playing the same role as in classical control theory, needs to be designed. The purpose of control is to restrict the system behaviour to a desirable subset of states and transitions (i.e. the specifications). This allows us to influence the evolution of system by prohibiting key events from occurring at certain points. By doing this, the supervisor ensures that the resulting *closed-loop* system satisfies the specified constraints.

2.3.1 Controlled DES and Nonblocking

We now formally introduce a means of control to DES G as follows. Firstly, the set of all *control patterns* for G is defined to be

$$\Gamma = \{\gamma \subseteq \Sigma \mid \gamma \supseteq \Sigma_u\},$$

where each control pattern $\gamma \in \Gamma$ is a subset of events to be enabled. We say that if $\sigma \in \gamma$, then σ is *enabled* by γ (permitted to occur), otherwise σ is *disabled* by γ (prohibited from occurring). The condition $\gamma \supseteq \Sigma_u$ means that uncontrollable events are always enabled. Then a *supervisory control* for G is any map

$$v : L(G) \longrightarrow \Gamma.$$

Thus a supervisory control is a map that specifies the set of events to be enabled for each string in the closed behaviour of G .

To formalise enablement and disablement on DES G , we extend the process G to

$$G_c = (Q, \Gamma \times \Sigma, \delta_c, q_0, Q_m),$$

where the transition function

$$\delta_c : Q \times \Gamma \times \Sigma \longrightarrow Q,$$

is a partial function according to

$$\delta_c(q, \gamma, \sigma) = \begin{cases} \delta(q, \sigma) & \text{if } \delta(q, \sigma) \text{ is defined and } \sigma \in \gamma \\ \text{undefined} & \text{otherwise.} \end{cases}$$

The process \mathbf{G}_c (often called *controlled DES*) is assumed to operate in the same way as the uncontrolled DES \mathbf{G} except that an event can occur only when it is enabled. Note that enabled events are not necessarily forced to occur but simply permitted to occur.

The closed-loop DES consisting of the plant \mathbf{G} under the supervision of a supervisory control v is denoted by (\mathbf{G}, v) or v/\mathbf{G} . The closed behaviour of v/\mathbf{G} , denoted $L(v/\mathbf{G})$, is defined inductively as follows:

- $\epsilon \in L(v/\mathbf{G})$,
- $\omega\sigma \in L(v/\mathbf{G})$ if and only if $\omega \in L(v/\mathbf{G})$, $\sigma \in v(\omega)$ and $\omega\sigma \in L(\mathbf{G})$.
- No other strings belong to $L(v/\mathbf{G})$.

Clearly, $L(v/\mathbf{G}) \subseteq L(\mathbf{G})$, and it is prefix-closed by definition. The marked behaviour of the supervised system is defined by

$$L_m(v/\mathbf{G}) = L(v/\mathbf{G}) \cap L_m(\mathbf{G}).$$

This is just the part of the original marked behaviour $L_m(\mathbf{G})$ that survives under supervision. Since $L_m(\mathbf{G})$ represents completed tasks, it indicates those tasks that will be completed under supervision.

We say a supervisory control v is *nonblocking* for \mathbf{G} if

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

Namely, if every string generated by the closed-loop process v/\mathbf{G} can be extended to reach the set of marker states of \mathbf{G} , then v is said to be nonblocking for \mathbf{G} . Otherwise, v is said to be *blocking*. The property of nonblocking of the closed-loop system indicates that the ability of a supervisory control v to take the system from its initial state to marker states. Since marker states represent completed tasks, blocking means that the closed-loop system cannot accomplish the execution of given tasks.

2.3.2 Controllability

The basic control problem in this framework is then as follows:

Problem 2.1 *For a given language $K \subseteq L(\mathbf{G})$, is there a nonblocking supervisor v such that $L_m(v/\mathbf{G}) = K$? In other words, what behaviour $K \subseteq L(\mathbf{G})$ can be achieved by the supervision of a nonblocking supervisor?* \square

The answer for this problem is in the concept of controllability [RW87b]. A language $K \subseteq \Sigma^*$ is *controllable* (with respect to $L(\mathbf{G})$ and Σ_u) if

$$\overline{K}\Sigma_u \cap L(\mathbf{G}) \subseteq \overline{K}.$$

This condition describes that for any prefix of a string in K , $s \in \overline{K}$, if s is extended by an uncontrollable event $\sigma \in \Sigma_u$ in $L(\mathbf{G})$, then it should be a prefix of a string in K , i.e., $s\sigma \in \overline{K}$. Since uncontrollable events cannot be prevented from occurring, it is intuitively clear that for K to be controllable if an uncontrollable event occurs along a ‘sample path’ in \overline{K} , then the extended sample path should also be in \overline{K} . Intuitively, this means that ‘if you have a string which cannot be prevented from occurring, then it has to be in the controlled behaviour’. Note that we can observe from the definition that K is controllable if and only if \overline{K} is controllable. It is easy to verify that ϕ , $L(\mathbf{G})$, and Σ^* are always controllable with respect to \mathbf{G} .

Then we have the following result ([RW87b]) which provides a necessary and sufficient condition for Problem 2.1.

Theorem 2.1 *Consider a DES \mathbf{G} where $\Sigma_u \subseteq \Sigma$ is the set of uncontrollable events. For a nonempty language $K \subseteq L_m(\mathbf{G})$, there exists a nonblocking supervisory control v for \mathbf{G} such that*

$$L_m(v/\mathbf{G}) = K$$

if and only if the following two conditions hold:

1. *K is controllable with respect to \mathbf{G} , i.e., $\overline{K}\Sigma_u \cap L(\mathbf{G}) \subseteq \overline{K}$.*
2. *K is $L_m(\mathbf{G})$ -closed, i.e., $K = \overline{K} \cap L_m(\mathbf{G})$.* \square

2.3.3 Specification Language

An uncontrolled DES G represents all possible strings generated in the open-loop process. In a controlled system, the behaviour of the system is usually restricted to a set of desirable strings which represent the user requirements for a given system. A process specification language is a set of such desirable strings within which a controlled system is permitted to evolve. Since in general an uncontrollable DES G is assumed to generate ‘illegal’ behaviour, a supervisory controller v is introduced so that the closed-loop system produce only the ‘legal’ behaviour given by a specification language. Like the case of system representation, a specification can be formally represented as a generator E which generates the specification language $L(E)$. After we account for all the user requirements which are imposed on the system, we can obtain specification languages E_i for $i = 1, 2, \dots, n$. They can be formally represented by generators E_i . The overall specification language is taken by the intersection⁴ of all these languages. If this language is not a subset of $L_m(G)$, then we take

$$E = L_m(G) \cap (\bigcap_i^n E_i).$$

2.3.4 Supremal Controllable Sublanguage

Let G be a controlled DES. Then consider an arbitrary language $E \subseteq \Sigma^*$. We introduce the set of all sublanguages of E that are controllable with respect to G . Later language E will play the role of a specification language for G .

Let $\mathcal{C}(E)$ be the set of all controllable (with respect to G) sublanguages of E :

$$\mathcal{C}(E) = \{K \subseteq E \mid K \text{ is controllable with respect to } G\}.$$

Some properties of this family of languages are [RW87b, Won96]: for $E_1, E_2 \in \mathcal{C}(E)$,

1. $E_1 \cup E_2$ is controllable.
2. $E_1 \cap E_2$ need not be controllable
3. if E_1 and E_2 are closed, then so is $E_1 \cap E_2$.

This family of languages is closed under arbitrary union ([RW87b, Proposition 7.1]). Since

⁴In some cases (like in decentralised control described in Section 2.6), we may have to take synchronous composition (\parallel) instead of the intersection

the empty language ϕ is controllable, it is a member of $\mathcal{C}(E)$. So $\mathcal{C}(E)$ is always nonempty. Hence, taking the union of members of $\mathcal{C}(E)$, the ‘supremal’ (in the sense of subset inclusion) controllable sublanguage of E always exists in $\mathcal{C}(E)$. In here, we denote this language by

$$\kappa_{L(\mathbf{G})}(E) := \sup \mathcal{C}(E).$$

Clearly $\phi \subseteq \kappa_{L(\mathbf{G})}(E) \subseteq E$. If a given language E is not controllable, then $\kappa_{L(\mathbf{G})}(E)$ is the largest possible or ‘optimal’ behaviour that can be synthesised by a supervisory control v and is contained in E . The role of supervisory control v is to restrict the behaviour of \mathbf{G} properly so that all the sequences generated by \mathbf{G} under v form a subset of the specification E . Thus the language $\kappa_{L(\mathbf{G})}(E)$ represents the *least restrictive* control that will guarantee to satisfy the specification E .

Now we have the following theorem [RW87b, Won96].

Theorem 2.2 *Let $E \subseteq \Sigma^*$ be $L_m(\mathbf{G})$ -closed. If $\kappa_{L(\mathbf{G})}(E)$ is nonempty, then there exists a nonblocking supervisory control v for \mathbf{G} such that $L_m(v/\mathbf{G}) = \kappa_{L(\mathbf{G})}(E)$. \square*

2.4 Realisation of a Supervisory Control

Let v be a nonblocking supervisory control for the controlled DES $\mathbf{G} = (Q, \Sigma, \delta, q_0, Q_m)$, with $L_m(v/\mathbf{G}) = K$ and $L(v/\mathbf{G}) = \bar{K}$. Now let \mathbf{S} be a DES such that

$$K = L_m(\mathbf{S}) \cap L_m(\mathbf{G}), \quad \bar{K} = L(\mathbf{S}) \cap L(\mathbf{G}).$$

If this relation holds, then we say that a DES \mathbf{S} *implements* v [Won96]. Formally, let a DES \mathbf{S} be

$$\mathbf{S} = (X, \Sigma, \xi, x_0, X_m),$$

where X is a finite state set, Σ is an alphabet, $\xi : X \times \Sigma \rightarrow X$ is a (partial) transition function, $x_0 \in X$ is the initial state and $X_m \subseteq X$ is the marker states. Assume that \mathbf{S} implements v .

Then the control action is exercised by \mathbf{S} on \mathbf{G} implicitly, by a *state feedback map*

$$\psi : X \rightarrow \Gamma$$

defined as follows: for $x \in X$,

$$\psi(x) := \{\sigma \in \Sigma \mid \xi(x, \sigma) \text{ is defined}\}.$$

A nonblocking generator S (i.e., $\overline{L_m(S)} = L(S)$) is said to be a *supervisor* for G [Won96] if

- (i) $L_m(S)$ is controllable with respect to G ,
- (ii) $\overline{L_m(S) \cap L_m(G)} = \overline{L_m(S)} \cap L(G)$.

If, in addition, S is trim (reachable and coreachable), then S is called a *proper supervisor* [Won96].

Let us denote the closed-loop system consisting of the plant G and the supervisor S by S/G (see Figure 2.4). The operation of S/G can be described as follows. The supervisor S starts at the initial state x_0 and executes state transition $\xi : X \times \Sigma \rightarrow X$ in response to events $\sigma \in \Sigma$ generated by G . At each state the map ψ decides whether σ is to be disabled or enabled at the corresponding state of G . The next possible event that can be generated by the closed-loop system S/G is any event $\sigma \in \Sigma$ such that only both $\xi(x, \sigma)$ and $\delta(q, \sigma)$ are defined. In this way, the supervisor S exerts some control over the future evolution of G (for detail, see [RW87b]). Note that the supervisor S does not force G to execute a particular event. It simply permits some events to occur. However, S can effectively force a particular event to occur by disabling all the other events at the given state except the desired one.

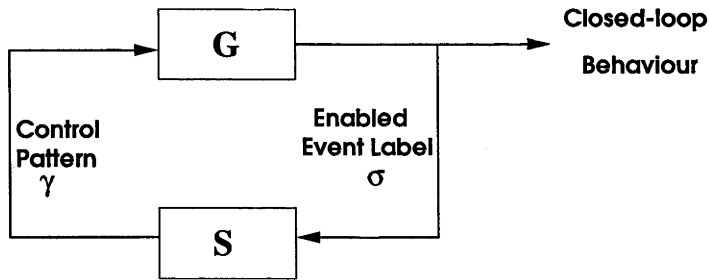


Figure 2.4: Supervisory control of DES

Now, denote the closed and marked behaviour of S/G to be $L(S/G)$ and $L_m(S/G)$ respectively. Then if a proper supervisor S which implements a supervisory control v is connected to G by the product operation, the result $S \times G$ represents exactly the behaviour of S/G , i.e.,

$$S/G = (X \times Q, \Sigma, \xi \times \delta, (x_0, q_0), X_m \times Q_m),$$

where

$$\xi \times \delta : X \times Q \times \Sigma \longrightarrow X \times Q,$$

according to, for $q \in Q$, $x \in X$ and $\sigma \in \Sigma$,

$$\xi \times \delta((x, q), \sigma) = \begin{cases} (\xi(x, \sigma), \delta(q, \sigma)) & \text{if } \xi(x, \sigma) \text{ is defined and } \delta(q, \sigma) \text{ is defined,} \\ \text{undefined} & \text{otherwise.} \end{cases}$$

So we have that

$$\begin{aligned} L(\mathbf{S}/\mathbf{G}) &= L(\mathbf{S} \times \mathbf{G}) = L(\mathbf{G}) \cap L(\mathbf{S}), \\ L_m(\mathbf{S}/\mathbf{G}) &= L_m(\mathbf{S} \times \mathbf{G}) = L_m(\mathbf{G}) \cap L_m(\mathbf{S}). \end{aligned}$$

For given DES \mathbf{G} and \mathbf{E} , there is an algorithm for computing a supervisor \mathbf{S} [WR87] such that

$$L_m(\mathbf{S}/\mathbf{G}) = \kappa_{L(\mathbf{G})}(L_m(\mathbf{E}) \cap L_m(\mathbf{G})).$$

The computational effort to obtain \mathbf{S} is $O(n^2 m^2)$, where n is the number of states in \mathbf{G} , and m is the number of states in \mathbf{E} [WR88].

2.5 Modular Control

In this section, we recall modular control as formalised in [WR88]. Often a complex system is composed of two or more simpler components. Likewise, a control task can be divided into several subtasks. Such systems are easily found in the case when modelling the concurrent operations of several asynchronous, or partially synchronous subtasks, for example, a flow rate control system consisting of tanks, valves and pumps. In this system, one control specification may be given as preventing overflow or underflow of tanks and another establishes the priorities of certain pumps or valves. In addition, a specification may also be required for which pump should be fixed before the others when several pumps are broken down at the same time. When we are dealing with these kinds of systems, one of the difficulties is that the number of states of the system as well as the overall specifications increase exponentially with the number of their components [Ram88, RW89]. To solve this problem, one can design a *local* subcontroller from the *local* specifications for each subtask and then the resultant subcontrollers are combined to form a solution to the original *global* problem. This approach is often called ‘divide and

conquer'. We refer to such a method as *modular synthesis* and the resulting supervisor as *modular supervisor*.

One advantage of this approach is that a modular supervisor can be synthesised more easily. Another advantage is that such a modular supervisor is more easily modified, updated and maintained. For example, when a subtask is changed, without modular synthesis, we need to design the entire supervisor again and this results in high computational cost. While with modular synthesis, we only need to redesign the corresponding subcontroller, which gives us more flexibility when designing a complex system. The main disadvantages of this approach is the problem of conflict between local systems when combined in the global level. The fact that each local systems are designed based only on local specification makes the design much simpler. However, since these local systems are to run concurrently in quasi-independent fashion on the basis of local information, the global system may exhibit conflict such as blocking (impossible to complete the global task) or deadlock (impossible to continue the operation). So the fundamental issue in the modular supervisory control is how to prevent conflict from occurring between local systems and how to ensure that the local supervisors achieve the same optimal control action as a single global supervisor when they are running concurrently in the global level.

2.5.1 Supervisor Conjunction and Nonconflicting

To characterise modular supervisors, we introduce a means of combining supervisors. Let $S_1 = (X, \Sigma, \xi, x_0, X_m)$ and $S_2 = (Y, \Sigma, \eta, y_0, Y_m)$ be supervisors for G . Then the *conjunction* of S_1 and S_2 is given by the supervisor

$$S_1 \wedge S_2 = Rch(S_1 \times S_2) = Rch(X \times Y, \Sigma, \xi \times \eta, (x_0, y_0), X_m \times Y_m),$$

where

$$\xi \times \eta(x, y, \sigma) = \begin{cases} (\xi(x, \sigma), \eta(y, \sigma)) & \text{if both } \xi(x, \sigma) \text{ and } \eta(y, \sigma) \text{ are defined,} \\ \text{undefined} & \text{otherwise,} \end{cases}$$

and $Rch(\dots)$ means the reachable component of an automaton. $S_1 \wedge S_2$ simply consists of the automata S_1 and S_2 operating together in parallel (modelled by $S_1 \times S_2$), in which the

supervisory action of $S_1 \wedge S_2$ is to enable the event σ only when σ is enabled by both S_1 and S_2 simultaneously.

Then it is straightforward to have the following [WR88]: let S_1 and S_2 are nonblocking supervisors for G . Then for the supervisor $S = S_1 \wedge S_2$,

- $L(S/G) = L(S_1/G) \cap L(S_2/G)$,
- $L_m(S/G) = L_m(S_1/G) \cap L_m(S_2/G)$.

However, since the nonblocking property of supervisors is not closed under conjunction, there is no guaranteed that combining of two nonblocking subsystems will yield a nonblocking system. Indeed it is clear that if two supervisors are implemented to achieve the contradictory objectives, their conjunction will end up in a blocking situation. The concept of nonconflicting languages introduced in [WR88] captures this notion. Consider two languages $H_1, H_2 \subseteq \Sigma^*$ to be specified as controllable languages with respect to G . Since it may usually be desired to satisfy two constraints simultaneously, the overall constraint is specified as $H_1 \cap H_2$. It is then always true that

$$\overline{H_1 \cap H_2} \subseteq \overline{H_1} \cap \overline{H_2}.$$

In the case of the equality, it is said to be *nonconflicting* [WR88], i.e.,

$$\overline{H_1 \cap H_2} = \overline{H_1} \cap \overline{H_2}.$$

In other words, two languages are nonconflicting if, whenever they share a prefix, they also share a string containing this prefix. For instance, any two closed languages are nonconflicting. If H_1 and H_2 fail to be nonconflicting, we say that they are *conflicting*. Now we have the following:

Proposition 2.1 ([Won96]) *Two nonblocking supervisors S_1 and S_2 running concurrently will form a proper nonblocking conjunctioned supervisor $S = S_1 \wedge S_2$ if and only if S is trim and $L_m(S_1/G)$ and $L_m(S_2/G)$ are nonconflicting. \square*

2.5.2 Modular Supervision

Now let $E_1, E_2 \subseteq L_m(G)$ be two specification languages. If the modular approach is taken, optimal subsupervisors S_1 and S_2 can be obtained for E_1 and E_2 respectively, i.e., $L_m(S_i/G) =$

$\kappa_{L(\mathbf{G})}(E_i)$, for $i = 1, 2$. Alternatively, an optimal supervisor \mathbf{S} can be synthesised for the overall specification $E_1 \cap E_2$, namely $L_m(\mathbf{S}/\mathbf{G}) = \kappa_{L(\mathbf{G})}(E_1 \cap E_2)$. Then the following is established in [WR88].

Theorem 2.3 *If $\kappa_{L(\mathbf{G})}(E_1) \cap \kappa_{L(\mathbf{G})}(E_2) \neq \phi$, and $\kappa_{L(\mathbf{G})}(E_1)$ and $\kappa_{L(\mathbf{G})}(E_2)$ are nonconflicting, then*

$$L_m((\mathbf{S}_1 \wedge \mathbf{S}_2)/\mathbf{G}) = L_m(\mathbf{S}/\mathbf{G}).$$

In other words,

$$\kappa_{L(\mathbf{G})}(E_1) \cap \kappa_{L(\mathbf{G})}(E_2) = \kappa_{L(\mathbf{G})}(E_1 \cap E_2). \quad \square$$

Theorem 2.3 indicates that in modular synthesis, if subsupervisors are designed for the corresponding local specifications and these subsupervisors are operated together, then the centralised supervisory control objective can be achieved under the condition that the subsupervisors are nonconflicting. If we use an operator $\kappa_{L(\mathbf{G})}$ to represent the process of synthesising an optimal supervisor restricted in $L(\mathbf{G})$, then Theorem 2.3 states the distributivity of the control synthesis operator κ over language intersection.

Finally, to see the benefit of modular synthesis, let us compare the computation complexity with nonmodular synthesis [WR88]. Suppose that $E_1, E_2 \subseteq L_m(\mathbf{G})$ and the state size of the generator \mathbf{G} is bounded by n and the state sizes of the generators for E_i are bounded by m . For the centralised synthesis, to compute $\kappa_{L(\mathbf{G})}(E_1 \cap E_2)$ directly requires $O(m^4 n^2)$ computations. In contrast, to find modular supervisors takes $O(m^2 n^2)$ computations and then to verify the nonconflicting condition also takes $O(m^2 n^2)$ computations. The total complexity of the modular synthesis is $O(m^2 n^2)$. Therefore, if the state size of constraints m is large, the computation savings of modular approach could be substantial.

2.6 Decentralised Supervisory Control

In one approach to decentralised supervisory control formalised by Lin and Wonham [LW88a], the ‘global’ behaviour is observed through a number of channels, modelled as natural projections. These channels may correspond to quasi-independent component activities (e.g., buffer operation vs. repair and maintenance). Each such a channel determines a projected ‘local’ plant model. A decentralised controller is designed for each local model according to a local

specification. Such a scheme is similar to modular supervisory control except that decentralised control has an additional constraint that synthesis is carried out based on partial information and partial control. In many applications, decentralised supervisory control is more suitable than centralised supervisory control. In particular when the global task splits into subtasks for which local supervisory controllers are fairly easy to obtain. However, like in modular supervisory control, the fundamental issue is how to ensure that the local supervisors achieve the same optimal control action as a single global supervisor when they are running concurrently in the global level without conflict problems among local systems.

2.6.1 Projection and Normality Condition

Here, we define the local projection of languages and the key property of normality. Let $\Sigma_o \subseteq \Sigma$ be a subalphabet of Σ with the assumption that $\Sigma_o \neq \phi$. Here Σ_o is interpreted as the set of events of \mathbf{G} which is observed by the local supervisor. Then define a natural projection

$$p : \Sigma^* \longrightarrow \Sigma_o^*,$$

according to

$$p(\epsilon) = \epsilon,$$

$$p(\sigma) = \begin{cases} \sigma & \text{if } \sigma \in \Sigma_o, \\ \epsilon & \text{otherwise, i.e., } \sigma \notin \Sigma_o, \end{cases}$$

$$p(s\sigma) = p(s)p(\sigma),$$

for $s \in \Sigma^*$ and $\sigma \in \Sigma$. The action of p on a string $s \in \Sigma^*$ is simply to erase the elements σ of s which do not belong to Σ_o . Denote by

$$p^{-1} : \mathcal{P}(\Sigma_o^*) \longrightarrow \mathcal{P}(\Sigma^*)$$

the inverse projection of p , where $\mathcal{P}(\cdot)$ is the power set of (\cdot) .

Let $K \subseteq L(\mathbf{G})$ be an arbitrary sublanguage of $L(\mathbf{G})$, Then under the natural projection $p : \Sigma^* \longrightarrow \Sigma_o^*$, where $\Sigma_o \subseteq \Sigma$, K appears as the language $p(K) \subseteq \Sigma_o^*$ at the local level. Notice that the inclusion

$$K \subseteq L(\mathbf{G}) \cap p^{-1}(p(K))$$

is always true since $K \subseteq L(\mathbf{G})$. In the case of equality, it is said that K is *normal* (with respect to $L(\mathbf{G})$ and p) [LW88a], i.e.,

$$K = L(\mathbf{G}) \cap p^{-1}(p(K)).$$

Thus K is normal if it can be recovered from its projection, $p(K)$, together with the global closed language $L(\mathbf{G})$. Equivalently, from a string $s \in K$, for $t \in L(\mathbf{G})$ if $p(t) = p(s)$, then one may infer that $t \in K$.

2.6.2 Local and Global Supervisors

The behaviours of a global plant \mathbf{G} viewed in the local level are simply $p(L(\mathbf{G}))$ and $p(L_m(\mathbf{G}))$. They can be represented by a *local DES model* \mathbf{G}_o , with $L(\mathbf{G}_o) = p(L(\mathbf{G}))$, and $L_m(\mathbf{G}_o) = p(L_m(\mathbf{G}))$. The locally controllable and locally uncontrollable events of \mathbf{G}_o are

$$\Sigma_{u,o} = \Sigma_o \cap \Sigma_u, \quad \Sigma_{c,o} = \Sigma_o \cap \Sigma_c.$$

Let $E_o \subseteq \Sigma_o^*$ be a nonempty closed sublanguage, representing a specification of legal behaviour at the local level. The corresponding specification E on the global level can be taken to be

$$E = (p|_{L(\mathbf{G})})^{-1}(E_o) = L(\mathbf{G}) \cap p^{-1}(E_o),$$

the inverse image of E_o [LW88a]. At this point, we can proceed in two different ways. In one way, we obtain a global supervisor \mathbf{S} that synthesises the supremal controllable sublanguage K of the global specification E ;

$$K := \kappa_{L(\mathbf{G})}(E) = \kappa_{L(\mathbf{G})}(L(\mathbf{G}) \cap p^{-1}(E_o)) \subseteq \Sigma^*.$$

Alternatively, a local supervisor \mathbf{S}_o is chosen in order to synthesise the supremal (locally) controllable sublanguage K_o for the local specification E_o ;

$$K_o := \kappa_{L(\mathbf{G}_o)}(p(L(\mathbf{G})) \cap E_o) \subseteq \Sigma_o^*.$$

By applying the default control rule in [LW88a] that permanently enables all events in $\Sigma_c - \Sigma_{c,o}$, a local supervisor \mathbf{S}_o can be extended to a global supervisor $\widehat{\mathbf{S}}_o$ for the global plant \mathbf{G} . The language synthesised by the local supervision at the global level is obtained as

$$\widehat{K}_o := L(\mathbf{G}) \cap p^{-1}(K_o) \subseteq \Sigma^*.$$

Then we have the following [LW88a].

Lemma 2.1 *The language synthesised by the local supervision is globally optimal i.e., $K =$*

\widehat{K}_o if and only if K is normal. Furthermore if K is normal., then

$$K(\Sigma - \Sigma_{c,o}) \cap L(\mathbf{G}) \subset K.$$

Equivalently, if $s \in K$, $\sigma \in \Sigma - \Sigma_{c,o}$, and $s\sigma \in L(\mathbf{G})$, then $s\sigma \in K$. \square

Lemma 2.1 implies that in the synthesis of K , controllable events that are not locally controllable (i.e., $\Sigma - \Sigma_{c,o}$) never need to be disabled if K is normal.

In practice, the use of local supervisor is more attractive than the global supervisor because of its relatively simple structure and ease of synthesis. So one would like to be able to justify the use of supervisor \widehat{S}_o without actually computing and comparing it with the global structure K since in general this may be more complicated and expensive to compute. Theorem 3.3 in [LW88a] show that this can be done on the basis of some sufficient conditions which do not require K to be explicitly computed.

2.6.3 Decentralised Supervision

We now return to the decentralised problem. Assuming that nonempty subalphabets $\Sigma_i \subseteq \Sigma$ ($i \in \{1, 2, \dots, n\}$) are given, not necessarily pairwise disjoint. Define the natural projection $p_i : \Sigma^* \rightarrow \Sigma_i^*$ as in the previous section. Let \mathbf{G}_i be the local models of \mathbf{G} , that is, \mathbf{G}_i is the generator of $p_i(L(\mathbf{G}))$. Let the local specifications be the closed languages $E_{i,o} \subseteq \Sigma_i^*$. The corresponding specifications on the global level are then $E_i = p_i^{-1}(E_{i,o}) \subseteq \Sigma^*$, with global supervisors S_i . The overall global legal specification on the behaviour of \mathbf{G} is then obtained by

$$E = \bigcap_{i=1}^n p_i^{-1}(E_{i,o}) \subset \Sigma^*.$$

Then we have the following [LW88a].

Theorem 2.4 *If each S_i is optimal for the corresponding legal specification, i.e.,*

$$(\forall i \in \{1, 2, \dots, n\}) L(S_i/\mathbf{G}) = \kappa_{L(\mathbf{G})}(L(\mathbf{G}) \cap p_i^{-1}(E_i)),$$

then

$$L\left(\left(\bigwedge_{i=1}^n S_i\right)/\mathbf{G}\right) = \kappa_{L(\mathbf{G})}(L(\mathbf{G}) \cap E).$$

\square

This result is interpreted as the concurrent operation of globally optimal decentralised supervisors is globally optimal. This result is most likely useful when the supervisors S_i can be designed and implemented in the local level. Thus for $i \in \{1, 2, \dots, n\}$, define

$$\begin{aligned} K_i &:= \kappa_{L(\mathbf{G})}(L(\mathbf{G}) \cap p_i^{-1}(E_{i,o})) \subseteq \Sigma^*, \\ K_{i,o} &:= \kappa_{i,o}(p_i(L(\mathbf{G})) \cap E_{i,o}) \subseteq \Sigma_i^*, \\ \widehat{K}_{i,o} &:= L(\mathbf{G}) \cap p_i^{-1}(K_{i,o}) \subseteq \Sigma^*, \\ L(S_{i,o}/\mathbf{G}_i) &= K_{i,o}, \end{aligned}$$

where,

- K_i : the supremal controllable sublanguage of the global specification for the i th subsystem,
- $K_{i,o}$: the supremal locally controllable sublanguage for local specification $E_{i,o}$,
- $\widehat{K}_{i,o}$: the global language resulting from local control for the i th subsystem.

Let \widehat{S}_i be the global version of a local supervisor $S_{i,o}$ (i.e. $L(\widehat{S}_i/\mathbf{G}) = \widehat{K}_{i,o}$). Thus \widehat{S}_i permanently enables all events that are not observed by $S_{i,o}$, otherwise behaves exactly as $S_{i,o}$ does. We say that \mathbf{G} is *locally controllable* with respect to the family of local sublanguages $E_{i,o}$, if $K_i = \widehat{K}_{i,o}$ for $i \in \{1, 2, \dots, n\}$. Then the following result is obtained [LW88a]:

Theorem 2.5 *Let \mathbf{G} be locally controllable with respect to the family $E_{i,o}$. Then*

$$L\left(\left(\bigwedge_i \widehat{S}_i\right)/\mathbf{G}\right) = \kappa_{L(\mathbf{G})}(L(\mathbf{G}) \cap E).$$

□

Theorem 2.5 states that if \mathbf{G} is locally controllable with respect to a set of local specifications, i.e., the local control for each of these specifications is globally optimal, then the combined supervisory actions of the local controllers achieve the overall optimality.

2.7 Simple Facts

In this section, we present some simple facts which will be used later in this thesis.

Lemma 2.2 Let $\Sigma_3 \subseteq \Sigma_2 \subseteq \Sigma_1$. Let $p_2^1 : \Sigma_1^* \rightarrow \Sigma_2^*$ and $p_3^1 : \Sigma_1^* \rightarrow \Sigma_3^*$ be the natural projections. Then

$$p_3^1 \circ p_2^1 = p_3^1,$$

where \circ denotes the composition of functions.

Proof: For the empty string ϵ , $p_3^1 \circ p_2^1(\epsilon) = p_3^1(p_2^1(\epsilon)) = p_3^1(\epsilon)$.

Assume that for $s \in \Sigma_1^*$, $p_3^1 \circ p_2^1(s) = p_3^1(s)$. Then consider a string $s\sigma \in \Sigma_1^*$, where $\sigma \in \Sigma_1$. If $\sigma \in (\Sigma_1 - \Sigma_2)$ (i.e., $\sigma \notin \Sigma_2$ and $\sigma \notin \Sigma_3$), then $p_3^1 \circ p_2^1(s\sigma) = p_3^1(p_2^1(s\sigma)) = p_3^1(p_2^1(s)) = p_3^1(s)$, and $p_3^1(s\sigma) = p_3^1(s)$. If $\sigma \in (\Sigma_2 - \Sigma_3)$ (i.e., $\sigma \in \Sigma_2$ and $\sigma \notin \Sigma_3$), then $p_3^1 \circ p_2^1(s\sigma) = p_3^1(p_2^1(s\sigma)) = p_3^1(p_2^1(s)\sigma) = p_3^1(p_2^1(s)) = p_3^1(s)$, and $p_3^1(s\sigma) = p_3^1(s)$. Finally, if $\sigma \in \Sigma_3$, then $p_3^1 \circ p_2^1(s\sigma) = p_3^1(p_2^1(s\sigma)) = p_3^1(p_2^1(s)\sigma) = p_3^1(p_2^1(s))\sigma = p_3^1(s)\sigma$, and $p_3^1(s\sigma) = p_3^1(s)\sigma$. Therefore, $p_3^1 \circ p_2^1 = p_3^1$. \square

Lemma 2.3 Let $\Sigma_3 \subseteq \Sigma_2 \subseteq \Sigma_1$. Let $p_3^2 : \Sigma_2^* \rightarrow \Sigma_3^*$ and $p_3^1 : \Sigma_1^* \rightarrow \Sigma_3^*$ be the natural projections. Then

$$p_3^2 = p_3^1|_{\Sigma_2^*},$$

where $p_3^1|_{\Sigma_2^*}$ denotes the restriction of p_3^1 to Σ_2^* .

Proof: For the empty string ϵ , $p_3^2(\epsilon) = \epsilon$, and $p_3^1|_{\Sigma_2^*}(\epsilon) = p_3^1(\epsilon) = \epsilon$ (since $\epsilon \in \Sigma_2^*$).

Suppose that for a string $s \in \Sigma_2^*$, $p_3^2(s) = p_3^1|_{\Sigma_2^*}(s)$. Then consider a string $s\sigma \in \Sigma_2^*$, where $\sigma \in \Sigma_2$. If $\sigma \in (\Sigma_2 - \Sigma_3)$, then $p_3^2(s\sigma) = p_3^2(s) = p_3^1|_{\Sigma_2^*}(s) = p_3^1(s)$, and $p_3^1|_{\Sigma_2^*}(s\sigma) = p_3^1(s\sigma) = p_3^1(s)$. If $\sigma \in \Sigma_3$, then $p_3^2(s\sigma) = p_3^2(s)\sigma = p_3^1|_{\Sigma_2^*}(s)\sigma = p_3^1(s)\sigma$, and $p_3^1|_{\Sigma_2^*}(s\sigma) = p_3^1(s\sigma) = p_3^1(s)\sigma$. Therefore $p_3^2(s\sigma) = p_3^1|_{\Sigma_2^*}(s\sigma)$. So, $p_3^2 = p_3^1|_{\Sigma_2^*}$. \square

Lemma 2.4 Let $\Sigma_1, \Sigma_2 \subseteq \Sigma$. Let $p_i : \Sigma^* \rightarrow \Sigma_i^*$ be the natural projection, for $i = 1, 2$. Then for $u \in \Sigma^*$

$$p_1 p_2(u) = p_2 p_1(u).$$

Proof: For the empty string ϵ , $p_1 p_2(\epsilon) = p_1(\epsilon) = \epsilon$ and $p_2 p_1(\epsilon) = p_2(\epsilon) = \epsilon$.

Suppose that for a string $s \in \Sigma^*$, $p_1 p_2(s) = p_2 p_1(s)$. Then consider a string $s\sigma \in \Sigma^*$.

i) If $\sigma \in \Sigma_1$ and $\sigma \notin \Sigma_2$, then $p_1 p_2(s\sigma) = p_1(p_2(s)p_2(\sigma)) = p_1(p_2(s)) = p_1 p_2(s)$ and

$p_2 p_1(s\sigma) = p_2(p_1(s)p_1(\sigma)) = p_2(p_1(s)\sigma) = p_2(p_1(s))p_2(\sigma) = p_2 p_1(s)$. So $p_1 p_2(s\sigma) = p_2 p_1(s\sigma)$. The case $\sigma \notin \Sigma_1$ and $\sigma \in \Sigma_2$ is similar.

ii) If $\sigma \in \Sigma_1$ and $\sigma \in \Sigma_2$, then $p_1 p_2(s\sigma) = p_1(p_2(s)p_2(\sigma)) = p_1(p_2(s)\sigma) = p_1 p_2(s)\sigma$ and $p_2 p_1(s\sigma) = p_2(p_1(s)p_1(\sigma)) = p_2(p_1(s)\sigma) = p_2(p_1(s))p_2(\sigma) = p_2 p_1(s)\sigma$. Hence $p_1 p_2(s\sigma) = p_2 p_1(s\sigma)$.

iii) The cases of $\sigma \notin \Sigma_1$ and $\sigma \notin \Sigma_2$ is trivial. □

Chapter 3

Structural Decentralised Control of Concurrent DES with Non-prefix-Closed Local Specifications

In this chapter, a problem in decentralised control of concurrent discrete event systems (DES) is formulated. Here we consider that the system is the synchronous composition of a number of subsystems. Specifications are to be given locally on the subsystems. We obtain two sufficient ‘structural’ conditions which ensure that, for a set of local specifications, local syntheses and controls achieve the same optimal behaviour as that would be obtained by a centralised control for the overall specification, and the control in one subsystem does not cause blocking in the other subsystems. Followed by an introduction in Section 3.1, a problem formulation is given in Section 3.2. Sufficient conditions are obtained in Section 3.3. In Section 3.4, examples are provided for illustration. Conclusions are presented in Section 3.5.

3.1 Introduction

In this chapter, we formulate a problem in decentralised control of concurrent discrete event systems (DES). The decentralised control considered in here is similar to that in [LW88a] with

the following differences. Firstly, in [LW88a] the local specifications are given by prefix-closed languages and as a result the issue of blocking in the concurrent supervisions of decentralised controllers never arises. Here, we allow local specifications to be non-prefix-closed. So, the question of blocking then needs to be addressed. Secondly, we are more interested in ‘structural decentralised control’, namely we are seeking conditions that are on the arrangement of the structures of local systems, and are generally independent of the specifications. That is, once the structural conditions are verified for a given system, then decentralised control is achieved for a set of specifications. Our approach is similar to that in [Als96]. In contrast, the conditions in [LW88a] are ‘specification-dependent’. That is, for each given local specification one has to check the consistency between local and global control. In this sense the present work is also different from that in [LW91] where conditions are obtained for ‘efficient’ verification of nonconflictingness between given decentralised supervisors.

As an initial attempt, we consider a more structural situation than that in [LW88a], namely that the overall system G is the synchronous composition of local systems, G_1, G_2, \dots, G_n , as studied in [WH91]. The event sets of G_i and G_j , for $i, j \in \{1, 2, \dots, n\}$ and $i \neq j$, need not to be disjoint, and the shared events may be controllable or uncontrollable. In [WH91] this situation is called *concurrent* DES.

We believe that despite the restriction that G is the synchronous composition of the subsystems G_1, G_2, \dots, G_n , the result that we obtain can still be applied to a large class of systems. For example, G_1, G_2, \dots, G_n might represent different workcells in a manufacturing plant or different departments in an organisation. There are often requirements that are specified locally for which local supervisors are synthesised. Since there are synchronisations through shared events, local control in one subsystem might affect the behaviour of the other subsystems. Thus the role of the conditions that we seek is to ensure that this potential interference of local controls does not result in conflict and the loss of optimality as compared to a globally designed supervisor for the combined local specifications. If, for instance, the structures of G_1, G_2, \dots, G_n and their synchronisation can be designed, then one can arrange these conditions to ensure that decentralised control is achieved. In our examples, this intuitively could be interpreted as to build appropriate ‘infrastructure’ of workcells or departments.

3.2 Problem Formulation

In this section we formulate a problem of decentralised control of concurrent DES as shown in Figure 3.1. For notational simplicity we present the problem formulation in terms of languages.

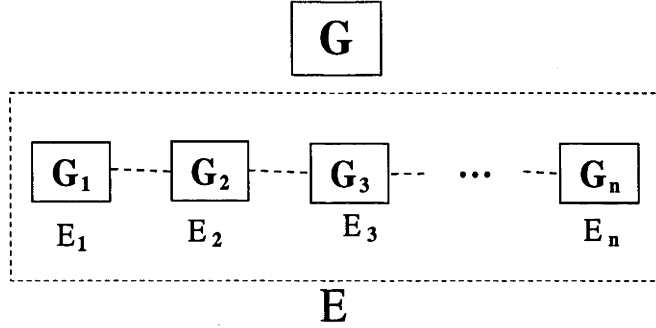


Figure 3.1: Decentralised control of concurrent DES

Let $\Sigma_1, \Sigma_2, \dots, \Sigma_n$ be the event alphabets of subplants, G_1, G_2, \dots, G_n , respectively. It is allowed that $\Sigma_i \cap \Sigma_j \neq \phi$, for $i, j \in \{1, 2, \dots, n\}$ and $i \neq j$. Assume that Σ_i is partitioned into controllable and uncontrollable events, i.e. $\Sigma_i = \Sigma_{ic} \dot{\cup} \Sigma_{iu}$. We assume that the following relation holds between the control status of events between two subsystems G_i and G_j :

$$\Sigma_{iu} \cap \Sigma_j = \Sigma_j \cap \Sigma_{iu}. \quad (3.1)$$

That is, G_i and G_j agree on the control status of shared events, for $i, j \in \{1, 2, \dots, n\}$ and $i \neq j$.

Let $\Sigma := \bigcup_{i=1}^n \Sigma_i$ be the event set of the global system, say G . Let $\Sigma_c := \bigcup_{i=1}^n \Sigma_{ic}$ and $\Sigma_u := \bigcup_{i=1}^n \Sigma_{iu}$. It can be easily checked that

$$\Sigma_{iu} = \Sigma_i \cap \Sigma_u, \quad (3.2)$$

for $i = 1, 2, \dots, n$. Let p_i be the natural projection from Σ^* to Σ_i^* . Let $L_{i,m}, L_i \subseteq \Sigma_i^*$ represent respectively the marked and closed behaviours of system G_i . We assume that $L_i = \overline{L_{i,m}}$. The

marked and closed behaviours of the overall system \mathbf{G} are respectively

$$\begin{aligned} L_m &= L_{1,m} \parallel L_{2,m} \parallel \cdots \parallel L_{n,m} \\ &= (p_1)^{-1}(L_{1,m}) \cap (p_2)^{-1}(L_{2,m}) \cap \cdots \cap (p_n)^{-1}(L_{n,m}) = \bigcap_{i=1}^n (p_i)^{-1}(L_{i,m}), \\ L &= L_1 \parallel L_2 \parallel \cdots \parallel L_n \\ &= (p_1)^{-1}(L_1) \cap (p_2)^{-1}(L_2) \cap \cdots \cap (p_n)^{-1}(L_n) = \bigcap_{i=1}^n (p_i)^{-1}(L_i). \end{aligned}$$

Now let $E_i \subseteq L_{i,m}$ be $L_{i,m}$ -closed language, not necessarily prefix-closed, representing a specification on the system \mathbf{G}_i . The corresponding specification on \mathbf{G} can be taken to be $(p_i|_L)^{-1}(E_i)$ (where $(p_i|_L)$ denotes the restriction of p_i to L), the inverse image of E_i , as done in [LW88a]. The overall specification then is

$$E := (p_1|_L)^{-1}(E_1) \cap (p_2|_L)^{-1}(E_2) \cap \cdots \cap (p_n|_L)^{-1}(E_n) = \bigcap_{i=1}^n (p_i|_L)^{-1}(E_i). \quad (3.3)$$

We can proceed in two different ways. Firstly, we synthesise local supervisors on \mathbf{G}_i whose closed-loop marked and closed behaviours are respectively $\kappa_{L_i}(E_i)$ and $\overline{\kappa_{L_i}(E_i)}$. By applying the default control rule in [LW88a] that permanently enables all events in $\Sigma_c - \Sigma_{i,c}$, the closed language synthesised on the system \mathbf{G} is $(p_i|_L)^{-1}(\overline{\kappa_{L_i}(E_i)})$. The closed behaviour under the concurrent supervisions of the local supervisors is

$$(p_1|_L)^{-1}(\overline{\kappa_{L_1}(E_1)}) \cap (p_2|_L)^{-1}(\overline{\kappa_{L_2}(E_2)}) \cap \cdots \cap (p_n|_L)^{-1}(\overline{\kappa_{L_n}(E_n)}) = \bigcap_{i=1}^n (p_i|_L)^{-1}(\overline{\kappa_{L_i}(E_i)}).$$

Secondly, we can synthesise for the global specification E and obtain $\overline{\kappa_L(E)}$ as the closed behaviour of a global supervisor. We are interested in the following.

Problem 3.1 *Under what condition is it true that, for $i = 1, 2, \dots, n$ and for any*

$$E_i \in \mathcal{F}_{L_{i,m}},$$

$$\bigcap_{i=1}^n (p_i|_L)^{-1}(\overline{\kappa_{L_i}(E_i)}) = \overline{\kappa_L(E)}, \quad (3.4)$$

and

$$p_i(\overline{\kappa_L(E)}) \text{ is nonblocking with respect to } L_{i,m} ? \quad (3.5)$$

We recall that $\mathcal{F}_{L_{i,m}}$ is the set of $L_{i,m}$ -closed languages. ◇

Thus the problem is to find conditions under which local syntheses and control for any $L_{i,m}$ -closed specifications do not result in the loss of optimality compared to the global synthesis as stated in Eq. 3.4, and control of one subsystem never incurs blocking in the other subsystems as stated in Eq. 3.5. Since $E_i \subseteq L_{i,m}$, it is easy to see that $\kappa_L(E) \subseteq E \subseteq L_m$. So $\kappa_L(E)$ is always nonblocking with respect to L_m .

The following examples show that the equality of Eq. 3.4 is generally not true.

Example 3.1 Consider the case when $n = 2$. Let $\Sigma_1 = \{\alpha, \beta\}$ and $\Sigma_2 = \{\beta\}$. Let $L_{1,m} = \{\alpha, \alpha\beta, \beta\}$, $L_{2,m} = \{\beta\}$, $L_1 = \overline{L_{1,m}} = \{\epsilon, \alpha, \alpha\beta, \beta\}$, and $L_2 = \{\epsilon, \beta\}$. Also let $L_m = L_{1,m} \parallel L_{2,m} = \{\alpha\beta, \beta\}$ and $L = L_1 \parallel L_2 = \{\epsilon, \alpha, \alpha\beta, \beta\}$. Let $\Sigma_c = \Sigma$ and $E_1 = \{\alpha, \beta\}$, $E_2 = \{\beta\}$. It can be checked that E_i is $L_{i,m}$ -closed for $i = 1, 2$. Since every event is controllable, $\kappa_{L_i}(E_i) = E_i$ for $i = 1, 2$. It can be checked that $(p_1|_L)^{-1}(\overline{\kappa_{L_1}(E_1)}) = \{\epsilon, \alpha, \beta\}$, and $(p_2|_L)^{-1}(\overline{\kappa_{L_2}(E_2)}) = \{\epsilon, \alpha, \alpha\beta, \beta\}$. Hence

$$(p_1|_L)^{-1}(\overline{\kappa_{L_1}(E_1)}) \cap (p_2|_L)^{-1}(\overline{\kappa_{L_2}(E_2)}) = \{\epsilon, \alpha, \beta\}.$$

From Eq. 3.3 one has $E = \{\beta\}$. Again since every event is controllable, $\kappa_L(E) = E$. Thus,

$$\overline{\kappa_L(E)} \subset (p_1|_L)^{-1}(\overline{\kappa_{L_1}(E_1)}) \cap (p_2|_L)^{-1}(\overline{\kappa_{L_2}(E_2)})$$

with strict inclusion. Clearly the behaviour under concurrent local controls

$$(p_1|_L)^{-1}(\overline{\kappa_{L_1}(E_1)}) \cap (p_2|_L)^{-1}(\overline{\kappa_{L_2}(E_2)})$$

is not nonblocking (with respect to L_m). This is also reflected in the conflict between the global languages associated with the goals the local supervisors, namely that the two languages

$$(p_1|_L)^{-1}(\overline{\kappa_{L_1}(E_1)}) = \{\alpha, \beta\} \text{ and } (p_2|_L)^{-1}(\overline{\kappa_{L_2}(E_2)}) = \{\alpha\beta, \beta\}$$

are not nonconflicting. Intuitively, for local supervisor 2 to complete its task, the event β needs to occur; however, for local supervisor 1, if α occurs then β will be disabled. Consequently, this prevents local supervisor 2 from reaching its marker state. \diamond

Example 3.2 Again, we consider the case when $n = 2$. Let $\Sigma_1 = \{\alpha, \beta\}$ and $\Sigma_2 = \{\beta, \gamma\}$. Let $\Sigma_c = \{\alpha, \gamma\}$ and $\Sigma_u = \{\beta\}$. Let $L_{1,m} = \{\epsilon, \alpha, \alpha\beta\}$, $L_{2,m} = \{\epsilon, \beta, \beta\gamma\}$, and $L_i = \overline{L_{i,m}}$. It can be checked that $L_m = L = \{\epsilon, \alpha, \alpha\beta, \alpha\beta\gamma\}$. Let $E_1 = \{\epsilon, \alpha\}$ and $E_2 = \{\epsilon\}$. Clearly E_i is $L_{i,m}$ -closed. It can be checked that $\kappa_{L_1}(E_1) = \{\epsilon\}$ and $\kappa_{L_2}(E_2) = \phi$. Thus $(p_1|_L)^{-1}(\overline{\kappa_{L_1}(E_1)}) \cap (p_2|_L)^{-1}(\overline{\kappa_{L_2}(E_2)}) = \phi$. We also have $E = (p_1|_L)^{-1}(E_1) \cap (p_2|_L)^{-1}(E_2) = \{\epsilon, \alpha\}$. Hence $\kappa_L(E) = \{\epsilon\}$. So

$$(p_1|_L)^{-1}(\overline{\kappa_{L_1}(E_1)}) \cap (p_2|_L)^{-1}(\overline{\kappa_{L_2}(E_2)}) \subset \overline{\kappa_L(E)}$$

with strict inclusion. Here information on synchronisation is exploited in synthesising the global supervisor that achieves a larger behaviour: the uncontrollable event β can only occur after the event α as specified by the structure of subsystem L_1 . \diamond

It is interesting to note that Eq. 3.4 can be rewritten as

$$\overline{\kappa_{L_1}(E_1)} \parallel \overline{\kappa_{L_2}(E_2)} \parallel \cdots \parallel \overline{\kappa_{L_n}(E_n)} = \overline{\kappa_{L_1 \parallel L_2 \cdots \parallel L_n}(E_1 \parallel E_2 \cdots \parallel E_n)}. \quad (3.6)$$

Roughly speaking, this equation can be seen as the distributivity of the control synthesis operator κ over the synchronous composition \parallel .

Remark 3.1 It may be informative to consider the complexity involved to see the benefit of achieving Eq. 3.6. Suppose that the state sizes of G_1, G_2, \dots, G_n are bounded above by k and the state sizes of the generators for E_1, E_2, \dots, E_n are bounded above by m . The sizes of the global system G and the generator for the overall specification E are respectively bounded above by k^n and m^n . The computation complexity for the synthesis of the left-hand side of Eq. 3.6 is $O(nk^2m^2)$ and that for the right-hand side is $O((k^n)^2(m^n)^2 = (k^2m^2)^n)$ (cf. [WR88, RW89]). Without Eq. 3.6 to guarantee that, for any $L_{i,m}$ -closed, local specifications, local syntheses and controls achieve the ‘optimal’ nonblocking behaviour, one is forced to carry out a global synthesis and global control. To do this, the computation may be exponentially more expensive. \diamond

3.3 Main Result

In this section, we present sufficient conditions for Problem 3.1. For this, we introduce the following definitions.

Firstly, let H_1, H_2, \dots, H_n be languages over the alphabet Σ . Then H_1, H_2, \dots, H_n are *nonconflicting* if

$$\overline{H_1 \cap H_2 \cap \cdots \cap H_n} = \overline{H_1} \cap \overline{H_2} \cap \cdots \cap \overline{H_n}.$$

In other words, the languages are said to be nonconflicting if, whenever they share a prefix, they also share a string containing this prefix. This generalises the notion of nonconflicting languages [WR88] to n languages.

Remark 3.2 Note that as will be seen in Examples 3.3 and 3.4, the nonconflictingness between any pair of given languages is not equivalent to the nonconflictingness among all the given languages, i.e.,

$$\begin{aligned} \overline{H_1 \cap H_2 \cap \dots \cap H_n} &= \overline{H_1} \cap \overline{H_2} \cap \dots \cap \overline{H_n} \\ \Leftrightarrow \overline{H_i \cap H_j} &= \overline{H_i} \cap \overline{H_j}, \forall i, j \in \{1, 2, \dots, n\} \text{ and } i \neq j. \end{aligned} \quad \diamond$$

Example 3.3 Consider the case when $n = 3$. Let

$$H_1 = \{\alpha, \alpha\alpha, \beta, \beta\alpha\}, H_2 = \{\beta, \beta\beta, \gamma, \gamma\beta\}, H_3 = \{\gamma, \gamma\gamma, \alpha, \alpha\gamma\}.$$

Then $H_1 \cap H_2 = \{\beta\}$. So, $\overline{H_1 \cap H_2} = \{\epsilon, \beta\}$. Also one can check that $\overline{H_1} \cap \overline{H_2} = \{\epsilon, \beta\}$. So H_1 and H_2 are nonconflicting. For the pair H_2 and H_3 , one can see that $\overline{H_2 \cap H_3} = \{\epsilon, \gamma\} = \overline{H_2} \cap \overline{H_3}$. So, H_2 and H_3 are nonconflicting. Also, since $\overline{H_3 \cap H_1} = \{\epsilon, \alpha\} = \overline{H_3} \cap \overline{H_1}$, one knows that H_3 and H_1 are nonconflicting. However, $H_1 \cap H_2 \cap H_3 = \phi$ and $\overline{H_1 \cap H_2 \cap H_3} = \{\epsilon\}$. So H_1, H_2 and H_3 are not nonconflicting. This shows that the nonconflictingness between H_i and H_j , for $i, j = 1, 2, 3$ and $i \neq j$, does not imply the nonconflictingness among H_1, H_2 and H_3 . \diamond

Example 3.4 Again we consider the case when $n = 3$. Let $H_1 = \{\alpha\alpha, \delta, \delta\alpha, \gamma\alpha\}$,

$$H_2 = \{\alpha\beta, \beta\beta, \delta, \delta\beta\}, H_3 = \{\beta\gamma, \delta, \delta\gamma, \gamma\gamma\}.$$

Hence, $H_1 \cap H_2 \cap H_3 = \{\delta\}$, and $\overline{H_1 \cap H_2 \cap H_3} = \{\epsilon, \delta\}$. Since one has that $\overline{H_1} \cap \overline{H_2} \cap \overline{H_3} = \{\epsilon, \delta\}$, H_1, H_2 and H_3 are nonconflicting. However, for the pair H_1 and H_2 , one has that $\overline{H_1 \cap H_2} = \{\epsilon, \delta\}$ and $\overline{H_1} \cap \overline{H_2} = \{\epsilon, \alpha, \delta\}$. Thus H_1 and H_2 are not nonconflicting. Similarly one could find that the pairs H_2 and H_3 , and H_1 and H_3 are also not nonconflicting. \diamond

Now we have the following result as an extension of Theorem 6.1 in [WR88].

Proposition 3.1 Let L be a closed language over Σ . Let $E_i \subseteq L$ for $i = 1, 2, \dots, n$. Suppose

that $\{\kappa_L(E_i) \mid i = 1, 2, \dots, n\}$ are nonconflicting. Then

$$\kappa_L\left(\bigcap_{i=1}^n E_i\right) = \bigcap_{i=1}^n \kappa_L(E_i).$$

Proof: Since $\bigcap_{i=1}^n E_i \subseteq E_i$, one has that $\kappa_L(\bigcap_{i=1}^n E_i) \subseteq \kappa_L(E_i)$, for $i = 1, 2, \dots, n$. Hence $\kappa_L(\bigcap_{i=1}^n E_i) \subseteq \bigcap_{i=1}^n \kappa_L(E_i)$.

For the reverse inclusion, since $\overline{\mathcal{C}}(L)$, the set of closed, controllable sublanguages of L , is a complete sublattice of $\mathcal{P}(L)$ and $\{\kappa_L(E_i) \mid i = 1, 2, \dots, n\}$ are nonconflicting,

$$\overline{\bigcap_{i=1}^n \kappa_L(E_i)} = \bigcap_{i=1}^n \overline{\kappa_L(E_i)} \in \overline{\mathcal{C}}(L),$$

where $\mathcal{P}(L)$ is the power set of L . Hence, $\bigcap_{i=1}^n \kappa_L(E_i) \in \mathcal{C}(L)$.

Also, since $\bigcap_{i=1}^n \kappa_L(E_i) \subseteq \bigcap_{i=1}^n E_i$, one has that

$$\bigcap_{i=1}^n \kappa_L(E_i) \subseteq \kappa_L\left(\bigcap_{i=1}^n E_i\right). \quad \square$$

Secondly, let Σ be an alphabet and H be a language over Σ . Let $\Sigma' \subseteq \Sigma$. We say that H marks Σ' if

$$\Sigma^* \Sigma' \cap \overline{H} \subseteq H \Sigma'.$$

Thus, a language H is said to mark a given set of events Σ' if, for any string $s \in \Sigma^*$ and any event $\sigma \in \Sigma'$ such that $s\sigma \in \overline{H}$, the string s is in H . In other words, a language H marks a set of events Σ' if all the strings that can be extended by Σ' -events in \overline{H} are in H , or if the ‘entering’ strings of all Σ' -events in \overline{H} are in H .

Finally, let $\Sigma_1, \Sigma_2, \dots, \Sigma_n$ be alphabets, not necessarily pairwise disjoint. Let p_i be the natural projection from Σ^* to Σ_i^* , where $\Sigma = \bigcup_{i=1}^n \Sigma_i$. Let Σ_i be partitioned into controllable and uncontrollable events (i.e., $\Sigma_i = \Sigma_{ic} \dot{\cup} \Sigma_{iu}$) such that Eq. 3.1 holds. Let $H_i \subseteq \Sigma_i^*$. Let $i, j \in \{1, 2, \dots, n\}$ and $i \neq j$. We say that H_i and H_j are *mutually controllable*¹ if

$$\begin{aligned} \overline{H_i}(\Sigma_{ju} \cap \Sigma_i) \cap p_i^{ij}((p_j^{ij})^{-1}(\overline{H_j})) &\subseteq \overline{H_i}, \text{ and} \\ \overline{H_j}(\Sigma_{iu} \cap \Sigma_j) \cap p_j^{ij}((p_i^{ij})^{-1}(\overline{H_i})) &\subseteq \overline{H_j}, \end{aligned}$$

where p_i^{ij} and p_j^{ij} are the natural projections from $(\Sigma_i \cup \Sigma_j)^*$ to Σ_i^* and Σ_j^* , respectively. Two languages H_i and H_j are mutually controllable if H_i is controllable with respect to $p_i^{ij}((p_j^{ij})^{-1}(\overline{H_j}))$ and the shared uncontrollable events $(\Sigma_{ju} \cap \Sigma_i)$, and H_j is controllable

¹The term ‘mutual controllability’ was suggested by Murray Wonham.

with respect to $p_j^{ij}((p_i^{ij})^{-1}(\overline{H}_i))$ and $(\Sigma_{iu} \cap \Sigma_j)$. Let G_i be a generator of H_i with event alphabet Σ_i . The language $(p_j^{ij})^{-1}(\overline{H}_j)$ models the behaviour of G_j with selflooping of the ‘internal’ events of G_i (i.e. $\Sigma_i - \Sigma_j$), and the language $p_i^{ij}((p_j^{ij})^{-1}(\overline{H}_j))$ represents the behaviour as seen by G_i . Roughly speaking $p_i^{ij}((p_j^{ij})^{-1}(\overline{H}_j))$ models the ‘external’ behaviour of G_j as seen by G_i . Thus, two systems are mutually controllable if the closed behaviour of one system is controllable with respect to the ‘external’ behaviour of the other system for the uncontrollable shared events, and vice versa.

The following theorem provides sufficient conditions for Problem 3.1.

Theorem 3.1 *Let $\Sigma_i, \Sigma_{ic}, \Sigma_{iu}, L_{i,m}, L_i$ ($i = 1, 2, \dots, n$), L_m , and L be given as in Section 3.2. Suppose that for $i, j \in \{1, 2, \dots, n\}$ and $i \neq j$,*

- i) $L_{i,m}$ and $L_{j,m}$ mark $\Sigma_i \cap \Sigma_j$,*
- ii) L_i and L_j are mutually controllable.*

Then Problem 3.1 is solved, namely, $E_i \in \mathcal{F}_{L_{i,m}}$, Eq. 3.4 and 3.5 hold. □

Thus, Problem 3.1 is solved if the marked behaviours of any two subsystems mark their shared events and any two subsystems are mutually controllable. We shall call the condition (i) as the ‘shared-event-marking’ condition and the condition (ii) as the ‘mutual controllability’ condition.

One intuitive interpretation of the shared-event-marking condition may be the following. Since a shared event can potentially be ‘disabled’ by supervisors for other subsystems, one way to ensure that this disablement does not cause blocking in a given subsystem may be to structure the subsystem in such a way that, synchronisations by shared events are permitted only after the subsystem has completed certain stages of its task, i.e., has reached a marker state. Thus, even if the synchronisation is held up by the other subsystem, it will not cause blocking in the given system.

Note that

$$L_{i,m} \text{ and } L_{j,m} \text{ mark } \Sigma_i \cap \Sigma_j \text{ (} i \neq j \text{),}$$

$$\iff L_{i,m} \text{ marks } (\Sigma_i \cap (\bigcup_{j \neq i} \Sigma_j)), \text{ for all } i = 1, 2, \dots, n.$$

The former statement says that any pair of systems marks their shared events and the lat-

ter states that a system marks all its shared events with all the other subsystems. They are equivalent. Since the shared events represent interactions with other subsystems, the set $(\Sigma_i \cap (\bigcup_{j \neq i} \Sigma_j))$ can be interpreted as all possible interactions of the plant G_i with the ‘outside world’.

For the mutual controllability condition in Theorem 3.1, one example of uncontrollable shared events may be the interrupts between an I/O device and the CPU in a computer. In this example the condition requires that a subsystem should be able to track interrupts generated by the other system. We note that the requirement in [WH91] that all shared events be controllable certainly implies our condition.

Since the mutual controllability condition involves projection, the complexity of verifying it might be exponential in the worst case [Rud88]. But the verification needs to be done only once for a given system. After the two conditions in Theorem 3.1 have been verified, decentralised control is achieved for all future syntheses. This cost in complexity may pay off in the long run. This agrees with the intuition that if the systems are structured properly then the operation of these systems will be easier. In the special situation when all the shared events are controllable, only the ‘shared-event-marking’ condition needs to be checked and the complexity for that is linear in the numbers of states and transitions.

For the proof of Theorem 3.1, one requires the following lemmas.

Lemma 3.1 *Let $\Sigma_1, \Sigma_2, \dots, \Sigma_n$ be event alphabets. It may be that $\Sigma_i \cap \Sigma_j \neq \phi$, for $i, j \in \{1, 2, \dots, n\}$ and $i \neq j$. Let $\Sigma = \bigcup_{i=1}^n \Sigma_i$ and p_i be the natural projection from Σ^* to Σ_i^* . Let $H_i \subseteq \Sigma_i^*$. Suppose that for $i, j \in \{1, 2, \dots, n\}$ and $i \neq j$,*

$$H_i \text{ and } H_j \text{ mark } \Sigma_i \cap \Sigma_j.$$

Then

$$\{p_i^{-1}(H_i) \mid i = 1, 2, \dots, n\} \text{ are nonconflicting.}$$

Furthermore,

$$p_i(H) \text{ is nonblocking with respect to } H_i \text{ for } i = 1, 2, \dots, n,$$

where $H := \bigcap_{i=1}^n p_i^{-1}(H_i)$.

Proof: Since one inclusion is always true, we will show only

$$\bigcap_{i=1}^n \overline{p_i^{-1}(H_i)} \subseteq \overline{\bigcap_{i=1}^n p_i^{-1}(H_i)}.$$

Let $s \in \overline{\bigcap_{i=1}^n p_i^{-1}(H_i)} = \overline{p_1^{-1}(H_1) \cap p_2^{-1}(H_2) \cap \dots \cap p_n^{-1}(H_n)}$. Then there exist $u_1, u_2, \dots, u_n \in \Sigma^*$ such that $su_i \in p_i^{-1}(H_i)$, for $i = 1, 2, \dots, n$. Thus

$$p_i(su_i) \in H_i, \text{ or } p_i(s)v_i \in H_i,$$

for some $v_i \in \Sigma_i^*$. Consider for the case $i = 1$. If there is an event $\sigma_1 \in (\Sigma_1 \cap (\bigcup_{j \neq 1} \Sigma_j))$ such that $v_1 = v'_1 \sigma_1 v''_1$ for some $v'_1, v''_1 \in \Sigma_1^*$, then pick v'_1 such that $v'_1 \in (\Sigma_1 - (\bigcup_{j \neq 1} \Sigma_j))^*$. Since $\sigma_1 \in (\bigcup_{j \neq 1} \Sigma_j)$, say $\sigma_1 \in \Sigma_2$. Thus by the assumption that H_1 marks $\Sigma_1 \cap \Sigma_2$,

$$p_1(s)v'_1 \sigma_1 \in \Sigma_1^*(\Sigma_1 \cap \Sigma_2) \cap \overline{H_1} \subseteq H_1(\Sigma_1 \cap \Sigma_2).$$

So $p_1(s)v'_1 \in H_1$. If not, then $p_1(s)v_1 \in H_1$ and $v_1 \in (\Sigma_1 - (\bigcup_{j \neq 1} \Sigma_j))^*$. Thus without loss of generality, one has that $p_1(s)v_1 \in H_1$ and $v_1 \in (\Sigma_1 - (\bigcup_{j \neq 1} \Sigma_j))^*$. Similarly, one has that $p_i(s)v_i \in H_i$, where $v_i \in (\Sigma_i - (\bigcup_{j \neq i} \Sigma_j))^*$. Thus $p_i(sv_1 v_2 \dots v_n) = p_i(s)v_i \in H_i$, for $i = 1, 2, \dots, n$. Hence $sv_1 v_2 \dots v_n \in \bigcap_{i=1}^n p_i^{-1}(H_i)$. Therefore

$$s \in \overline{\bigcap_{i=1}^n p_i^{-1}(H_i)}.$$

It is clear that $p_i(H)$ is prefix-closed. To show that $p_i(H)$ is nonblocking with respect to H_i , one only needs to show that $p_i(H) \subseteq \overline{p_i(H) \cap H_i}$ as the other inclusion is always true. We show the case $i = 1$. Let $s \in p_1(H)$. Then there is a string $u \in H$ such that $s = p_1(u)$. Since $u \in H = \bigcap_{i=1}^n p_i^{-1}(\overline{H_i})$, one has that $p_i(u) \in \overline{H_i}$ for $i = 1, 2, \dots, n$. So there is a string $v_i \in \Sigma_i^*$ such that $p_i(u)v_i \in H_i$. If there is an event $\sigma_i \in (\Sigma_i \cap (\bigcup_{j \neq i} \Sigma_j))$ such that $v_i = v'_i \sigma_i v''_i$ for some $v'_i \in (\Sigma_i - (\bigcup_{j \neq i} \Sigma_j))^*$ and $v''_i \in \Sigma_i^*$, by the assumption that H_i and H_j marks $\Sigma_i \cap \Sigma_j$, and the same argument as above, one has that $p_i(u)v'_i \in H_i \subseteq \overline{H_i}$. If not, again $p_i(u)v_i \in H_i \subseteq \overline{H_i}$ and $v_i \in (\Sigma_i - (\bigcup_{j \neq i} \Sigma_j))^*$. Without loss of generality, one has that $p_i(u)v_i \in \overline{H_i}$ and $v_i \in (\Sigma_i - (\bigcup_{j \neq i} \Sigma_j))^*$. Thus $p_i(uv_1 v_2 \dots v_n) = p_i(u)v_i \in H_i$, for $i = 1, 2, \dots, n$. Therefore

$$uv_1 v_2 \dots v_n \in p_1^{-1}(\overline{H_1}) \cap p_2^{-1}(\overline{H_2}) \cap \dots \cap p_n^{-1}(\overline{H_n}) = H.$$

Thus

$$p_1(uv_1 v_2 \dots v_n) = p_1(u)v_1 = sv_1 \in p_1(H).$$

Hence

$$sv_1 \in p_1(H) \cap H_1, \text{ or } s \in \overline{p_1(H) \cap H_1}.$$

The other cases can be shown similarly. □

Lemma 3.2 Let $H := \bigcap_{i=1}^n p_i^{-1}(H_i)$ and $E_i \subseteq H_i$, for $i = 1, 2, \dots, n$. Suppose that $p_1^{-1}(E_1), p_2^{-1}(E_2), \dots, p_n^{-1}(E_n)$ are nonconflicting. Then

$(p_1|_H)^{-1}(E_1), (p_2|_H)^{-1}(E_2), \dots, (p_n|_H)^{-1}(E_n)$ are nonconflicting.

Proof: As the other inclusion is obvious, we only need to show the following.

$$\begin{aligned}
& \overline{(p_1|_H)^{-1}(E_1) \cap (p_2|_H)^{-1}(E_2) \cap \dots \cap (p_n|_H)^{-1}(E_n)} \\
\subseteq & \overline{p_1^{-1}(E_1) \cap p_2^{-1}(E_2) \cap \dots \cap p_n^{-1}(E_n)} \\
= & \overline{p_1^{-1}(E_1) \cap p_2^{-1}(E_2) \cap \dots \cap p_n^{-1}(E_n)} && \text{since } p_1^{-1}(E_1), p_2^{-1}(E_2), \dots, \\
& && p_n^{-1}(E_n) \text{ are nonconflicting} \\
= & \overline{p_1^{-1}(E_1) \cap p_2^{-1}(E_2) \cap \dots \cap p_n^{-1}(E_n) \cap H} \\
= & \overline{(p_1|_H)^{-1}(E_1) \cap (p_2|_H)^{-1}(E_2) \cap \dots \cap (p_n|_H)^{-1}(E_n)}.
\end{aligned}$$

□

Let H be a language over Σ and $\Sigma' \subseteq \Sigma$. The property that H marks Σ' is generally not closed under inclusion, i.e. for $F \subseteq H$ it is generally not true that F marks Σ' as shown in the following example.

Example 3.5 Let $\Sigma = \{\alpha, \beta\}$ and $\Sigma' = \{\beta\}$. Let $H = \{\alpha, \alpha\beta\alpha\}$ and $F = \{\alpha\beta\alpha\}$. Then $\alpha\beta \in \Sigma^*\Sigma' \cap \overline{F}$ but $\alpha\beta \notin F\Sigma'$, i.e. F does not mark Σ' . ◇

However, we do have the following.

Lemma 3.3 Let H be a language over Σ and $\Sigma' \subseteq \Sigma$. Suppose that H marks Σ' and $F \in \mathcal{F}_H$, i.e. F is H -closed. Then F marks Σ'

Proof: Let $s\sigma \in \Sigma^*\Sigma' \cap \overline{F}$. Since $F \subseteq H$, $s\sigma \in \Sigma^*\Sigma' \cap \overline{H}$. Since H marks Σ' , $s\sigma \in H\Sigma'$. Thus $s \in H$. Since $F \in \mathcal{F}_H$, $s \in \overline{F} \cap H = F$. □

Thus, if a language H marks a set of events Σ' , then the property of marking Σ' is inherited by all the sublanguages that are closed relative to the language H .

Lemma 3.4 *Under the conditions in Theorem 3.1, for $i = 1, 2, \dots, n$ and for $H \in \mathcal{F}_{L_{i,m}}$,*

$$\overline{(p_i|_L)^{-1}(H)} = (p_i|_L)^{-1}(\overline{H}).$$

With reference to Proposition 8 in [WW96a], $(p_i|_L)$ may be seen as to have a restricted observer property.

Proof: We show the case $i = 1$. Note that for all $A \subseteq \Sigma_i^*$, $\overline{p_i^{-1}(A)} = p_i^{-1}(\overline{A})$. Let $H \in \mathcal{F}_{L_{1,m}}$.

As $\overline{(p_1|_L)^{-1}(H)} \subseteq (p_1|_L)^{-1}(\overline{H})$ is always true, one only needs to show the other inclusion.

One has that

$$\begin{aligned} (p_1|_L)^{-1}(\overline{H}) &= p_1^{-1}(\overline{H}) \cap L \\ &= p_1^{-1}(\overline{H}) \cap p_1^{-1}(L_1) \cap p_2^{-1}(L_2) \cap \dots \cap p_n^{-1}(L_n) \\ &= p_1^{-1}(\overline{H} \cap L_1) \cap p_2^{-1}(L_2) \cap \dots \cap p_n^{-1}(L_n) \\ &= p_1^{-1}(\overline{H}) \cap p_2^{-1}(\overline{L_{2,m}}) \cap \dots \cap p_n^{-1}(\overline{L_{n,m}}) \\ &= \overline{p_1^{-1}(H) \cap p_2^{-1}(L_{2,m}) \cap \dots \cap p_n^{-1}(L_{n,m})}. \end{aligned}$$

Since $L_{i,m}$ and $L_{j,m}$ mark $\Sigma_i \cap \Sigma_j$, for $i, j \in \{1, 2, \dots, n\}$ and $i \neq j$ (an assumption in Theorem 3.1), and H is $L_{1,m}$ -closed, by Lemma 3.3, H also marks $\Sigma_1 \cap \Sigma_i$ for $i = 2, 3, \dots, n$.

So, by Lemma 3.1

$$\begin{aligned} (p_1|_L)^{-1}(\overline{H}) &= \overline{p_1^{-1}(H) \cap p_2^{-1}(L_{2,m}) \cap \dots \cap p_n^{-1}(L_{n,m})} \\ &= \overline{p_1^{-1}(H) \cap p_2^{-1}(L_{2,m}) \cap \dots \cap p_n^{-1}(L_{n,m})} \\ &= \overline{p_1^{-1}(H \cap L_{1,m}) \cap p_2^{-1}(L_{2,m}) \cap \dots \cap p_n^{-1}(L_{n,m})} \quad \text{since } H \subseteq L_{1,m} \\ &= \overline{p_1^{-1}(H) \cap p_1^{-1}(L_{1,m}) \cap p_2^{-1}(L_{2,m}) \cap \dots \cap p_n^{-1}(L_{n,m})} \\ &= \overline{p_1^{-1}(H) \cap L_m} \\ &\subseteq \overline{p_1^{-1}(H) \cap L} \\ &= \overline{(p_1|_L)^{-1}(H)}. \end{aligned}$$

□

The following lemma provides a technical result which will be used later.

Lemma 3.5 *For three event sets Σ_1, Σ_2 , and Σ_3 , let $\Sigma_{23} = (\Sigma_2 \cup \Sigma_3)$ and $\Sigma = (\Sigma_1 \cup \Sigma_2 \cup \Sigma_3)$.*

Let $p_i : \Sigma^ \rightarrow \Sigma_i^*$ ($i = 1, 2, 3$) and $p_i^{23} : \Sigma_{23}^* \rightarrow \Sigma_i^*$ ($i = 2, 3$) be the natural projections.*

Then for $L \subseteq \Sigma_3^$,*

$$p_2((p_3)^{-1}(L)) = p_2^{23}((p_3^{23})^{-1}(L)).$$

Proof: For the inclusion (\subseteq), let $s \in p_2((p_3)^{-1}(L)) \subseteq \Sigma_2^*$. Then there exists a string $u \in (p_3)^{-1}(L)$, such that $s = p_2(u)$. So, $p_3(u) \in L$. By Lemma 2.2 in Section 2.7, $p_2(u) = p_2 \circ p_{23}(u)$, where p_{23} is the natural projection from Σ^* to Σ_{23}^* . Let $v := p_{23}(u) \in \Sigma_{23}^*$. Then $s = p_2(u) = p_2 \circ p_{23}(u) = p_2(v)$. Also by Lemma 2.3 in Section 2.7, $p_2^{23}(v) = p_2|_{\Sigma_{23}^*}(v) = p_2(v)$ (since $v \in \Sigma_{23}^*$). So $s = p_2(u) = p_2(v) = p_2^{23}(v)$. Also, since $p_3^{23}(v) = p_3^{23}p_{23}(u) = p_3(u) \in L$, $v \in (p_3^{23})^{-1}(L)$. Therefore, $s = p_2(v) = p_2^{23}(v) \in p_2^{23}((p_3^{23})^{-1}(L))$.

For the other inclusion, let $s \in p_2^{23}((p_3^{23})^{-1}(L))$. Then there exists a string $u \in (p_3^{23})^{-1}(L)$ such that $s = p_2^{23}(u)$. Since $u \in \Sigma_{23}^*$, by Lemma 2.3, one has that $s = p_2^{23}(u) = p_2|_{\Sigma_{23}^*}(u) = p_2(u)$. Since $u \in (p_3^{23})^{-1}(L) \subseteq (p_3)^{-1}(L)$, $u \in (p_3)^{-1}(L)$. Therefore $s = p_2(u) \in p_2((p_3)^{-1}(L))$. \square

Let $\mathcal{C}(L)$ and $\mathcal{C}(L_i)$ be respectively the sets of controllable sublanguages of L and L_i .

Lemma 3.6 *Under the conditions in Theorem 3.1,*

- (1) for all $K \in \mathcal{C}(L_i) \cap \mathcal{F}_{L_i, m}$, $(p_i|_L)^{-1}(K) \in \mathcal{C}(L)$;
- (2) for all $K \in \mathcal{C}(L)$, $p_i(K) \in \mathcal{C}(L_i)$.

Proof: To show (1), let $K \in \mathcal{C}(L_i) \cap \mathcal{F}_{L_i, m}$ and $K' := (p_i|_L)^{-1}(K)$. To show that K' is controllable with respect to L , let $s\sigma \in \overline{K'}\Sigma_u \cap L$. Thus $s \in \overline{K'}$, $\sigma \in \Sigma_u$, and $s\sigma \in L$.

So $s \in \overline{(p_i|_L)^{-1}(K)} \subseteq (p_i|_L)^{-1}(\overline{K})$, or $p_i(s) \in \overline{K}$. If $\sigma \in \Sigma_i$ then

$$p_i(s\sigma) = p_i(s)\sigma \in p_i(L) \subseteq L_i.$$

By Eq. 3.2, $\sigma \in \Sigma_i \cap \Sigma_u = \Sigma_{iu}$. Thus, since $K \in \mathcal{C}(L_i)$,

$$p_i(s\sigma) = p_i(s)\sigma \in \overline{K}\Sigma_{iu} \cap L_i \subseteq \overline{K}, \text{ or } s\sigma \in p_i^{-1}(\overline{K}).$$

Since $s\sigma \in L$, $s\sigma \in p_i^{-1}(\overline{K}) \cap L = (p_i|_L)^{-1}(\overline{K})$. Since $K \in \mathcal{F}_{L_i, m}$, by Lemma 3.4

$$s\sigma \in (p_i|_L)^{-1}(\overline{K}) = \overline{(p_i|_L)^{-1}(K)} = \overline{K'}.$$

If $\sigma \notin \Sigma_i$ then $p_i(s\sigma) = p_i(s) \in \overline{K}$. By the same argument as above, $s\sigma \in \overline{K'}$. Hence $K' \in \mathcal{C}(L)$.

To show (2), we show the case $i = 1$. Let $K \in \mathcal{C}(L)$ and $K' = p_1(K)$. To show $K' \in \mathcal{C}(L_1)$, let $s\sigma \in \overline{K'}\Sigma_{1u} \cap L_1$. Thus $s \in \overline{K'}$, $\sigma \in \Sigma_{1u}$, and $s\sigma \in L_1$. So $s \in \overline{p_1(K)} = p_1(\overline{K})$. There is a string $u \in \overline{K}$ such that $s = p_1(u)$. Since $u \in \overline{K} \subseteq L = \bigcap_{i=1}^n p_i^{-1}(L_i)$, one has that $p_i(u) \in L_i$, for $i = 1, 2, \dots, n$. Since $\sigma \in \Sigma_{1u} \subseteq \Sigma_1$ and $p_1(u) = s$,

$$p_1(u\sigma) = p_1(u)\sigma = s\sigma \in L_1, \text{ or } u\sigma \in p_1^{-1}(L_1).$$

For $j = 2, 3, \dots, n$, if $\sigma \notin \Sigma_j$, then $p_j(u\sigma) = p_j(u) \in L_j$, i.e., $u\sigma \in p_j^{-1}(L_j)$; if $\sigma \in \Sigma_j$ then $\sigma \in \Sigma_{1u} \cap \Sigma_j$. Thus $p_j(u\sigma) = p_j(u)\sigma \in L_j(\Sigma_{1u} \cap \Sigma_j)$. Since $u\sigma \in p_1^{-1}(L_1)$ and by Lemma 3.5, $p_j(u\sigma) \in p_j((p_1)^{-1}(L_1)) = p_j^{1j}((p_1^{1j})^{-1}(L_1))$. So by the mutual controllability of L_1 and L_j (an assumption in Theorem 3.1),

$$p_j(u\sigma) \in L_j(\Sigma_{1u} \cap \Sigma_j) \cap p_j^{1j}((p_1^{1j})^{-1}(L_1)) \subseteq L_j.$$

So $u\sigma \in p_j^{-1}(L_j)$. Thus $u\sigma \in \bigcap_{i=1}^n p_i^{-1}(L_i) = L$. Since $\sigma \in \Sigma_{1u} \subseteq \Sigma_u$ and K is controllable, $u\sigma \in \overline{K}\Sigma_u \cap L \subseteq \overline{K}$.

Thus $p_1(u\sigma) = p_1(u)\sigma = s\sigma \in p_1(\overline{K}) = \overline{p_1(K)} = \overline{K'}$. So $K' \in \mathcal{C}(L_1)$. \square

Lemma 3.7 *Under the conditions in Theorem 3.1, for $i = 1, 2, \dots, n$ and for $E_i \in \mathcal{F}_{L_i, m}$,*

$$(p_i|_L)^{-1}\kappa_{L_i}(E_i) = \kappa_L(p_i|_L)^{-1}(E_i).$$

Proof: Let $E_i \in \mathcal{F}_{L_i, m}$ for $i = 1, 2, \dots, n$. Then $\kappa_{L_i}(E_i) \in \mathcal{C}(L_i)$. Since $L_{i, m}$ -closedness is closed under κ_{L_i} (Proposition 6.1 in [WR88]),

$$\kappa_{L_i}(E_i) \in \mathcal{C}(L_i) \cap \mathcal{F}_{L_i, m}.$$

By (1) in Lemma 3.6, $(p_i|_L)^{-1}\kappa_{L_i}(E_i) \in \mathcal{C}(L)$. Since $(p_i|_L)^{-1}\kappa_{L_i}(E_i) \subseteq (p_i|_L)^{-1}(E_i)$,

$$(p_i|_L)^{-1}\kappa_{L_i}(E_i) \subseteq \kappa_L(p_i|_L)^{-1}(E_i).$$

For the reverse inclusion, since $\kappa_L(p_i|_L)^{-1}(E_i) \in \mathcal{C}(L)$, by (2) in Lemma 3.6,

$$p_i\kappa_L(p_i|_L)^{-1}(E_i) \in \mathcal{C}(L_i).$$

Since $p_i\kappa_L(p_i|_L)^{-1}(E_i) \subseteq p_i(p_i|_L)^{-1}(E_i) \subseteq E_i$,

$$p_i\kappa_L(p_i|_L)^{-1}(E_i) \subseteq \kappa_{L_i}(E_i).$$

Thus $\kappa_L(p_i|_L)^{-1}(E_i) \subseteq p_i^{-1}\kappa_{L_i}(E_i)$. Since $\kappa_L(p_i|_L)^{-1}(E_i) \subseteq L$,

$$\kappa_L(p_i|_L)^{-1}(E_i) \subseteq (p_i|_L)^{-1}\kappa_{L_i}(E_i). \quad \square$$

Now, one is ready to prove Theorem 3.1.

Proof of Theorem 3.1: Let $E_i \in \mathcal{F}_{L_i, m}$, for $i = 1, 2, \dots, n$. Since $L_{i, m}$ -closedness is closed under κ_{L_i} (Proposition 6.1 [WR88]), $\kappa_{L_i}(E_i) \in \mathcal{F}_{L_i, m}$. By the assumption that $L_{i, m}$ and $L_{j, m}$ mark the shared events $\Sigma_i \cap \Sigma_j$ for $i, j \in \{1, 2, \dots, n\}$ and $i \neq j$, and Lemma 3.3, $\kappa_{L_i}(E_i)$ and $\kappa_{L_j}(E_j)$ also mark $\Sigma_i \cap \Sigma_j$. By Lemma 3.1,

$$\{p_i^{-1}(\kappa_{L_i}(E_i)) \mid i = 1, 2, \dots, n\} \text{ are nonconflicting.}$$

By Lemma 3.2,

$$\{(p_i|_L)^{-1}(\kappa_{L_i}(E_i)) \mid i = 1, 2, \dots, n\} \text{ are nonconflicting.} \quad (3.7)$$

Since $E_i \in \mathcal{F}_{i,m}$, by Lemma 3.7 $(p_i|_L)^{-1}\kappa_{L_i}(E_i) = \kappa_L(p_i|_L)^{-1}(E_i)$. Hence

$$\{\kappa_L((p_i|_L)^{-1}(E_i)) \mid i = 1, 2, \dots, n\} \text{ are nonconflicting.} \quad (3.8)$$

Now it can be shown in turn, for $i = 1, 2, \dots, n$,

$$\begin{aligned} & \overline{\bigcap_i (p_i|_L)^{-1}(\kappa_{L_i}(E_i))} \\ = & \overline{\bigcap_i ((p_i|_L)^{-1}(\kappa_{L_i}(E_i)))} && \text{since } \kappa_{L_i}(E_i) \in \mathcal{F}_{L_i,m} \\ & && \text{and by Lemma 3.4,} \\ = & \overline{\bigcap_i (p_i|_L)^{-1}(\kappa_{L_i}(E_i))} && \text{by Eq. 3.7,} \\ = & \overline{\bigcap_i (\kappa_L(p_i|_L)^{-1}(E_i))} && \text{since } E_i \in \mathcal{F}_{i,m} \\ & && \text{and by Lemma 3.7,} \\ = & \overline{\kappa_L(\bigcap_i ((p_i|_L)^{-1}(E_i)))} && \text{by Eq. 3.8 and} \\ & && \text{Proposition 3.1} \\ = & \overline{\kappa_L(E)}. \end{aligned}$$

To show that $p_i(\overline{\kappa_L(E)})$ is nonblocking with respect to $L_{i,m}$, we observe that $\overline{\kappa_L(E)} = p_1^{-1}(\overline{\kappa_{L_1}(E_1)}) \cap p_2^{-1}(\overline{\kappa_{L_2}(E_2)}) \cap \dots \cap p_n^{-1}(\overline{\kappa_{L_n}(E_n)})$. By Lemma 3.1, $p_i(\overline{\kappa_L(E)})$ is nonblocking with respect to $\kappa_{L_i}(E_i)$. Since $\kappa_{L_i}(E_i) \subseteq L_{i,m}$, it follows that $p_i(\overline{\kappa_L(E)})$ is nonblocking with respect to $L_{i,m}$. \square

Since $\kappa_{L_i}(E_i)$ for $i = 1, 2, \dots, n$, is controllable and $L_{i,m}$ -closed, by the standard theory [RW89] a nonblocking supervisor S_i can be synthesised so that its closed-loop marked behaviour with G_i is precisely $\kappa_{L_i}(E_i)$. Provided that the assumptions of $L_{i,m}$ marking the shared events and mutual controllability hold, by our result these supervisors implement the behaviour of an ‘optimal’ supervisor that is synthesised for the overall specification. In particular there will not be conflicts between the ‘global’ implementations of the S_i ’s and control of the supervisor in one subsystem will never cause blocking in the other.

3.4 Example

In this section we illustrate our result with two examples. All the computations in this section are carried out with TCT [Won96]. In the first example, we consider two machine cells G_1 and G_2 that need to cooperate at some stage for processing certain parts. Let the models of the two machine cells be given by the generators in Figure 3.2. Here a state is represented by a circle and an event is described by an arrow from an exit state to an entrance state with an event label attached. The initial state is labelled with an entering arrow ($\rightarrow\circ$), and a marker state is labelled with an exiting arrow ($\circ\rightarrow$). A double arrow ($\circ\leftrightarrow$) indicates that the initial state is also a marker state. The arrow with a 'tick' indicates that the event is controllable.

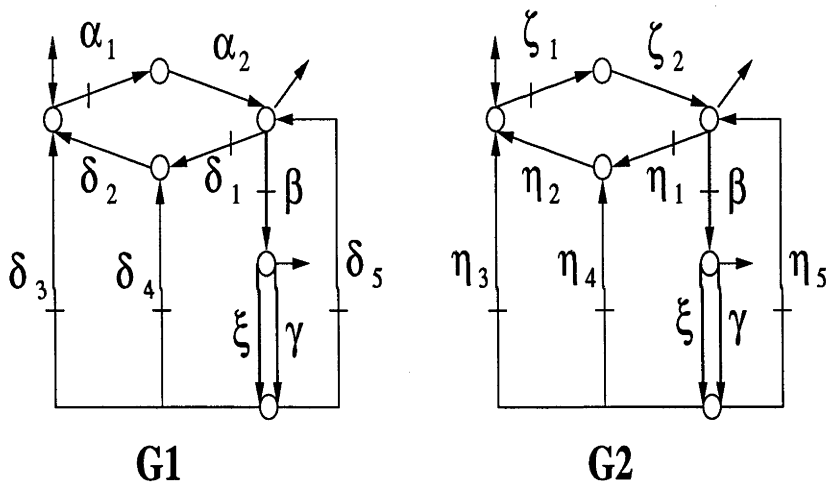


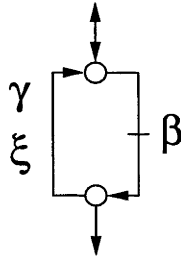
Figure 3.2: Generators for G_1 and G_2

In G_1 , the cycle of α_1 , α_2 , δ_1 , and δ_2 represents the cycle of processing that requires no cooperation with other machine cells. The event β represents the start of the cooperation. The event γ indicates that the cooperation ends successfully, whereas ξ indicates that the cooperation is unsuccessful. The events δ_3 , δ_4 , and δ_5 represent respectively the event of returning to the initial state, the event of rejoining the process in mid-stream, and the event of returning to the state before cooperation. A similar explanation for events in G_2 applies.

The event sets for local systems are

$$\begin{aligned}\Sigma_1 &= \{\alpha_1, \alpha_2, \beta, \gamma, \xi, \delta_1, \delta_2, \delta_3, \delta_4, \delta_5\} \\ \Sigma_2 &= \{\zeta_1, \zeta_2, \beta, \gamma, \xi, \eta_1, \eta_2, \eta_3, \eta_4, \eta_5\}\end{aligned}$$

The shared events $\Sigma_1 \cap \Sigma_2 = \{\beta, \gamma, \xi\}$. The shared uncontrollable events are $\{\gamma, \xi\}$. Let $L_{i,m} := L_m(\mathbf{G}_i)$ and $L_i := L(\mathbf{G}_i)$. It is easy to see that the ‘shared-event-marking’ assumption in Theorem 3.1 holds in this example since all the state before the shared events are marked. To check the mutual controllability condition (i.e. $L_i(\Sigma_{iu} \cap \Sigma_j) \cap p_i p_j^{-1}(L_j) \subseteq L_i$), the language $p_1 p_2^{-1}(L_2)$ is computed and is displayed in Figure 3.3. It can be verified that



Selfloop α_i and δ_j for $i=1,2$ and $j=1,2, \dots, 5$.

Figure 3.3: A generator for $p_1 p_2^{-1}(L_2)$

$L_1\{\gamma, \xi\} \cap p_1 p_2^{-1}(L_2) \subseteq L_1$ indeed holds. The other inclusion for L_2 holds by symmetry.

Let the specifications E_1 and E_2 be the marked behaviours of the generators in Figure 3.4. Clearly E_i is $L_{i,m}$ -closed for $i = 1, 2$. Thus in both specifications after successful cooperation (the event γ) the systems are returned to their respective initial states and perhaps new parts are accepted into the workcells. If the cooperation is not successful, then in E_1 a re-attempt at cooperation immediately afterwards is possible and in E_2 an immediate re-attempt at cooperation will not be made.

It is checked that $\kappa_{L_i}(E_i) = E_i$. We also compute $L = L_1 \parallel L_2$, $L_m = L_{1,m} \parallel L_{2,m}$, and $E = (p_1|_L)^{-1}(E_1) \cap (p_2|_L)^{-1}(E_2)$. The global synthesis is carried out. It can be verified that Eq. 3.4 and 3.5 hold.

To illustrate our result with a more practical example, we consider a chemical batch reactor described in [Als96]. The reactor is comprised of a feed valve V_1, a feed pump P_1, a drain

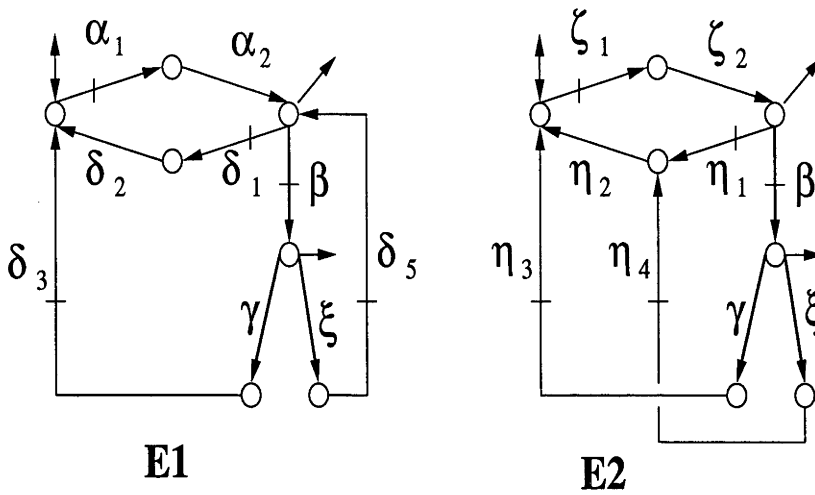


Figure 3.4: Generators for E_1 and E_2

valve V_2, a drain pump P_2, a weight measurement unit W_1, a timer T_1, a heater with a temperature probe TP_1 and a continuous reaction controller C_1 (see Figure 3.5). DES models for the elementary components are presented in Figure 3.6.

The batch reaction process operates as follows. Firstly, the feed valve V_1 is opened (event α_1) and then feed pump P_1 is turned on (event ξ_1) to fill the reactor tank with material until W_1 indicates that the weight is 100kg (event η_3). When the weight reaches 100kg, the feed pump is turned off (event λ_1) and the feed valve is closed (event β_1). Then the heater is turned on to warm up the material. When the material temperature reaches $30^\circ\text{C} < T < 60^\circ\text{C}$ (event ω_1), the reaction controller C_1 is enabled (event γ). The duration of the reaction is timed by the timer which can be set for 3 minutes (event μ_1), 5 minutes (event μ_2) and 7 minutes (event μ_3). Assume that the controller is required to operate for 5 minutes at the material temperature $30^\circ\text{C} < T < 60^\circ\text{C}$. So, μ_2 is chosen. During the operation, the timer can be held at its current time (event μ_4) to handle some situations such as emergency repairs. From the *held* state, the timer can either be reset (event μ_8) to the *idle* state or re-released (event μ_5). After the set time has expired (uncontrollable event μ_6), the controller C_1 is disabled (event δ) and the timer is reset (event μ_7). We assume that now the reaction will take 5 minutes more at the material temperature $T > 60^\circ\text{C}$. So the material temperature is increased to $T > 60^\circ\text{C}$ (event ω_3) and controller is enabled again. Then, event μ_2 is again chosen. When the time has expired (event

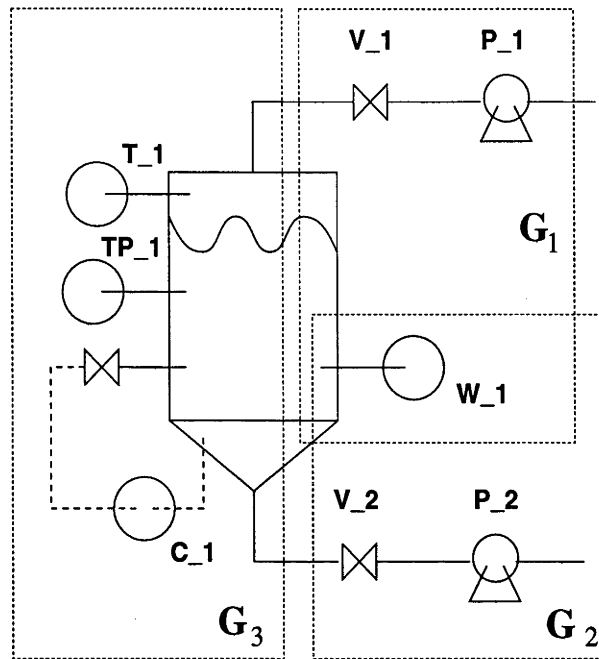


Figure 3.5: A simple batch reactor

μ_6), the controller C_1 is disabled (event δ) and the timer is reset (event μ_7). Before beginning to empty the tank, it is required that the material temperature should be less than 30°C (event ω_2). Then the drain valve V_2 is opened (event α_2) and the drain pump P_2 is turned on (event ξ_2) to discharge the finished product from the tank until W_1 indicates that the tank is empty ($W < 1\text{kg}$) by the event η_2 . When this happens, the drain pump is turned off (event λ_2) and the drain valve is closed (event β_2). This completes one cycle of the batch reaction process.

The synchronous composition of all the elementary components generates the possible system behaviours. However, there exist some physical constraints among the elementary components and they will restrict the system behaviour by deleting infeasible states and transitions. Physical constraints are often resulted from conservation of mass, energy and momentum, gravitational consideration and mechanical connections among the components. The automata displayed in Figure 3.7 represent two physical constraints, namely that the weight can only be increased after the feed pump is turned on (constraint 1) and can only be decreased after the drain pump is turned on (constraint 2).

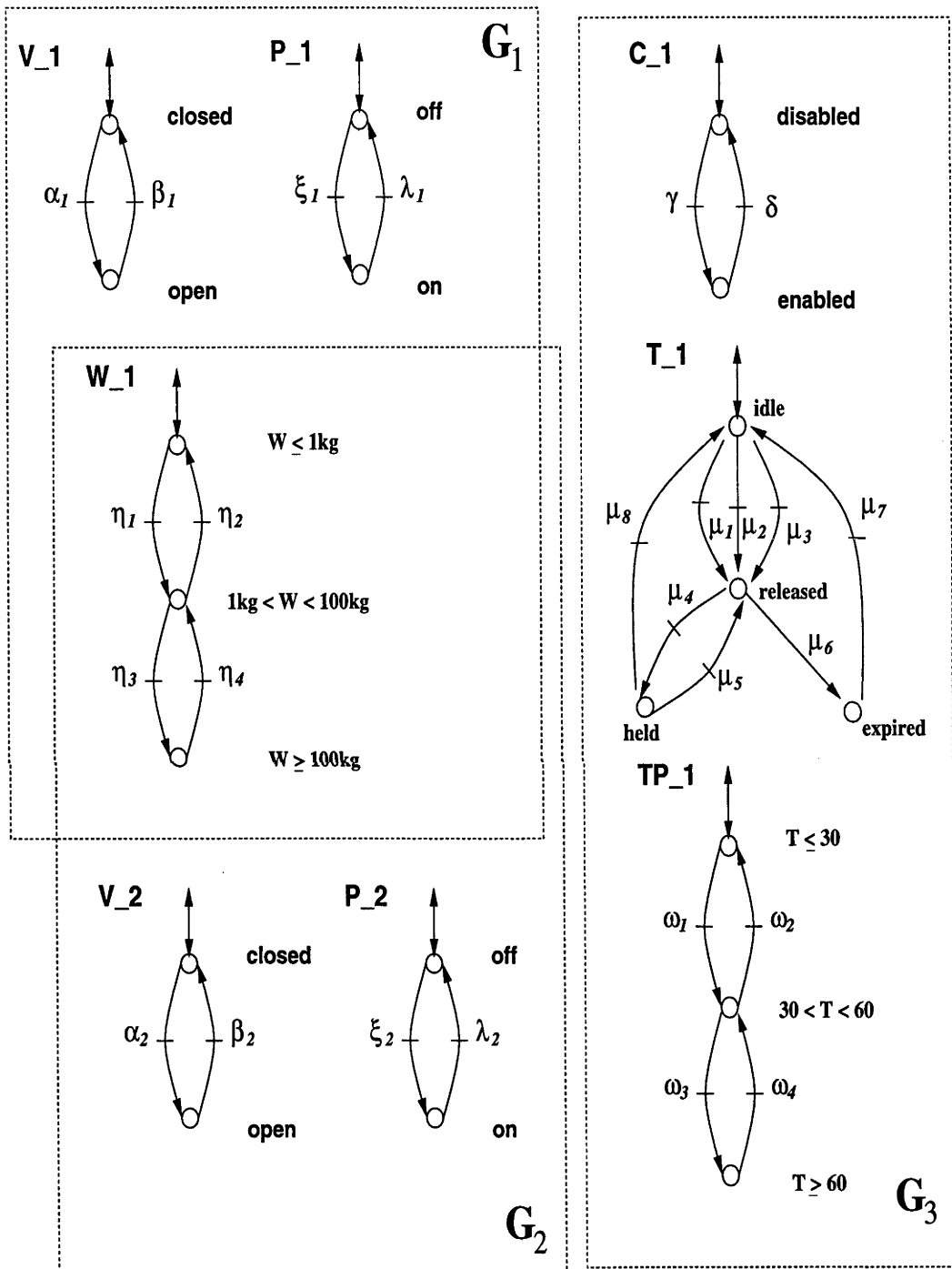


Figure 3.6: DES models for the elementary components of the batch reactor

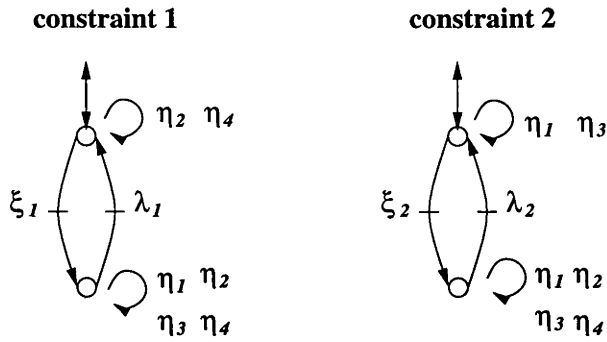


Figure 3.7: Physical constraints of the batch reactor

The complete centralised model formed by the synchronous composition of these elementary components has 1152 states with 10368 transitions.

The DES model of specification for the whole process is given in Figure 3.8. The optimal

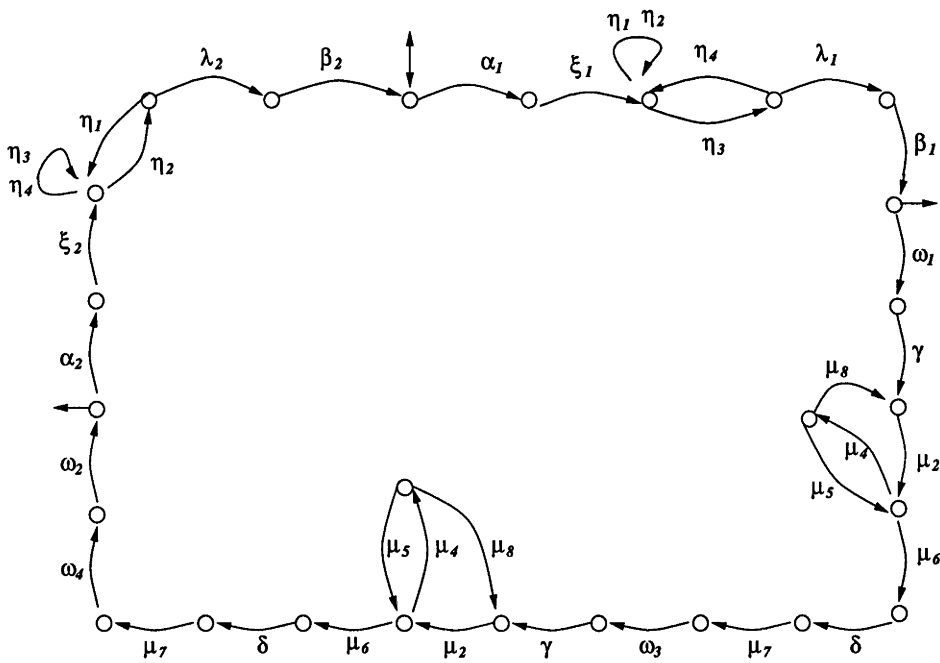


Figure 3.8: The specification for a centralised control of the batch reactor

centralised supervisor which satisfies the given specification is obtained and has 28 states with 30 transitions.

We now consider decentralised control for this example. The batch reaction process is decomposed naturally into three subplants: the filling subplant (G_1), the draining subplant (G_2) and the reaction subplant (G_3). The subplants G_1 , G_2 and G_3 consist of, respectively, V_1, P_1 and W_1; V_2, P_2 and W_1; and T_1, TP_1 and C_1. Note that the component W_1 is the shared component of plants G_1 and G_2 . The operation of the batch process is also divided naturally into three subprocesses according to the decomposition of the plant: the filling process, the reaction process, and the draining process. To enforce that these three processes are operating serially, we introduce three controllable shared events ($\sigma_1, \sigma_2, \sigma_3$). These shared events indicate the completion of the current process and allow the next process to proceed. For example, the event σ_1 represents that the filling process has been completed and the chemical reaction process can now proceed. The event σ_2 represents the completion of the chemical reaction, while σ_3 represents that the draining has finished and indicates a complete cycle of the batch reaction. These controllable synchronised events are represented as 'flag's (Figure 3.9) and considered as a part of the plants.

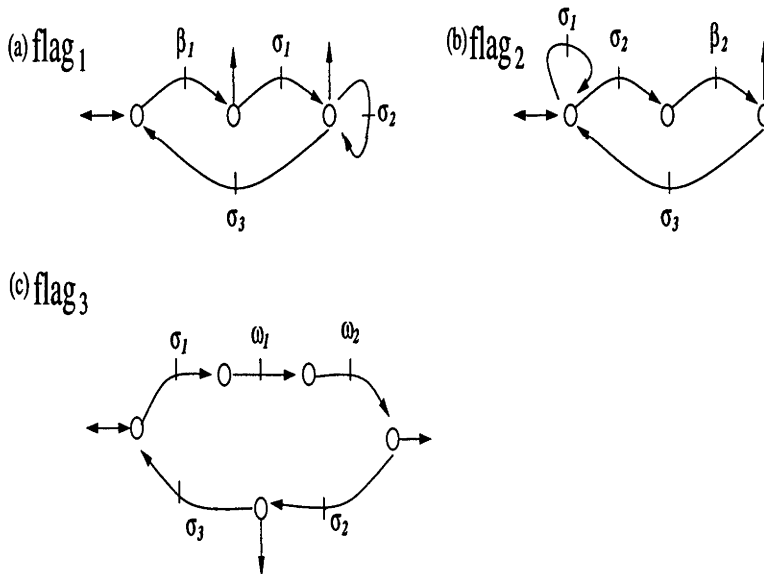


Figure 3.9: Synchronisation flags for decentralised control of the batch reactor

So, the subplants now are;

- G_1 : V_1, P_1, W_1, flag_1.

- G_2 : V_2, P_2, W_1, flag_2.
- G_2 : T_1, TP_1, C_1, flag_3.

The sizes of the plants G_1 and G_2 are 36 states with 132 transitions, while G_3 has 48 states with 200 events. Comparing these with the size of centralised counterpart shows that decentralised control can offer significant savings on computational effort.

The DES models of specifications, E_1 , E_2 , and E_3 (for G_1 , G_2 and G_3 , respectively) are given in Figure 3.10. The corresponding decentralised supervisors for the local specifications are computed. The supervisors for G_1 and G_2 both have 17 states with 29 transitions and that for G_3 has 19 states with 23 events.

The event sets for the subplants are

$$\begin{aligned}\Sigma_1 &= \{\alpha_1, \beta_1, \eta_1, \eta_2, \eta_3, \eta_4, \xi_1, \lambda_1, \sigma_1, \sigma_2, \sigma_3\} \\ \Sigma_2 &= \{\alpha_2, \beta_2, \eta_1, \eta_2, \eta_3, \eta_4, \xi_2, \lambda_2, \sigma_1, \sigma_2, \sigma_3\} \\ \Sigma_3 &= \{\gamma, \delta, \mu_1, \mu_2, \mu_3, \mu_4, \mu_5, \mu_6, \mu_7, \mu_8, \omega_1, \omega_2, \omega_3, \omega_4, \sigma_1, \sigma_2, \sigma_3\}.\end{aligned}$$

The shared events are

$$\begin{aligned}\Sigma_1 \cap \Sigma_2 &= \{\eta_1, \eta_2, \eta_3, \eta_4, \sigma_1, \sigma_2, \sigma_3\} \\ \Sigma_2 \cap \Sigma_3 &= \{\sigma_1, \sigma_2, \sigma_3\} \\ \Sigma_3 \cap \Sigma_1 &= \{\sigma_1, \sigma_2, \sigma_3\}.\end{aligned}$$

We verify that the system structure of this example (G_1 , G_2 and G_3) satisfies the conditions given in Theorem 3.1. Since all shared events are controllable, the ‘mutual controllability’ condition is satisfied trivially. For the ‘shared-event-marking’ condition, we mark all the states before the shared events in G_i and E_i , for $i = 1, 2, 3$. The marking of the shared events seem natural in this case. For example, the marking of the states before the event σ_1 represents the completion of the filling process and the beginning of the chemical reaction process. It can be checked that specification E_i is $L_{i,m}$ -closed language, where $L_{i,m}$ are the marked behaviours of G_i , for $i = 1, 2, 3$. Then, we compute the global system behaviours, $L = L_1 || L_2 || L_3$, $L_m = L_{1,m} || L_{2,m} || L_{3,m}$, and the global specification, $E = (p_1|L)^{-1}(E_1) \cap (p_2|L)^{-1}(E_2) \cap (p_3|L)^{-1}(E_3)$. The global synthesis is now carried out. The result shows that Eq. 3.4 and 3.5 hold.

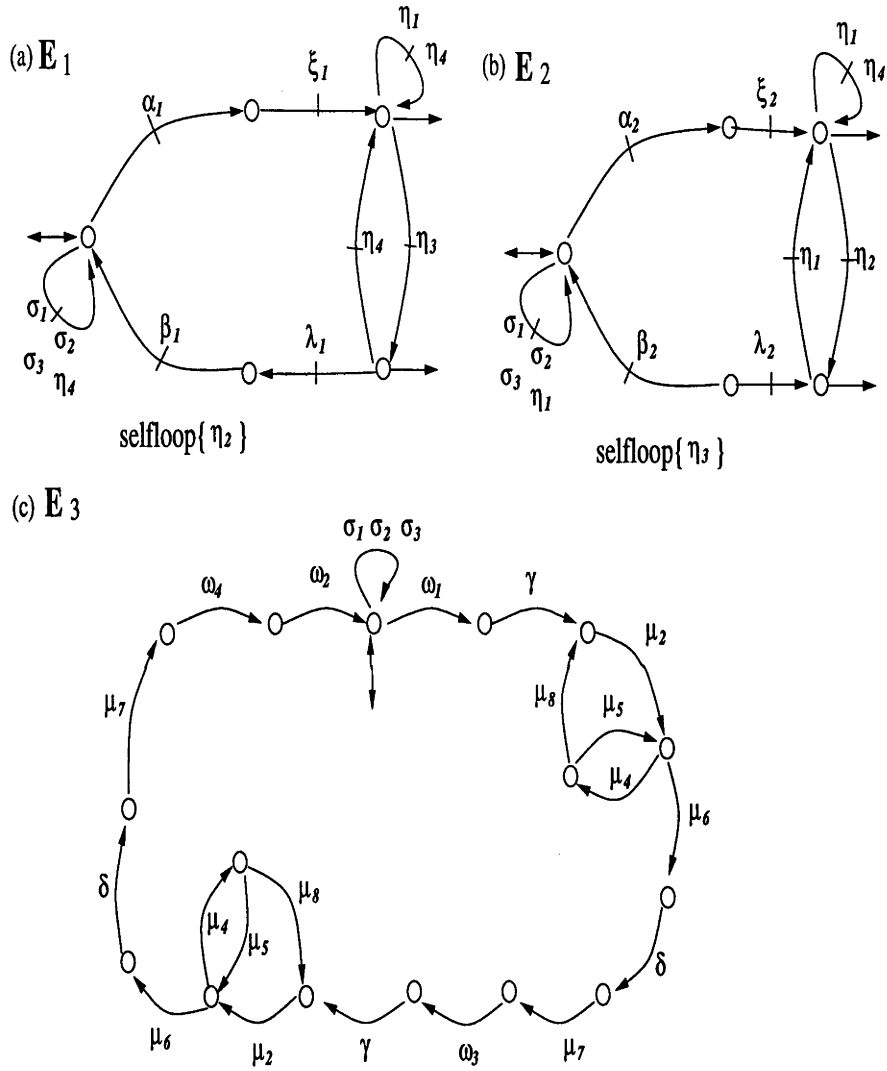


Figure 3.10: Local specifications for decentralised control of the batch reactor

Note that the generator for L after we project out the synchronisation events is not the same as the system model obtained as the synchronous composition of all the elementary components. This is because the synchronisation events σ_1 , σ_2 and σ_3 impose certain orders in the behaviour. For example, without these synchronisation events the draining of material from the tank is possible at the start of the operation, whereas with these synchronisation events, draining is allowed only after the reaction in the tank has been completed. However, the specification E is the same as the specification in Figure 3.8 if the synchronisation events σ_1 , σ_2 and σ_3 are projected out from E . Thus, by introducing the ‘high-level’ order of feeding-reaction-draining into the system structure using these controllable synchronisation events, the specification is decomposed accordingly. With this decomposition, the synthesis may be carried out more efficiently.

Intuitively, it is the requirements within the subprocesses that are often changed (e.g. for different chemicals), whereas the high-level structure like feeding-reaction-draining remains constant for many different requirements. Thus, the other advantage of this decomposition is that any change in specification in a subprocess can be localised and adapted to efficiently by re-computing only the corresponding sub-supervisor. In this case, after verifying the synchronisation structure with the result in Theorem 3.1, the structure can be used for any $L_{i,m}$ -closed local specifications. As a simple illustration, let us consider that the specification for G_2 is changed as follows:

1. Warm up the material to $T > 60^\circ C$ (event ω_3).
2. Enable the controller (event γ) for 3 minutes (event μ_1).
3. After the timer has expired, disable the controller, reset the timer, and then let the material cool down to $30^\circ C < T < 60^\circ C$ (event ω_4).
4. Set the timer and enable C_1 (event γ) for 7 minutes (event μ_3).
5. After the timer has expired, disable the controller and reset the timer.
6. Wait until the material is cooled down to $T < 30^\circ C$ (event ω_2).

DES model for this specification is given in Figure 3.11. Since it has already been verified

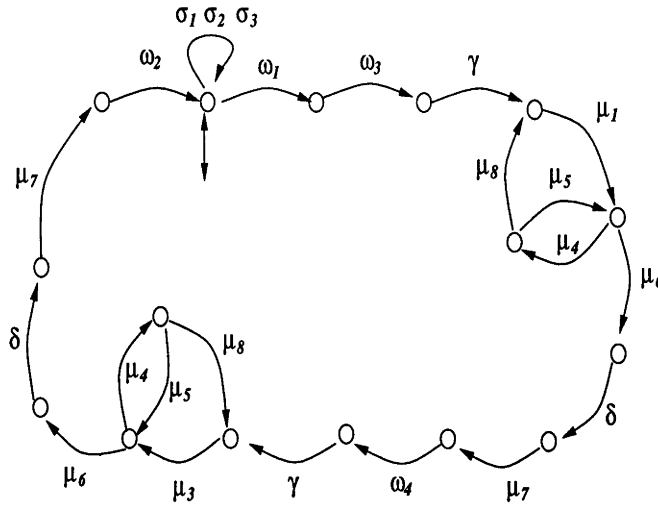


Figure 3.11: Modified local specification for G_2

that our conditions are satisfied, we only need to check whether this specification language is $L_{2,m}$ -closed. It can be verified (using TCT) that this is the case, and the new supervisor for G_2 is then synthesised.

3.5 Conclusions

In this chapter, we have formulated a problem of structural decentralised control of concurrent discrete event systems. In particular, we have considered a system G as the synchronous composition of subsystems G_1, G_2, \dots, G_n with shared events. If the marked behaviours of the subsystems mark the shared events and each pair of subsystems is mutually controllable, then for any $L_{i,m}$ -closed specifications on the subsystems, local syntheses and controls will achieve the same optimal behaviour as that would be obtained by a centralised control for the overall specification, and the control in one subsystem will not cause blocking in the other subsystems. Without the result in Theorem 3.1, one is forced to carry out a global synthesis and global control, and its computational complexity may be exponentially more expensive. Even though we have restricted that the plant is the synchronous composition of the local systems, the result can still be used for many practical applications such as a CIP process presented in Chapter 7.

Chapter 4

Task Rescheduling using Coordinator

In many applications of decentralised control, after the local supervisors have been designed for a given task, rescheduling of task may be handled by a coordinator. In this chapter, we investigate this idea. The result in the previous chapter is used here to establish such a coordination scheme. Examples are provided for illustration.

4.1 Introduction

In many applications, after a decentralised control design has already been established, one might need to reschedule certain jobs in local subsystems to meet different requirements such as the production of different types of products. In some cases, a coordinator can be used to handle such rescheduling. As an example, consider the following.

Example 4.1 Consider two machines modelled as in Figure 4.1(a). Assume that the systems need to cooperate at some stage so that certain products can be processed. The event sets are

$$\Sigma_1 = \{\alpha_1, \beta_1, \beta_2, \beta_3, \delta_1, \delta_2, \delta_3, \delta_4\},$$

$$\Sigma_2 = \{\alpha_1, \alpha_2, \beta_1, \beta_2, \beta_3, \lambda_1, \lambda_2, \lambda_3\}.$$

The shared event set is $\Sigma_1 \cap \Sigma_2 = \{\alpha_1, \beta_1, \beta_2, \beta_3\}$. Assume that all the events are controllable. Let p_i be the natural projection from $(\Sigma_1 \cup \Sigma_2)^*$ to Σ_i^* and let the closed and marked behaviour

of the plant G_i be respectively L_i and $L_{i,m}$ for $i = 1, 2$. In this case, one knows that $L_i = \overline{L_{i,m}}$. Assume that the system produces two types of parts, say part A and B. Assume that for part A the synchronisation of the event β_1 is required, while for part B the event β_2 is required. The specifications (say E_1 and E_2) are given in Figure 4.1(b). Here, after the shared event α_1 occurs, the system G_2 needs one more event (α_2) before the synchronisation (β_1 or β_2). The events $\delta_1, \delta_2, \delta_3, \delta_4$ in G_1 and $\lambda_1, \lambda_2, \lambda_3$ in G_2 represent certain processes required before the completion of one cycle of the whole operation. The shared event β_3 indicates the completion of the operation. Local supervisors (say S_1 and S_2) can now be designed for given local specifications (E_1 and E_2). Let the marked and closed behaviours of the closed-loop local system S_i/G_i be respectively $\kappa_{L_i}(E_i)$ and $\overline{\kappa_{L_i}(E_i)}$ for $i = 1, 2$. Since the systems satisfy the conditions given in Theorem 3.1, the concurrent actions of decentralised supervisors will achieve the global task.

Let G_h be a DES such that $L(G_h) = p_s(L)$ and $L_m(G_h) = p_s(L_m)$ where $p_s : (\Sigma_1 \cup \Sigma_2)^* \rightarrow (\Sigma_1 \cap \Sigma_2)^*$, as represented in Figure 4.1(c). Here after the event α_1 both β_1 and β_2 are allowed to occur. Write L_h for $p_s(L)$. The system G_h models the interactions of G_1 and G_2 , and is usually simpler than the G_i 's. Now we consider the following situation. Assume that the system is required to produce only one type of product for some period of time, and to prohibit the production of the other product due to, for example, market demands or problems in the next assembly line. This requirement could be achieved in the plant G_h by prohibiting the shared event β_1 or β_2 from occurring. The model given in Figure 4.1(d) represents the specification (denoted by E_{ha}) for the situation that only part A is produced. A supervisor, denoted by S_{ha} , is then computed. Let the marked and closed behaviours of the closed-loop system S_{ha}/G_h be respectively $\kappa_{L_h}(E_{ha})$ and $\overline{\kappa_{L_h}(E_{ha})}$.

Let

$$E' = (p_1|_L)^{-1}(E_1) \cap (p_2|_L)^{-1}(E_2) \cap (p_s|_L)^{-1}(E_{ha}).$$

Here E' represents the modified global specification. In this case, it can be verified that the control action of S_{ha} with the concurrent actions of local supervisors will achieve the control objective. That is,

$$\overline{\kappa_L(E')} = (p_1|_L)^{-1}(\overline{\kappa_{L_1}(E_1)}) \cap (p_2|_L)^{-1}(\overline{\kappa_{L_2}(E_2)}) \cap (p_s|_L)^{-1}(\overline{\kappa_{L_h}(E_{ha})}). \quad (4.1)$$

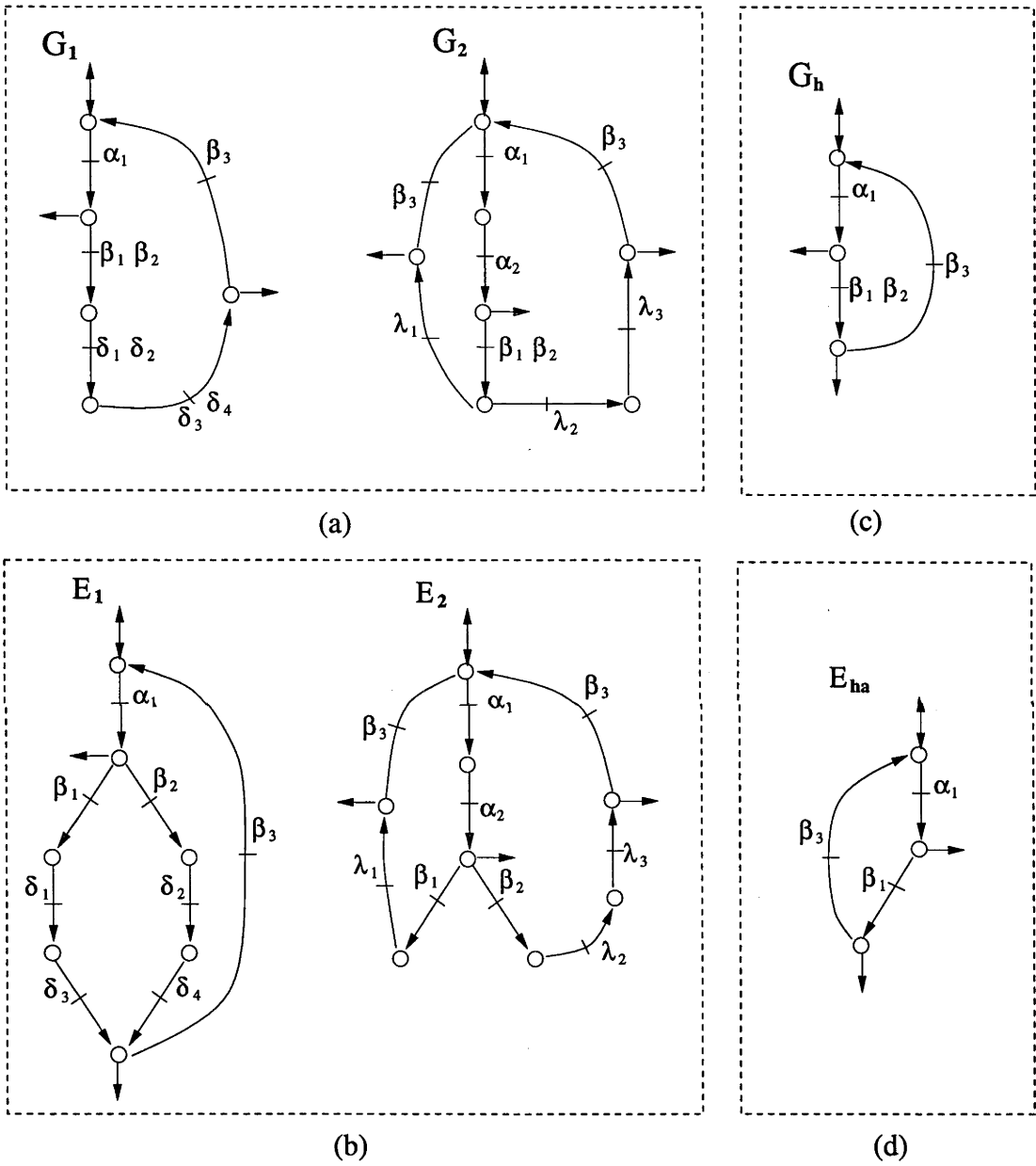


Figure 4.1: An example of coordination

For different requirements such as only part B is produced, a supervisor S_{hb} can be computed and an equation similar to Eq 4.1 can be verified. This example shows that in some situations, modifications of the given specification can be carried out by a supervisor designed for the plant G_h . We may call this supervisor a *coordinator*. \diamond

The concept of coordination in supervisory control of DES was introduced in [LW90]. In that paper, the authors consider a situation in which after decentralised supervisors have been designed, a coordinating supervisor, called a *coordinator*, at a higher or a second level may supervise the interactions of the decentralised supervisors. They have established the conditions for the existence of such a coordinator. In the paper [Lin91], the author considers the case that the decentralised control alone cannot achieve the overall task. Using a coordinator (another supervisor), the overall task can be achieved by allocating subtasks to the coordinator and local supervisors. That is, in such a case, decentralised control can achieve only a part of the overall task and a coordinator is used to achieve the rest of the overall task and this intervention by the coordinator should be minimal. In the paper [WW96c], the authors investigate a scheme of hierarchical coordination for resolving potential conflicts between the modular supervisors in a ‘non-intrusive manner’. Once a conflict situation is detected, the high-level controller will ‘disable’ those high-level transitions that will inevitably lead to conflict. In practice, a command is sent down to a low-level coordinator which will carry out the disablement so as to prevent the low-level system from evolving into a conflict situation.

In this chapter, we consider the case that the global system G is obtained by the synchronous composition of a number of local subsystems, and we assume that the local systems satisfy the conditions given in Theorem 3.1. We will take the model, denoted by G_h , of only the synchronisation events in G as a high-level plant for coordination. For a specification given on G_h , a coordinator can be computed. By using different coordinators, we will show that under some conditions it is possible to achieve the control objectives for different tasks.

4.2 A Coordination Scheme

Firstly we introduce a definition of *coordinator*. Consider a system as given in Figure 4.2. Let

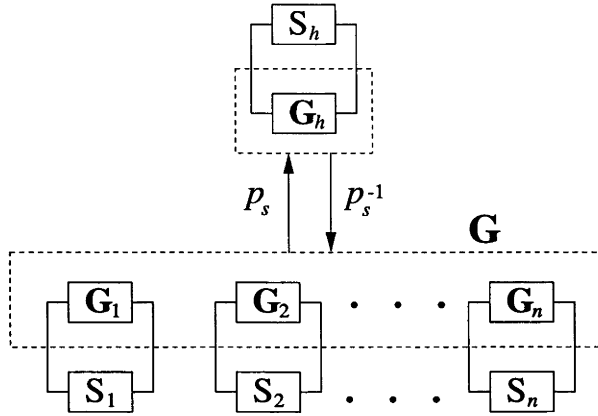


Figure 4.2: A coordination scheme

Σ_i be the event alphabets for the plants G_i for $i = 1, 2, \dots, n$. It is allowed that $\Sigma_i \cap \Sigma_j \neq \phi$ for $i, j \in \{1, 2, \dots, n\}$ and $i \neq j$. Let $\Sigma := \bigcup_{i=1}^n \Sigma_i$ be the event set of the overall system G . Let $\Sigma_c := \bigcup_{i=1}^n \Sigma_{ic}$ and $\Sigma_u := \bigcup_{i=1}^n \Sigma_{iu}$. Let p_i be the natural projection from Σ^* to Σ_i^* . Let $L_{i,m}, L_i \subseteq \Sigma_i^*$ represent respectively the marked and closed behaviours of the local system G_i . We assume that $L_i = \overline{L_{i,m}}$. The marked and closed behaviours of the overall system G are respectively

$$\begin{aligned} L_m &= L_{1,m} \parallel L_{2,m} \parallel \dots \parallel L_{n,m} = \bigcap_{i=1}^n (p_i)^{-1}(L_{i,m}), \\ L &= L_1 \parallel L_2 \parallel \dots \parallel L_n = \bigcap_{i=1}^n (p_i)^{-1}(L_i). \end{aligned}$$

Now let $E_i \subseteq L_{i,m}$ be an $L_{i,m}$ -closed language, not necessarily prefix-closed, representing a specification on the system G_i . As in the previous chapter, the overall specification is

$$E := \bigcap_{i=1}^n (p_i|_L)^{-1}(E_i).$$

Local supervisors, denoted S_i , can be designed for given local specifications (E_i) . The marked and closed behaviours of the closed-loop local system S_i/G_i are respectively $\kappa_{L_i}(E_i)$ and $\overline{\kappa_{L_i}(E_i)}$. Assume that the systems G_i and G_j , for $i, j \in \{1, 2, \dots, n\}$ and $i \neq j$, satisfy the conditions in Theorem 3.1. Then one has that

$$\overline{\kappa_L(E)} = \bigcap_{i=1}^n (p_i|_L)^{-1}(\overline{\kappa_{L_i}(E_i)}),$$

where $\overline{\kappa_L(E)}$ is the closed behaviour of a global closed-loop system (say S/G).

Let the shared event set be $\Sigma_s = \bigcup_i (\Sigma_i \cap (\bigcup_{j \neq i} \Sigma_j)) = \bigcup_{i \neq j} (\Sigma_i \cap \Sigma_j)$, and let p_s be the natural projection from Σ^* to Σ_s^* . Consider a system \mathbf{G}_h such that $L(\mathbf{G}_h) = p_s(L)$ and $L_m(\mathbf{G}_h) = p_s(L_m)$. Write L_h for $p_s(L)$ and $L_{h,m}$ for $p_s(L_m)$. Let a specification on the system \mathbf{G}_h , $E_h \subseteq L_{h,m}$, be an $L_{h,m}$ -closed language. Now compute a supervisor \mathbf{S}_h for the specification E_h on the plant \mathbf{G}_h . Then the marked and closed behaviours of the closed-loop system $\mathbf{S}_h/\mathbf{G}_h$ are $\kappa_{L_h}(E_h)$ and $\overline{\kappa_{L_h}(E_h)}$, respectively.

The modified global specification is

$$E' = \bigcap_{i=1}^n (p_i|_L)^{-1}(E_i) \cap (p_s|_L)^{-1}(E_h) = E \cap (p_s|_L)^{-1}(E_h).$$

We call \mathbf{S}_h a *coordinator* if

$$\overline{\kappa_L(E')} = \bigcap_{i=1}^n (p_i|_L)^{-1}(\overline{\kappa_{L_i}(E_i)}) \cap (p_s|_L)^{-1}(\overline{\kappa_{L_h}(E_h)}). \quad (4.2)$$

That is, a coordinator is a supervisor on \mathbf{G}_h which coordinates the local supervisions such that the combined supervision will achieve the modified global control objective.

Now, suppose that for a given system \mathbf{G} , any pair of subsystems \mathbf{G}_i and \mathbf{G}_j satisfies that

$$\begin{aligned} & i) L_{i,m} \text{ and } L_{j,m} \text{ mark } \Sigma_i \cap \Sigma_j, \\ & ii) L_i \text{ and } L_j \text{ are mutually controllable,} \end{aligned} \quad (4.3)$$

where $i, j \in \{1, 2, \dots, n\}$ and $i \neq j$. From Chapter 3, we know that, if for all pairs of the systems $\mathbf{G}_1, \mathbf{G}_2, \dots, \mathbf{G}_n$, and \mathbf{G}_h satisfy the shared-event-marking condition and the mutual controllability condition, then for any $L_{h,m}$ -closed language E_h , Eq. 4.2 is true, i.e., the supervisor \mathbf{S}_h for E_h on \mathbf{G}_h is a coordinator.

Proposition 4.1 shows that under given the assumptions in Eq. 4.3, the pair of the systems $(\mathbf{G}_i, \mathbf{G}_h)$, for $i = 1, 2, \dots, n$, satisfies the shared-event-marking condition.

Proposition 4.1 *Let $\Sigma_i, \Sigma_s, L_{i,m}, L_{h,m}$ be defined as above, for $i = 1, 2, \dots, n$. Then*

$$L_{i,m} \text{ and } L_{h,m} \text{ mark } \Sigma_i \cap \Sigma_s.$$

To prove Proposition 4.1, we firstly establish the following lemmas.

Lemma 4.1 *Assume that $L_i, L_{i,m}$ and p_s are given as above where $i = 1, 2, \dots, n$. Then*

$$p_s(L_i) = p_s(L_{i,m}).$$

Proof: Since one inclusion is always true, one only needs to show that $p_s(L_i) \subseteq p_s(L_{i,m})$. Let $s \in p_s(L_i)$. Then there exists a string $u \in L_i$ such that $s = p_s(u)$. Since $L_i = \overline{L_{i,m}}$, there exists a string $v \in \Sigma_i^*$ such that $uv \in L_{i,m}$. Hence $p_s(uv) \in p_s(L_{i,m})$. we consider two cases. If $v \in (\Sigma_i - \Sigma_s)^*$, then $p_s(uv) = p_s(u)p_s(v) = p_s(u) = s \in p_s(L_{i,m})$. Otherwise, there are events $\sigma_1, \sigma_2, \dots, \sigma_n \in \Sigma_s$ and $w_1, w_2, \dots, w_n, w_{n+1} \in (\Sigma_i - \Sigma_s)^*$ such that $v = w_1\sigma_1w_2\sigma_2 \dots w_n\sigma_nw_{n+1}$. Since $uv \in L_{i,m}$,

$$uv = uw_1\sigma_1w_2\sigma_2 \dots w_n\sigma_nw_{n+1} \in L_{i,m}. \text{ Hence } uw_1\sigma_1 \in \overline{L_{i,m}}.$$

Also, since $u \in L_i$ and $w_1 \in (\Sigma_i - \Sigma_s)^*$, $uw_1 \in \Sigma_i^*$. Since $\sigma \in \Sigma_s = \bigcup_{i \neq j} (\Sigma_i \cap \Sigma_j)$, $\sigma \in \Sigma_i \cap \Sigma_j$ for some j . By the assumption of the shared event marking condition between two systems \mathbf{G}_i and \mathbf{G}_j , one has that $uw_1 \in L_{i,m}$. Therefore,

$$p_s(uw_1) = p_s(u)p_s(w_1) = p_s(u) = s \in p_s(L_{i,m}). \quad \square$$

Next two lemmas are used to establish Lemma 4.4.

Lemma 4.2 ¹ *For alphabets $\Sigma_o, \Sigma_1, \Sigma_2$ with $\Sigma = \Sigma_1 \cup \Sigma_2$ and $\Sigma_o \subseteq \Sigma$, let $L_1 \subseteq \Sigma_1^*$, $L_2 \subseteq \Sigma_2^*$, and let $P_o : \Sigma^* \rightarrow \Sigma_o^*$ be the natural projection. Then*

$$P_o(L_1 \parallel L_2) \subseteq (P_oL_1) \parallel (P_oL_2).$$

Note that $P_oL_i \subseteq (\Sigma_o \cap \Sigma_i)^$.*

Proof : Recall that $L_1 \parallel L_2 = p_1^{-1}(L_1) \cap p_2^{-1}(L_2)$, where $p_i : \Sigma^* \rightarrow \Sigma_i^*$ and $(P_oL_1) \parallel (P_oL_2) = (p_1^o)^{-1}(p_o(L_1)) \cap (p_2^o)^{-1}(p_o(L_2))$, where $p_i^o : \Sigma_o^* \rightarrow (\Sigma_o \cap \Sigma_i)^*$. To show $P_o(L_1 \parallel L_2) \subseteq (P_oL_1) \parallel (P_oL_2)$, consider a string $s \in P_o(L_1 \parallel L_2)$. Then there exists a string $u \in L_1 \parallel L_2$ such that $s = p_o(u)$. Hence $u \in p_1^{-1}(L_1) \cap p_2^{-1}(L_2)$ and $s = p_o(u)$. So, $p_1(u) \in L_1$ and $p_2(u) \in L_2$. Therefore,

$$p_op_1(u) \in p_o(L_1) \text{ and } p_op_2(u) \in p_o(L_2).$$

Note that in this case $p_i^o(s) = p_i^op_o(u) = p_ip_o(u) = p_op_i(u)$. Hence, $p_1^o(s) \in p_o(L_1)$ and $p_2^o(s) \in p_o(L_2)$. So,

$$s \in (p_1^o)^{-1}p_o(L_1) \cap (p_2^o)^{-1}p_o(L_2) = (P_oL_1) \parallel (P_oL_2). \quad \square$$

¹This lemma is Exercise 3.3.3 in [Won96].

Lemma 4.3 *Let $\Sigma_o, \Sigma_1, \Sigma_2, p_o, p_i, p_i^o$ be defined as in Lemma 4.2. Suppose that*

$$\Sigma_1 \cap \Sigma_2 \subseteq \Sigma_o.$$

Then

$$P_o(L_1 \parallel L_2) = (P_o L_1) \parallel (P_o L_2).$$

Proof : One inclusion (\subseteq) is already shown in Lemma 4.2. For the other inclusion, we firstly show the following claims.

Claim 4.1 *For $L \subseteq \Sigma_i^*$ (for $i = 1, 2$)*

$$(p_i^o)^{-1} p_o(L) = p_o(p_i)^{-1}(L),$$

where $p_i^o : \Sigma_o^ \rightarrow (\Sigma_o \cap \Sigma_i)^*$. Note that $P_o(L) \subseteq (\Sigma_o \cap \Sigma_i)^*$.*

Proof of the claim: We show the case $i = 1$. For one inclusion (\supseteq), let $s \in p_o(p_1)^{-1}(L)$. Then there exists a string $u \in (p_1)^{-1}(L)$ such that $s = p_o(u)$. Hence $p_1(u) \in L$. So $p_o(p_1(u)) \in p_o(L)$. Note that $p_1^o(s) = p_1^o p_o(u) = p_1 p_o(u) = p_o p_1(u) \in p_o(L)$. Hence $s \in (p_1^o)^{-1} p_o(L)$. For the reverse inclusion, let $s \in (p_1^o)^{-1} p_o(L) \in \Sigma_o^*$. Hence $p_1^o(s) \in p_o(L)$. We have two cases. If $s \in (\Sigma_o - \Sigma_1)^*$, then $p_1^o(s) = \epsilon \in p_o(L)$. So, there exists a string $u \in L$ such that $p_o(u) = \epsilon$. Hence $u \in (\Sigma_1 - \Sigma_o)^*$. Since $p_1(su) = u \in L$, one has $su \in p_1^{-1}(L)$. Hence $p_o(su) = s \in p_o p_1^{-1}(L)$. If $s \notin (\Sigma_o - \Sigma_1)^*$, one can consider a string

$$s = w_1 \sigma_1 w_2 \sigma_2 \cdots w_m \sigma_m w_{m+1}, \text{ where } \sigma_1, \sigma_2, \cdots, \sigma_m \in (\Sigma_1 \cap \Sigma_o),$$

$$w_1, w_2, \cdots, w_m, w_{m+1} \in (\Sigma_o - \Sigma_1)^*.$$

Since $p_1^o(s) \in p_o(L)$, one has that $p_1^o(s) = \sigma_1 \sigma_2 \cdots \sigma_m \in p_o(L)$. Hence there exists a string $u \in L$ such that $p_1^o(s) = \sigma_1 \sigma_2 \cdots \sigma_m = p_o(u)$. Since $u \in L \subseteq \Sigma_1^*$, one has that

$$u = u_1 \sigma_1 u_2 \sigma_2 \cdots u_m \sigma_m u_{m+1}, \text{ where } u_1, u_2, \cdots, u_m, u_{m+1} \in (\Sigma_1 - \Sigma_o)^*.$$

Define a string $v := w_1 u_1 \sigma_1 w_2 u_2 \sigma_2 \cdots w_m u_m \sigma_m w_{m+1} u_{m+1}$. Then

$$p_1(v) = u_1 \sigma_1 u_2 \sigma_2 \cdots u_m \sigma_m u_{m+1} = u \in L. \text{ Hence } v \in p_1^{-1}(L). \text{ Thus } p_o(v) \in p_o p_1^{-1}(L).$$

Also $p_o(v) = w_1 \sigma_1 w_2 \sigma_2 \cdots w_m \sigma_m w_{m+1} = s$. Therefore $s = p_o(v) \in p_o p_1^{-1}(L)$. This proves the claim.

Claim 4.2 *Suppose that $\Sigma_1 \cap \Sigma_2 \subseteq \Sigma_o$. Then*

$$p_o p_1^{-1}(L_1) \cap p_o p_2^{-1}(L_2) \subseteq p_o(p_1^{-1}(L_1) \cap p_2^{-1}(L_2)).$$

Proof of the claim: Let $s \in p_o p_1^{-1}(L_1) \cap p_o p_2^{-1}(L_2)$. Then $s \in p_o p_1^{-1}(L_1)$ and $s \in p_o p_2^{-1}(L_2)$. Hence there exist a string $u \in p_1^{-1}(L_1)$ such that $s = p_o(u)$. So $p_1(u) \in L_1$ and $s = p_o(u)$. Also there is a string $v \in p_2^{-1}(L_2)$ such that $s = p_o(v)$. So $p_2(v) \in L_2$ and $s = p_o(v)$.

If $s = \epsilon$, then $u, v \in (\Sigma - \Sigma_o)^*$. Since $\Sigma_1 \cap \Sigma_2 \subseteq \Sigma_o$, one has that $\Sigma - \Sigma_o \subseteq \Sigma - (\Sigma_1 \cap \Sigma_2) = (\Sigma_1 - \Sigma_2) \cup (\Sigma_2 - \Sigma_1)$. So, $u, v \in ((\Sigma_1 - \Sigma_2) \cup (\Sigma_2 - \Sigma_1))^*$. Therefore

$$p_1(u) \in (\Sigma_1 - \Sigma_2)^* \text{ and } p_2(v) \in (\Sigma_2 - \Sigma_1)^*.$$

Let $w := p_1(u)p_2(v)$. Then $p_1(w) = p_1(u) \in L_1$ and $p_2(w) = p_2(v) \in L_2$. Hence $w \in p_1^{-1}(L_1) \cap p_2^{-1}(L_2)$. So $p_o(w) = p_o(p_1(u)p_2(v)) = p_o p_1(u) p_o p_2(v) = p_1 p_o(u) p_2 p_o(v) = p_1(\epsilon) p_2(\epsilon) = \epsilon = s$. Hence $s \in p_o(p_1^{-1}(L_1) \cap p_2^{-1}(L_2))$.

If $s \neq \epsilon$, then $s = \sigma_1 \sigma_2 \cdots \sigma_n$, for some $n \geq 1$ and $\sigma_i \in \Sigma_o$. Write

$$u = u_1 \sigma_1 u_2 \sigma_2 \cdots u_n \sigma_n u_{n+1} \text{ and } v = v_1 \sigma_1 v_2 \sigma_2 \cdots v_n \sigma_n v_{n+1},$$

for some $u_i, v_i \in (\Sigma - \Sigma_o)^*$. Since $\Sigma_1 \cap \Sigma_2 \subseteq \Sigma_o$, one has that $\Sigma - \Sigma_o \subseteq (\Sigma_1 - \Sigma_2) \cup (\Sigma_2 - \Sigma_1)$.

Therefore

$$p_1(u_i) \in (\Sigma_1 - \Sigma_2)^* \text{ and } p_2(v_i) \in (\Sigma_2 - \Sigma_1)^*.$$

Define a string

$$w := p_1(u_1)p_2(v_1) \left\{ \begin{array}{ll} \sigma_1 & \text{if } \sigma_1 \in (\Sigma_1 \cap \Sigma_2) \\ p_1(\sigma_1)p_2(\sigma_1) & \text{if } \sigma_1 \in \Sigma - (\Sigma_1 \cap \Sigma_2) \end{array} \right\} p_1(u_2)p_2(v_2) \cdots \\ p_1(u_n)p_2(v_n) \left\{ \begin{array}{ll} \sigma_n & \text{if } \sigma_n \in (\Sigma_1 \cap \Sigma_2) \\ p_1(\sigma_n)p_2(\sigma_n) & \text{if } \sigma_n \in \Sigma - (\Sigma_1 \cap \Sigma_2) \end{array} \right\} p_1(u_{n+1})p_2(v_{n+1}).$$

So,

$$p_1(w) = p_1(u_1)p_1(\sigma_1)p_1(u_2)p_1(\sigma_2) \cdots p_1(u_n)p_1(\sigma_n)p_1(u_{n+1}) \in L_1$$

$$p_2(w) = p_2(v_1)p_2(\sigma_1)p_2(v_2)p_2(\sigma_2) \cdots p_2(v_n)p_2(\sigma_n)p_2(v_{n+1}) \in L_2.$$

Hence $w \in p_1^{-1}(L_1) \cap p_2^{-1}(L_2)$. So $p_o(w) = \sigma_1 \sigma_2 \cdots \sigma_n = s$. Therefore

$$s \in p_o(p_1^{-1}(L_1) \cap p_2^{-1}(L_2)).$$

This proves the claim.

Proof for Lemma 4.3: We now show that $(P_o L_1) \parallel (P_o L_2) \subseteq P_o(L_1 \parallel L_2)$.

$$\begin{aligned} (P_o L_1) \parallel (P_o L_2) &= (p_1^o)^{-1} p_o(L_1) \cap (p_2^o)^{-1} p_o(L_2) \\ &= p_o p_1^{-1}(L_1) \cap p_o p_2^{-1}(L_2) && \text{(by Claim 4.1)} \\ &\subseteq p_o(p_1^{-1}(L_1) \cap p_2^{-1}(L_2)) && \text{(by Claim 4.2)} \\ &\subseteq P_o(L_1 \parallel L_2). \end{aligned}$$

□

Lemma 4.4 *Let Σ_i be the event alphabets for the plant G_i for $i = 1, 2, \dots, n$ and let $\Sigma := \bigcup_{i=1}^n \Sigma_i$. It is allowed that $\Sigma_i \cap \Sigma_j \neq \phi$ for $i, j \in \{1, 2, \dots, n\}$ and $i \neq j$. Let $\Sigma_s = \bigcup_{i \neq j} (\Sigma_i \cap \Sigma_j)$, and let p_i and p_s be the natural projection from Σ^* to Σ_i^* and to Σ_s^* , respectively. Then for $L_i \subseteq \Sigma_i^*$,*

$$p_s(L_1 \parallel L_2 \parallel \dots \parallel L_n) = p_s(L_1) \parallel p_s(L_2) \parallel \dots \parallel p_s(L_n).$$

Proof: One has that

$$\begin{aligned} & p_s(L_1 \parallel L_2 \parallel \dots \parallel L_n) \\ = & p_s(L_1) \parallel p_s(L_2 \parallel \dots \parallel L_n) \quad (\text{since } \Sigma_1 \cap (\bigcup_{j \neq 1} \Sigma_j) \subseteq \Sigma_s \text{ and by Lemma 4.3}) \\ = & p_s(L_1) \parallel p_s(L_2) \parallel p_s(L_3 \parallel \dots \parallel L_n) \quad (\text{by Lemma 4.3}) \\ & \vdots \\ = & p_s(L_1) \parallel p_s(L_2) \parallel \dots \parallel p_s(L_{n-1} \parallel L_n) \quad (\text{by Lemma 4.3}) \\ = & p_s(L_1) \parallel p_s(L_2) \parallel \dots \parallel p_s(L_n). \end{aligned}$$

□

It follows from Lemmas 4.1 and 4.4, we have the following.

Lemma 4.5 *Let $L_{h,m} = P_s(L_m)$ and $L_h = P_s(L)$. Then $L_h = L_{h,m}$.*

Proof: It can be shown in turn

$$\begin{aligned} L_h &= p_s(L_1 \parallel L_2 \parallel \dots \parallel L_n) \\ &= p_s(L_1) \parallel p_s(L_2) \parallel \dots \parallel p_s(L_n) \quad (\text{by Lemma 4.4}) \\ &= p_s(L_{1,m}) \parallel p_s(L_{2,m}) \parallel \dots \parallel p_s(L_{n,m}) \quad (\text{by Lemma 4.1}) \\ &= p_s(L_{1,m} \parallel L_{2,m} \parallel \dots \parallel L_{n,m}) \quad (\text{by Lemma 4.4}) \\ &= L_{h,m}. \end{aligned}$$

□

Now we show Proposition 4.1.

Proof of Proposition 4.1: It is obvious that $L_{i,m}$ marks $\Sigma_i \cap \Sigma_s$ since we assume that $L_{i,m}$ for $i = 1, 2, \dots, n$, marks its shared events. Given Lemma 4.5, $L_{h,m}$ also marks $\Sigma_i \cap \Sigma_s$. □

The proposition 4.1 shows that some of the structural properties of the given subsystems G_i are inherited by the coordination plant G_h . However, for the mutual controllability condition, we could only show the following.

Proposition 4.2 Assume that $\Sigma = \Sigma_1 \cup \Sigma_2$, $\Sigma_i = \Sigma_{ic} \dot{\cup} \Sigma_{iu}$, and $\Sigma_s = \Sigma_1 \cap \Sigma_2$. Let p_i and p_s be the natural projections from Σ^* to Σ_i^* and to Σ_s^* . Let $L_i \subseteq \Sigma_i^*$ be prefix-closed. Define $L = L_1 \parallel L_2$. Let $L_h = P_s(L)$. Then one has that

$$L_i(\Sigma_{su} \cap \Sigma_i) \cap p_i^{is}(p_s^{is})^{-1}(L_h) \subseteq L_i,$$

where p_i^{is} and p_s^{is} are the natural projections from $(\Sigma_i \cup \Sigma_s)^*$ to Σ_i^* and Σ_s^* respectively. \square

We need the following lemma to prove Proposition 4.2.

Lemma 4.6 Let Σ , L_i , p_i , p_s and L_h be defined as above for $i = 1, 2$. Then for $L_j \subseteq \Sigma_j^*$ and $j \neq i$

$$p_i p_s^{-1}(L_h) \subseteq p_i p_j^{-1}(L_j).$$

Proof: We only need to consider the case that L_h is not empty. We note that

$$\begin{aligned} p_i p_s^{-1}(L_h) = & \{u \mid u \in (\Sigma_i - \Sigma_s)^*\} \cup \\ & \{u \mid (\exists \sigma_1, \sigma_2, \dots, \sigma_n \in \Sigma_s) (\exists u_1, u_2, \dots, u_n, u_{n+1} \in (\Sigma_i - \Sigma_s)^*) \\ & \quad \sigma_1 \sigma_2 \dots \sigma_n \in L_h \text{ and } u = u_1 \sigma_1 u_2 \sigma_2 \dots u_n \sigma_n u_{n+1}\}. \end{aligned}$$

Let $s \in p_i p_s^{-1}(L_h)$. If $s \in (\Sigma_i - \Sigma_s)^*$, then $s \in p_j^{-1}(L_j)$. Thus $s = p_i(s) \in p_i p_j^{-1}(L_j)$.

For the other case $s = u_1 \sigma_1 u_2 \sigma_2 \dots u_n \sigma_n u_{n+1}$, for some $u_1, u_2, \dots, u_n, u_{n+1} \in (\Sigma_i - \Sigma_s)^*$ and $\sigma_1, \sigma_2, \dots, \sigma_n \in \Sigma_s$, such that $\sigma_1 \sigma_2 \dots \sigma_n \in L_h$.

Since $L_h = p_s(p_1^{-1}(L_1) \cap p_2^{-1}(L_2)) \subseteq p_s(p_j)^{-1}(L_j)$, for $j = 1, 2$, one has that

$$\sigma_1 \sigma_2 \dots \sigma_n \in p_s(p_j^{-1}(L_j)).$$

Thus there exist strings $v_1, v_2, \dots, v_{n+1} \in ((\Sigma_i - \Sigma_j) \cup (\Sigma_j - \Sigma_i))^*$ such that

$$v_1 \sigma_1 v_2 \sigma_2 \dots v_n \sigma_n v_{n+1} \in p_j^{-1}(L_j) \text{ and } p_s(v_1 \sigma_1 v_2 \sigma_2 \dots v_n \sigma_n v_{n+1}) = \sigma_1 \sigma_2 \dots \sigma_n.$$

Therefore,

$$p_j(v_1 \sigma_1 v_2 \sigma_2 \dots v_n \sigma_n v_{n+1}) = p_j(v_1) \sigma_1 p_j(v_2) \sigma_2 \dots p_j(v_n) \sigma_n p_j(v_{n+1}) \in L_j.$$

Note that $p_j(v_k) \in (\Sigma_j - \Sigma_i)^*$ for $k = 1, 2, \dots, n+1$. Let

$$w := u_1 p_j(v_1) \sigma_1 u_2 p_j(v_2) \sigma_2 \dots u_n p_j(v_n) \sigma_n u_{n+1} p_j(v_{n+1}).$$

Since $u_k \in (\Sigma_i - \Sigma_j)^*$ for $1 \leq k \leq n+1$, $w \in p_j^{-1}(L_j)$. Hence

$$p_i(w) = u_1 \sigma_1 u_2 \sigma_2 \dots u_n \sigma_n u_{n+1} \in p_i p_j^{-1}(L_j).$$

Therefore $s = p_i(w) \in p_i p_j^{-1}(L_j)$. \square

Proof of Proposition 4.2: By Lemma 3.5 in Chapter 3, one has that

$$p_i^{is}(p_s^{is})^{-1}(L_h) = p_i(p_s)^{-1}(L_h) \text{ and } p_i^{ij}(p_j^{ij})^{-1}(L_j) = p_i(p_j)^{-1}(L_j).$$

Also, by Lemma 4.6, and by the assumption that L_i and L_j are mutually controllable, one has that

$$\begin{aligned} L_i(\Sigma_{iu} \cap \Sigma_s) \cap p_i^{is}(p_s^{is})^{-1}(L_h) &= L_i(\Sigma_{iu} \cap \Sigma_s) \cap p_i(p_s)^{-1}(L_h) \\ &\subseteq L_i(\Sigma_{iu} \cap \Sigma_s) \cap p_i p_j^{-1}(L_j) \\ &= L_i(\Sigma_{iu} \cap \Sigma_s) \cap p_i^{ij}(p_j^{ij})^{-1}(L_j) \\ &\subseteq L_i. \end{aligned}$$

□

At this point, it is still an open question whether the mutual controllability condition is satisfied in general, given our assumptions on the structure of the subsystems G_i . However, under some additional conditions, the mutual controllability is also inherited by coordination plant as well (See Proposition 4.3 below). But in general we could state the following.

Theorem 4.1 *Under the foregoing assumptions, suppose that any pair of the systems in $\{G_1, G_2, \dots, G_n\}$ satisfies the assumptions given in Eq. 4.3 and suppose further that G_i and G_h are mutually controllable. Then any supervisor for a $L_{h,m}$ -closed sublanguage is a coordinator.*

□

Corollary 4.1 *Under the same assumptions as in Theorem 4.1, suppose that $n = 2$ and $L_h(\Sigma_{iu} \cap \Sigma_s) \cap p_i^{is}(p_s^{is})^{-1}(L_i) \subseteq L_h$. Then the same conclusion follows.*

Proof : This follows from Proposition 4.2 and the assumption.

□

Now, we consider a special structure which under the assumptions in Eq. 4.3, mutual controllability follows automatically. Assume that $\Sigma_{iu} \cap \Sigma_j = \Sigma_{ju} \cap \Sigma_i$. for $i, j \in \{1, 2, \dots, n\}$ and $L_i = \overline{L_{i,m}}$. Then we have the following.

Proposition 4.3 *Suppose that*

$$(\forall \sigma \in \Sigma_i) (\exists s \in L_i) s\sigma \in L_i,$$

and

$$p_s(L_1) = p_s(L_2) = \cdots = p_s(L_n).$$

Then L_h and L_i are mutually controllable.

Proof: Firstly, we show that $L_h(\Sigma_{su} \cap \Sigma_i) \cap p_s^{is}(p_i^{is})^{-1}(L_i) \subseteq L_h$. Let $s\sigma \in L_h(\Sigma_{su} \cap \Sigma_i) \cap p_s^{is}((p_i^{is})^{-1}(L_i))$, for $i = 1, 2, \dots, n$. Thus $s \in L_h$, $\sigma \in (\Sigma_{su} \cap \Sigma_i)$ and $s\sigma \in p_s^{is}((p_i^{is})^{-1}(L_i))$. By Lemma 3.5 in Chapter 3, $p_s^{is}((p_i^{is})^{-1}(L_i)) = p_s(p_i)^{-1}(L_i)$. We now show the following.

Claim 4.3 $\Sigma_s = \Sigma_i \cap \Sigma_j$, for $i \neq j$.

Proof of Claim: It suffices to show $\Sigma_i \cap \Sigma_j = \Sigma_i \cap \Sigma_k$, where i, j, k are distinct. Let $\sigma \in \Sigma_i \cap \Sigma_j$. Hence $\sigma \in \Sigma_j$. By the first assumption in this Proposition, there exists $t \in L_j$ such that $t\sigma \in L_j$. Since $\sigma \in \Sigma_i \cap \Sigma_j \subseteq \Sigma_s$, one has that $p_s(t\sigma) = p_s(t)\sigma \in p_s(L_j) = p_s(L_k)$. Hence $\sigma \in \Sigma_k$. Therefore $\sigma \in \Sigma_i \cap \Sigma_k$. This proves the claim.

It follows from the claim that $\Sigma_s \subseteq \Sigma_i$. Hence, $p_s(p_i)^{-1}(L_i) = p_s(L_i)$. Hence, one has that $s\sigma \in p_s((p_i)^{-1}(L_i)) = p_s(L_i)$. Consider a string $s\sigma = \sigma_1\sigma_2 \cdots \sigma_m\sigma \in p_s(L_1) = p_s(L_2) = \cdots = p_s(L_n)$, where $\sigma_1, \sigma_2, \dots, \sigma_m, \in \Sigma_s$. Therefore, there exist strings

$$v'_1, v''_1, \dots, v_1^m, v_1^{m+1} \in (\Sigma_1 - \Sigma_s)^*,$$

$$v'_2, v''_2, \dots, v_2^m, v_2^{m+1} \in (\Sigma_2 - \Sigma_s)^*,$$

⋮

$$v'_n, v''_n, \dots, v_n^m, v_n^{m+1} \in (\Sigma_n - \Sigma_s)^*,$$

such that

$$v'_1\sigma_1 v''_1\sigma_2 \cdots v_1^m\sigma_m v_1^{m+1}\sigma \in L_1,$$

$$v'_2\sigma_1 v''_2\sigma_2 \cdots v_2^m\sigma_m v_2^{m+1}\sigma \in L_2,$$

⋮

$$v'_n\sigma_1 v''_n\sigma_2 \cdots v_n^m\sigma_m v_n^{m+1}\sigma \in L_n.$$

Define

$$\omega := v'_1 v'_2 \cdots v'_n \sigma_1 v''_1 v''_2 \cdots v''_n \sigma_2 \cdots v_1^m v_2^m \cdots v_n^m \sigma_m v_1^{m+1} v_2^{m+1} \cdots v_n^{m+1} \sigma.$$

One has that

$$p_1(\omega) = v'_1 \sigma_1 v''_1 \sigma_2 \cdots v_1^m \sigma_m v_1^{m+1} \sigma \in L_1,$$

$$p_2(\omega) = v'_2 \sigma_1 v''_2 \sigma_2 \cdots v_2^m \sigma_m v_2^{m+1} \sigma \in L_2,$$

$$\vdots$$

$$p_n(\omega) = v'_n \sigma_1 v''_n \sigma_2 \cdots v_n^m \sigma_m v_n^{m+1} \sigma \in L_n.$$

Hence $\omega \in p_1^{-1}(L_1) \cap p_2^{-1}(L_2) \cap \cdots \cap p_n^{-1}(L_n) = L$. So

$$p_s(\omega) = \sigma_1 \sigma_2 \cdots \sigma_m \sigma = s\sigma \in p_s(L) = L_h.$$

Now we show that $L_i(\Sigma_{su} \cap \Sigma_i) \cap p_i^{is}(p_s^{is})^{-1}(L_h) \subseteq L_i$. Firstly, we establish that for $L_j \in \Sigma_j^*$ and $j \neq i$, $p_i p_s^{-1}(L_h) \subseteq p_i p_j^{-1}(L_j)$: by Claim 4.3, we know that $\Sigma_s \subseteq \Sigma_i$. Then the above statement can be shown similar to Lemma 4.6. Then the rest of the proof follows similarly as that of Proposition 4.2. \square

To arrange the mutual controllability condition in Theorem 4.1, we could use algorithms developed in the next chapter. We may point out that given L_i , for $i = 1, 2, \dots, n$, one has that $L_h = p_s(L)$ where L is synchronous composition of the L_i 's. Now suppose we apply the algorithms in Chapter 5 to obtain L'_i and L'_h for which the mutual controllability condition is satisfied. It may be the case that $p_s(L') \neq L'_h$ where L' is synchronous composition of the L'_i 's. However, this would cause no problems as far as coordination is concerned.

4.3 Example

In this section, we consider an example for illustrating the results in this chapter. Consider again the chemical reactor shown in Figure 3.5 and described in Section 3.4. At this time, we assume that the reactor can produce several different products depending on different materials and catalysts supplied and also the process methods chosen. Figure 4.3 is a schematic diagram of the plant.

The reactor is comprised of the following elementary components: the reaction tank, three material feed valves V_1, V_2 and V_3, three material feed pumps P_1, P_2 and P_3, three drain valves V_4, V_5 and V_6, three drain pumps P_4, P_5 and P_6, two supply valves for catalysts V_7 and V_8, a low and high level sensors WL_1 and WH_1 for the tank, a heater with a temperature probe TP_1, a continuous reaction controller C_1 and two timers T_1 and T_2.

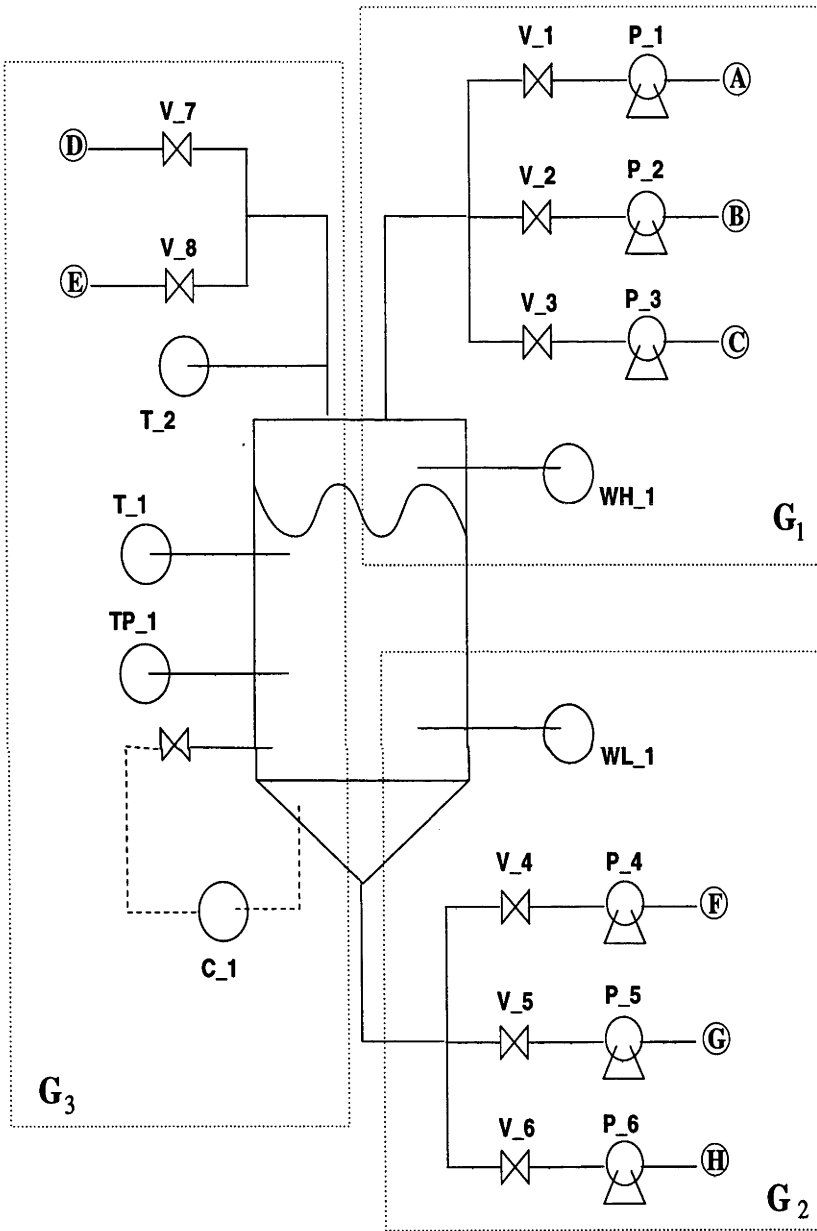


Figure 4.3: Modified batch reactor

The duration of the chemical reaction is timed by the timer T_1, which can be set for 10 or 15 minutes. A 30-second timer T_2 is for the timing requirement of adding catalysts.

Like the case in Section 3.4, the whole plant can be naturally decomposed into three subplants as shown in Figure 4.3: the filling subplant (G_1), the reaction subplant (G_2) and the draining subplant (G_3). Note that there are no shared components among subplants. Also, the whole operation can be divided naturally into three subprocesses according to the decomposition of the plant: the filling process, the reaction process, and the draining process. To enforce that these three processes are operating serially, we introduce controllable shared events which indicate the beginning ($\sigma_1, \sigma_2, \dots, \sigma_{10}$) or the ending ($\lambda_1, \lambda_2, \lambda_3$) of a certain process. They will be described in detail later. These controllable synchronised events are represented as DES models, called ‘flag’s, and considered as a part of the plants.

DES models of elementary components of G_1 are shown in Figure 4.4. The automaton

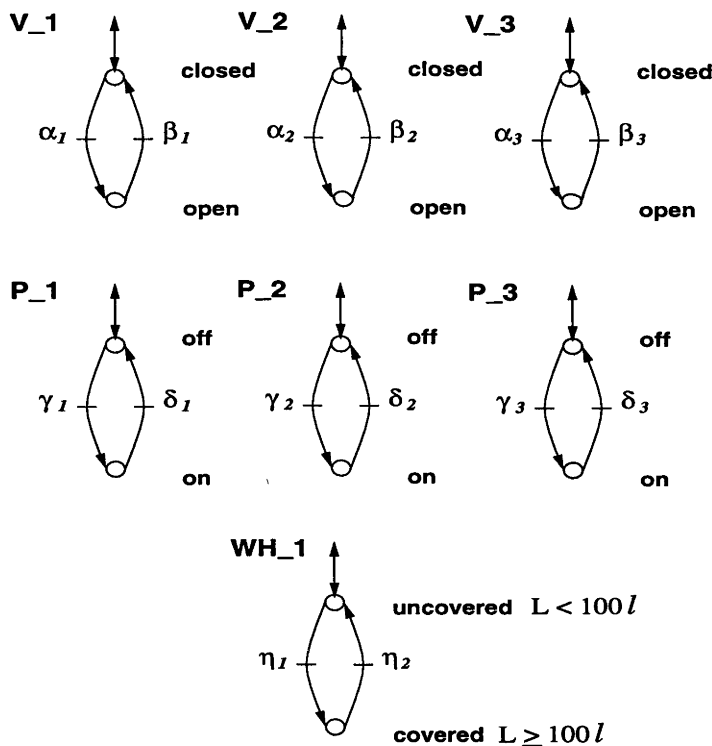


Figure 4.4: DES models for the elementary components of the plant G_1

presented in Figure 4.5(a) represents a physical constraint, namely that the level in the tank

cannot be increased after all the feed valves have been closed. The DES model for flag_1 given in Figure 4.5(b) represents that after the shared events ($\sigma_1, \sigma_2, \text{ or } \sigma_3$) have occurred, the feed valves are allowed to be closed ($\beta_1, \beta_2 \text{ and } \beta_3$). Then with the occurrences of the events λ_1 and λ_3 , a cycle of the operation is completed. For this example we assume that we always have a combination of two materials from materials (A), (B) and (C) (see Figure 4.3) for the chemical reaction. But this assumption is made for convenience and it is not crucial. Other combinations of the materials can be specified in a similar manner. Now, we explain the roles of the shared events $\sigma_1, \sigma_2, \sigma_3, \lambda_1$ and λ_3 . The shared event σ_1 is for indicating that only the materials (A) and (B) are used for the production, while σ_2 and σ_3 respectively indicate that the materials (B) and (C), and (A) and (C) are used. The shared event λ_1 represents that the filling process has been completed and the chemical reaction process can now proceed. The event λ_3 represents a complete cycle of the batch reaction. The selfloops of the shared events at the state after λ_1 in Figure 4.5 (b) are interpreted as that the system G_1 monitors the progress of the status the other plants but takes no action. The DES model for the plant G_1 is obtained by synchronous

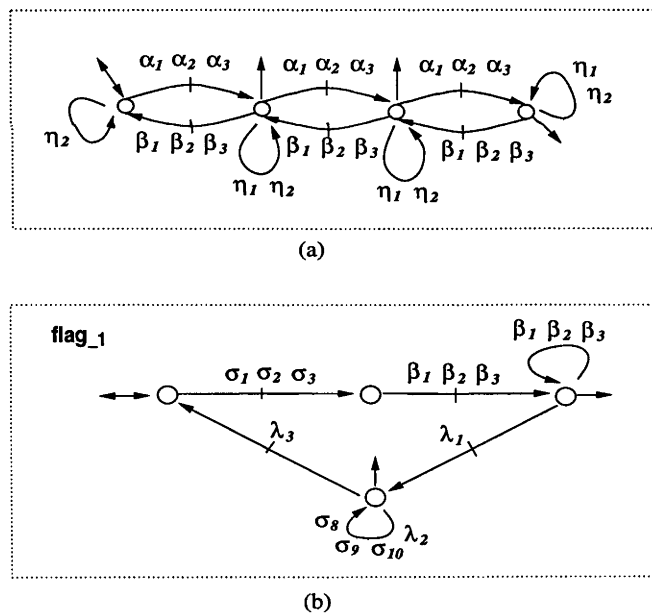


Figure 4.5: DES models for physical constraint and synchronisation flag for the plant G_1

composition of the elementary components and the physical constraint. Since it is assumed that the mixture of any two materials is used for the chemical reaction, the specification for the

plant G_1 , denoted by E_1 , is as follows:

1. For the mixture of the materials \textcircled{A} and \textcircled{B} , after the event σ_1 (indicating this mixture) has occurred, open the valves V_1 and V_2 (α_1, α_2) and turn on the pumps P_1 and P_2 (γ_1, γ_2).
2. For the mixture of the materials \textcircled{B} and \textcircled{C} , after the event σ_2 has occurred, open the valves V_2 and V_3 (α_2, α_3) and turn on the pumps P_2 and P_3 (γ_2, γ_3).
3. For the mixture of the materials \textcircled{A} and \textcircled{C} , after the event σ_3 has occurred, open the valves V_1 and V_3 (α_1, α_3) and turn on the pumps P_1 and P_3 (γ_1, γ_3).
4. When the level in the tank reaches $L \geq 100\text{ l}$ (η_1), turn off the pumps ($\delta_1, \delta_2, \delta_3$) and close the valves ($\beta_1, \beta_2, \beta_3$) whichever necessary. Then send a signal (λ_1) to the other plants.
5. The selfloops at the initial state mean that the subplant G_1 does not restrict the behaviours of the other subplants.

DES model of E_1 is given in Figure 4.6. Then a local supervisor S_1 for E_1 is designed.

DES models of elementary components for the chemical reaction process, G_2 , are shown in Figure 4.7. The duration of the reaction is timed by the timer T_1 which can be set for 10 (event μ_1) or 15 minutes (event μ_2). Assume that there are four possible ways of how the chemical reaction will proceed. The details will be presented below. The automaton displayed in Figure 4.8 represents flag_2 . Here after the shared event λ_1 indicating that the filling process has been completed, the shared events ($\sigma_4, \sigma_5, \sigma_6, \sigma_7$) are allowed to occur. These shared events indicate that different chemical reactions will take place. The chemical reaction starts with the increase of temperature of the material in the tank to $T > 30^\circ\text{C}$ (ω_1). After the temperature returns to $T < 30^\circ\text{C}$ (ω_2), the shared event λ_2 indicating a completion of the chemical reaction is permitted to occur. The selfloops of the shared events at the initial state can be interpreted similarly as the case in G_1 . The DES model for the plant G_2 is obtained by synchronous composition of the elementary components.

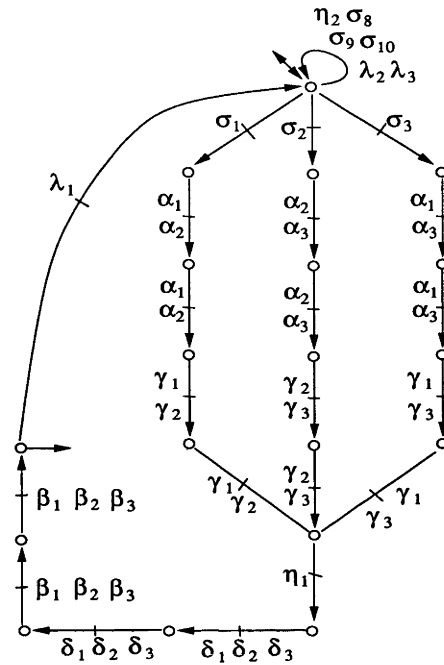


Figure 4.6: Specification E_1 for the plant G_1

It is assumed that there are four different chemical reactions. We denote them by methods I, II, III and IV. The shared controllable events $\sigma_4, \sigma_5, \sigma_6$ and σ_7 are used respectively to indicate that which method is taken for the chemical reaction. With reference to Figure 4.9, for method I, we assume that after the occurrence of λ_1 the requirements are:

1. After the shared event σ_4 has occurred, increase the temperature of mixed material in the tank to $30^\circ\text{C} < T < 60^\circ\text{C}$ (ω_1). Then enable the reaction controller C_1 (ξ_1).
2. Release the timer T_1. Assume that the controller is required to operate for 10 minutes at the material temperature $30^\circ\text{C} < T < 60^\circ\text{C}$. So μ_1 is chosen.
3. During the operation, the timer can be held at its current time (μ_3) to handle some situations such as emergency repairs. From the *held* state, the timer can either be reset (μ_7) to the *idle* state or re-released (μ_4).
4. After the set time has expired (uncontrollable event μ_5), the controller C_1 is disabled (ξ_2) and the timer T_1 is reset (μ_6).

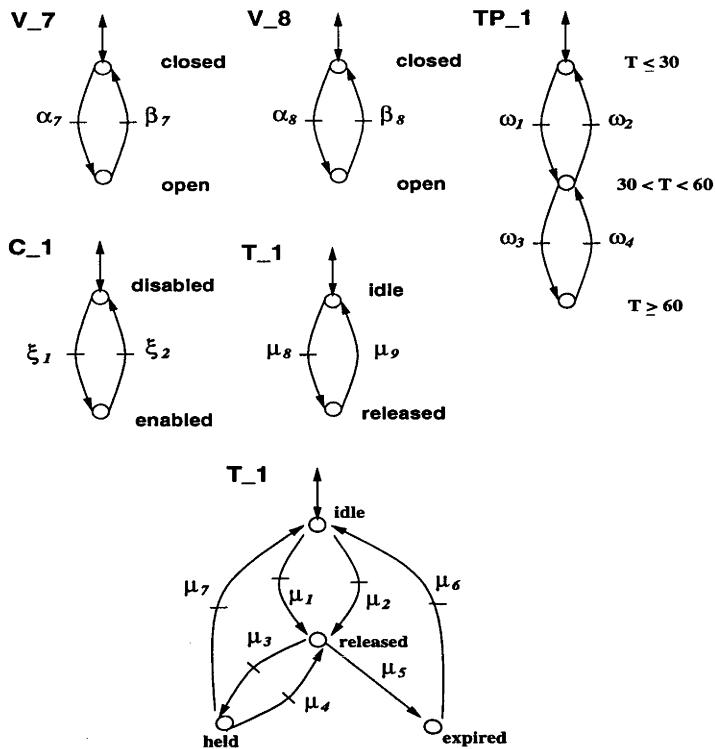


Figure 4.7: DES models for the elementary components of the plant G_2

5. To supply the catalyst \textcircled{D} , open the valve V_7 (α_7); then release the 30-second timer T_2 (μ_8).
6. After T_2 has expired (uncontrollable event μ_9), close the valve V_7 (β_7).
7. Then increase the temperature to $T > 60^\circ\text{C}$ (ω_3) and enable the reaction controller C_1 (ξ_1) again.
8. Release the timer T_1. At this time, we assume that the reaction is required to operate for 15 minutes. So μ_2 is chosen.
9. When the time has expired (μ_5), the controller C_1 is disabled (ξ_2) and the timer is reset (μ_6).
10. Wait until the material temperature is cooled down to $T < 30^\circ\text{C}$ (ω_4, ω_2).
11. The selfloops at the initial state mean that the subplant G_2 does not restrict the behaviours

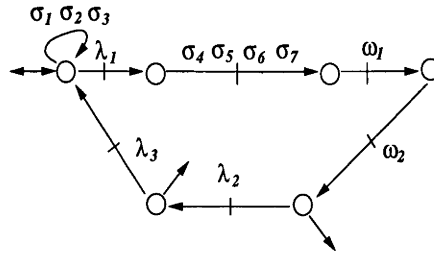


Figure 4.8: Synchronisation flag (flag_2) for the plant G_2

of the other subplants.

The requirements for the other methods, II, III and IV, can also be established similarly. DES models of the requirements for G_2 , denoted by E_2 , is given in Figure 4.9. A local supervisor S_2 for E_2 is then computed.

Similarly, one can obtain a local supervisor S_3 for the plant G_3 . The elementary components of G_3 are modelled as shown in 4.10. A physical constraint is represented as an automaton given in Figure 4.11 (a), namely, the level in the tank cannot be decreased after all the drain valves are closed. The DES model for flag_3 displayed in Figure 4.11(b) represents that after the shared event λ_2 indicating a completion of the chemical reaction, the synchronisation events ($\sigma_8, \sigma_9, \sigma_{10}$) are permitted and then the drain valves will be closed (β_4, β_5 and β_6). Then with the occurrence of the event λ_3 , a cycle of the operation is completed. In here, we explain the roles of the shared events σ_8, σ_9 and σ_{10} . The shared event σ_8 represents that the drain exit \textcircled{F} (see Figure 4.3) will be used for the draining of the product. The events σ_9 and σ_{10} respectively indicate that the drain exits \textcircled{C} and \textcircled{H} will be used. The selfloops of the shared events at the initial state can be interpreted similarly as the case in G_1 . The DES model for the plant G_3 is obtained by synchronous composition. The specification for G_3 , denoted by E_3 , is as follows: after occurrence of the event λ_2 ,

1. For draining to the exit \textcircled{F} , after the event σ_8 has occurred, open the valve V_4 (α_4) and turn on the pump P_4 (γ_4).
2. For draining to \textcircled{C} , after the event σ_9 has occurred, open the vale V_5 (α_5) and turn on the pump P_5 (γ_5).

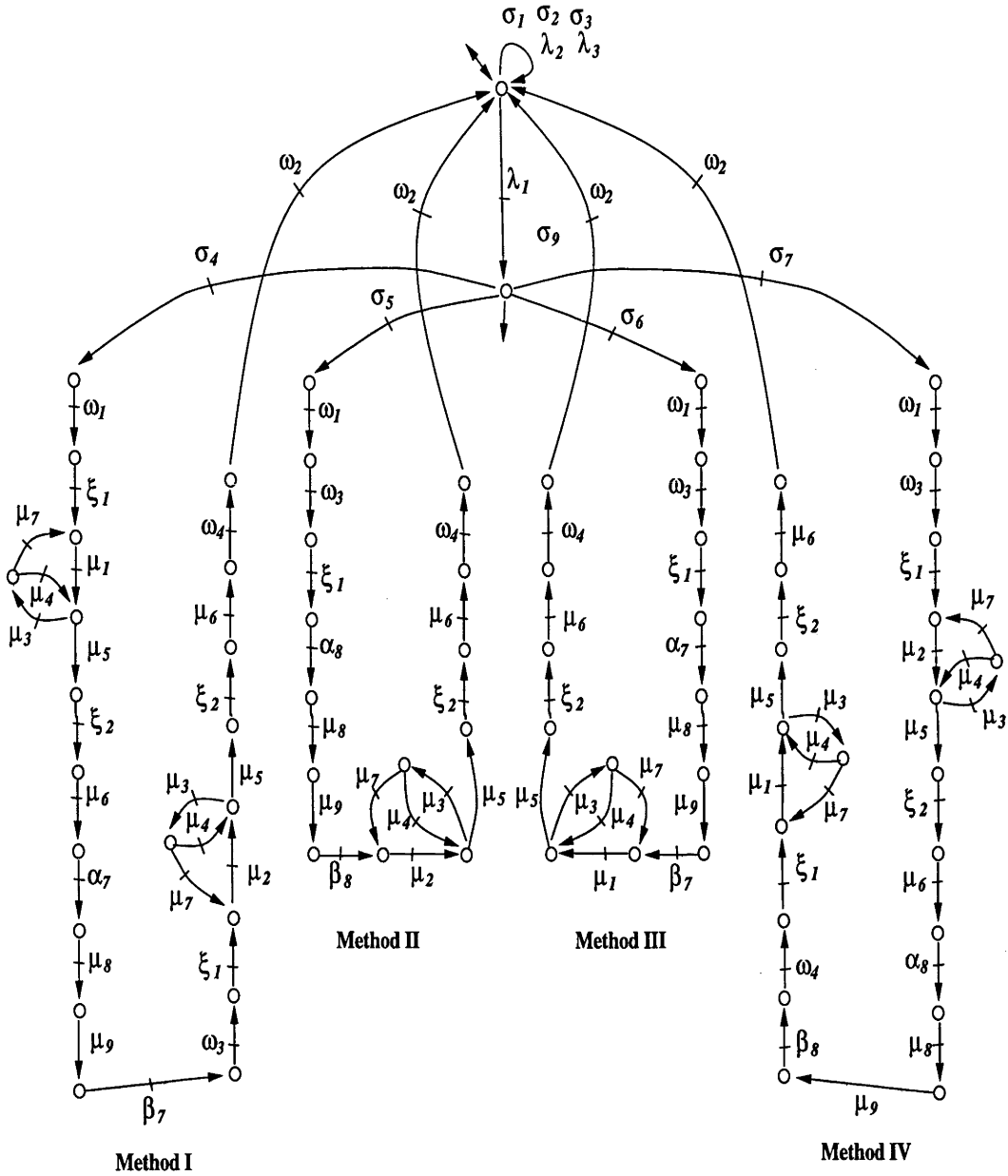


Figure 4.9: Specification E_2 for the plant G_2

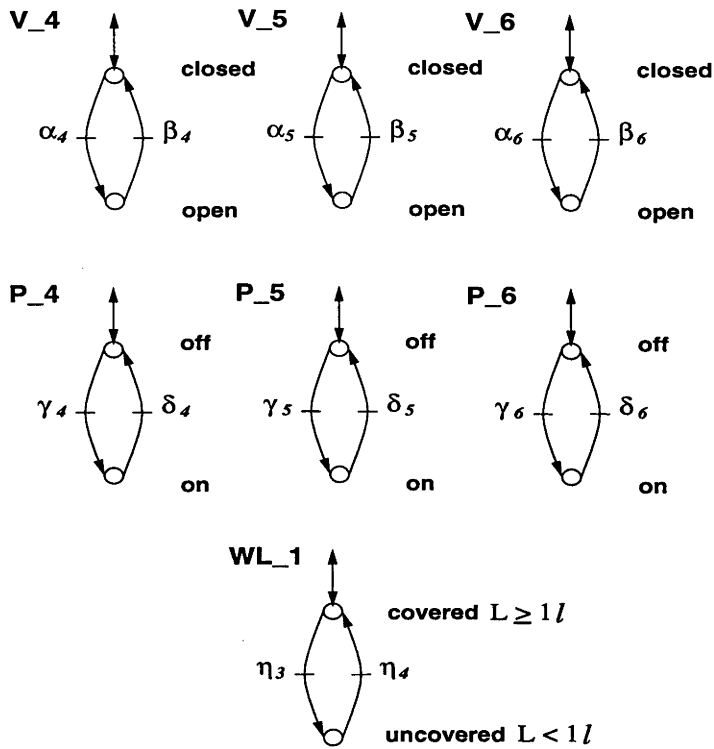


Figure 4.10: DES models for the elementary components of the plant G_3

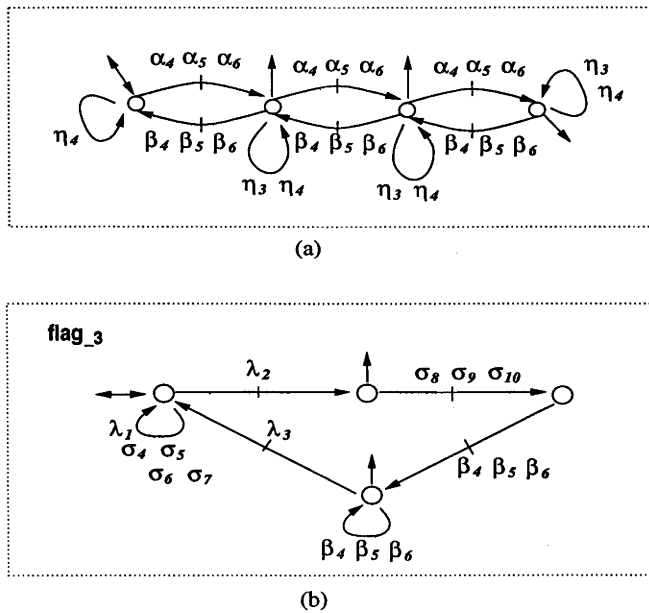


Figure 4.11: DES models for physical constraint and synchronisation flag for the plant G_3

3. For draining to \textcircled{H} , after the event σ_{10} has occurred, open the valve V_6 (α_6) and turn on the pump P_6 (γ_6).
4. When the level in the tank reaches $L < 1l$ (η_3), turn off the pump ($\delta_4, \delta_5, \delta_6$) and close the valve ($\beta_4, \beta_5, \beta_6$) whichever necessary. Then send a signal (λ_3) to the other plants.
5. The selfloops at the initial state mean that the subplant G_3 does not restrict the behaviours of the other subplants.

DES model of E_3 is given in Figure 4.12. A local supervisor S_3 for E_3 is then computed.

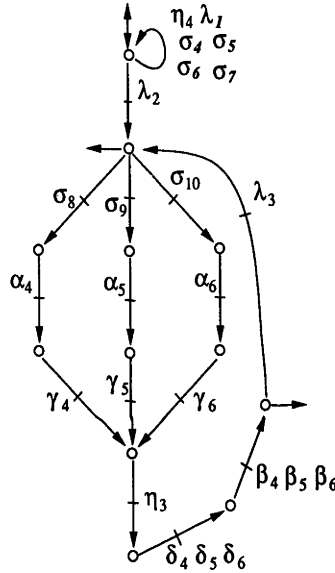


Figure 4.12: Specification E_3 for the plant G_3

The event sets for the subplants are

$$\Sigma_1 = \{\alpha_1, \alpha_2, \alpha_3, \beta_1, \beta_2, \beta_3, \gamma_1, \gamma_2, \gamma_3, \delta_1, \delta_2, \delta_3, \eta_1, \eta_2, \sigma_1, \sigma_2, \sigma_3, \sigma_8, \sigma_9, \sigma_{10}, \lambda_1, \lambda_2, \lambda_3\},$$

$$\Sigma_2 = \{\alpha_7, \alpha_8, \beta_7, \beta_8, \xi_1, \xi_2, \omega_1, \omega_2, \omega_3, \omega_4, \mu_1, \mu_2, \mu_3, \mu_4, \mu_5, \mu_6, \mu_7, \mu_8, \mu_9, \\ \sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5, \sigma_6, \sigma_7, \lambda_1, \lambda_2, \lambda_3\},$$

$$\Sigma_3 = \{\alpha_4, \alpha_5, \alpha_6, \beta_4, \beta_5, \beta_6, \gamma_4, \gamma_5, \gamma_6, \delta_4, \delta_5, \delta_6, \eta_3, \eta_4, \sigma_4, \sigma_5, \sigma_6, \sigma_7, \sigma_8, \sigma_9, \sigma_{10}, \lambda_1, \lambda_2, \lambda_3\}.$$

The shared event set is

$$\Sigma_s = \{\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5, \sigma_6, \sigma_7, \sigma_8, \sigma_9, \sigma_{10}, \lambda_1, \lambda_2, \lambda_3\}.$$

We verify that the system structure of this example (G_1 , G_2 , and G_3) satisfies the required

conditions in Theorem 3.1. Since all the shared events are controllable, the mutual controllability conditions are satisfied trivially. For the ‘shared event marking’ condition, we mark all the states before the shared events. It can be checked that the local specification E_i is $L_{i,m}$ -closed language, where $L_{i,m}$ is the marked behaviour of G_i for $i = 1, 2, 3$.

Now we obtain the coordination system G_h by natural projection $p_s : \Sigma^* \rightarrow \Sigma_s^*$. Figure 4.14(a) represents a DES model for the plant G_h . Assume that the system can produce four kinds of products as presented in Figure 4.13. Here the product type I is produced from the mixed materials (A) and (B), with the catalyst (D) by the reaction method I, and then the product will be drained through the drain exit (F). For the product types II, III, and IV, it can be interpreted similarly. The specification for this in G_h is displayed in Figure 4.14(b).

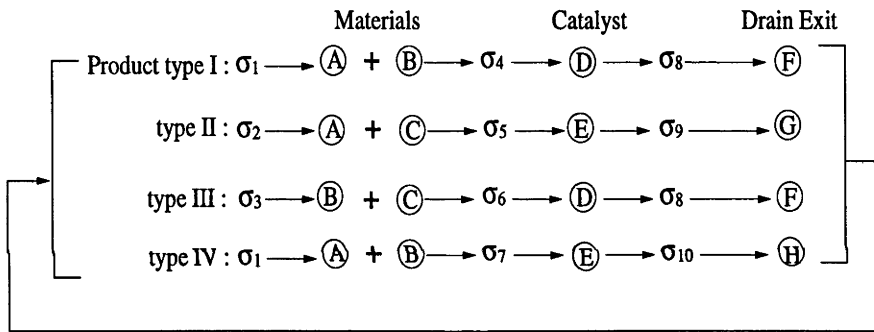


Figure 4.13: Processes for the batch reaction

Now assume that due to, for example, market demands, the system is required to produce only two types of products, type I and another type of product, say the product type V. Assume that the product type V is made from the mixed materials (B) and (C), with the catalyst (E) by the reaction method IV which starts with the shared event σ_7 , and then the product will be drained through the drain exit (H). To meet with this requirements, this can be done in the plant G_h by prohibiting certain shared events from occurring. We change the specification E_h as shown in Figure 4.14(c) and design S_h based on the modified E_h . By Theorem 4.1, the concurrent actions of S_h and S_i will guarantee to achieve the modified requirements.

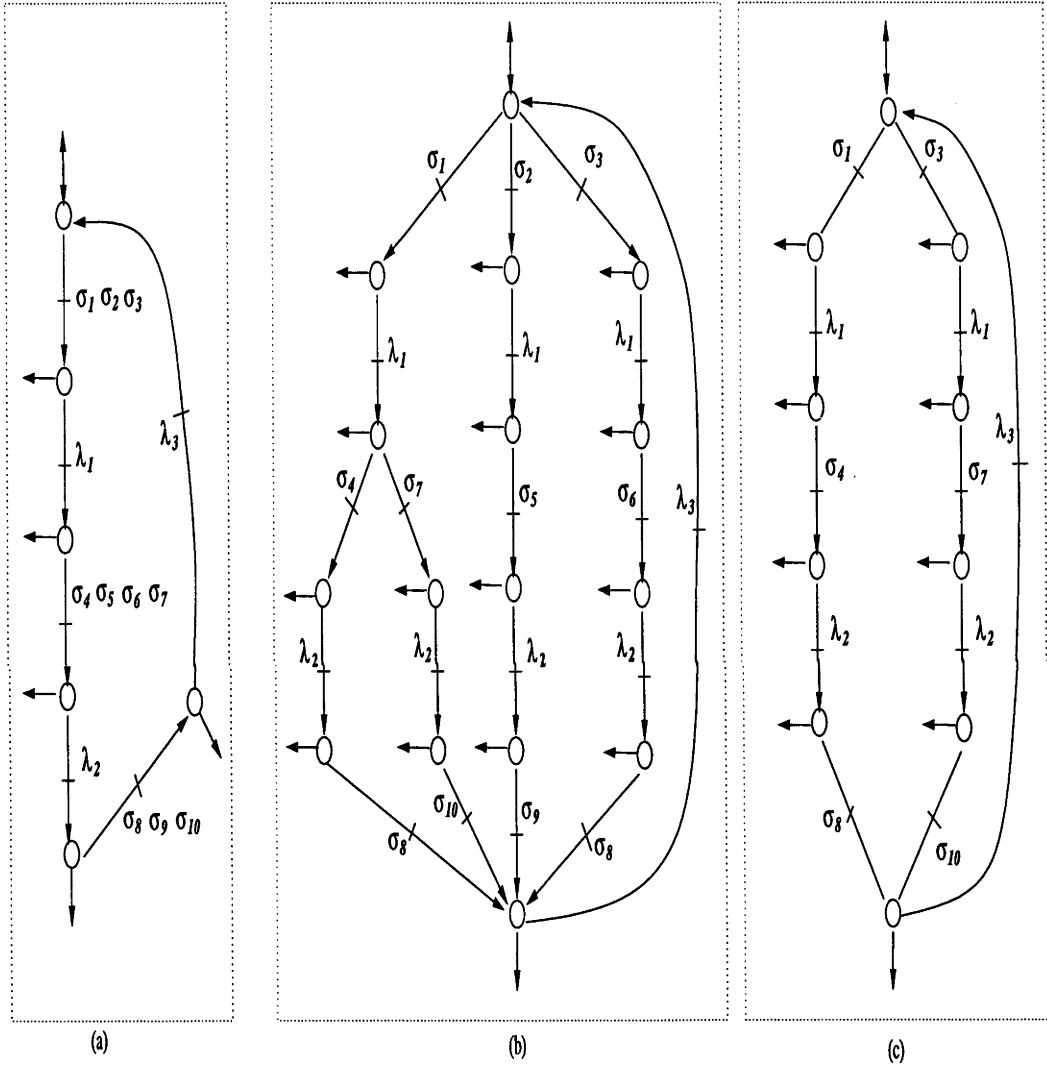


Figure 4.14: A coordination plant G_h

4.4 Conclusions

In this chapter, we have shown that after the local supervisors have been designed for a given task, a coordinator can be used to solve some rescheduling problems among local plants G_i . The coordination system G_h models the interactions of local plants, and is usually simpler than the G_i 's. Then a coordinator is designed based on the specification given for the system G_h . Under the conditions given in Theorem 4.1, the combined concurrent actions of the coordinator with the existing local supervisors will achieve the rescheduling requirements. By using different coordinators, we have shown that under some conditions it is possible to achieve the control objectives for different tasks, as seen in the example provided.

Chapter 5

Structure Synthesis

For a given system G obtained by synchronous composition of n subsystems, in Chapter 3 we have shown that if the system G possesses the structural properties given in Theorem 3.1, we can achieve decentralised control. In this chapter we investigate one way to arrange the structural conditions. In Section 5.1, some basic concepts used in this chapter are introduced. We present methods to arrange the shared-event-marking condition in Section 5.2, and the mutual controllability condition in Section 5.3. Examples are provided in Section 5.4 for illustration. This chapter ends with some concluding remarks in Section 5.5

5.1 Preliminaries

In this section, we introduce some useful concepts for subsequent discussions. Recall that throughout this thesis, we assume that a DES has finite sets of states and events. Firstly, we define *selfloop operation*. A selfloop is a transition for which the exit state and the entrance state are the same. Formally, for a DES $G = (Q, \Sigma, \delta, q_o, Q_m)$, we say that a transition (q, σ, q') in G is a *selfloop* if $q' = q$. Then the *Selfloop Operation* for a given generator $G_1 = (Q_1, \Sigma_1, \delta_1, q_{10}, Q_{1,m})$, where $\Sigma_1 \subseteq \Sigma$, is defined as follows: for a state $q^s \in Q_1$ and a set of events, $\Sigma_o \subseteq \Sigma$,

$$\mathbf{S}_{\Sigma_o}^{q^s}(G_1) := (Q_1, \Sigma_1 \cup \Sigma_o, \delta_1^s, q_{10}, Q_{1,m}),$$

where the partial transition function $\delta_1^s : Q_1 \times (\Sigma_1 \cup \Sigma_o) \rightarrow Q_1$ is defined as follows:

for $q \in Q_1, \sigma \in (\Sigma_1 \cup \Sigma_o)$,

1. if $\delta_1(q, \sigma)$ is defined, then

$$\delta_1^s(q, \sigma) \text{ is defined and } \delta_1^s(q, \sigma) = \delta_1(q, \sigma),$$

2. if $q = q^s, \sigma \in \Sigma_o$ and $\delta_1(q, \sigma)$ is not defined, then

$$\delta_1^s(q, \sigma) \text{ is defined and } \delta_1^s(q, \sigma) = q,$$

3. undefined, otherwise.

Selfloop operation is just to add additional selfloop transitions to a specific state in the original structure of a given generator. Since no selfloop transition $\sigma \in \Sigma_o$ is added to the states of G_1 at which σ is already defined, the selfloop operation does not destroy the property of determinism of G_1 . Also, the reachable, coreachable and trim properties are preserved by the selfloop operation.

Secondly, let $G_i = (Q_i, \Sigma_i, \delta_i, q_{i0}, Q_{i,m})$ for $i = 1, 2$ be trim, finite automata. Assume that $L_m(G_2) \subseteq L_m(G_1)$. Then we say that G_2 *refines* G_1 [WR87] if

$$(\forall s, t \in \overline{L_m(G_2)}), \delta_2(q_{20}, s) = \delta_2(q_{20}, t) \text{ implies } \delta_1(q_{10}, s) = \delta_1(q_{10}, t).$$

Also, if G_2 refines G_1 , then there exists a unique function

$$h : Q_2 \rightarrow Q_1,$$

satisfying

$$h \circ \delta_2(q_{20}, s) = \delta_1(q_{10}, s) \text{ for } s \in \overline{L_m(G_2)}.$$

The following will be used later.

Lemma 5.1 ([WR87]) *Let $G_i := (Q_i, \Sigma_i, \delta_i, q_{i0}, Q_{i,m})$ for $i = 1, 2$ be trim, finite automata. Let $G = G_1 \times G_2$. Then G refines G_1 and G_2 . \square*

5.2 Arrangement for the Shared-Event-Marking Condition

In this section, we present a procedure to arrange the system structure for the shared-event-marking condition. Let $G_i = (Q_i, \Sigma_i, \delta_i, q_{i0}, Q_{i,m})$ be generators over the alphabet $\Sigma_i \subseteq \Sigma$ for $i = 1, 2, \dots, n$. Let $\Sigma_i = \Sigma_{ic} \cup \Sigma_{iu}$. Let L_i and $L_{i,m}$ be respectively the closed and

marked behaviours of G_i . We assume that $\overline{L_{i,m}} = L_i$. Suppose that the shared-event-marking condition in Theorem 3.1 is not satisfied for some G_i , for $i \in \{1, 2, \dots, n\}$. In other words, $L_{i,m}$ does not mark $\Sigma_i \cap \Sigma_j$ for some j and $j \neq i$. For this situation, we develop the following procedure to modify the system G_i so that the resultant generators $G'_i = (Q_i, \Sigma_i, \delta_i, q_{i0}, Q'_{i,m})$ will satisfy the shared-event-marking condition. That is, for all $i, j \in \{1, 2, \dots, n\}$ and $i \neq j$,

$$\Sigma_i^*(\Sigma_i \cap \Sigma_j) \cap \overline{L'_{i,m}} \subseteq L'_{i,m}(\Sigma_i \cap \Sigma_j), \quad (5.1)$$

where $L'_{i,m}$ is the marked behaviour of G'_i . The new marker states of G'_i , $Q'_{i,m} \subseteq Q_i$, is obtained by the following procedure. Assume that all the states are numbered, i.e., $Q_i = \{q_0, q_1, \dots, q_{l_i}\}$, where $l_i + 1$ is the number of states of G_i . Let $\Sigma_{G_i}(q_k)$ be the active event set at the state $q_k \in Q_i$.

Procedure I (Shared-Event-Marking Condition)

$Q'_{i,m} := Q_{i,m}$.

$k := 0$.

For $k \leq l_i$ do

If $q_k \notin Q_{i,m}$ and $\Sigma_{G_i}(q_k) \cap (\bigcup_{j \neq i}(\Sigma_j)) \neq \phi$, then

$Q'_{i,m} := Q'_{i,m} \cup \{q_k\}$.

end if

$k := k + 1$.

end for

$G'_i := (Q_i, \Sigma_i, \delta_i, q_{i0}, Q'_{i,m})$. □

This procedure simply adds markings to the unmarked states of G_i which contain some shared events in their active event sets. The property of nonblocking in the original structure is preserved, i.e., $\overline{L'_{i,m}} = L'_i$. It is assumed that the generators G_i have finite sets of transitions and states. Therefore, the procedure will stop in a finite number of steps. Lemma 5.2 shows that the resultant DES G'_i obtained by Procedure I will satisfy the shared-event-marking condition.

Lemma 5.2 *Suppose that for given systems G_i , for $i = 1, 2, \dots, n$, DES G'_i are constructed by Procedure I. Then G'_i marks $\Sigma_i \cap \Sigma_j$ for any $i \neq j$.*

Proof: Let $s\sigma \in \Sigma_i^*(\Sigma_i \cap \Sigma_j) \cap \overline{L'_{i,m}}$. That is $s \in \Sigma_i^*$, $\sigma \in (\Sigma_i \cap \Sigma_j)$ and $s\sigma \in \overline{L'_{i,m}} = L'_i$. So, $s \in L'_i$. Let $q := \delta_i(q_{i0}, s)$. Since $s\sigma \in L'_i$, i.e., $\delta_i(q_{i0}, s\sigma)$ is defined, $\sigma \in \Sigma_{G_i}(q)$. Therefore, $\Sigma_{G_i}(q) \cap (\bigcup_{j \neq i}(\Sigma_j)) \neq \phi$. So, $q \in Q'_{i,m}$. That is $\delta_i(q_{i0}, s) \in Q'_{i,m}$. Hence $s \in L'_{i,m}$. Therefore, $s\sigma \in L'_{i,m}(\Sigma_i \cap \Sigma_j)$. \square

Remark 5.1 In some situations, another way to arrange the shared-event-marking condition may be to ‘internalise’ some of the shared events if that is meaningful and feasible in a particular application. Intuitively, internalisation of a shared event may be interpreted as follows: consider a system with two machine cells. Assume that there is one robot which transfers the products of the two cells to the next assembly line. Since they share the robot, the events which model the behaviours of the robot are shared events. However, if one more robot is available and is used to transfer the products of one system and the existing robot is used for the other system, then the events for the robot behaviour are not shared events any more. In this way, some shared events can be internalised. \diamond

Remark 5.2 It may be the case that many of the states need to be marked for the shared-event-marking condition to be satisfied. In some applications the author found that this is indeed the case. To deal with this there may be two approaches: one is to weaken the shared event marking condition in our result. We will attempt to do this in Chapter 6. The other may be to minimise the number of shared events by appropriate decomposition of the system. This may be seen as minimising interactions among subsystems. This approach seems to agree with the philosophy in the papers [Par72, PCW85]. Internalisation of the shared events might be another way to minimise the interactions among subsystems. \diamond

5.3 Arrangement for the Mutual Controllability Condition

In this section we present algorithms to arrange the system for the mutual controllability condition. Let $G_i = (Q_i, \Sigma_i, \delta_i, q_{i0})$ be the minimal, reachable generator for L_i for $i = 1, 2, \dots, n$. Thus, $L(G_i) = L_i$. Assume that $\Sigma_i = \Sigma_{iu} \dot{\cup} \Sigma_{ic}$ and $\Sigma_{iu} \cap \Sigma_j = \Sigma_{ju} \cap \Sigma_i = \Sigma_{iu} \cap \Sigma_{ju}$, for $i, j \in \{1, 2, \dots, n\}$ and $i \neq j$. Suppose that the mutual controllability condition between

G_i and G_j are not satisfied. We develop an algorithm to modify the given systems so that the resultant systems, G'_i and G'_j , will satisfy the mutual controllability condition. That is, for all $i, j \in \{1, 2, \dots, n\}$ and $i \neq j$,

$$\begin{aligned} L'_i(\Sigma_{ju} \cap \Sigma_i) \cap p_i^{ij}(p_j^{ij})^{-1}L'_j &\subseteq L'_i, \\ L'_j(\Sigma_{iu} \cap \Sigma_j) \cap p_j^{ij}(p_i^{ij})^{-1}L'_i &\subseteq L'_j, \end{aligned} \quad (5.2)$$

where $L'_i = L(G'_i)$, and p_i^{ij} and p_j^{ij} are natural projections from $(\Sigma_i \cup \Sigma_j)^*$ to Σ_i^* and to Σ_j^* , respectively.

Firstly, we introduce the following selfloop algorithm. Let $G_a := (Q_a, \Sigma_a, \delta_a, q_{a0})$ and $G_b := (Q_b, \Sigma_b, \delta_b, q_{b0})$ be the finite, reachable generators. Assume that $\Sigma_a = \Sigma_{au} \dot{\cup} \Sigma_{ac}$, $\Sigma_b = \Sigma_{bu} \dot{\cup} \Sigma_{bc}$, $\Sigma_{au} \cap \Sigma_b = \Sigma_{bu} \cap \Sigma_a$ and $\Sigma = \Sigma_a \cup \Sigma_b$. Recall that $\Sigma_{G_i}(q)$ represents the active event set at the state $q \in Q_i$.

Selfloop Algorithm

$$G_{ab} := G_a \times G_b = (Q_a \times Q_b, \Sigma, \delta_{ab}, (q_{a0}, q_{b0})).$$

Define

$$\begin{aligned} h : Q_a \times Q_b &\longrightarrow Q_b, & g : Q_a \times Q_b &\longrightarrow Q_a, \\ (q_a, q_b) &\longmapsto q_b, & (q_a, q_b) &\longmapsto q_a. \end{aligned}$$

Number $Q_a \times Q_b$: $Q_a \times Q_b = \{q_0, q_1, \dots, q_l\}$.

$$G_a^s(-1) := G_a.$$

$$k := 0.$$

For $k \leq l$ do

$$\Sigma_{sk} := \Sigma_{G_b}(h(q_k)) \cap (\Sigma_{au} \cap \Sigma_b) - \Sigma_{G_{ab}}(q_k).$$

If $\Sigma_{sk} \neq \phi$, then

$$G_a^s(k) := S_{\Sigma_{sk}}^{g(q_k)}(G_a^s(k-1)).$$

end if

$$k = k + 1$$

end for

$$G_a^s := G_a^s(k). \quad \square$$

We denote this algorithm as the following function for notational simplicity:

$$G_a^s = S(G_a, G_b).$$

This algorithm simply adds selfloops of shared uncontrollable events to some states of DES G_a when necessary. Hence the general structure of DES G_a is not changed except that selfloops are added to some states.

We note that G_a^s is generally not controllable with respect to G_b as shown in Example 5.1.

Example 5.1 Consider two systems G_a and G_b over the alphabet $\Sigma = \{\alpha, \beta, \gamma, \delta\}$. Assume that all the events are shared uncontrollable events and all the states are marked. Let G_a

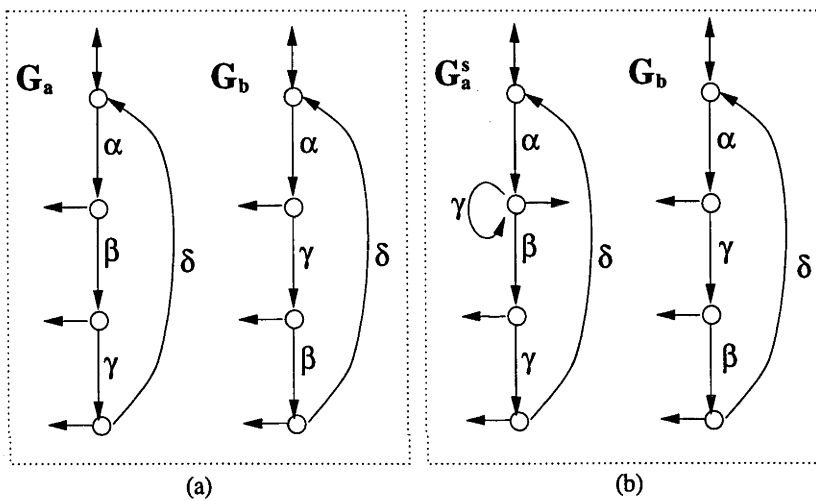


Figure 5.1: Example for controllability of G_a^s and G_b

and G_b be given as in Figure 5.1(a). So $L(G_a \times G_b) = \{\epsilon, \alpha\}$. By the selfloop algorithm, $G_a^s = S(G_a, G_b)$ can be obtained and is displayed in Figure 5.1(b). Clearly, G_a^s is not controllable with respect to G_b . ◇

Let $G_a := (Q_a, \Sigma_a, \delta_a, q_{a0})$ and $G_b := (Q_b, \Sigma_b, \delta_b, q_{b0})$ be defined as in the selfloop algorithm. Let p_a and p_b be the natural projections from $(\Sigma_a \cup \Sigma_b)^*$ to Σ_a^* and to Σ_b^* , respectively. The following algorithm produces two systems such that the closed behaviour of one system is controllable with respect to the ‘external’ behaviour of the other system and the shared uncontrollable events (for one direction only):

Algorithm I (One Directional Controllability)

1. $\mathbf{G}_b^P := (Q_b^P, \Sigma_a, \delta_b^P, q_{b0}^P)$ such that $L(\mathbf{G}_b^P) = p_a(p_b)^{-1}L(\mathbf{G}_b)$.
2. Obtain $\mathbf{G}_{as}^1 := \mathbf{S}(\mathbf{G}_a, \mathbf{G}_b^P)$.
3. $k := 1$.
4. Find $\mathbf{G}_{as}^{k+1} = \mathbf{S}(\mathbf{G}_{as}^k, \mathbf{G}_b^P)$.
5. If $\mathbf{G}_{as}^{k+1} = \mathbf{G}_{as}^k$, then $\mathbf{G}_f := \mathbf{G}_{as}^{k+1}$ and stop. □
6. Otherwise, $k := k + 1$ and go to Step 4.

Again, for notational convenience, we denote Algorithm I as a function:

$$\mathbf{G}_f = \mathbf{F}_1(\mathbf{G}_a, \mathbf{G}_b).$$

Figure 5.2 is the flow chart of Algorithm I.

Algorithm I keeps adding selfloops of shared uncontrollable events to some states in \mathbf{G}_a until $L(\mathbf{G}_a^k)$ is controllable with respect to $p_a(p_b)^{-1}L(\mathbf{G}_b)$ and $(\Sigma_{au} \cap \Sigma_b)$. Note that since we have assumed that the generators \mathbf{G}_i have finite numbers of states and transitions, Algorithm I will converge in a finite number of steps. Proposition 5.1 shows that $L(\mathbf{G}_f)$ is controllable.

Proposition 5.1 *Let $\mathbf{G}_a := (Q_a, \Sigma_a, \delta_a, q_{a0})$ and $\mathbf{G}_b := (Q_b, \Sigma_b, \delta_b, q_{b0})$ be defined as in the selfloop algorithm. Let p_a and p_b be natural projections from $(\Sigma_a \cup \Sigma_b)^*$ to Σ_a^* and to Σ_b^* , respectively. Let $\Sigma_u = \Sigma_{au} \cap \Sigma_b = \Sigma_{bu} \cap \Sigma_a$. Let $\mathbf{G}_b^P := (Q_b^P, \Sigma_a, \delta_b^P, q_{b0}^P)$ be such that $L(\mathbf{G}_b^P) = p_a(p_b)^{-1}L(\mathbf{G}_b)$. Let $\mathbf{G}_f := \mathbf{F}_1(\mathbf{G}_a, \mathbf{G}_b)$. Then $L(\mathbf{G}_f)$ is controllable with respect to $L(\mathbf{G}_b^P)$ and Σ_u , i.e.,*

$$L(\mathbf{G}_f)\Sigma_u \cap L(\mathbf{G}_b^P) \subseteq L(\mathbf{G}_f). \quad \square$$

We need the following lemmas to prove Proposition 5.1.

Lemma 5.3 *Let $\mathbf{G}_i := (Q_i, \Sigma_i, \delta_i, q_{i0})$, for $i = 1, 2$, be the reachable generators. Let $\Sigma = \Sigma_1 \cup \Sigma_2$ and $\Sigma = \Sigma_u \dot{\cup} \Sigma_c$. Suppose that \mathbf{G}_1 refines \mathbf{G}_2 . (Thus, there exists a unique function $h : Q_1 \rightarrow Q_2$ such that $h \circ \delta_1(q_{10}, s) = \delta_2(q_{20}, s)$ for $s \in L(\mathbf{G}_1)$.)*

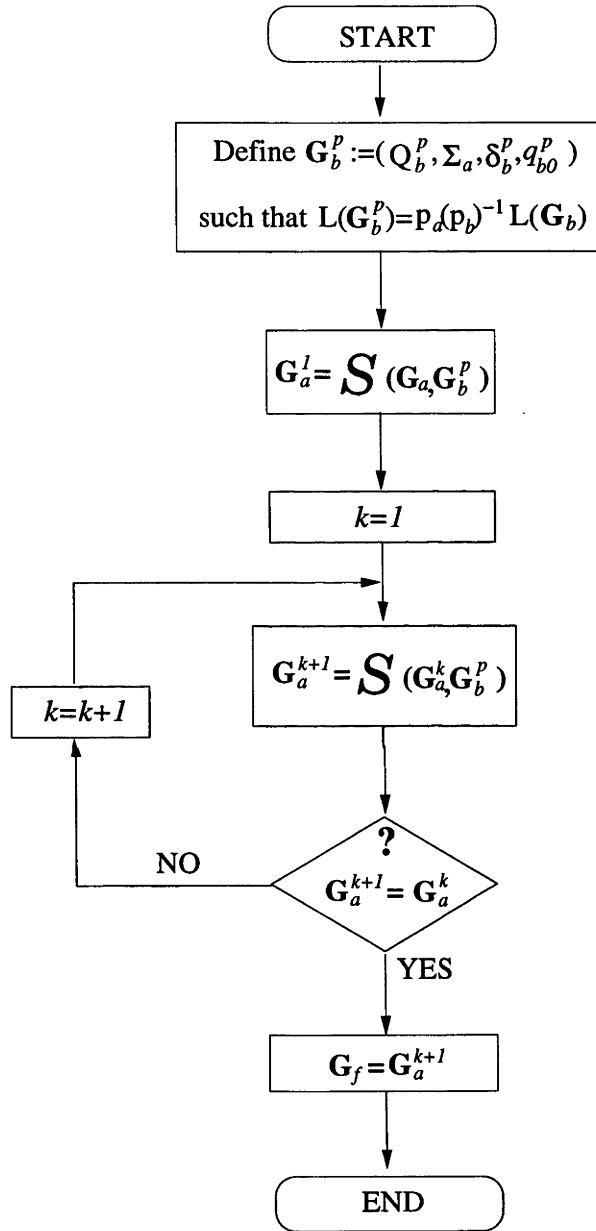


Figure 5.2: Flow chart for Algorithm I: $G_f = F_1(G_a, G_b)$

Then

$$L(\mathbf{G}_1)(\Sigma_u) \cap L(\mathbf{G}_2) \subseteq L(\mathbf{G}_1)$$

if and only if

$$(\forall q \in Q_1), \Sigma_{\mathbf{G}_2}(h(q)) \cap \Sigma_u \subseteq \Sigma_{\mathbf{G}_1}(q).$$

Proof: For one implication (\implies), consider an event σ such that, for $q \in Q_1$,

$$\sigma \in \Sigma_{\mathbf{G}_2}(h(q)) \cap \Sigma_u.$$

Since $q \in Q_1$ and \mathbf{G}_1 is reachable, there exists a string $s \in L(\mathbf{G}_1)$ such that

$$q = \delta_1(q_{10}, s) \in Q_1.$$

Also, $\sigma \in \Sigma_{\mathbf{G}_2}(h(q)) = \Sigma_{\mathbf{G}_2}(h(\delta_1(q_{10}, s))) = \Sigma_{\mathbf{G}_2}(\delta_2(q_{20}, s))$. That is

$$\delta_2(q_{20}, s\sigma) \text{ is defined, or } s\sigma \in L(\mathbf{G}_2).$$

Therefore

$$s\sigma \in L(\mathbf{G}_1)(\Sigma_u) \cap L(\mathbf{G}_2) \subseteq L(\mathbf{G}_1).$$

Hence $\delta_1(q_{10}, s\sigma) \in Q_1$ or $\delta_1(q_{10}, s\sigma)$ is defined. So $\sigma \in \Sigma_{\mathbf{G}_1}(\delta_1(q_{10}, s)) = \Sigma_{\mathbf{G}_1}(q)$.

For the reverse implication, consider a string $s\sigma \in L(\mathbf{G}_1)(\Sigma_u) \cap L(\mathbf{G}_2)$, i.e.,

$$s \in L(\mathbf{G}_1), \sigma \in (\Sigma_u), \text{ and } s\sigma \in L(\mathbf{G}_2).$$

Hence $\delta_1(q_{10}, s) \in Q_1$. Therefore by assumption,

$$\Sigma_{\mathbf{G}_2}(h(\delta_1(q_{10}, s))) \cap \Sigma_u \subseteq \Sigma_{\mathbf{G}_1}(\delta_1(q_{10}, s)),$$

or,

$$\Sigma_{\mathbf{G}_2}(\delta_2(q_{20}, s)) \cap \Sigma_u \subseteq \Sigma_{\mathbf{G}_1}(\delta_1(q_{10}, s)).$$

Since $s\sigma \in L(\mathbf{G}_2)$, one has that $\delta_2(q_{20}, s\sigma) \in Q_2$. That is

$$\sigma \in \Sigma_{\mathbf{G}_2}(\delta_2(q_{20}, s)).$$

Therefore, since $\sigma \in \Sigma_u$,

$$\sigma \in \Sigma_{\mathbf{G}_2}(\delta_2(q_{20}, s)) \cap \Sigma_u \subseteq \Sigma_{\mathbf{G}_1}(\delta_1(q_{10}, s)).$$

That is

$$\delta_1(q_{10}, s\sigma) \text{ is defined, or } s\sigma \in L(\mathbf{G}_1). \quad \square$$

Lemma 5.4 For given languages $K, L \subseteq \Sigma^*$,

$$\overline{K}\Sigma_u \cap L \subseteq \overline{K} \text{ if and only if } (\overline{K} \cap L)\Sigma_u \cap L \subseteq (\overline{K} \cap L).$$

Proof: For one implication (\implies),

$$\begin{aligned}
 (\overline{K} \cap L)\Sigma_u \cap L &= \overline{K}\Sigma_u \cap L\Sigma_u \cap L \\
 &\subseteq \overline{K}\Sigma_u \cap L \\
 &= \overline{K}\Sigma_u \cap L \cap L \\
 &\subseteq \overline{K} \cap L. \quad (\text{by assumption})
 \end{aligned}$$

For the reverse inclusion, Let $s\sigma \in \overline{K}\Sigma_u \cap L$. That is,

$$s \in \overline{K}, \sigma \in \Sigma_u \text{ and } s\sigma \in L.$$

Hence, $s \in \overline{K} \cap L$. Therefore,

$$s\sigma \in (\overline{K} \cap L)\Sigma_u \cap L \subseteq (\overline{K} \cap L) \quad (\text{by assumption}).$$

So, $s\sigma \in \overline{K}$. □

Now, one is ready to prove Proposition 5.1.

Proof of Proposition 5.1: Recall that the selfloop algorithm $\mathbf{S}(\mathbf{G}_a, \mathbf{G}_b^P)$ simply adds self-loops of some shared uncontrollable events to some states of the original system \mathbf{G}_a . So, when one obtains \mathbf{G}_f in Step 5 of Algorithm I, one has that $\mathbf{G}_f = \mathbf{G}_{as}^k = \mathbf{G}_{as}^{k+1}$. Let us define

$$\mathbf{G}_{as}^k := (Q_a, \Sigma_a, \delta_a^k, q_{a0}) \text{ and } \mathbf{G}_{as}^{k+1} := (Q_a, \Sigma_a, \delta_a^{k+1}, q_{a0}),$$

where δ_a^k and δ_a^{k+1} are the original transition function δ_a plus some selfloops added by the selfloop algorithm. In the first step of the selfloop algorithm, one has that

$$\mathbf{G}_{ab}^k = \mathbf{G}_{as}^k \times \mathbf{G}_b^P, \text{ and hence } \mathbf{G}_{ab}^k \text{ refines } \mathbf{G}_{as}^k \text{ and } \mathbf{G}_b^P.$$

Therefore, there are unique functions

$$h : Q_a \times Q_b^P \longrightarrow Q_b^P, \quad g : Q_a \times Q_b^P \longrightarrow Q_a.$$

Since $\mathbf{G}_{as}^k = \mathbf{G}_{as}^{k+1}$, one has that $\delta_a^k = \delta_a^{k+1}$. Thus, for $q \in Q_a \times Q_b^P$, one has that $\Sigma_{\mathbf{G}_b^P}(h(q)) \cap \Sigma_u \subseteq \Sigma_{\mathbf{G}_{ab}^k}(q)$. Since \mathbf{G}_{ab}^k refines \mathbf{G}_{as}^k and \mathbf{G}_b^P , by Lemma 5.3,

$$L(\mathbf{G}_{ab}^k)\Sigma_u \cap L(\mathbf{G}_b^P) \subseteq L(\mathbf{G}_{ab}^k).$$

Since $L(\mathbf{G}_{ab}^k) = L(\mathbf{G}_{as}^k) \cap L(\mathbf{G}_b^P)$ and $\mathbf{G}_f = \mathbf{G}_{as}^k$, by Lemma 5.4

$$L(\mathbf{G}_f)\Sigma_u \cap L(\mathbf{G}_b^P) \subseteq L(\mathbf{G}_f). \quad \square$$

We now establish the algorithm for the mutual controllability condition for a given pair of systems, say \mathbf{G}_1 and \mathbf{G}_2 . The flow chart for the algorithm is given in Figure 5.3. Again for notational convenience, we denote the following algorithm as a function:

$$(\mathbf{G}'_1, \mathbf{G}'_2) = \mathbf{F}_2(\mathbf{G}_1, \mathbf{G}_2).$$

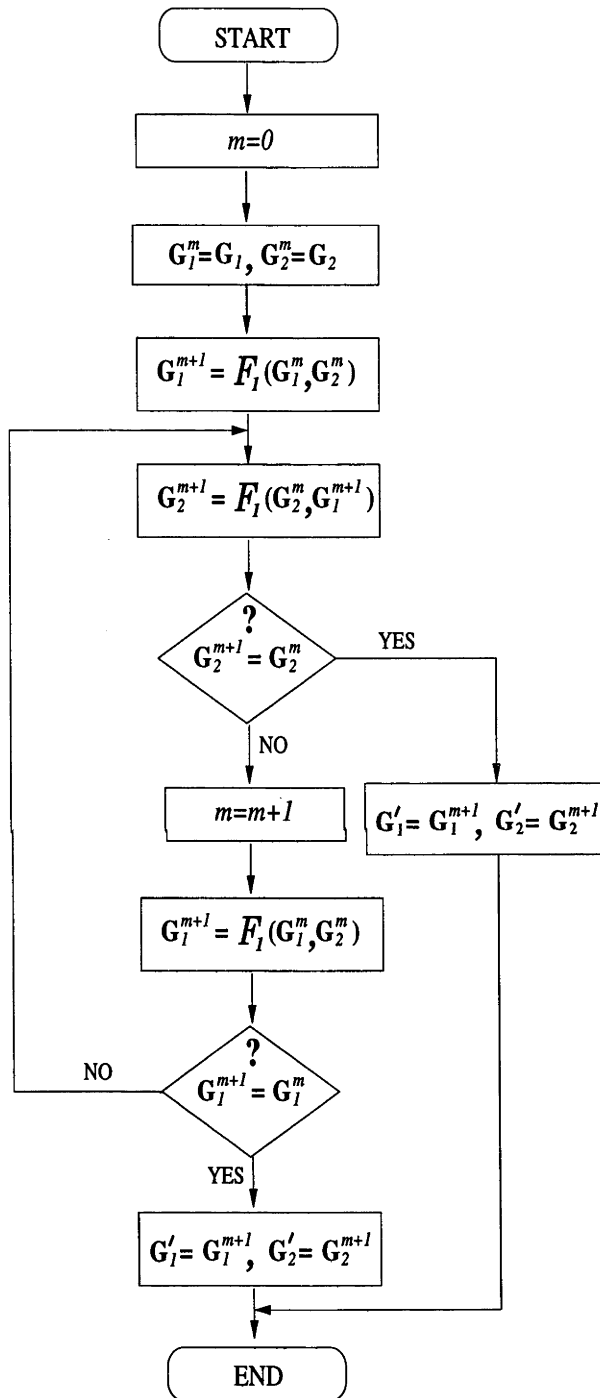


Figure 5.3: Flow chart for Algorithm II: $(G'_1, G'_2) = F_2(G_1, G_2)$

Algorithm II (Mutual Controllability for a pair of systems)

1. Let $m = 0$.
2. Let $G_1^m = G_1, G_2^m = G_2$.
3. Using Algorithm I, find $G_1^{m+1} = F_1(G_1^m, G_2^m)$.
4. Again using Algorithm I, find $G_2^{m+1} = F_1(G_2^m, G_1^{m+1})$.
5. If $G_2^{m+1} = G_2^m$, then let $G_1' = G_1^{m+1}$ and $G_2' = G_2^{m+1}$ and stop.
6. Otherwise $m = m + 1$ and go to Step 7.
7. Find $G_1^{m+1} = F_1(G_1^m, G_2^m)$.
8. If $G_1^{m+1} = G_1^m$, then let $G_1' = G_1^{m+1}$ and $G_2' = G_2^{m+1}$ and stop.
9. Otherwise go to Step 4. □

Algorithm II is the procedure which adds selfloops of shared uncontrollable events to some states in a given pair of systems G_1 and G_2 repeatedly until the mutual controllability condition, Eq 5.2, is satisfied. Since the given pair of DES G_1 and G_2 are assumed to have finite number of states and transitions, Algorithm II will stop in a finite number of steps. The proof that L_1' and L_2' are mutually controllable is given in Proposition 5.2, where $L_i' = L(G_i')$ for $i = 1, 2$.

Proposition 5.2 *For a given pair of systems G_1 and G_2 , suppose that DES G_1' and G_2' are obtained by Algorithm II, i.e., $(G_1', G_2') = F_2(G_1, G_2)$. Then L_1' and L_2' are mutually controllable.*

Proof: The Algorithm II proceeds explicitly as in Figure 5.4. At the beginning, we obtain G_1^1 from the pair (G_1, G_2) , using Algorithm I (Step Ⓐ). So, by Proposition 5.1, one knows that $L(G_1^1)$ is controllable with respect to the 'external' behaviours of $L(G_2)$ and $\Sigma_{2u} \cap \Sigma_1$. Then again using Algorithm I, we obtain G_2^1 from the pair (G_2, G_1^1) , (Step Ⓑ). Again by Proposition

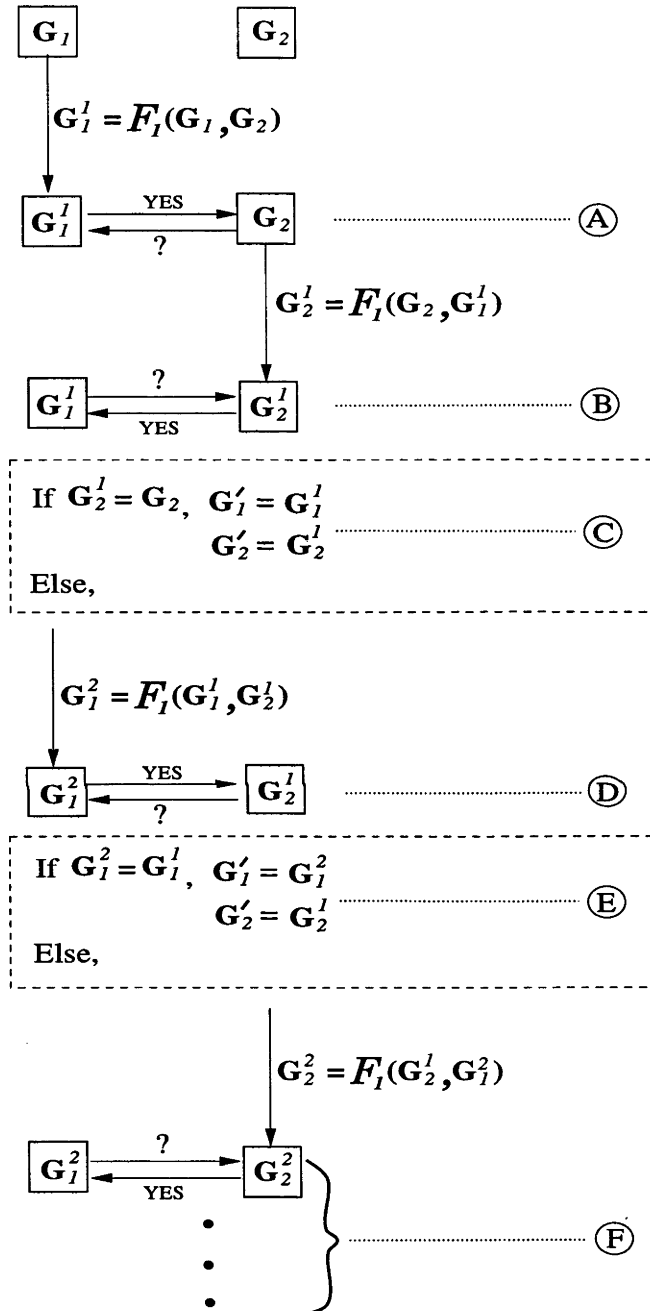


Figure 5.4: Procedure of Algorithm II

5.1, $L(\mathbf{G}_2^1)$ is controllable with respect to the ‘external’ behaviours of $L(\mathbf{G}_1^1)$ and $\Sigma_{1u} \cap \Sigma_2$. If $\mathbf{G}_2^1 = \mathbf{G}_2$, one has that $L(\mathbf{G}_1^1)$ and $L(\mathbf{G}_2^1)$ are mutually controllable by the controllability obtained in Step (A) and (B). So, we denote $\mathbf{G}'_1 = \mathbf{G}_1^1$ and $\mathbf{G}'_2 = \mathbf{G}_2^1$ (Step (C)). Otherwise, using procedure II again, we obtain \mathbf{G}_1^2 (Step (D)). One knows that $L(\mathbf{G}_1^2)$ is controllable with respect to the ‘external’ behaviours of $L(\mathbf{G}_2^1)$ and $\Sigma_{2u} \cap \Sigma_1$. In here, if $\mathbf{G}_1^2 = \mathbf{G}_1^1$, then $L(\mathbf{G}_1^2)$ and $L(\mathbf{G}_2^1)$ are mutually controllable. So, we denote $\mathbf{G}'_1 = \mathbf{G}_1^2$ and $\mathbf{G}'_2 = \mathbf{G}_2^1$ (Step (E)). We repeat the procedure until we find a pair which satisfies the mutual controllability condition (Step (F)). So the resultant pair L'_1 and L'_2 are mutually controllable. \square

Finally the algorithm to arrange the mutual controllability condition for a system with n subsystems is as follows:

Algorithm III (Mutual Controllability for a system with n subplants)

1. Let $\mathbf{G}_j^0 = \mathbf{G}_j$ for $j = 1, 2, \dots, n$.
2. Let $k = 0, i = 1$.
3. $(\mathbf{G}_i^{k+1}, \mathbf{G}_{i+1}^{k+1}) = \mathbf{F}_2(\mathbf{G}_i^k, \mathbf{G}_{i+1}^k)$.
4. Do $(\mathbf{G}_{i+1}^{k+2}, \mathbf{G}_{i+2}^{k+2}) = \mathbf{F}_2(\mathbf{G}_{i+1}^{k+1}, \mathbf{G}_{i+2}^k)$, $i = i + 1$, while $i \leq n - 2$.
5. $(\mathbf{G}_n^{k+2}, \mathbf{G}_1^{k+2}) = \mathbf{F}_2(\mathbf{G}_n^{k+1}, \mathbf{G}_1^{k+1})$.
6. If $\mathbf{G}_i^{k+2} = \mathbf{G}_i^k$ for all $i = 1, 2, \dots, n$ then goto Step 8.
7. Otherwise let $k = k + 2$ and $i = 1$, and go to Step 3.
8. Let $\mathbf{G}'_i = \mathbf{G}_i^{k+2}$ for all $i = 1, 2, \dots, n$. \square

Algorithm III proceeds as displayed in Figure 5.5. Algorithm III is to arrange all the pairs of subsystems in a given plant using Algorithm II so that each pair is mutually controllable. Since the number of plants is finite and each subsystem is assumed to have finite number of states and transitions, Algorithm III is guaranteed to converge in a finite number of steps. The following proposition shows that each pair of the L'_i is mutually controllable.

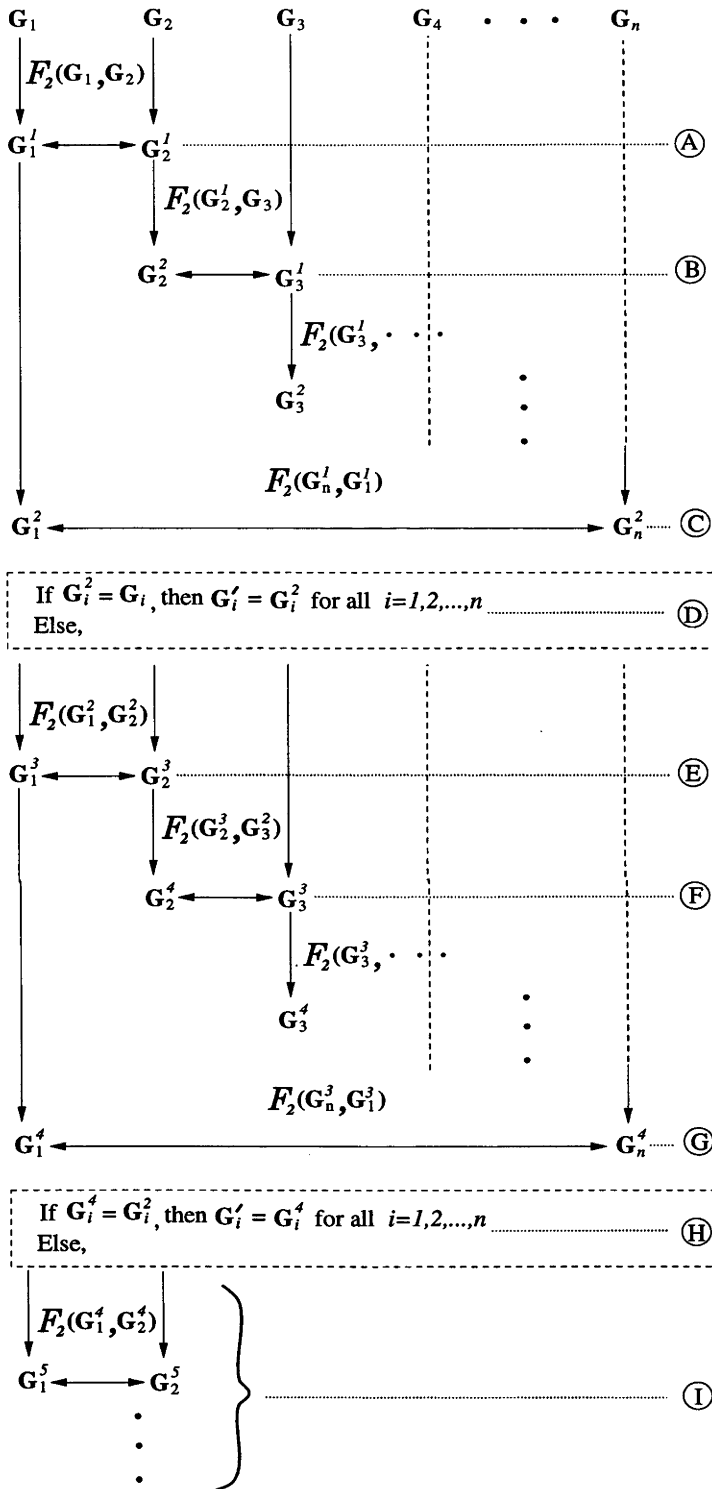


Figure 5.5: Procedure of Algorithm III

Proposition 5.3 *For given systems G_i , for $i = 1, 2, \dots, n$, suppose that the systems G_i^l are obtained by Algorithm III. Then each pair of the L_i^l is mutually controllable.*

Proof: Since the number of plants is finite and we assume that each subsystem has finite numbers of states and transitions, Algorithm III will stop. Thus for some k , one has that $G_i^k = G_i^{k+2}$, meaning that no additional selfloops are required for the mutual controllability conditions. Hence one has that $G_i^k = G_i^{k+1} = G_i^{k+2}$. Therefore, clearly all pairs are mutually controllable. \square

There are some special situations in which no additional selfloops are required for the mutual controllability condition. For instance, see the first example in Section 3.4. For another example, consider a waste neutralisation system, borrowed from [Als96], as given in Figure 5.6 (a). A continuous stream of alkaline industrial waste is neutralised by an injection of acid via pump P_1 and the valve V_1. Acidic wastes are neutralised by the basic liquid via P_2 and V_2. The pH level of the stream is measured continuously by pH probe H_1. Their DES models are displayed in Figure 5.6 (b). The plant is naturally decomposed into two subplants; the subplant G_1 for the treatment of basic waste, and the other subplant G_2 for acidic waste. Their event sets are

$$\begin{aligned}\Sigma_1 &= \{\alpha_1, \beta_1, \gamma_1, \delta_1, \eta_1, \eta_2, \eta_3, \eta_4\}, \\ \Sigma_2 &= \{\alpha_2, \beta_2, \gamma_2, \delta_2, \eta_1, \eta_2, \eta_3, \eta_4\}.\end{aligned}$$

Note that H_1 is the shared component and all the events of H_1 are uncontrollable. The DES models for subplants, G_1 and G_2 , are obtained by synchronous composition. It can be verified that G_1 and G_2 are mutually controllable. In fact mutual controllability holds in general for this type of systems as shown in Proposition 5.4.

Proposition 5.4 *Let G_i be generators over the alphabet Σ_i , for $i = 1, 2, 3$. Let L_i be the closed behaviour of the plant G_i . Let G_a and G_b such that their closed behaviours are respectively $L_a = L_1 \parallel L_2$ and $L_b = L_2 \parallel L_3$. Suppose that $\{\Sigma_i \mid i = 1, 2, 3\}$ is pairwise disjoint, i.e., $\Sigma_i \cap \Sigma_j = \phi$ for $i \neq j$. Then L_a and L_b are mutually controllable.*

Proof: Consider a string $s\sigma \in L_a(\Sigma_{bu} \cap \Sigma_a) \cap p_a(p_b)^{-1}(L_b)$, where $\Sigma_a = \Sigma_1 \cup \Sigma_2$ and $\Sigma_b = \Sigma_2 \cup \Sigma_3$, and p_a and p_b are the natural projections from Σ^* to $(\Sigma_a)^*$ and to $(\Sigma_b)^*$, respectively.

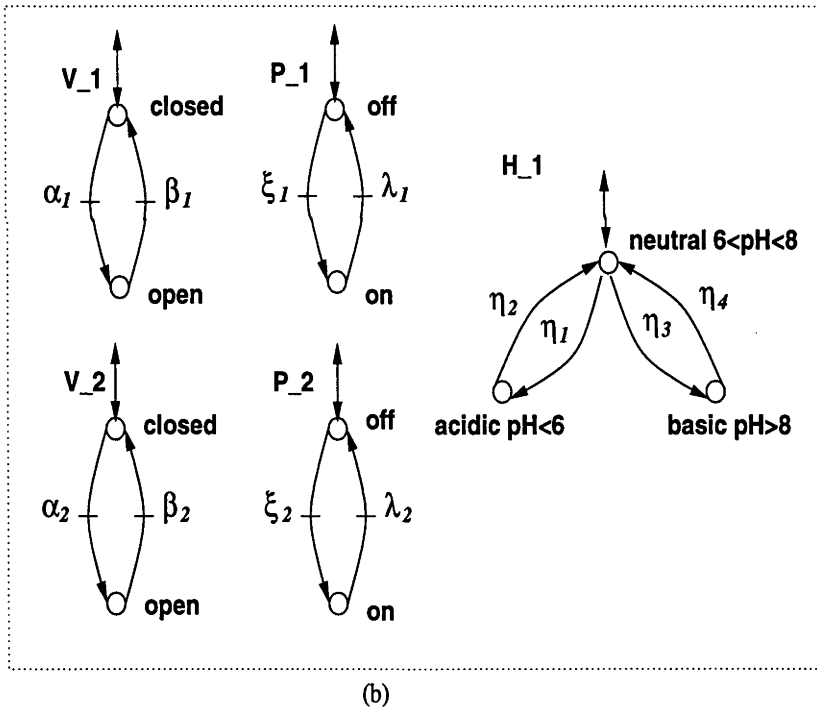
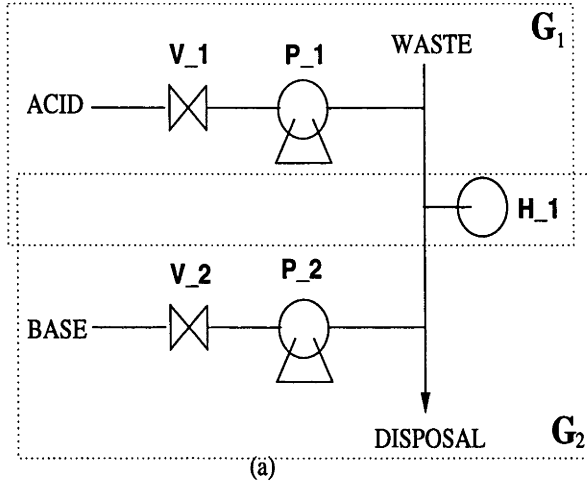


Figure 5.6: Waste neutralisation plant and DES models for its components

Hence $s \in L_a$, $\sigma \in (\Sigma_{bu} \cap \Sigma_a)$ and $s\sigma \in p_a(p_b)^{-1}(L_b)$. Since $\{\Sigma_i \mid i = 1, 2, 3\}$ is pairwise disjoint, $\Sigma_{bu} \cap \Sigma_a = \Sigma_{2u}$ and $p_a(p_b)^{-1}(L_b) = (p_2^a)^{-1}(L_2)$. So, $\sigma \in \Sigma_{2u}$ and $s\sigma \in (p_2^a)^{-1}(L_2)$. Also $s \in L_a = (P_1^a)^{-1}(L_1) \cap (P_2^a)^{-1}(L_2)$ where P_1^a and P_2^a are the natural projections from Σ_a^* to Σ_1^* and to Σ_2^* respectively. So, $s \in (P_1^a)^{-1}(L_1)$. Since $\sigma \in \Sigma_{2u}$ and $\{\Sigma_i \mid i = 1, 2, 3\}$ is pairwise disjoint, $s\sigma \in (P_1^a)^{-1}(L_1)$. Therefore, $s\sigma \in (P_1^a)^{-1}(L_1) \cap (P_2^a)^{-1}(L_2) = L_a$. \square

Remark 5.3 Intuitively, selflooping of shared uncontrollable events on one system G_1 might be interpreted as to represent the ‘monitoring’ of the other subsystem G_2 by G_1 , without G_1 taking any actions. In short one system simply allows shared uncontrollable events in the other subsystems to occur. \diamond

Remark 5.4 Here we consider other possible solutions for the arrangement of the mutual controllability condition. Firstly we recall the following. For given languages, $L \subseteq M \subseteq \Sigma^*$, the supremal controllable sublanguage of L with respect to M , denoted by L^\uparrow ,¹ may be interpreted to represent a modification of L so that the modified language, L^\uparrow , is controllable with respect to M . There is a ‘dual’ concept to the supremal controllable sublanguage, namely the infimal closed, controllable superlanguage [LW88b, LC90]. The infimal closed, controllable superlanguage of a language $L \subseteq M$, denoted L^\downarrow , is defined as

$$L^\downarrow = \inf\{K \mid \bar{L} \subseteq K, K = \bar{K} \text{ and } \bar{K}\Sigma_u \cap M \subseteq \bar{K}\},$$

where $\Sigma_u \subseteq \Sigma$ is the set of uncontrollable events. The closed-form expression of L^\downarrow is [LC90]

$$L^\downarrow = M \cap \bar{L}\Sigma_u^*.$$

In summary, we have the following inclusions:

$$\phi \subseteq L^\uparrow \subseteq L \subseteq \bar{L} \subseteq L^\downarrow \subseteq \Sigma^*.$$

Assume that we are given two languages $L_1, L_2 \subseteq \Sigma^*$ such that L_1 is not controllable with respect to the ‘external’ behaviour of L_2 , $p_1 p_2^{-1}(L_2)$, and vice versa. In other words, L_1 and L_2 are not mutually controllable. Let $L_i^p = p_j p_i^{-1}(L_i)$ for $i \neq j$. To modify L_1 and L_2 so that their modifications are mutually controllable, one way would be to use κ operator as shown in Figure 5.7. If the algorithm displayed in Figure 5.7 stops, then we have a mutually controllable pair of languages. However, it is still an open question whether this algorithm will stop in a

¹We use the notation $\kappa_M(L)$ throughout this thesis, but in here we use this notation for a clearer comparison.

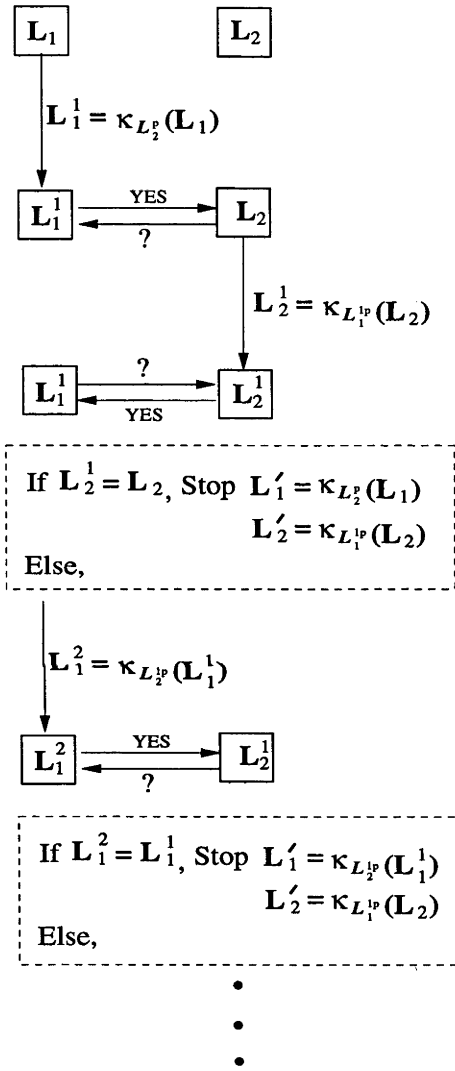


Figure 5.7: Algorithm for arranging mutual controllability using κ operator

finite number of steps.

The other way would be to use the infimal closed, controllable superlanguages of the given languages:

$$L_1^\downarrow = L_1 \Sigma_u^*, \quad L_2^\downarrow = L_2 \Sigma_u^*.$$

It can be shown that L_1^\downarrow and L_2^\downarrow are mutually controllable. Note that the language L'_i obtained by Algorithm II may be bigger than L_i^\downarrow . Figure 5.8 displays an example of the generators obtained by Algorithm II (b) and by infimal operator \downarrow (c). One can see that L_1^\downarrow could get into

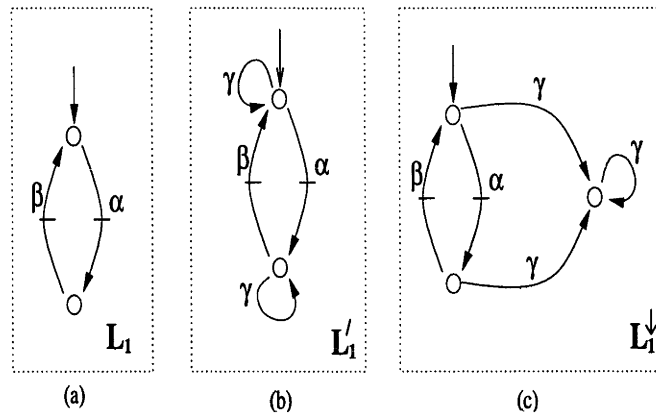


Figure 5.8: Algorithms for arranging mutual controllability

a ‘blocking’ situation. That is, once γ occurs, transitions in the original structure of L_1 are no longer possible. This may not be desirable. In a similar vein, the generators obtained by κ operator, if exist, also do not in general preserve the original structure of the given systems. However, L'_1 preserves the original structure and has an intuitive interpretation as discussed in Remark 5.3. ◇

Remark 5.5 If one needs to arrange a given system for both conditions, one should first arrange the system for the mutual controllability condition and then carry out the arrangement for the shared-event-marking condition. The reason is that the selflooping of shared uncontrollable events might change the active event sets of some of the states. This might require to mark additional states for the shared-event-marking condition. ◇

5.4 Example

In this section, we illustrate our results with two examples.

The first example illustrates the arrangement for the shared-event-marking condition. We consider a system with three subplants (as in Figure 5.9(a)). The event sets for three subplants are

$$\begin{aligned}\Sigma_1 &= \{\alpha_1, \alpha_2, \alpha_3, \alpha_4, \beta_1, \gamma_1\}, & \Sigma_2 &= \{\alpha_5, \alpha_6, \beta_1, \beta_2, \gamma_1\}, \\ \Sigma_3 &= \{\alpha_7, \alpha_8, \alpha_9, \mu_1, \beta_2, \gamma_1\}.\end{aligned}$$

Assume that all the events are controllable. Clearly, this system does not satisfy the shared-event-marking condition. Therefore, we need to modify the system to arrange for the desired condition. In \mathbf{G}_1 , the state A_0 is already marked. So $A_0 \in Q'_{1,m}$. The active event set for the state A_1 , $\Sigma_{\mathbf{G}_1}(A_1) = \{\beta_1\}$. So, $\Sigma_{\mathbf{G}_1}(A_1) \cap (\bigcup_{j \neq 1}(\Sigma_j)) \neq \phi$. Hence, $A_1 \in Q'_{1,m}$. Also, for the state A_2 , $\Sigma_{\mathbf{G}_1}(A_2) = \{\gamma_1\}$. So, $\Sigma_{\mathbf{G}_1}(A_2) \cap (\bigcup_{j \neq 1}(\Sigma_j)) \neq \phi$. Hence, $A_2 \in Q'_{1,m}$. However, for the state A_3 , $\Sigma_{\mathbf{G}_1}(A_3) = \{\alpha_2, \alpha_4\}$. So, $\Sigma_{\mathbf{G}_1}(A_3) \cap (\bigcup_{j \neq 1}(\Sigma_j)) = \phi$. Therefore, $A_3 \notin Q'_{1,m}$. Similarly, $A_4 \notin Q'_{1,m}$. The systems \mathbf{G}_2 and \mathbf{G}_3 can also be arranged similarly. The resultant systems \mathbf{G}'_1 , \mathbf{G}'_2 and \mathbf{G}'_3 are presented in Figure 5.9(b).

To illustrate the arrangement for the mutual controllability condition for a pair of systems, Algorithm II, we consider two systems modelled as in Figure 5.10(a). Assume that all the states are marked. Let \mathbf{G}_1 and \mathbf{G}_2 be generators of L_1 and L_2 respectively, defined as follows:

$$\mathbf{G}_1 := (Q_1, \Sigma_1, \delta_1, q_{10}), \quad \mathbf{G}_2 := (Q_2, \Sigma_2, \delta_2, q_{20}),$$

where

$$\begin{aligned}Q_1 &= \{A_0, A_1, A_2, A_3\}, & Q_2 &= \{B_0, B_1, B_2, B_3\}, \\ \Sigma_1 &= \{\alpha_1, \alpha_2, \alpha_3\}, & \Sigma_2 &= \{\alpha_1, \alpha_2, \alpha_4\}, \\ \delta_1 : (A_0, \alpha_1) &\mapsto A_1, & \delta_2 : (B_0, \alpha_1) &\mapsto B_1, \\ &(A_1, \alpha_2) \mapsto A_2, & &(B_1, \alpha_2) \mapsto B_1, \\ &(A_2, \alpha_3) \mapsto A_3, & &(B_1, \alpha_4) \mapsto B_2, \\ &(A_3, \alpha_2) \mapsto A_3, & &(B_2, \alpha_2) \mapsto B_3, \\ q_{10} &= A_0, & q_{20} &= B_0.\end{aligned}$$

The shared event set is $\Sigma_1 \cap \Sigma_2 = \{\alpha_1, \alpha_2\}$. Assume that α_1 is a controllable event, while α_2 is an uncontrollable event. So, $\Sigma_{1u} \cap \Sigma_{2u} = \{\alpha_2\}$. Then define a new generator \mathbf{G}_2^P such that

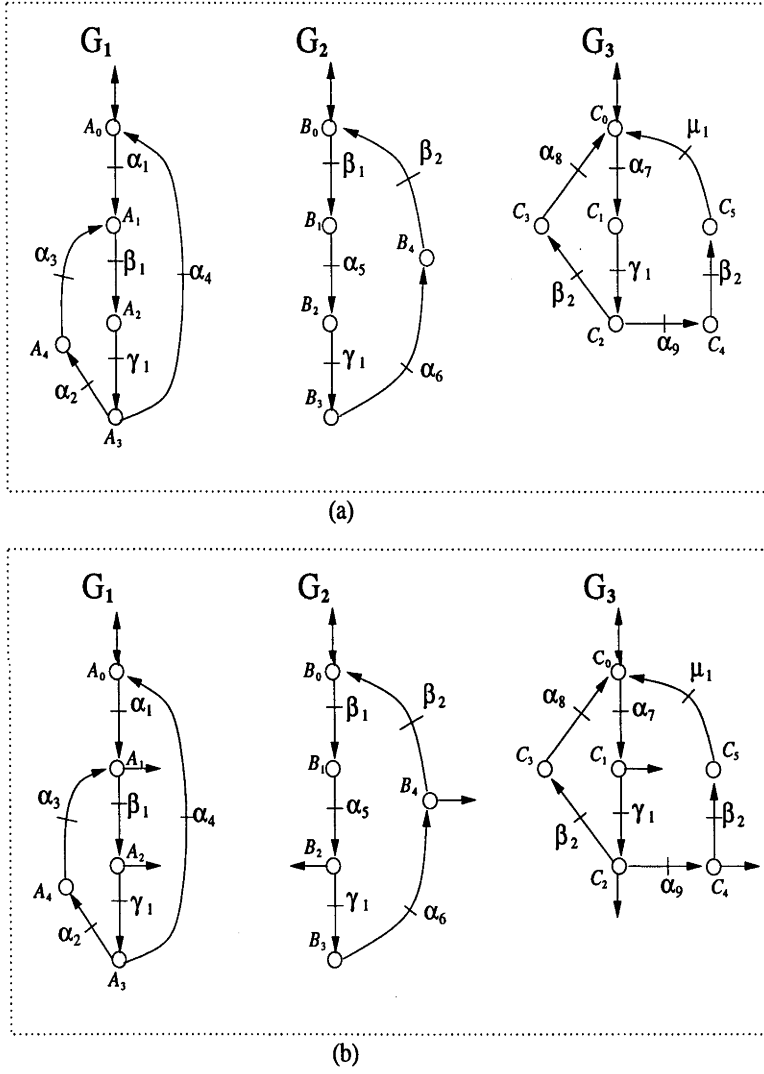


Figure 5.9: Example: arrange the system for the shared-event-marking condition

$L(\mathbf{G}_2^P) = p_1(p_2)^{-1}L(\mathbf{G}_2)$ (Figure 5.10(b));

$$\mathbf{G}_2^P := (Q_2^P, \Sigma_1, \delta_2^P, q_{20}^P),$$

where

$$\begin{aligned} Q_2^P &= \{C_0, C_1\}, \\ \delta_2^P : (C_0, \alpha_1) &\mapsto C_1, & (C_0, \alpha_3) &\mapsto C_0, \\ (C_1, \alpha_2) &\mapsto C_1, & (C_1, \alpha_3) &\mapsto C_1, \\ q_{20}^P &= C_0. \end{aligned}$$

Then compute the product $\mathbf{G}_{12}^P = \mathbf{G}_1 \times \mathbf{G}_2^P$ (Figure 5.10(c)):

$$\mathbf{G}_{12}^P = \mathbf{G}_1 \times \mathbf{G}_2^P = (Q_1 \times Q_2^P, \Sigma_1, \delta_1 \times \delta_2^P, (q_{10}, q_{20}^P)),$$

where

$$\begin{aligned} Q_1 \times Q_2^P &= \{(A_0, C_0), (A_1, C_1), (A_2, C_1), (A_3, C_1)\}, \\ \delta_1 \times \delta_2^P : ((A_0, C_0), \alpha_1) &\mapsto (A_1, C_1), & ((A_1, C_1), \alpha_2) &\mapsto (A_2, C_1), \\ ((A_2, C_1), \alpha_3) &\mapsto (A_3, C_1), & ((A_3, C_1), \alpha_2) &\mapsto (A_3, C_1), \\ (q_{10}, q_{20}^P) &= (A_0, C_0). \end{aligned}$$

Since \mathbf{G}_{12}^P refines \mathbf{G}_1 and \mathbf{G}_2^P , there exist unique functions

$$h_1 : Q_1 \times Q_2^P \longrightarrow Q_2^P \quad \text{and} \quad g_1 : Q_1 \times Q_2^P \longrightarrow Q_1,$$

such that

$$\begin{aligned} h_1(\delta_1, \delta_2^P)((q_{10}, q_{20}^P), s) &= \delta_2^P(q_{20}^P, s), \\ g_1(\delta_1, \delta_2^P)((q_{10}, q_{20}^P), t) &= \delta_1(q_{10}, t), \end{aligned}$$

where the strings s, t are in $\overline{L_m(\mathbf{G}_{12}^P)}$, the closure of the marked behaviour of \mathbf{G}_{12}^P . So one can easily find the functions h_1 and g_1 given as follows:

$$\begin{aligned} h_1(A_0, C_0) &\mapsto C_0, & h_1(A_1, C_1) &\mapsto C_1, \\ h_1(A_2, C_1) &\mapsto C_1, & h_1(A_3, C_1) &\mapsto C_1, \\ g_1(A_0, C_0) &\mapsto A_0, & g_1(A_1, C_1) &\mapsto A_1, \\ g_1(A_2, C_1) &\mapsto A_2, & g_1(A_3, C_1) &\mapsto A_3. \end{aligned}$$

Then, using the selfloop algorithm, $\mathbf{G}_{1s}^1 = \mathbf{S}(\mathbf{G}_1, \mathbf{G}_2^P)$ is computed (Step 2 in Algorithm I).

Firstly, for a state $(A_0, C_0) \in Q_1 \times Q_2^P$, since $(\Sigma_{1u} \cap \Sigma_{2u}) = \{\alpha_2\}$, $\Sigma(A_0, C_0) = \{\alpha_1\}$ and $\Sigma(h_1(A_0, C_0)) = \Sigma(C_0) = \{\alpha_1, \alpha_3\}$, one has that $\Sigma_s = \Sigma(h_1(A_0, C_0)) \cap (\Sigma_{1u} \cap$

$\Sigma_{2u}) - \Sigma(A_0, C_0) = \phi$. So, no selfloop is required at the state $g_1(A_0, C_0) = A_0 \in Q_1$. At the state (A_1, C_1) , α_2 is already defined. For the state (A_2, C_1) , since $\Sigma(A_2, C_1) = \{\alpha_3\}$ and $\Sigma(C_1) = \{\alpha_2, \alpha_3\}$, one has that $\Sigma_s = \{\alpha_2\}$. So one needs to selfloop α_2 at the state $g_1(A_2, C_1) = A_2 \in Q_1$. Similarly, one can find that $\Sigma_s = \phi$ at the state (A_3, C_1) . So one obtains $G_{1s}^1 = (Q_1, \Sigma_1, \delta_1^1, q_{10})$ where δ_1^1 consists of all the transitions of δ_1 with an additional selfloop $(A_2, \alpha_2) \rightarrow A_2$ (Figure 5.10(d)). Then again, one can find $G_{1s}^2 = \mathbf{S}(G_1, G_2^P)$ (Step 4 in Algorithm I). It is found that $G_{1s}^1 = G_{1s}^2$. So let $G_1^1 = G_{1s}^2$ (Step 5 in Algorithm I). Hence by Proposition 5.1, one knows that $L(G_1^1)$ is controllable with respect to $L(G_2^P)$ (Step 3 in Algorithm II).

Then for the pair G_1^1 and G_2 , we define G_1^{1P} such that $L(G_1^{1P}) = p_2(p_1)^{-1}L(G_1^1)$ (Figure 5.10(e)):

$$G_1^{1P} := (Q_1^P, \Sigma_1, \delta_1^{1P}, q_{10}^P),$$

where

$$\begin{aligned} Q_1^P &= \{D_0, D_1\}, \\ \delta_1^{1P} : (D_0, \alpha_1) &\mapsto D_1, & (D_0, \alpha_4) &\mapsto D_0, \\ &(D_1, \alpha_2) &\mapsto D_1, & (D_1, \alpha_4) &\mapsto D_1, \\ q_{20}^P &= D_0. \end{aligned}$$

Then find the product $G_{21}^{1P} = G_2 \times G_1^{1P}$ as follows (Figure 5.10(f)):

$$G_{21}^{1P} = G_2 \times G_1^{1P} = (Q_2 \times Q_1^P, \Sigma_2, \delta_2 \times \delta_1^{1P}, (q_{20}, q_{10}^P)),$$

where

$$\begin{aligned} Q_2 \times Q_1^P &= \{(B_0, D_0), (B_1, D_1), (B_2, D_1), (B_3, D_1)\}, \\ \delta_2 \times \delta_1^{1P} : &((B_0, D_0), \alpha_1) \mapsto (B_1, D_1), & ((B_1, D_1), \alpha_2) &\mapsto (B_1, D_1), \\ &((B_1, D_1), \alpha_4) \mapsto (B_2, D_1), & ((B_2, D_1), \alpha_2) &\mapsto (B_3, D_1), \\ (q_{20}, q_{10}^P) &= (B_0, D_0). \end{aligned}$$

One knows that G_{21}^{1P} refines G_2 and G_1^{1P} . So one has the unique functions

$$h_1 : Q_2 \times Q_1^P \rightarrow Q_1^P \quad \text{and} \quad g_1 : Q_2 \times Q_1^P \rightarrow Q_2,$$

as follows:

$$\begin{array}{ll}
h_1(B_0, D_0) \mapsto D_0, & h_1(B_1, D_1) \mapsto D_1, \\
h_1(B_2, D_1) \mapsto D_1, & h_1(B_3, D_1) \mapsto D_1, \\
g_1(B_0, D_0) \mapsto B_0, & g_1(B_1, D_1) \mapsto B_1, \\
g_1(B_2, D_1) \mapsto B_2, & g_1(B_3, D_1) \mapsto B_3.
\end{array}$$

Then, by the selfloop algorithm, $\mathbf{G}_{2s}^1 = \mathbf{S}(\mathbf{G}_2, \mathbf{G}_1^{1P})$ is computed (Step 2 in Algorithm I). One can find that for the state (B_3, D_1) , $\Sigma_s = \{\alpha_2\}$. So one needs to selfloop α_2 at the state $g_1(B_3, D_1) = B_3 \in Q_2$. So $\mathbf{G}_{2s}^1 = (Q_2, \Sigma_2, \delta_2^1, q_{20})$ is obtained where δ_2^1 consists of all the transitions of δ_2 with an additional selfloop $(B_3, \alpha_2) \rightarrow B_3$. Again one obtains $\mathbf{G}_{2s}^2 = \mathbf{S}(\mathbf{G}_2, \mathbf{G}_1^{1P})$ (Step 4 in Algorithm I). It is found that $\mathbf{G}_{2s}^2 = \mathbf{G}_{2s}^1$. So, let $\mathbf{G}_2^1 = \mathbf{G}_{2s}^2$ (Step 5 in Algorithm I). One knows that $L(\mathbf{G}_2^1)$ is controllable with respect to $L(\mathbf{G}_1^{1P})$. Since $\mathbf{G}_2^1 \neq \mathbf{G}_2$, one needs to repeat Algorithm II for a pair \mathbf{G}_1^1 and \mathbf{G}_2^1 (Step 6 in Algorithm II). Then one obtains \mathbf{G}_1^2 (Step 7 in Algorithm II). It is found that $\mathbf{G}_1^2 = \mathbf{G}_1^1$. Therefore, the resultant DESs are given by $\mathbf{G}'_1 = \mathbf{G}_1^2$ and $\mathbf{G}'_2 = \mathbf{G}_2^1$ (Step 8 in Algorithm II). The closed behaviours of the resultant DESs \mathbf{G}'_1 and \mathbf{G}'_2 are given in Figure 5.10(g). Clearly, one can see that the languages $L(\mathbf{G}'_1)$ and $L(\mathbf{G}'_2)$ are mutually controllable.

5.5 Conclusions

In this chapter, for the systems which do not satisfy the conditions in Theorem 3.1, we have developed a method to modify the systems so that the resultant systems will possess the desired structural properties. With the arranged structure, one can carry out decentralised synthesis and control. For the shared-event-marking condition, we have shown a systematic method in which all the necessary states are marked. For the mutual controllability condition, we firstly introduced selfloop operation and then we have presented an algorithm to add selfloops of shared uncontrollable events so that the resultant systems are pairwise mutually controllable.

The procedure is rather complicated. However, this procedure is applied to the structure of the systems and so it needs to be done only once. Then for any family of $L_{i,m}$ -closed specification languages, one can use decentralised control without further checking.

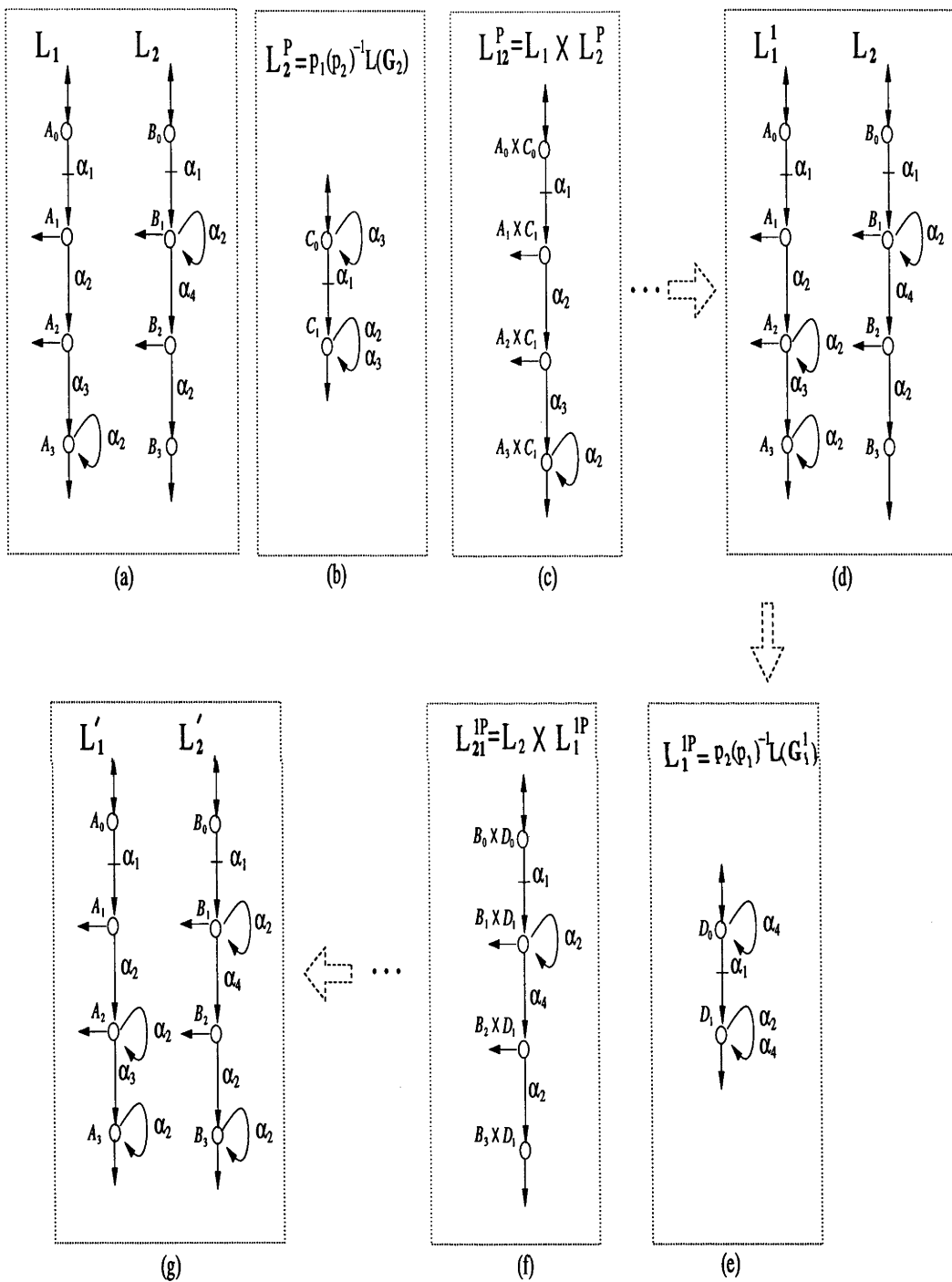


Figure 5.10: Example: arrange the system for the mutual controllability condition

Chapter 6

Weakening the Shared-Event-Marking Condition

In this chapter, we describe two ways to weaken the shared-event-marking condition. In Section 6.1, some basic concepts used in this chapter are introduced. In Section 6.2, we formulate a problem. In Section 6.3, bisimulation is used for weakening the condition. In Section 6.4 we explain how observer properties are applied to weaken the condition. An example is presented for illustration in Section 6.5. This chapter ends with some concluding remarks in Section 6.6

6.1 Preliminary

In Chapter 3, we have presented two structural sufficient conditions, the shared-event-marking condition and the mutual controllability condition, for a decentralised control of concurrent DES. In some applications, however, we have found that too many of the states for given systems need to be marked for the shared-event-marking condition to be satisfied. Hence, it is desirable if we can relax this condition. In this chapter, we investigate this idea using two different but closely related concepts. We consider a global system that is the synchronous composition of two subsystems. We obtain sufficient conditions for given local, not necessary prefix-closed, specification languages. However, the conditions now become specification-

dependent conditions. So, like conditions obtained in [LW88a], if a specification is changed, our conditions are required to be verified again for the changed specification. However, still unlike in [LW88a], we allow non-prefix-closed specification languages and hence the problem of blocking will be addressed.

Firstly, we recall the following concept of bisimulation [Arn94, Fer89, Mil80]. For two transition systems, the concept of bisimulation is one way to describe an equivalence relation between them. Let Σ_i be the event alphabet of a subplant G_i , for $i = 1, 2$. It is allowed that $\Sigma_i \cap \Sigma_j \neq \emptyset$ for $i \neq j$. Let $\Sigma := \Sigma_1 \cup \Sigma_2$ and $\Sigma_s := \Sigma_1 \cap \Sigma_2$. Let p_s be the natural projections from Σ^* to Σ_s^* . Let $H_i \subseteq \Sigma_i^*$ for $i = 1, 2$. Define a binary relation $\equiv_{H_1, H_2} \subseteq \overline{H_1} \times \overline{H_2}$ as follows:

Definition 6.1 For $s_1 \in \overline{H_1}$ and $s_2 \in \overline{H_2}$, we say that $s_1 \equiv_{H_1, H_2} s_2$ if and only if $p_s(s_1) = p_s(s_2)$. \square

Now we recall a concept of *bisimulation* [Arn94, Mil80].

Definition 6.2 Let $\mathcal{S} \subseteq \overline{H_1} \times \overline{H_2}$. We say that \mathcal{S} is a bisimulation on H_1 and H_2 with respect to $\Sigma_1 \cap \Sigma_2$, if for $(s_1, s_2) \in \mathcal{S}$,

i) $(\forall \sigma_1 \in \Sigma_1) s_1 \sigma_1 \in \overline{H_1}$ implies $(\exists u \in \Sigma_2^*) p_s(u) = p_s(\sigma_1), s_2 u \in \overline{H_2}$
and $(s_1 \sigma_1, s_2 u) \in \mathcal{S}$.

ii) $(\forall \sigma_2 \in \Sigma_2) s_2 \sigma_2 \in \overline{H_2}$ implies $(\exists v \in \Sigma_1^*) p_s(v) = p_s(\sigma_2), s_1 v \in \overline{H_1}$
and $(s_1 v, s_2 \sigma_2) \in \mathcal{S}$.

iii) $s_1 \in H_1$ implies $(\exists u \in \Sigma_2^*) p_s(u) = \epsilon$ and $s_2 u \in H_2$ and $(s_1, s_2 u) \in \mathcal{S}$.

iv) $s_2 \in H_2$ implies $(\exists v \in \Sigma_1^*) p_s(v) = \epsilon$ and $s_1 v \in H_1$ and $(s_1 v, s_2) \in \mathcal{S}$. \square

We say that H_1 and H_2 are *bisimilar* with respect to $\Sigma_1 \cap \Sigma_2$ if \equiv_{H_1, H_2} is a bisimulation. Intuitively, two languages are bisimilar with respect to their shared events if a shared event is possible at some point in one language, that shared event should also be possible in the other language and vice versa; and if a marked string is reached in one language, then from a corresponding string in the other language, a marked string is also reachable.

Now we recall some concepts of observers in the supervisory control of DES framework.

For detail, refer [WW96a, WTHM95, ZW90]. Let $L \subseteq \Sigma^*$ be a prefix-closed language, representing the behaviour of a system G . Let T be another set of event labels.

Definition 6.3 A map $\theta : L \rightarrow T^*$ is a causal reporter map [ZW90] if

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

$$\theta(s\sigma) = \begin{cases} \theta(s)\tau & \text{for some } \tau \in T, \\ \theta(s) & \text{otherwise,} \end{cases}$$

for $s \in L$ and $\sigma \in \Sigma$.

The map θ preserves the history of an event sequence such that if $s \leq u$, then $\theta(s) \leq \theta(u)$. A causal reporter map can be characterised as follows [WW96a]:

$$\theta \text{ is a causal reporter map iff for all } H \subseteq L, \theta(\overline{H}) = \overline{\theta(H)}.$$

Let $M := \theta(L)$. An *observer* property of θ is introduced in [WW96a] and can be characterised as follows:

$$\theta \text{ is an observer} \iff (\forall K \subseteq M) \theta^{-1}(\overline{K}) = \overline{\theta^{-1}(K)}.$$

Let $H \subseteq L$ and $N := \theta(H)$.

Definition 6.4 We say that θ is an H-observer if for all $N' \subseteq N$,

$$\overline{\theta_H^{-1}(N')} = \theta^{-1}(\overline{N'}),$$

where, θ_H denotes the restriction of θ to H , i.e., $\theta_H = \theta|_H$.

If $H = L$, then θ is an observer. We recall some properties of H-observer [WTHM95]. Let θ be a causal reporter map $\theta : L \rightarrow T^*$ and $H \subseteq L$. Then

1. θ is an H-observer

$$\iff (\forall s \in L) (\forall t \in T^*) \theta(s)t \in \theta(H) \implies (\exists u \in \Sigma^*) su \in H \text{ and } \theta(su) = \theta(s)t. \quad (6.1)$$

2. If θ is an H-observer, then $\theta^{-1}\theta(\overline{H}) = \overline{H}$. If $\overline{H} = L$, then θ is an observer. (6.2)

6.2 Problem Formulation

Let Σ_i be the event alphabet of a subplant G_i , for $i = 1, 2$. It is allowed that $\Sigma_1 \cap \Sigma_2 \neq \emptyset$. Assume that $\Sigma_i = \Sigma_{ic} \dot{\cup} \Sigma_{iu}$. We assume that the control status of shared events are the same,

i.e.,

$$\Sigma_{1u} \cap \Sigma_2 = \Sigma_2 \cap \Sigma_{1u}.$$

Let $\Sigma := \Sigma_1 \cup \Sigma_2$ be the event set of the global system, say G . Let $\Sigma_c := \Sigma_{1c} \cap \Sigma_{2c}$, $\Sigma_u := \Sigma_{1u} \cap \Sigma_{2u}$ and $\Sigma_s := \Sigma_1 \cap \Sigma_2$. Let p_i and p_s be the natural projections from Σ^* to Σ_i^* and to Σ_s^* , respectively. Let $L_{i,m}, L_i \subseteq \Sigma_i^*$ represent respectively the marked and closed behaviours of system G_i . We assume that $L_i = \overline{L_{i,m}}$. The marked and closed behaviours of the system G are respectively

$$L_m = L_{1,m} || L_{2,m}, \text{ and } L = L_1 || L_2.$$

Now let $E_i \subseteq L_{i,m}$ be the local specification language on the subsystem G_i , not necessarily prefix-closed. The overall specification is then

$$E := (p_1|_L)^{-1}(E_1) \cap (p_2|_L)^{-1}(E_2).$$

For a decentralised control approach, we synthesise local supervisors on G_i whose closed-loop marked and closed behaviours are respectively $\kappa_{L_i}(E_i)$ and $\overline{\kappa_{L_i}(E_i)}$. The closed behaviour under the concurrent supervisions of the local supervisors is then

$$(p_1|_L)^{-1}(\overline{\kappa_{L_1}(E_1)}) \cap (p_2|_L)^{-1}(\overline{\kappa_{L_2}(E_2)}).$$

For a centralised control approach, we can synthesise for the global specification E and obtain $\overline{\kappa_L(E)}$ as the closed behaviour of a global supervisor.

Now we establish the following problem.

Problem 6.1 For given $E_i \subseteq L_{i,m}$, where $i = 1, 2$, under what condition is it true that

$$(p_1|_L)^{-1}(\overline{\kappa_{L_1}(E_1)}) \cap (p_2|_L)^{-1}(\overline{\kappa_{L_2}(E_2)}) = \overline{\kappa_L(E)}, \quad (6.3)$$

and

$$p_i(\overline{\kappa_L(E)}) \text{ is nonblocking with respect to } L_{i,m} ? \quad (6.4)$$

◇

Note that this problem is different from Problem 3.1, in that the conditions that we seek depend on the given specifications. We have developed two ways to solve this problem.

6.3 Using Bisimulation

The following theorem provides sufficient conditions for Problem 6.1.

Theorem 6.1 *Let $\Sigma_i, \Sigma_{ic}, \Sigma_{iu}, L_{i,m}$ and L_i , for $i = 1, 2$, be given as in Section 6.2. Suppose that L_1 and L_2 are mutually controllable. Let $E_i \subseteq L_{i,m}$. Suppose further that*

- i) $\kappa_{L_1}(E_1)$ and $\kappa_{L_2}(E_2)$ are bisimilar with respect to Σ_s ,
- ii) $\kappa_{L_i}(E_i)$ and $L_{j,m}$ are bisimilar with respect to Σ_s , for $i \neq j$.

Then Eq. 6.3 and 6.4 hold.

Note that the mutual controllability condition remains a structural condition. For given specifications, now one needs to check whether local supervisors are bisimilar with each other and the marked behaviour of the other plant.

To prove Theorem 6.1, one requires the following lemmas.

Lemma 6.1 *Let $H_i \subseteq \Sigma_i^*$ for $i = 1, 2$. Suppose that H_1 and H_2 are bisimilar with respect to $\Sigma_1 \cap \Sigma_2$. Then*

$p_1^{-1}(H_1)$ and $p_2^{-1}(H_2)$ are nonconflicting.

Furthermore,

$p_i(H)$ is nonblocking with respect to H_i ,

where $H := p_1^{-1}(\overline{H_1}) \cap p_2^{-1}(\overline{H_2})$.

Proof: We will show only one inclusion since the other inclusion is always true. Let $s \in \overline{p_1^{-1}(H_1)} \cap \overline{p_2^{-1}(H_2)}$. Then $p_1(s) \in \overline{H_1}$ and $p_2(s) \in \overline{H_2}$. Let $s_1 := p_1(s)$ and $s_2 := p_2(s)$.

Then one has that

$$p_s(s_1) = p_s p_1(s) = p_s(s) \text{ and } p_s(s_2) = p_s p_2(s) = p_s(s).$$

So, $p_s(s_1) = p_s(s_2)$. Therefore by definition, one has that

$$s_1 \equiv_{H_1, H_2} s_2.$$

Since we assume that H_1 and H_2 are bisimilar, $s_1 \equiv_{H_1, H_2} s_2$ is a bisimulation. Also, since $s_1 \in \overline{H_1}$, there is a string $u \in \Sigma_1^*$ such that

$$s_1 u \in H_1.$$

If $u = \epsilon$, since $s_1 \equiv_{H_1, H_2} s_2$ there is a string $t \in \Sigma_2^*$ such that $p_s(t) = \epsilon$, $s_2 t \in H_2$ and $(s_1, s_2 t) \in \equiv_{H_1, H_2}$. Thus for a string $st \in \Sigma^*$,

$$p_1(st) = p_1(s)\epsilon = s_1 u \in H_1 \text{ and } p_2(st) = p_2(s)t = s_2 t \in H_2.$$

Hence $st \in p_1^{-1}(H_1) \cap p_2^{-1}(H_2)$. Therefore,

$$s \in \overline{p_1^{-1}(H_1) \cap p_2^{-1}(H_2)}.$$

Otherwise, write $u = \sigma_1 \sigma_2 \cdots \sigma_n$, where $\sigma_1, \sigma_2, \dots, \sigma_n \in \Sigma_1$ for some n . Since $s_1 \equiv_{H_1, H_2} s_2$ and $s_1 \sigma_1 \in \overline{H_1}$, there is a string $v_1 \in \Sigma_2^*$ such that

$$p_s(v_1) = p_s(\sigma_1), s_2 v_1 \in \overline{H_2}, \text{ and } (s_1 \sigma_1, s_2 v_1) \in \equiv_{H_1, H_2}.$$

Repeating this for $\sigma_2, \sigma_3, \dots, \sigma_n$, one has $v_2, v_3, \dots, v_n \in \Sigma_2^*$ such that

$$p_s(v_i) = p_s(\sigma_i), s_2 v_1 v_2 \cdots v_i \in \overline{H_2}, \text{ and } (s_1 \sigma_1 \sigma_2 \cdots \sigma_i, s_2 v_1 v_2 \cdots v_i) \in \equiv_{H_1, H_2}, \quad (6.5)$$

for $i = 2, 3, \dots, n$. Let $v := v_1 v_2 \cdots v_n \in \Sigma_2^*$. Since $s_1 u \in H_1$ and $(s_1 u, s_2 v) \in \equiv_{H_1, H_2}$, there is a string $t \in \Sigma_2^*$ such that

$$p_s(t) = \epsilon, s_2 vt \in H_2 \text{ and } (s_1 u, s_2 vt) \in \equiv_{H_1, H_2}.$$

Thus

$$s_1 u = p_1(s)u \in H_1 \text{ and } s_2 vt = p_2(s)vt \in H_2,$$

where $u \in \Sigma_1^*$, $v \in \Sigma_2^*$ and $t \in (\Sigma_2 - \Sigma_1)^*$. For $\sigma_i \in \Sigma_1 \cap \Sigma_2$, write $v_i = w_i \sigma_i r_i$, where $w_i, r_i \in (\Sigma_2 - \Sigma_1)^*$. Note that if $\sigma_i \notin \Sigma_1 \cap \Sigma_2$, i.e., $\sigma_i \in \Sigma_1 - \Sigma_2$, then $p_s(\sigma_i) = \epsilon$. So, by Eq. 6.5, one has that $p_s(v_i) = \epsilon$, i.e., $v_i \in (\Sigma_2 - \Sigma_1)^*$. Let

$$I_{\Sigma_1 \cap \Sigma_2} := \{i \mid 1 \leq i \leq n, \sigma_i \in \Sigma_1 \cap \Sigma_2\}.$$

That is, if $i \in I_{\Sigma_1 \cap \Sigma_2}$, then $v_i = w_i \sigma_i r_i$, $w_i, r_i \in (\Sigma_2 - \Sigma_1)^*$ and $\sigma_i \in \Sigma_1 \cap \Sigma_2$, and if $i \notin I_{\Sigma_1 \cap \Sigma_2}$, then $v_i \in (\Sigma_2 - \Sigma_1)^*$. Write $I_{\Sigma_1 \cap \Sigma_2} = \{i_1, i_2, \dots, i_m\}$. Let

$$x := \sigma_1 v_1 \cdots \sigma_{i_1-1} v_{i_1-1} v_{i_1} \sigma_{i_1+1} v_{i_1+1} \cdots \sigma_{i_2-1} v_{i_2-1} v_{i_2} \sigma_{i_2+1} v_{i_2+1} \cdots \\ \cdots \sigma_{i_m-1} v_{i_m-1} v_{i_m} \sigma_{i_m+1} v_{i_m+1} \cdots \sigma_n v_n t.$$

Thus

$$\begin{aligned} p_1(sx) &= p_1(s)p_1(x) \\ &= s_1 \sigma_1 \cdots \sigma_{i_1-1} \sigma_{i_1} \sigma_{i_1+1} \cdots \sigma_{i_2-1} \sigma_{i_2} \sigma_{i_2+1} \cdots \sigma_{i_m-1} \sigma_{i_m} \sigma_{i_m+1} \cdots \sigma_n \\ &= s_1 u \in H_1. \end{aligned}$$

Also,

$$\begin{aligned} p_2(sx) &= p_2(s)p_2(x) \\ &= s_2(s)v_1 \cdots v_{i_1-1} v_{i_1} v_{i_1+1} \cdots v_{i_2-1} v_{i_2} v_{i_2+1} \cdots v_{i_m-1} v_{i_m} v_{i_m+1} \cdots v_n t \\ &= s_2 vt \in H_2. \end{aligned}$$

Hence $sx \in p_1^{-1}(H_1) \cap p_2^{-1}(H_2)$. Therefore,

$$s \in \overline{p_1^{-1}(H_1) \cap p_2^{-1}(H_2)}.$$

Using a similar construction as the above, one can show that $p_i(H)$ is nonblocking with respect to H_i . \square

Lemma 6.2 *Let $\Sigma_1, \Sigma_{ic}, \Sigma_{iu}, L_{i,m}, L_i, E_i$ and $\kappa_{L_i}(E_i)$, for $i = 1, 2$, be given as in Section 6.2. Suppose that the conditions given in Theorem 6.1 hold. Then*

$$\overline{(p_i|_L)^{-1}(\kappa_{L_i}(E_i))} = (p_i|_L)^{-1}(\overline{\kappa_{L_i}(E_i)}).$$

Proof: This can be shown using Lemma 6.1 as in Lemma 3.4 in Section 3.3. \square

Lemma 6.3 *For given $E_i \subseteq L_{i,m}$, for $i = 1, 2$, suppose that the conditions given in Theorem 6.1 hold. Then*

$$(p_i|_L)^{-1}(\kappa_{L_i}(E_i)) \in \mathcal{C}(L),$$

where $\mathcal{C}(L)$ is the set of controllable sublanguages of L .

Proof: The proof can proceed similarly as in Lemma 3.6 in Section 3.3. \square

Note that the preservation of controllability from the global system to the local system (cf. (ii) in Lemma 3.6) is due entirely to mutual controllability of the subsystems. Therefore it remains valid in here.

Lemma 6.4 *For given $E_i \subseteq L_{i,m}$, for $i = 1, 2$, suppose that the conditions given in Theorem 6.1 hold. Then*

$$(p_i|_L)^{-1}\kappa_{L_i}(E_i) = \kappa_L(p_i|_L)^{-1}(E_i).$$

Proof: The proof can proceed similarly as in Lemma 3.7 in Section 3.3. \square

Now one can prove Theorem 6.1.

Proof of Theorem 6.1: The proof is similar to that of Theorem 3.1 in Section 3.3. \square

6.4 Using H-Observers

In this section, we present sufficient conditions for Problem 6.1 using the concepts of H -observer. Under the same assumption as in Section 6.2. Then we have the following.

Theorem 6.2 *Under the foregoing assumption, suppose that L_1 and L_2 are mutually controllable. Let $E_i \subseteq L_{i,m}$ and let $K_i := \kappa_{L_i}(E_i)$. Suppose further that*

1. $p_s^1(K_1)$ and $p_s^2(K_2)$ are nonconflicting, and
 $p_s^i(K_i)$ and $p_s^j(L_{j,m})$ are nonconflicting, for $i \neq j$.
2. $p_s^i|_{\overline{K_i}}$ is a K_i -observer and $p_s^i|_{L_i}$ is an $L_{i,m}$ -observer, for $i = 1, 2$.

Then Eq. 6.3 and 6.4 hold. □

Remark 6.1 The first condition is concerned with nonconflictingness between the shared-event images of the two local supervisors and between those of a supervisor and the marked behaviour of the other subsystem. The second condition basically states that the shared-event image is a sufficiently ‘accurate’ model of the subsystem. Namely, if an event is possible in a state in the shared-event image, then it should be possible in any of the corresponding states in the subsystem. ◇

To prove Theorem 6.2, one needs the following lemmas.

Lemma 6.5 *Suppose that $H_i \subseteq L_{i,m} \subseteq \Sigma_i^*$ for $i = 1, 2$,*

1. $p_s^1(H_1)$ and $p_s^2(H_2)$ are nonconflicting,
2. $p_s^i|_{\overline{H_i}}$ is H_i -observer.

Then, $p_1^{-1}(H_1)$ and $p_2^{-1}(H_2)$ are nonconflicting. Furthermore,

$p_i(H)$ is nonblocking with respect to H_i ,

where $H := p_1^{-1}(\overline{H_1}) \cap p_2^{-1}(\overline{H_2})$.

Proof: We will show only one inclusion since the other inclusion is always true. Let $s \in \overline{p_1^{-1}(H_1)} \cap \overline{p_2^{-1}(H_2)}$. So there exist strings $u', v' \in \Sigma^*$ such that $su' \in p_1^{-1}(H_1)$ and

$sv' \in p_2^{-1}(H_2)$. Thus $p_1(sv') \in H_1$ and $p_2(sv') \in H_2$. So for some strings $u'' \in \Sigma_1^*$ and $v'' \in \Sigma_2^*$, $p_1(s)u'' \in H_1 \subseteq \overline{H_1}$ and $p_2(s)v'' \in H_2 \subseteq \overline{H_2}$. Hence

$$p_1(s) \in \overline{H_1} \text{ and } p_2(s) \in \overline{H_2}.$$

Then, one has $p_s^1(p_1(s)) \in p_s^1(\overline{H_1})$ and $p_s^2(p_2(s)) \in p_s^2(\overline{H_2})$. Also, since $p_s^i(\overline{H_i}) = \overline{p_s^i(H_i)}$, one has that

$$p_s^1(p_1(s)) \in p_s^1(\overline{H_1}) = \overline{p_s^1(H_1)} \text{ and } p_s^2(p_2(s)) \in p_s^2(\overline{H_2}) = \overline{p_s^2(H_2)}.$$

Also, we have $p_s^1(p_1(s)) = p_s^2(p_2(s))$. Since it is assumed that $p_s^1(H_1)$ and $p_s^2(H_2)$ are non-conflicting, i.e., $\overline{p_s^1(H_1)} \cap \overline{p_s^2(H_2)} = \overline{p_s^1(H_1) \cap p_s^2(H_2)}$, one has that

$$p_s^1(p_1(s)) = p_s^2(p_2(s)) \in \overline{p_s^1(H_1) \cap p_s^2(H_2)} = \overline{p_s^1(H_1)} \cap \overline{p_s^2(H_2)}.$$

Hence there exists a string $\omega \in (\Sigma_1 \cap \Sigma_2)^*$ such that

$$p_s^1(p_1(s))\omega = p_s^2(p_2(s))\omega \in p_s^1(H_1) \cap p_s^2(H_2).$$

So $p_s^1(p_1(s))\omega \in p_s^1(H_1)$ and $p_s^2(p_2(s))\omega \in p_s^2(H_2)$. Since $p_i(s) \in \overline{H_i}$ and $\omega \in (\Sigma_1 \cap \Sigma_2)^*$, one has that

$$\begin{aligned} p_s^1(p_1(s))\omega &= p_s^1|_{\overline{H_1}}(p_1(s))\omega \in p_s^1|_{\overline{H_1}}(H_1), \\ p_s^2(p_2(s))\omega &= p_s^2|_{\overline{H_2}}(p_2(s))\omega \in p_s^2|_{\overline{H_2}}(H_2). \end{aligned}$$

Then, since we assume that $p_s^i|_{\overline{H_i}}$ is H_i -observer, by Eq. 6.1,

$$\begin{aligned} (\exists u \in \Sigma_1^*) p_1(s)u \in H_1 \text{ and } p_s^1(p_1(s))\omega &= p_s^1(p_1(s))\omega, \\ (\exists v \in \Sigma_2^*) p_2(s)v \in H_2 \text{ and } p_s^2(p_2(s))\omega &= p_s^2(p_2(s))\omega. \end{aligned}$$

So one has $p_s^1(p_1(s))p_s^1(u) = p_s^1(p_1(s))\omega$, i.e., $\omega = p_s^1(u)$. Likewise, one has that $\omega = p_s^2(v)$. Therefore, $p_s^1(u) = p_s^2(v)$. Now we consider two cases, $\omega = \epsilon$ and $\omega \neq \epsilon$.

Firstly, the case of $\omega = \epsilon$ implies that $u \in (\Sigma_1 - \Sigma_2)^*$ and $v \in (\Sigma_2 - \Sigma_1)^*$. Consider a string $suv \in \Sigma^*$. Then

$$p_1(suv) = p_1(s)u \in H_1, \text{ and } p_2(suv) = p_2(s)v \in H_2.$$

So, $suv \in p_1^{-1}(H_1) \cap p_2^{-1}(H_2)$. Therefore,

$$s \in \overline{p_1^{-1}(H_1) \cap p_2^{-1}(H_2)}.$$

For the case of $\omega \neq \epsilon$, let $\omega = p_s^1(u) = p_s^2(v) = \sigma_1\sigma_2 \cdots \sigma_n$. Consider strings u, v as follows:

$$u = u_1\sigma_1u_2\sigma_2 \cdots u_n\sigma_nu_{n+1} \in \Sigma_1^* \text{ and } v = v_1\sigma_1v_2\sigma_2 \cdots v_n\sigma_nv_{n+1} \in \Sigma_2^*,$$

where

$$\begin{aligned} u_1, u_2, \dots, u_{n+1} &\in (\Sigma_1 - \Sigma_2)^*, \quad v_1, v_2, \dots, v_{n+1} \in (\Sigma_2 - \Sigma_1)^*, \text{ and} \\ \sigma_1, \sigma_2, \dots, \sigma_n &\in \Sigma_1 \cap \Sigma_2. \end{aligned}$$

Then consider a string $sx \in \Sigma^*$ such that

$$sx = su_1v_1\sigma_1u_2v_2\sigma_2 \cdots u_nv_n\sigma_nu_{n+1}v_{n+1}.$$

Then

$$p_1(sx) = p_1(s)u_1\sigma_1u_2\sigma_2 \cdots \sigma_nu_{n+1} = p_1(s)u \in H_1,$$

$$p_2(sx) = p_2(s)v_1\sigma_1v_2\sigma_2 \cdots \sigma_nv_{n+1} = p_2(s)v \in H_2.$$

Hence $sx \in p_1^{-1}(H_1) \cap p_2^{-1}(H_2)$. Therefore,

$$s \in \overline{p_1^{-1}(H_1) \cap p_2^{-1}(H_2)}.$$

Using a similar construction as the above, one can show that $p_i(H)$ is nonblocking with respect to H_i . \square

Lemma 6.6 *Let $\Sigma_1, \Sigma_{ic}, \Sigma_{iu}, L_{i,m}, L_i, E_i$ and $\kappa_{L_i}(E_i)$, for $i = 1, 2$, be given as in Section 6.2. Suppose that the conditions given in Theorem 6.2 hold. Then*

$$\overline{(p_i|_L)^{-1}(\kappa_{L_i}(E_i))} = (p_i|_L)^{-1}(\overline{\kappa_{L_i}(E_i)}).$$

Proof: This can be shown using Lemma 6.1 as in Lemma 3.4 in Section 3.3. \square

Lemma 6.7 *For given $E_i \subseteq L_{i,m}$, for $i = 1, 2$, suppose that the conditions given in Theorem 6.2 hold. Then*

$$(p_i|_L)^{-1}(\kappa_{L_i}(E_i)) \in C(L).$$

Proof: The proof can proceed similarly as in Lemma 3.6 in Section 3.3. \square

Lemma 6.8 *For given $E_i \subseteq L_{i,m}$, for $i = 1, 2$, suppose that the conditions given in Theorem 6.2 hold. Then*

$$(p_i|_L)^{-1}\kappa_{L_i}(E_i) = \kappa_L(p_i|_L)^{-1}(E_i).$$

Proof: The proof can proceed similarly as in Lemma 3.7 in Section 3.3. \square

Now one can prove Theorem 6.2.

Proof of Theorem 6.2: The proof is similar to that of Theorem 3.1 in Section 3.3. \square

Remark 6.2 If there is conflicting in the shared-event model, one could use a coordination scheme presented in [WW96c]. This may be another direction of future research. The concepts

of bisimulation and H -observer are closely related as shown in the paper [WW96a]. \diamond

6.5 Example

Consider the first example given in Section 3.4. The DES models for the local plants and local specifications are given in Figures 3.2 and 3.4, respectively. Here, we do not need the marking before the shared event. So the only marker state is the initial state. The local supervisors are obtained and it turns out that in this case, the behaviours of the local supervisors are the same as the specification languages. It can be checked that the conditions in Theorem 6.1 and 6.2 hold.

For a more practical example, consider again a simplified version of the chemical reactor shown in Figure 3.5 and described in Section 3.4. Figure 6.1 is a schematic diagram of the plant. The reactor is comprised of a feed valve V_1 , a drain valve V_2 , a weight measurement unit

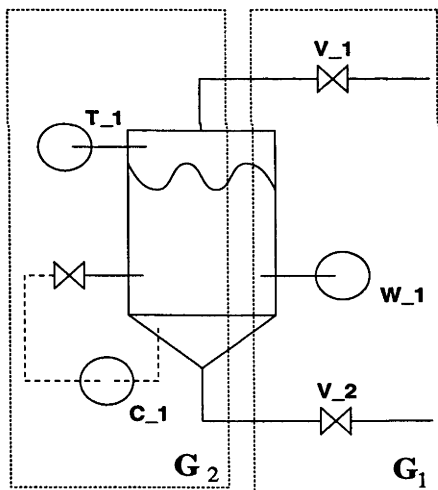


Figure 6.1: Simplified batch reactor

W_1 , a timer T_1 , and a continuous reaction controller C_1 . DES models for the components are the same as presented in Figure 3.6 in Section 3.4 except W_1 (shown in Figure 6.2(a)).

At this time, we divide the whole plant into two subplants, the filling-draining subplant (G_1) and the reaction subplant (G_2). Note that there are no shared components among sub-

plants. Also, like the case in Section 3.4, to enforce the process proceeds in a specified order, we introduce controllable shared events ($\sigma_1, \sigma_2, \sigma_3$) which indicate a completion of a certain process. These controllable synchronised events are represented as ‘flag’s and the DES models are shown in Figure 6.2(b) and (c).

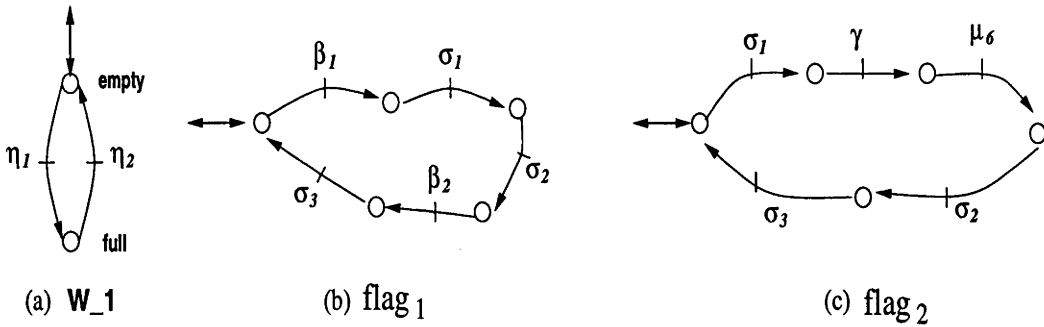


Figure 6.2: DES models for W_1 and synchronisation flags

The process will proceed similarly as described in Section 3.4. Also, the user requirements can be established similarly. The automata displayed in Figure 6.3 represent specifications (E_1 and E_2) for the subplants. It can be checked that $E_i \subseteq L_{i,m}$. The supervisors are obtained and

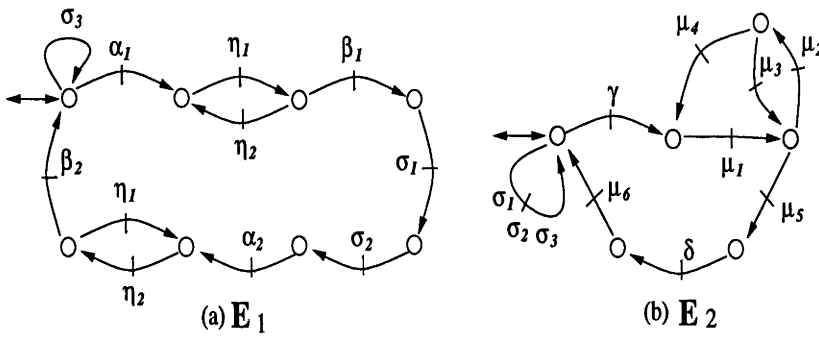


Figure 6.3: DES models for local specifications

their marked behaviours ($\kappa_{L_1}(E_1)$ and $\kappa_{L_2}(E_2)$) are displayed in Figure 6.4. It can be checked that in this case, the conditions in Theorem 6.1 and 6.2 hold.

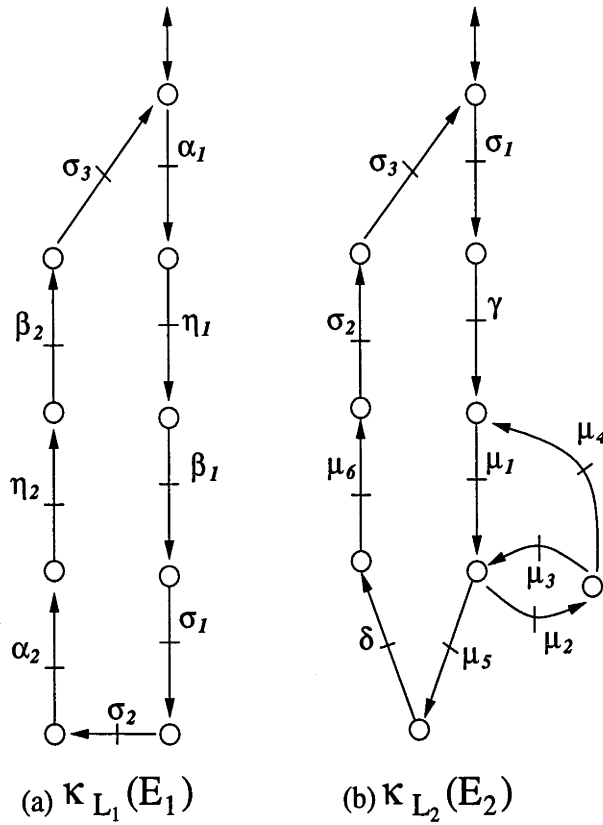


Figure 6.4: Behaviours of the local supervisors

6.6 Conclusions

In this chapter, we have presented two ways to relax the shared-event-marking condition. Unlike the work in Chapter 3, the conditions are specification-dependent. That means now some of our conditions are on-line. So for given specification languages, our conditions are required to be verified. The issue of how to establish these conditions as structural conditions, namely they hold for a set of specifications, is still an open question.

Chapter 7

Application

In this chapter, we describe an application of the result in Chapter 3 to the automation of a cleaning-in-place (CIP) process for a small scale multipurpose, multiproduct batch plant ([LM93, Als96]). Three suboperations of the CIP process, Water Rinse, Detergent Preparation and Detergent Rinse, are considered.

7.1 Introduction

The purpose of the CIP process is to clean the equipments and their associated pipelines using detergent solutions. In many chemical processes, CIP process is important for safety reasons and product qualities. However, in many cases the problem is that the CIP process takes a large portion of the total production time. So the reduction of the CIP operation time is an important issue in the efficient production of the final product. The automation of the CIP process is one way to reduce the operating time.

In the following, we examine a DES control synthesis of the CIP process for a small batch plant. Since the CIP process is typically running in a batch-mode (so it is inherently discrete), the process can be suitably modelled as a DES. Furthermore, the CIP process often requires flexible configurations. For example, the cleaning process for one part of the plant is often different from the cleaning of another part, and the cleaning procedure might need to be changed

depending on the materials being processed in the plant. Therefore, the decentralised approach is more suitable in this case.

The plant employed in this example is a small scale batch process for food and pharmaceutical products. The schematic diagram of the plant is presented in Figure 7.1¹. The main components of the plant are two feed preparation vessels (T_2, T_3), a multipurpose batch reactor (T_4), two product storage vessels (T_5,T_6), and two heat exchangers (HE_1, HE_2). These components are connected with pipelines, valves and pumps. In addition to these main components, the tank T_1 is provided to prepare the hot caustic detergent solution for the CIP process. The flexible operation of the CIP system enables parts of the plant to be cleaned separately. In here, we consider the automatic process of cleaning the feed preparation tank T_2 and its associated pipelines.

The components involved in the CIP operation of T_2 are comprised of pumps (P_1–P_4), valves (DV_1,DV_2,V_1–V_15, V_22), a temperature controller in the heat exchanger HE_1 (C_1), a conductivity level sensor which monitors the caustic level in the detergent solution (CS_1), a sensor for the lid position of the tank T_2 (PO_1), a temperature probe for the tank T_1 (TP_1), a low level sensor for the tank T_1 (LT_1), a high level sensor for the tank T_1 (HT_1), and a continuous level sensor for the tank T_2 (LT_2). The number of the elementary components is 29. So if each component is assumed to have 2 states, the total number of states is more than 5.0×10^8 . This is generally too big to analyse as a whole. This is another reason to use the decentralised control approach.

7.2 Decomposition of the CIP process

The CIP process of the tank T_2 consists of complex sequences of discrete-event driven activities. However, it is possible to divide the whole operation into the following four sequential suboperations.

1. Water Pre-Rinse of T_2: the tank T_2 will be cleaned with high pressured water for 10

¹This diagram is adapted from [Als96] with slight modifications.

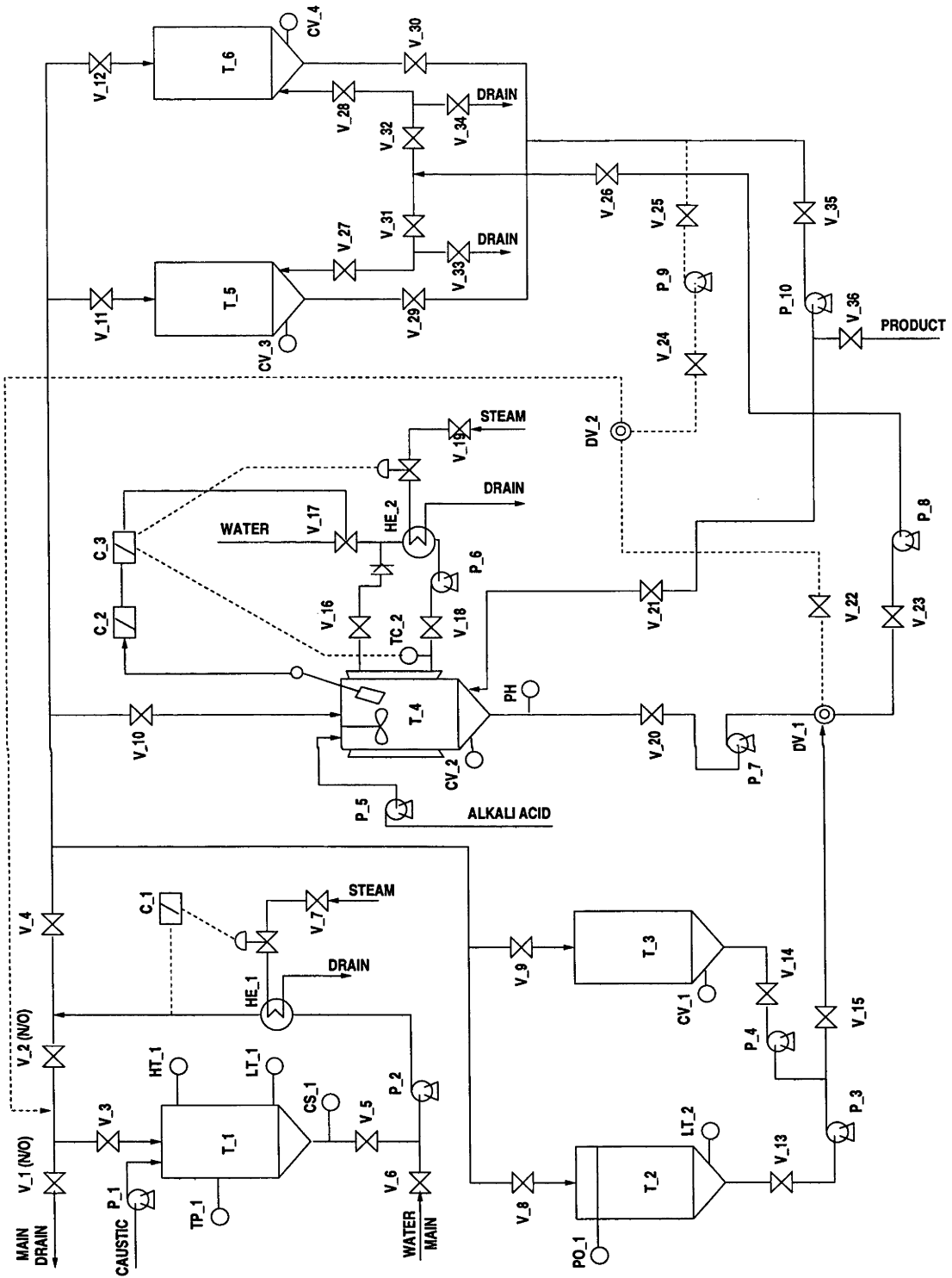


Figure 7.1: Schematic diagram of the batch process

minutes.

2. Detergent Preparation in T.1: the hot caustic detergent solution is prepared in the tank T.1.
3. Detergent Rinse of T.2: the tank T.2 is cleaned with high pressure spray of the detergent solution for 10 minutes.
4. Water Rinse of T.2: Clean T.2 with high pressure water spray to dissolve the residual detergent for 10 minutes. This operation is the same as the first operation, water pre-rinse.

Since the suboperations are operated in sequence, it is possible to assume that each suboperation is operated in isolation from each other, i.e., the supervisor of one operation can be designed independently from the supervisors of the other suboperations. The DES models of the elementary components for the CIP operation of T.2 are presented in Figures 7.2, 7.3 and 7.4. Note that the initial states of the valves V.1 and V.2 are 'open' while those of the other valves are 'closed'. A timer (TS.10) is included to deal with the timing requirements.

The DES model of each suboperation is obtained by the synchronous composition of those elementary components involved in that particular operation. In addition, since these suboperations should be operated in a specific sequence, it is necessary to have an additional DES model to enforce this sequential requirement. To this end, we introduce four controllable shared events, $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ (see Figure 7.5). These shared events indicate the completion of the current suboperation and allow the next suboperation to proceed. For example, the event λ_1 is a shared event of the water pre-rinse operation and the detergent preparation. It represents that the water pre-rinse operation has been completed and the detergent preparation can now proceed. The event λ_4 is an event shared by all suboperations, representing a complete cycle of the CIP operation of the tank T.2. The synchronisation events, $\lambda_1, \lambda_2, \lambda_3, \lambda_4$, are represented as 'flag's in the modelling process and are considered as a part of the plant. In the following sections, we synthesise a supervisor for each suboperation.

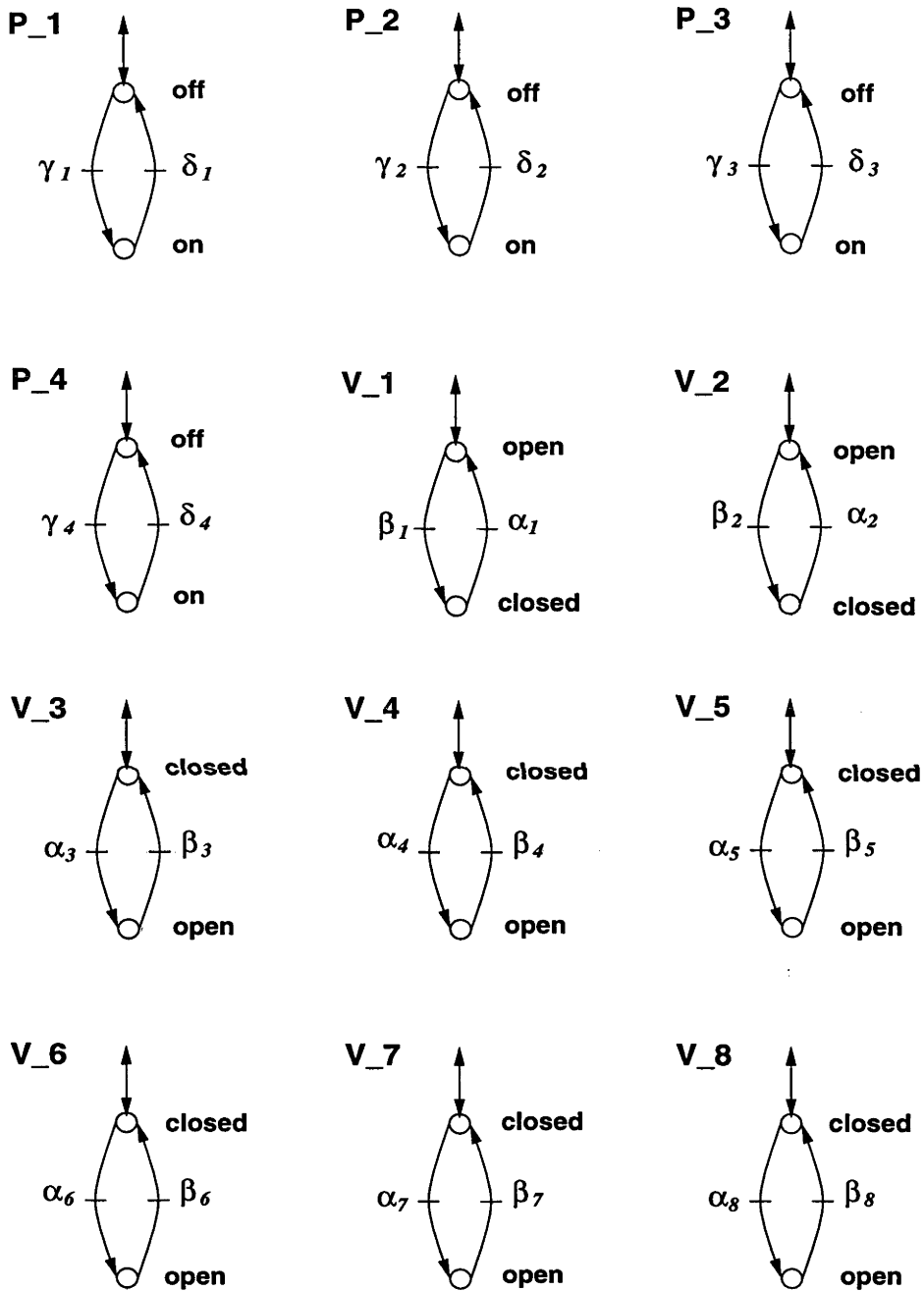


Figure 7.2: DES models for the elementary components of CIP process I

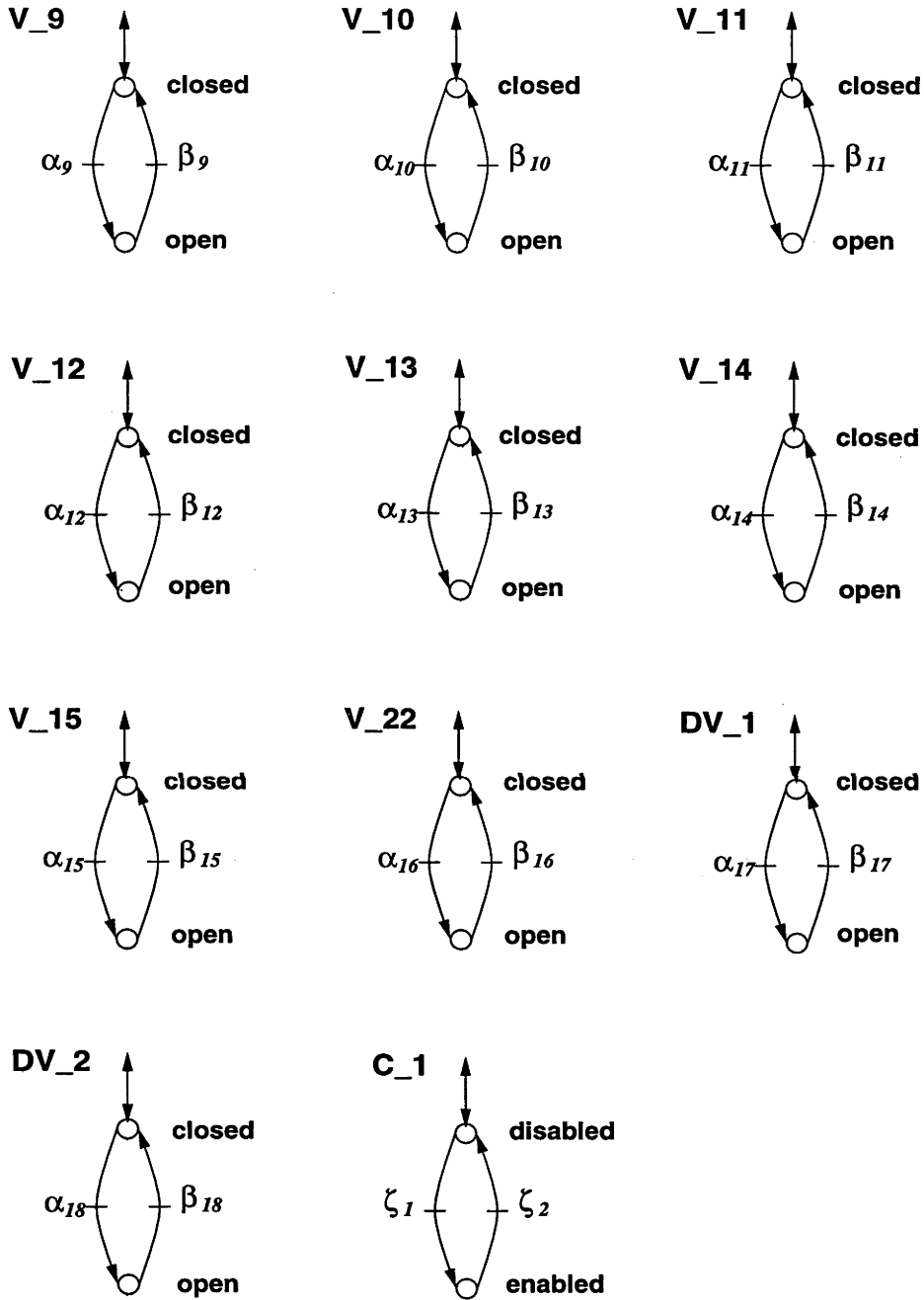


Figure 7.3: DES models for the elementary components of CIP process II

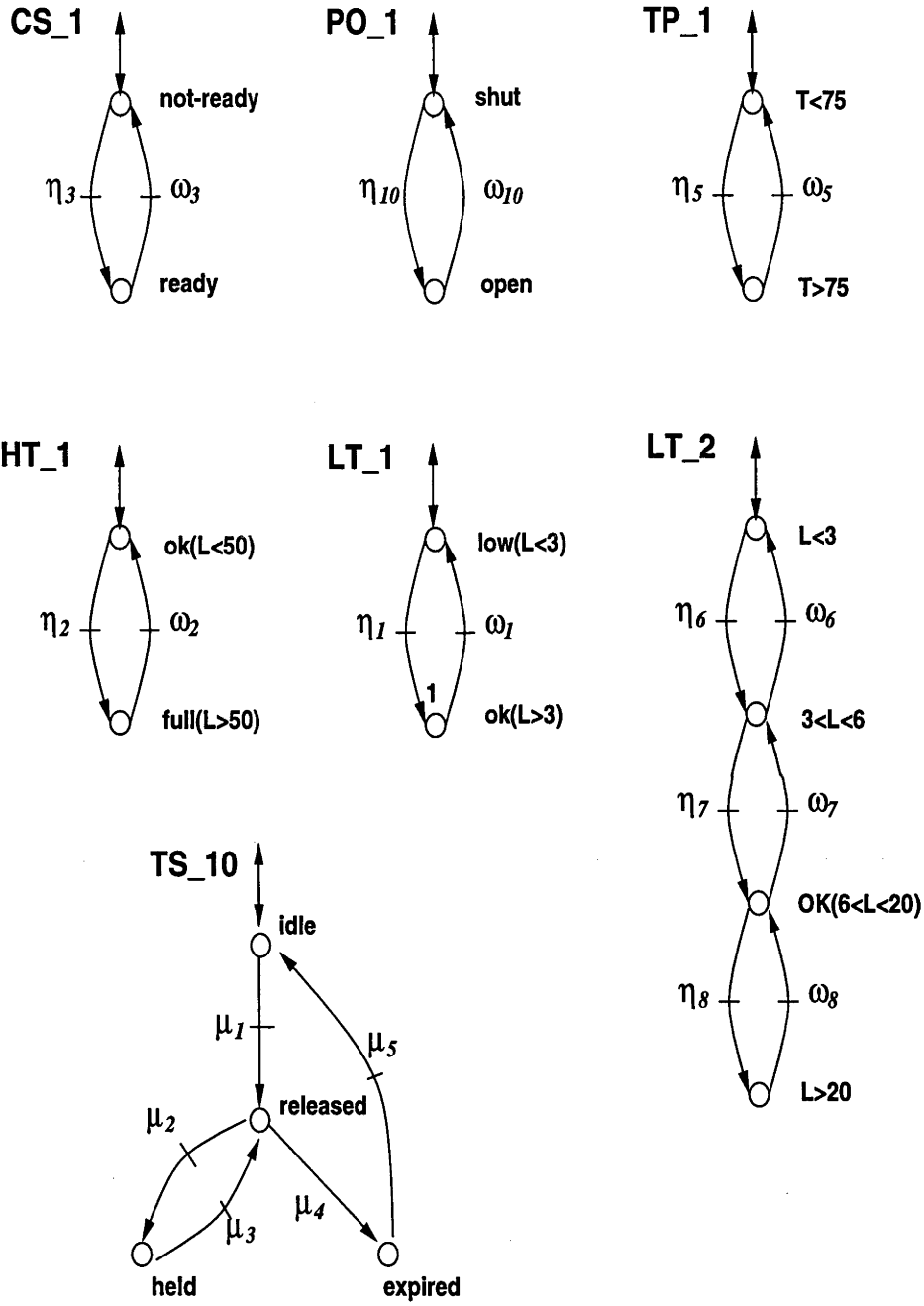


Figure 7.4: DES models for the elementary components of CIP process III

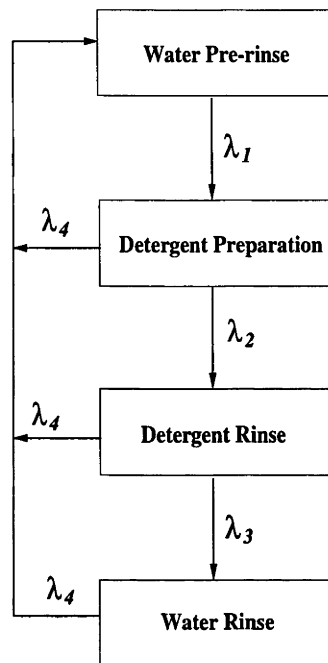


Figure 7.5: Sequential control of the CIP process

7.3 Water-Rinse Operation

In the water-rinse operation, the residual solids in the tank T.2 are removed by intermittent bursts of spraying water at high pressure from pump P.2 for 10 minutes. The water-rinse operation proceeds as follows. See Figure 7.6 for the elementary components involved in this operation. At the beginning of the operation, each elementary component is in its initial state, the tank T.2 is empty and the lid of T.2 is closed. The operation is started with opening the feed route from the water main to T.2 (via V.4, V.6, V.8), and the drain route from T.2 to the main drain (via V.13, V.15, DV.1, V.22, DV.2, V.1). T.2 is cleaned with high-pressure water spray using the pump P.2 for 10 minutes. During the normal operation, the water level inside T.2 should remain between 6L to 20L. If the lid of T.2 is opened at any time during the operation, water feed should be stopped and not be permitted to restart until the lid is shut. When the lid is closed, water feed can be started again only after the water in the tank T.2 is fully drained. Water is drained from T.2 by the pump P.3. P.3 should be stopped if the water level of T.2 is less than 3L. After 10 minutes of the above operations, T.2 is drained until the

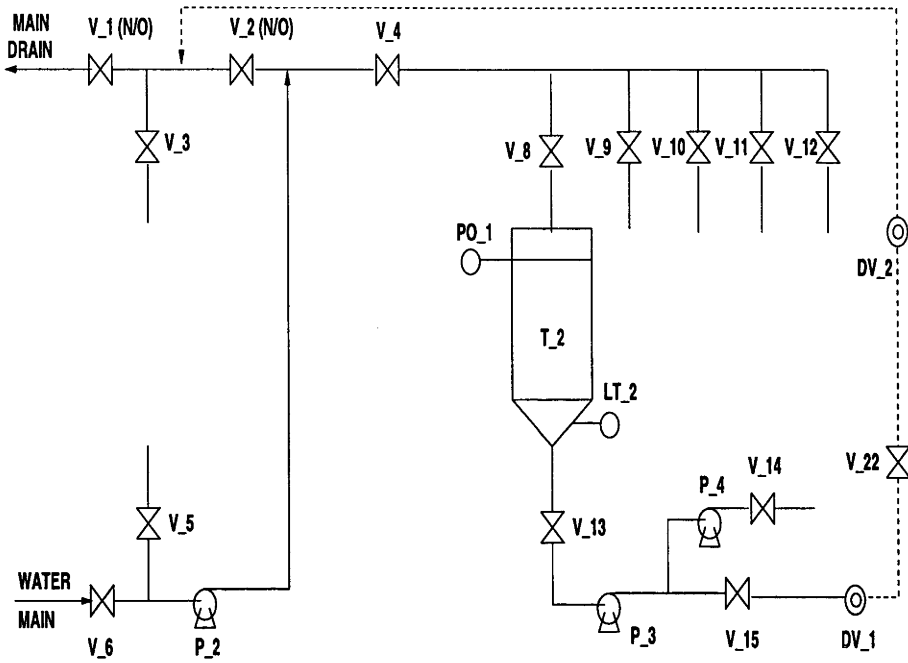


Figure 7.6: Water-Rinse operation

water level is less than 3L. Finally all the components are returned to their initial states.

7.3.1 Decomposition of the Components in the Water-Rinse Operation

The number of components involved in the water-rinse operation is 23. So the size of the whole plant is still too big to analyse. The elementary components are partitioned naturally into two groups: one group of components which actively participate in the water-rinse operation of the tank T_2 (designated as G_{yp}) and the other group which are required to remain at their initial states during the operation (G_{id}). Furthermore, in the plant (G_{yp}), the components involved in the preparation of the feed or drain route (G_{pr}) are separated from those involved in the actual cleaning and draining (G_{pa}). Finally, in subplants G_{pr} , G_{pa} and G_{id} , the components involved in the water feed, from the water main to the tank T_2 (G_{fpr} , G_{fpa} , G_{fid}), are separated from those components involved in the water drain, from T_2 to the main drain (G_{dpr} , G_{dpa} , G_{did}). The partitioning table of the elementary components is presented in Table 7.1. Note that only LT_2 is a shared component between partitioned subsystems, G_{fpa} and G_{dpa} .

G_{wr}	G_{yp}	G_{pr}	G_{fpr}	V_2, V_4, V_6	Preparation water feed
			G_{dpr}	DV_1, DV_2, V_15, V_22	Preparation water drain
		G_{pa}	G_{fpa}	PO_1, LT_2, P_2, V_8, TS_10	Water feeding and cleaning
			G_{dpa}	LT_2, P_3, V_13	Water draining
	G_{id}	G_{fid}	V_5, V_9, V_10, V_11, V_12	Idle in water feed part	
		G_{did}	P_4, V_1, V_3, V_14	Idle in water drain part	

Table 7.1: Partition table for water-rinse operation

The water-rinse operation is also divided naturally into the following three sequential processes like the decomposition of the elementary components:

1. Prepare the feed and drain routes.
2. Clean the tank T_2 for 10 minutes.
3. Return the feed and drain routes into their initial states.

So, it is necessary to enforce that these three processes are operating in sequence. To this end, three controllable shared events ($\sigma_1, \sigma_2, \sigma_3$) are introduced. These shared events play the same role as the shared events between suboperations ($\lambda_1-\lambda_4$). The event σ_1 represents that the feed route is ready, while the event σ_2 represents the drain route is ready. The event σ_3 represents that the water spray to the tank T_2 is completed and now all the components can return to their initial states.

7.3.2 Modelling and Synthesis for G_{pr}

The elementary components of G_{pr} are those involved in the preparation of the water feed and drain routes. The subplant G_{pr} is further decomposed since it is easier for analysis and synthesis (before the decomposition, G_{pr} has 724 states with 1151 transitions). Naturally G_{pr} can be divided into two subplants, G_{fpr} for the elementary components involved in the feed route and G_{dpr} for those components involved in the drain route. Also, since the synchronisations with the other subplants are required, the events $\sigma_1, \sigma_2, \sigma_3$ as well as λ_1, λ_4 are included in the flags (see Figure 7.7). The plant G_{fpr} consists of V_6, V_2, V_4 and flag $_{fpr}$, and the subplant G_{dpr} is comprised of DV_1, DV_2, V_15, V_22 and flag $_{dpr}$.

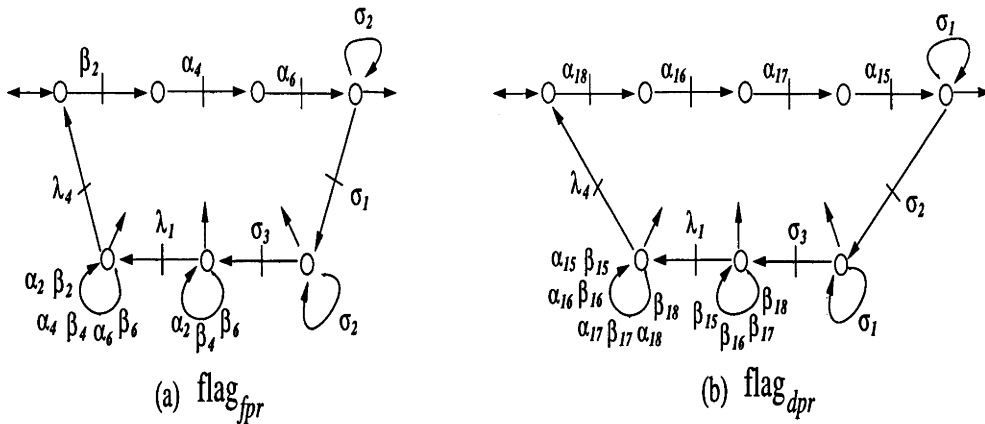


Figure 7.7: Synchronisation flags for G_{pr}

The interpretation of flag_{fpr} , shown in Figure 7.7 (a), is as follows: when the valve V_2 is closed (β_2), and V_4 and V_6 are opened (α_4, α_6), the feed route is ready (σ_1). After high pressured water spray to T_2 has finished (σ_3), all valves are allowed to return to their initial states ($\alpha_2, \beta_4, \beta_6$). Then the water rinse operation is finished (λ_1). The event λ_4 represents a completion of the CIP process of T_2. The DES flag_{dpr} in Figure 7.7 (b) is interpreted similarly. Here after the valves V_15, V_16, V_17, V_18 have been opened ($\alpha_{15}, \alpha_{16}, \alpha_{17}, \alpha_{18}$), send the drain route ready signal (σ_2). After σ_3 has occurred, all the valves are allowed to be closed ($\beta_{15}, \beta_{16}, \beta_{17}, \beta_{18}$). The orders of valve operations are important to avoid an unnecessary waste of water. The selfloops of events after the event λ_1 in both figures are essential for the events to be allowed to occur in the other suboperations (for example, Detergent Rinse and Detergent Preparation).

The DES models for G_{fpr} and G_{dpr} are obtained by synchronous composition and the event sets are

$$\begin{aligned}\Sigma_{fpr} &= \{\alpha_2, \beta_2, \alpha_4, \beta_4, \alpha_6, \beta_6, \sigma_1, \sigma_2, \sigma_3, \lambda_1, \lambda_4\}, \\ \Sigma_{dpr} &= \{\alpha_{15}, \beta_{15}, \alpha_{16}, \beta_{16}, \alpha_{17}, \beta_{17}, \alpha_{18}, \beta_{18}, \sigma_1, \sigma_2, \sigma_3, \lambda_1, \lambda_4\}.\end{aligned}$$

The shared events are $\Sigma_{fpr} \cap \Sigma_{dpr} = \{\sigma_1, \sigma_2, \sigma_3, \lambda_1, \lambda_4\}$ and $\Sigma_{pr} = \Sigma_{fpr} \cup \Sigma_{dpr}$. The size of G_{fpr} is 32 states and 63 transitions while G_{dpr} has 64 states with 147 transitions.

The system requirements for the subplant $G_{fpr}(E_{fpr})$ are:

1. Close the valve V_2 (β_2) and open the valves V_4, V_6 (α_4, α_6). Then send the feed-route-ready signal (σ_1).
2. Keep remaining at this state until the water spraying of T_2 is finished (σ_3).
3. When the water spraying has finished (σ_3), return V_2, V_4, V_6 to their initial states ($\alpha_2, \beta_4, \beta_6$). Then send the signals λ_1 and λ_4 .
4. The selfloops at the state after λ_1 mean that the subplant G_{fpr} does not restrict the behaviours of the other subplants.

Formally, they can be modelled as in Figure 7.8(a). The specifications for G_{dpr} (E_{dpr}) are similar to those for G_{fpr} and its DES model is given in Figure 7.8(b). The supervisors are computed. The supervisor for G_{fpr} has 17 states with 36 transitions, and the supervisor for G_{dpr} has 27 states with 78 transitions.

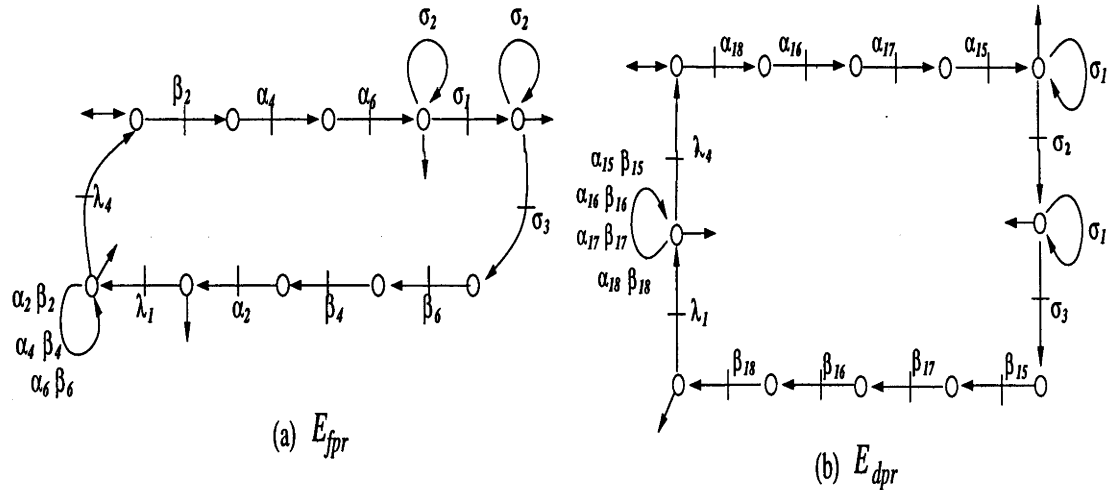


Figure 7.8: Specifications for G_{pr}

7.3.3 Modelling and Synthesis for G_{pa}

The subplant G_{pa} consists of the elementary components which actively participate in the water-rinse operation of the tank T_2. The subplant G_{pa} is still very complex; it has 1344

states with 7616 transitions. Thus, G_{pa} is further decomposed into two subplants; G_{fpa} and G_{dpa} . G_{fpa} consists of the elementary components involved in the water feed while G_{dpa} comprises of those involved in the water drain. The plant G_{fpa} consists of PO_1, LT_2, P_2, V_8, TS_10 and flag_{fpa} which models the behaviour of synchronisation events as shown in Figure 7.9(a). Similarly, the subplant G_{dpa} consists of P_3, V_13, LT_2 and flag_{dpa} (Figure 7.9(b)).

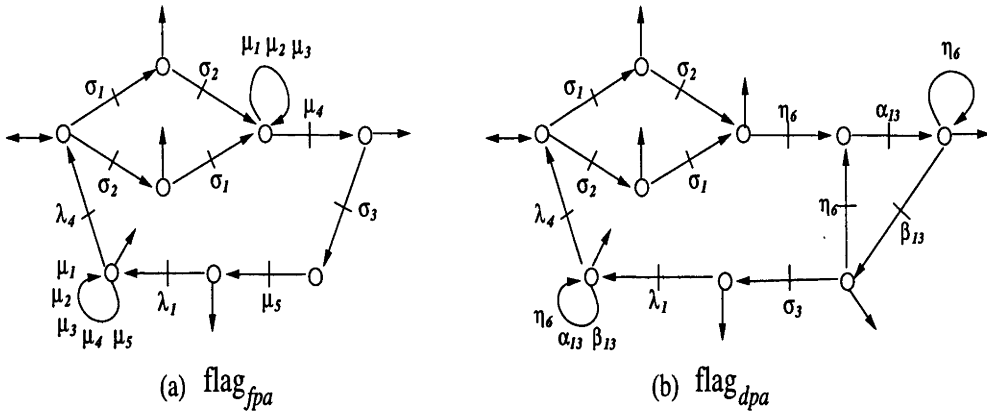


Figure 7.9: Synchronisation flags for G_{pa}

The interpretation of flag_{fpa} , shown in Figure 7.9(a), is as follow: when the feed route is ready (σ_1) and the drain route is ready (σ_2), the timer TS_10 can be started (μ_1). After the timer has expired (μ_4), send a signal σ_3 to the other subplants. Then after the timer has been reset (μ_5), the water rinse operation is completed with the occurrence of the synchronisation event λ_1 and the detergent preparation can now proceed. Finally, one cycle of CIP process of T_2 is finished with the signalling of the event λ_4 . Figure 7.9(b) can be interpreted similarly. In here, after the events σ_1 and σ_2 , once the amount of water in T_2 is more than 3L (η_6), then the valve V_13 is opened (α_{13}). After V_13 is closed (β_{13}), the process is finished with the event σ_3 followed by the events λ_1, λ_4 . Like in the case of G_{pr} , the selfloops of events after the event λ_1 in both figures are necessary to allow the events to occur in the other suboperations.

The synchronous composition of all the components involved generates the possible system behaviours. However, there exist some physical constraints among the elementary components and they will restrict the system behaviour by deleting infeasible states and transitions. The automata shown in Figure 7.10 (a) and (b) represent physical constraints for G_{fpa} , namely that

the level of the tank T_2 can only be increased after the feed pump P_2 is turned on (Figure 7.10(a)) and the feed valve V_8 is opened (Figure 7.10(b)). Figure 7.10 (c) and (d) represent similar physical constraints for the plant G_{dpa} .

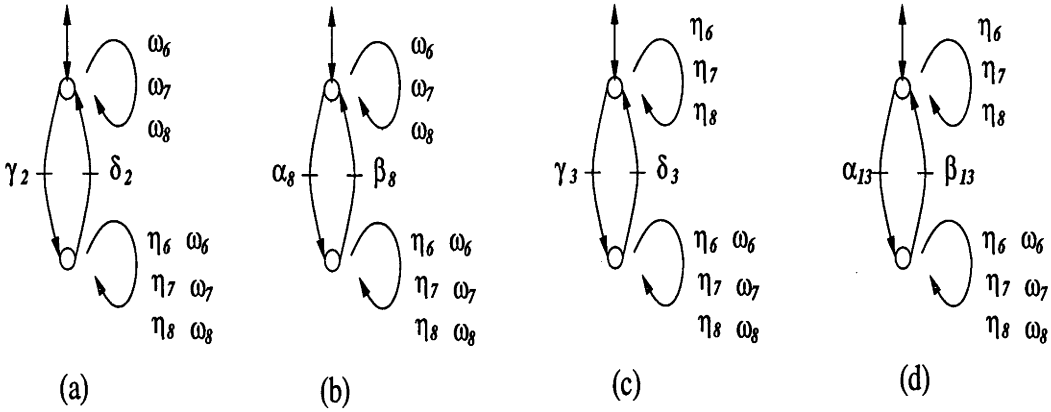


Figure 7.10: Physical constraints of G_{pa}

The synchronous composition of physical constraints with the elementary components gives the DES models of the subplant G_{fpa} and G_{dpa} . The event sets are

$$\Sigma_{fpa} = \{\alpha_8, \beta_8, \gamma_2, \delta_2, \eta_6, \omega_6, \eta_7, \omega_7, \eta_8, \omega_8, \eta_{10}, \omega_{10}, \mu_1, \mu_2, \mu_3, \mu_4, \mu_5, \sigma_1, \sigma_2, \sigma_3, \lambda_1, \lambda_4\}$$

$$\Sigma_{dpa} = \{\gamma_3, \delta_3, \alpha_{13}, \beta_{13}, \eta_6, \omega_6, \eta_7, \omega_7, \eta_8, \omega_8, \sigma_1, \sigma_2, \sigma_3, \lambda_1, \lambda_4\}.$$

The shared events are $\Sigma_{fpa} \cap \Sigma_{dpa} = \{\eta_6, \omega_6, \eta_7, \omega_7, \eta_8, \omega_8, \sigma_1, \sigma_2, \sigma_3, \lambda_1, \lambda_4\}$ and $\Sigma_{pa} = \Sigma_{fpa} \cup \Sigma_{dpa}$. The size of G_{fpa} is 736 states with 3922 transitions while G_{dpa} has 118 states and 337 transitions.

The specifications for G_{dpa} (E_{dpa}) are:

1. Wait until receiving signals that indicate the feed and drain routes are ready (σ_1, σ_2).
2. If the water level in the tank T_2 is more than 3L (η_6), open V_13 (α_{13}) and then turn on P_3 (γ_3).
3. If the water level is decreased to less than 3L (ω_6), turn off P_3 (δ_3) and then close V_13 (β_{13}).

4. The synchronisation event σ_3 is possible only after P_3 is turned off (δ_3) and V_13 is closed (β_{13}).
5. Finish the operation with sending the signals λ_1 and λ_4 .
6. The selfloops at the state after λ_1 mean that the subplant G_{dpa} does not restrict the behaviours of the other subplants.

The DES models for these specifications are given in Figure 7.11. Similarly, the system re-

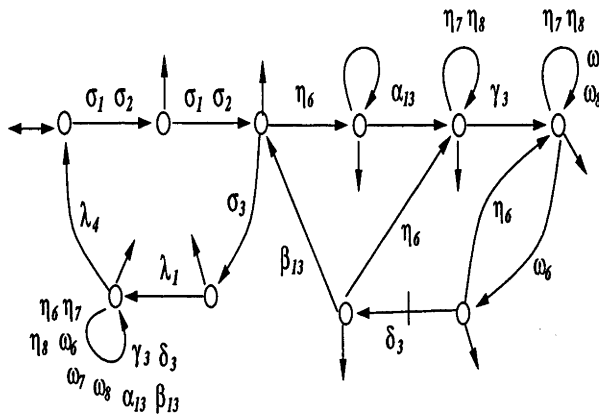


Figure 7.11: Specification for G_{dpa} (E_{dpa})

quirements for the subplant G_{fpa} (E_{fpa}) are:

1. Wait until receiving signals that indicate the feed and drain routes are ready(σ_1, σ_2).
2. If the lid of T_2 is opened (η_{10}), wait until the lid is shut (ω_{10}).
3. Open the valve V_8 first (α_8) and turn on the pump P_2 (γ_2). Then release the timer TS_10 (μ_1).
4. The pump P_2 should be turned off (δ_2) before the valve V_8 is closed (β_8).
5. If the lid of T_2 is opened (η_{10}), hold the timer (μ_2) and turn off P_2 (δ_2). Then close V_8 (β_8).
6. If the lid of T_2 is shut (ω_{10}), drain all water in T_2 (ω_6) and open V_8 (α_8). Then turn on P_2 (γ_2) and re-release the timer (μ_3).

7. After the timer is expired (μ_4), turn off P_2 (δ_2) and then close V_8 (β_8).
8. If the water level in T_2 is increased to more than 20L (η_8), turn off P_2 (δ_2) and then close V_8 (β_8).
9. If the water level is decreased to less than 6L (ω_7), open V_8 (α_8) and turn on P_2 (γ_2).
10. After the timer expired, turn off P_2 (δ_2) and close V_8 (β_8). Then send the signal σ_3 .
11. The timer TS_10 can be reset (μ_5) only after the lid of T_2 is shut (ω_{10}), P_2 is off (δ_2) and the timer is expired (μ_4).
12. Finish the operation with sending the signals λ_1 and λ_4 .
13. The selfloops at the state after λ_1 mean that the subplant G_{fpa} does not restrict the behaviours of the other subplants.

Formally, they can be modelled as in Figure 7.12. Supervisors satisfying these requirements are computed. The sizes of the supervisors are respectively 95 states with 221 transitions for G_{fpa} and 33 states with 75 transitions for G_{dpa} .

7.3.4 Modelling and Synthesis for G_{fid} and G_{did}

The subsystem G_{fid} comprises of the elementary components which are required to remain open or closed in the feed route during the water-rinse operation, while G_{did} consists of those in the drain route. The flags representing synchronisation events are presented in Figure 7.13. The size of G_{fid} is 64 states with 384 transitions, while G_{did} has 32 states with 160 transitions. The event sets of G_{fid} and G_{did} are

$$\begin{aligned}\Sigma_{fid} &= \{\alpha_5, \beta_5, \alpha_9, \beta_9, \alpha_{10}, \beta_{10}, \alpha_{11}, \beta_{11}, \alpha_{12}, \beta_{12}, \lambda_1, \lambda_4\}, \\ \Sigma_{did} &= \{\gamma_4, \delta_4, \alpha_1, \beta_1, \alpha_3, \beta_3, \alpha_{14}, \beta_{14}, \lambda_1, \lambda_4\}.\end{aligned}$$

The common events of Σ_{fid} and Σ_{did} are λ_1, λ_4 , and $\Sigma_{id} = \Sigma_{fid} \cup \Sigma_{did}$. For these two systems G_{fid} and G_{did} , the specifications are to prohibit the occurrences of all the events at their initial states except λ_1 and λ_4 , as shown in Figure 7.14. Like in the previous cases, the selfloops are included after the event λ_1 in both figures. The sizes of the supervisors are respectively 64 states with 304 transitions for G_{fid} and 32 states with 128 transitions for G_{did} .

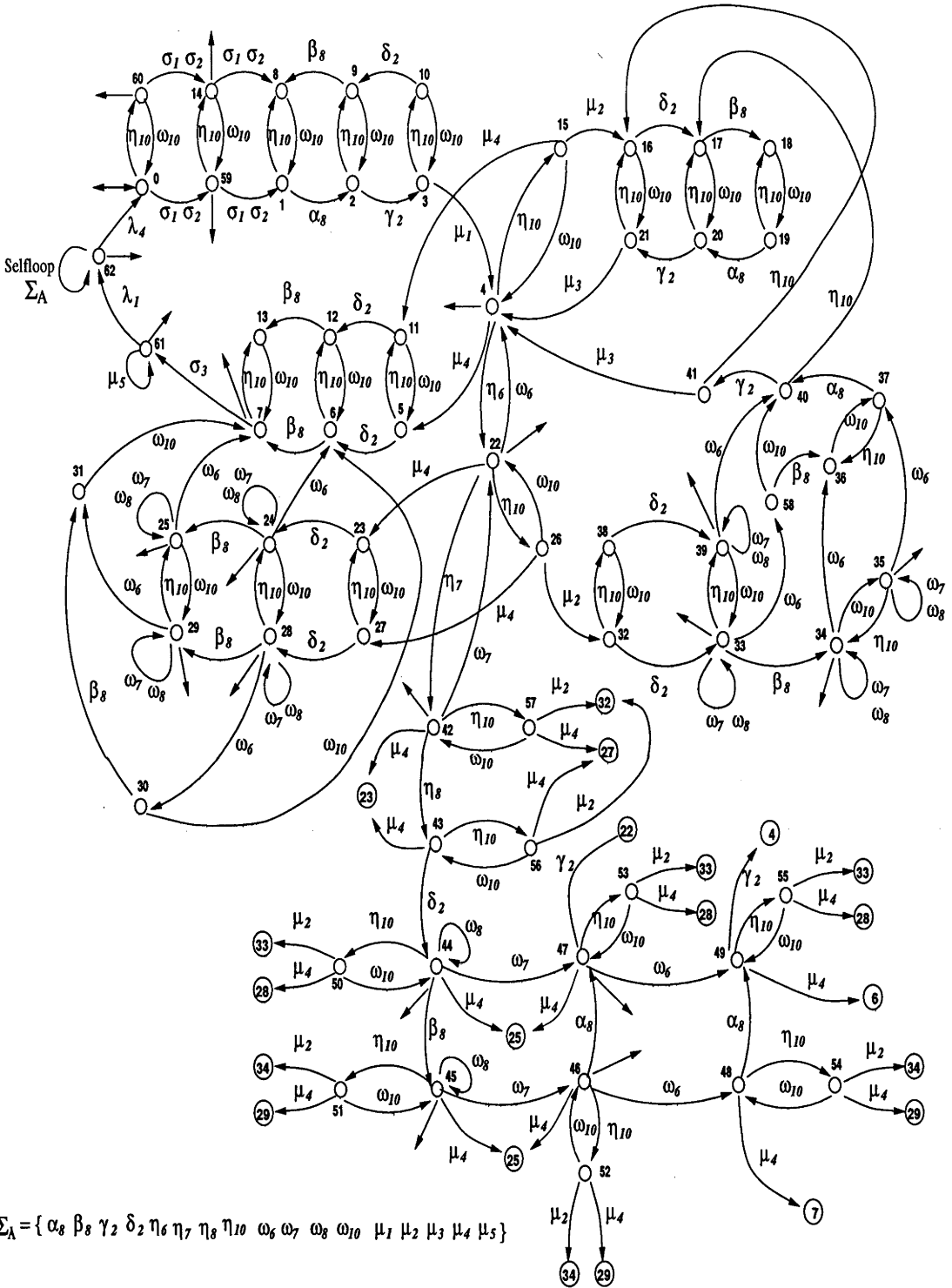


Figure 7.12: Specification for $G_{fpa} (E_{fpa})$

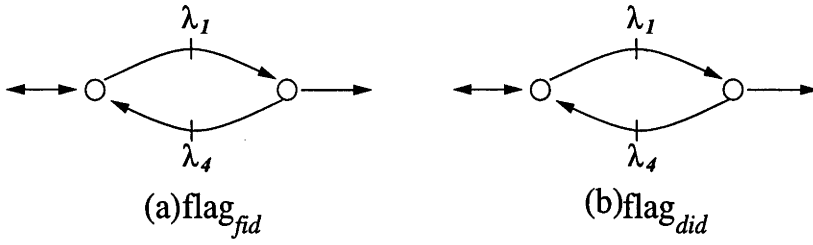


Figure 7.13: Synchronisation flags for G_{fid} and G_{did}

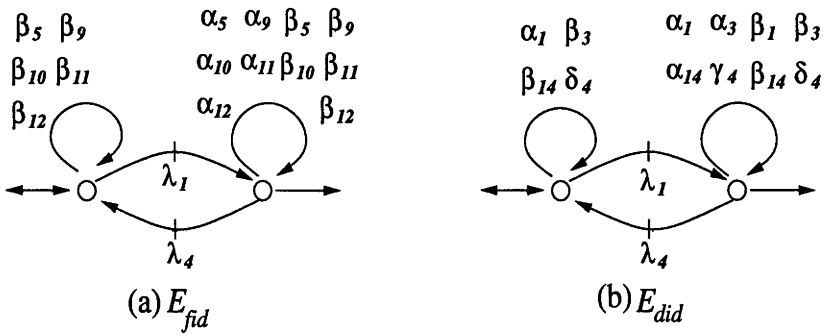


Figure 7.14: Specifications for G_{fid} and G_{did}

7.3.5 Verification for Water-Rinse Operation

In the previous sections, we have obtained the DES models of the subplants, specified local requirements, and then computed the corresponding local supervisors for the water rinse operation. In this section we will verify that the plant DESs of water rinse operation satisfy the sufficient conditions given in Chapter 3.

The event sets of the plants are

$$\begin{aligned}
\Sigma_{fpa} &= \{\alpha_8, \beta_8, \gamma_2, \delta_2, \eta_6, \omega_6, \eta_7, \omega_7, \eta_8, \omega_8, \eta_{10}, \omega_{10}, \mu_1, \mu_2, \mu_3, \mu_4, \mu_5, \sigma_1, \sigma_2, \sigma_3, \lambda_1, \lambda_4\} \\
\Sigma_{dpa} &= \{\gamma_3, \delta_3, \alpha_{13}, \beta_{13}, \eta_6, \omega_6, \eta_7, \omega_7, \eta_8, \omega_8, \sigma_1, \sigma_2, \sigma_3, \lambda_1, \lambda_4\}, \\
\Sigma_{fpr} &= \{\alpha_2, \beta_2, \alpha_4, \beta_4, \alpha_6, \beta_6, \sigma_1, \sigma_2, \sigma_3, \lambda_1, \lambda_4\}, \\
\Sigma_{dpr} &= \{\alpha_{15}, \beta_{15}, \alpha_{16}, \beta_{16}, \alpha_{17}, \beta_{17}, \alpha_{18}, \beta_{18}, \sigma_1, \sigma_2, \sigma_3, \lambda_1, \lambda_4\}, \\
\Sigma_{fid} &= \{\alpha_5, \beta_5, \alpha_9, \beta_9, \alpha_{10}, \beta_{10}, \alpha_{11}, \beta_{11}, \alpha_{12}, \beta_{12}, \lambda_1, \lambda_4\}, \\
\Sigma_{did} &= \{\gamma_4, \delta_4, \alpha_1, \beta_1, \alpha_3, \beta_3, \alpha_{14}, \beta_{14}, \lambda_1, \lambda_4\}.
\end{aligned}$$

The shared event set of each system, $\Sigma_{s_i} = \Sigma_i \cap (\bigcup_{k \neq i} (\Sigma_k))$, are as follows:

$$\begin{aligned}
\Sigma_{s_{fpa}} &= \{\eta_6, \omega_6, \eta_7, \omega_7, \eta_8, \omega_8, \sigma_1, \sigma_2, \sigma_3, \lambda_1, \lambda_4\}, \\
\Sigma_{s_{dpa}} &= \{\eta_6, \omega_6, \eta_7, \omega_7, \eta_8, \omega_8, \sigma_1, \sigma_2, \sigma_3, \lambda_1, \lambda_4\}, \\
\Sigma_{s_{fpr}} &= \{\sigma_1, \sigma_2, \sigma_3, \lambda_1, \lambda_4\}, \\
\Sigma_{s_{dpr}} &= \{\sigma_1, \sigma_2, \sigma_3, \lambda_1, \lambda_4\}, \\
\Sigma_{s_{fid}} &= \{\lambda_1, \lambda_4\}, \\
\Sigma_{s_{did}} &= \{\lambda_1, \lambda_4\}.
\end{aligned}$$

Since all the shared events are controllable, the mutual controllability conditions are verified trivially. For the shared-event-marking conditions, we mark all the states before the shared events. In addition, the conditions in Theorem 3.1 require that local specification languages should be $L_{i,m}$ -closed. We verify this by checking if $E_i = \overline{E_i} \cap L_{i,m}$, where E_i is the specification given on the plant G_i , and i is the index of the plant. Therefore, all the requirements in Theorem 3.1 are now satisfied. This guarantees that the concurrent actions of the decentralised supervisors achieve the same optimal behaviour as the centralised counterpart without blocking problems.

7.4 Detergent Preparation Operation

The detergent preparation operation is the procedure for preparing the hot caustic detergent solution in the tank T.1. Figure 7.15 is a schematic diagram of the elementary components involved in this operation. The operation proceeds as follows: at the beginning of the operation,

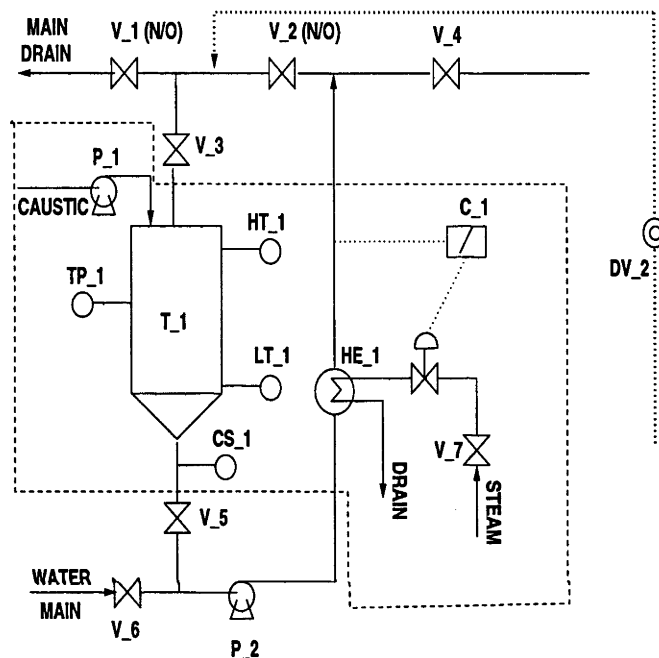


Figure 7.15: Detergent Preparation operation.

each elementary component is in its initial state and T.1 is with full of water. The operation is started with opening the circulating route around T.1. This will be done by closing V.1, and opening V.3 and V.5. The cleaning solution is to be circulated around the tank T.1 by the pump P.2. To heat up the solution, steam is admitted to HE.1 by opening the steam inlet valve V.7. The controller C.1 will keep the steam outlet temperature under 80°C . The steam valve V.7 is to be closed if the level in T.1 is less than 3L or the circulation pump P.2 is turned off. The solution will be heated to about 75°C . Concentrated caustic solution is to be supplied to T.1 by the pump P.1. The caustic solution is mixed with the solution circulating around T.1. If the required caustic level in the detergent solution is reached, P.1 is stopped. Also, P.1 should be turned off if the level in T.1 is less than 3L or the circulation pump P.2 is stopped.

G_{det}	G_{ap}	G_{cr}	V_1, V_3, V_5, P_2, LT_1, HT_1		Circulating the solution
		G_{hc}	G_{cs}	LT_1, CS_1, P_1, P_2	Caustic dosing
			G_{he}	LT_1, TP_1, V_7, C_1, P_2	Heating the solution
	G_{dnp}	V_2, V_4, V_6, DV_2			Idle

Table 7.2: Partition table for detergent preparation

7.4.1 Decomposition of the Components in the Detergent Preparation

The number of components involved in the detergent preparation is 15. So, the number of states in DES model is still about 4.9×10^4 which is fairly big to analyse and to synthesise a centralised supervisor. Therefore, we partition the component DESs into subplants to reduce the number of states to a manageable size. The first decomposition is to divide the components into two groups: one group of components which actively participates in the detergent preparation (designated as G_{ap}) and the other group which does not (G_{dnp}). The plant G_{ap} is further decomposed into the components involved in circulating the solution around the tank T_1 (G_{cr}) and those for the heating and caustic dosing (G_{hc}). Finally, the plant G_{hc} is partitioned into the heating part (G_{he}) and the caustic supplying part (G_{cs}). The partition table is presented in Table 7.2. Note that LT_1 and P_2 are shared components of G_{he} , G_{cs} and G_{cr} .

In addition to the elementary components, like in the water-rinse operation, we introduce the synchronisation events, $\lambda_1, \lambda_2, \lambda_4, \sigma_4, \sigma_5$. Now, the detergent preparation is operated as follows: after the synchronisation event λ_1 has occurred indicating the completion of water-rinse operation, appropriate valves in the detergent preparation operation are opened and then the circulation pump P_2 is turned on. The pump P_2 will be turned off when the synchronisation event σ_4 (indicating that the caustic level in the detergent solution reaches a set value) and σ_5 (indicating that the right temperature is reached) occur. The synchronisation event λ_2 indicates the completion of the detergent preparation. The event λ_4 signals that one cycle of the CIP process for the tank T_2 has completed.

7.4.2 Modelling and Synthesis for G_{ap}

The plant G_{ap} consists of the subplants G_{cr} and G_{hc} , and the latter comprises of G_{cs} and G_{he} . The subplant G_{cr} consists of the elementary components for the circulation of the detergent solution around the tank T.1. The subplant G_{cs} comprises of those components for dosing concentrated caustic solution, while G_{he} consists of those components for heating the solution. The DES ‘flag’ including the synchronisation events are presented in Figure 7.16.

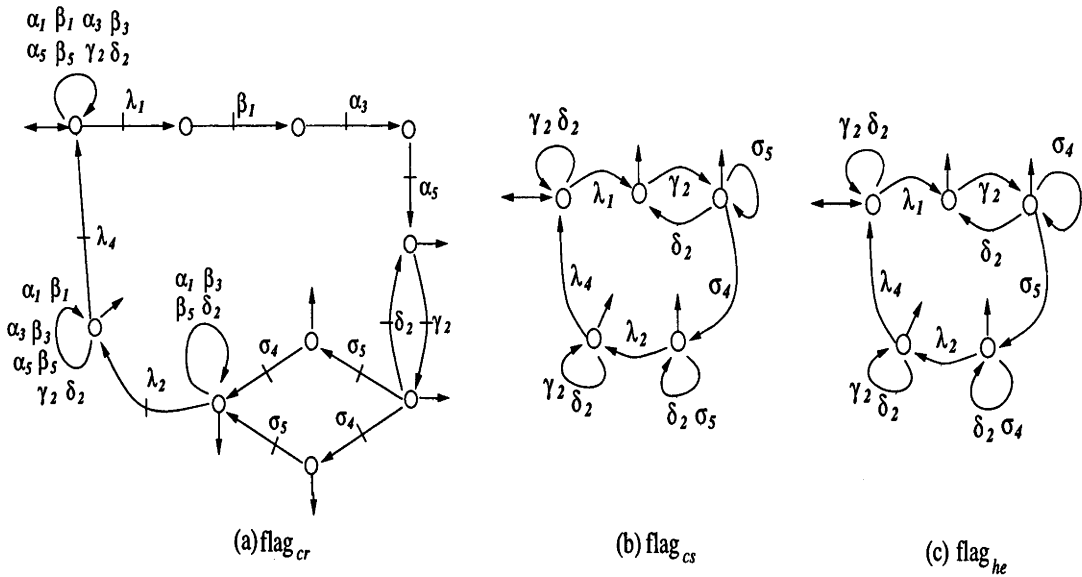


Figure 7.16: synchronisation flags for G_{ap}

Figure 7.16 (a) represents the following: after the water rinse operation has finished (λ_1), the valve V.1 is closed (β_1), and V.3 and V.5 are opened (α_3, α_5). Then the pump P.2 is turned on (γ_2). After the caustic level is ready (σ_4) and the detergent temperature is ready (σ_5), all the valves and pump are allowed to return to their initial states ($\alpha_1, \beta_3, \beta_5, \delta_2$). The detergent preparation is completed with occurrence of the event λ_2 . This is followed by the completion of the CIP process (λ_4). Figure 7.16 (b) and (c) are interpreted similarly. When the pump P.2 is on (γ_2), the events σ_4 (in (b)) and σ_5 (in (c)) are permitted to occur. Like in Water rinse process, the selfloops after the event λ_2 and before λ_4 are necessary to allow for those events to occur in the other subprocesses.

The automata shown in Figure 7.17 represent three physical constraints of G_{cr} . Figure 7.17 (a) represents that the level of the tank T_1 can only be increased after the feed valve V_3 has been opened and can only be decreased after the drain valve V_5 has been opened. Figure 7.17 (b) represents that the level of T_1 can be increased only after P_2 has been turned off. The DES for the physical constraint between the low level and the high level sensors of T_1 is given in Figure 7.17 (c). Four physical constraints are identified for the subplant G_{hc}

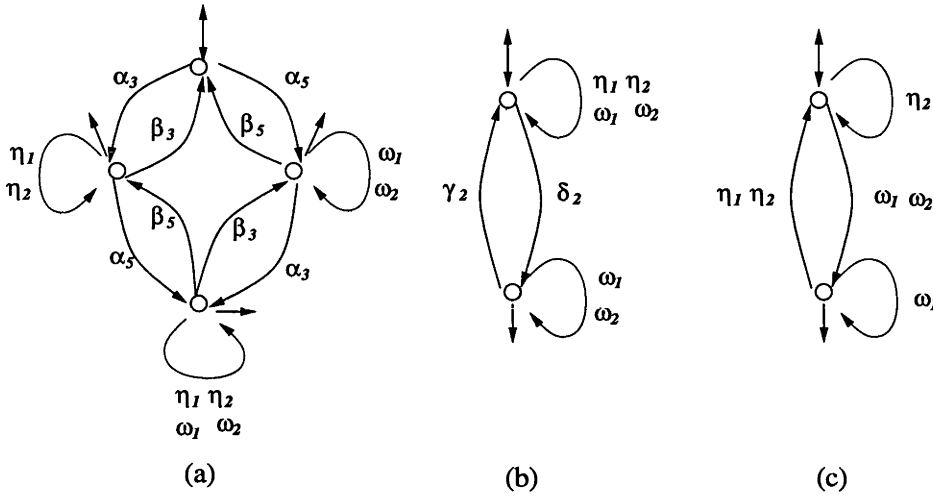


Figure 7.17: Physical constraints of G_{cr}

and given in Figure 7.18. Physical constraint represented in Figure 7.18(a) says that the level of T_1 can be increased only after P_2 has been turned on. The constraints modelled as the automata shown in Figure 7.18 (b), (c) and (d) are similar to (a) except that (b) is for the caustic level, and (c) and (d) are for the temperature of the solution. The plant G_{cs} requires (a) and (b), while G_{he} needs (a), (c) and (d). Note that the level increase of the tank T_1 by caustic supply is minimal and thus is ignored.

The DES G_{cr} , G_{cs} and G_{he} are computed using the synchronous compositions. Their event sets are

$$\Sigma_{cr} = \{\alpha_1, \beta_1, \alpha_3, \beta_3, \alpha_5, \beta_5, \gamma_2, \delta_2, \omega_1, \eta_1, \omega_2, \eta_2, \sigma_4, \sigma_5, \lambda_1, \lambda_2, \lambda_4\},$$

$$\Sigma_{cs} = \{\gamma_1, \delta_1, \gamma_2, \delta_2, \omega_1, \eta_1, \omega_3, \eta_3, \sigma_4, \sigma_5, \lambda_1, \lambda_2, \lambda_4\},$$

$$\Sigma_{he} = \{\alpha_7, \beta_7, \gamma_2, \delta_2, \omega_1, \eta_1, \omega_5, \eta_5, \zeta_1, \zeta_2, \sigma_4, \sigma_5, \lambda_1, \lambda_2, \lambda_4\}.$$

Here, $\Sigma_{hc} = \Sigma_{cs} \cup \Sigma_{he}$, $\Sigma_{ap} = \Sigma_{cr} \cup \Sigma_{hc}$, and the shared events are $\Sigma_{cs} \cap \Sigma_{he} = \Sigma_{cr} \cap$

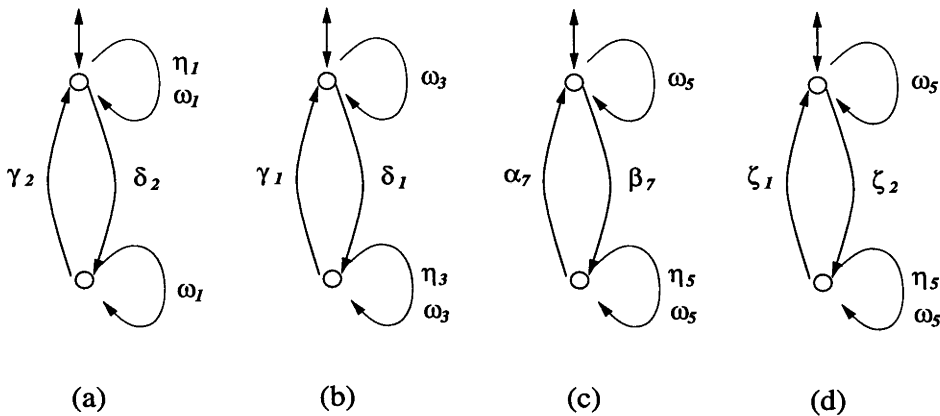


Figure 7.18: Physical constraints of G_{hc} (G_{cs} and G_{he})

$\Sigma_{cs} = \Sigma_{cr} \cap \Sigma_{he} = \{\gamma_2, \delta_2, \omega_1, \eta_1, \lambda_1, \sigma_4, \sigma_5, \lambda_2, \lambda_4\}$. The subplant G_{cr} has 486 states with 1668 transitions and G_{cs} has 72 states with 314 transitions, while G_{he} has 144 states with 754 transitions.

The requirements for G_{cr} (E_{cr}) are as follows:

1. After the synchronisation event λ_1 (completion of the water rinse operation) has occurred, close V_1 (β_1), and open V_3 and V_5 (α_3, α_5). Then turn on P_2 (γ_2).
2. If the level of T_1 is less than 3L (ω_1), turn off P_2 (δ_2).
3. If the level is increased to more than 3L (η_1), turn on P_2 (γ_2).
4. After both the synchronisation events σ_4 (caustic level ready) and σ_5 (temperature ready) have occurred, turn off P_2 (δ_2).
5. Return all valves to their initial states ($\alpha_1, \beta_3, \beta_5$). Then the detergent preparation operation is completed with the occurrence of the synchronisation event λ_2 .
6. The selfloops at the initial state mean that the subplant G_{cr} does not restrict the behaviours of the other subplants.

Formally, they are represented as an automaton given in Figure 7.19.

The requirements for G_{cs} (E_{cs}) are:

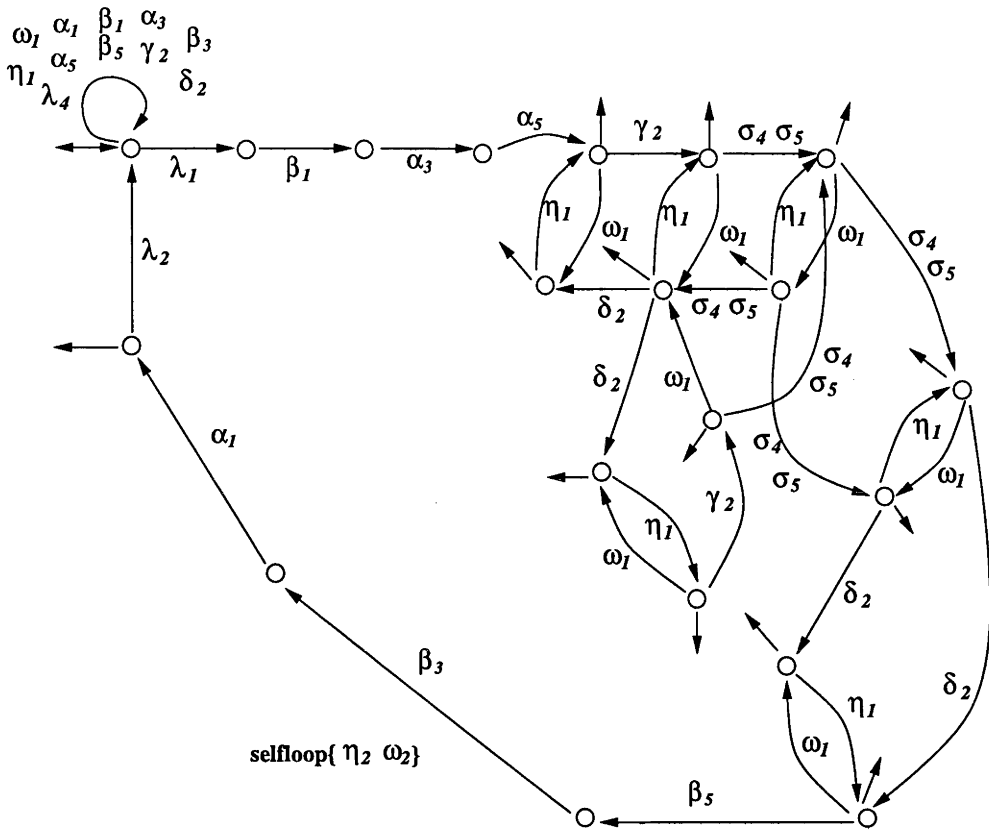


Figure 7.19: Specification for $G_{cr} (E_{cr})$

1. After the synchronisation event λ_1 has occurred, turn on the circulation pump P_2 (γ_2). Then turn on the caustic supplying pump P_1 (γ_1).
2. During the operation, if P_2 is turned off (δ_2) or the level of the tank T_1 is less than 3L (ω_1), turn off P_1 (δ_1).
3. If the caustic level is 'ready', turn off P_1 (δ_1) and send the signal σ_4 to the other plants.
4. Turn off P_2 (δ_2) and finish the operation with the occurrence of the synchronisation event λ_2 .
5. The selfloops at the initial state mean that the subplant G_{cs} does not restrict the behaviours of the other subplants.

The DES model for these requirements is presented in Figure 7.20. Similarly, the specification

for $G_{he} (E_{he})$ can be constructed and the DES model for the specifications is presented in Figure 7.21.

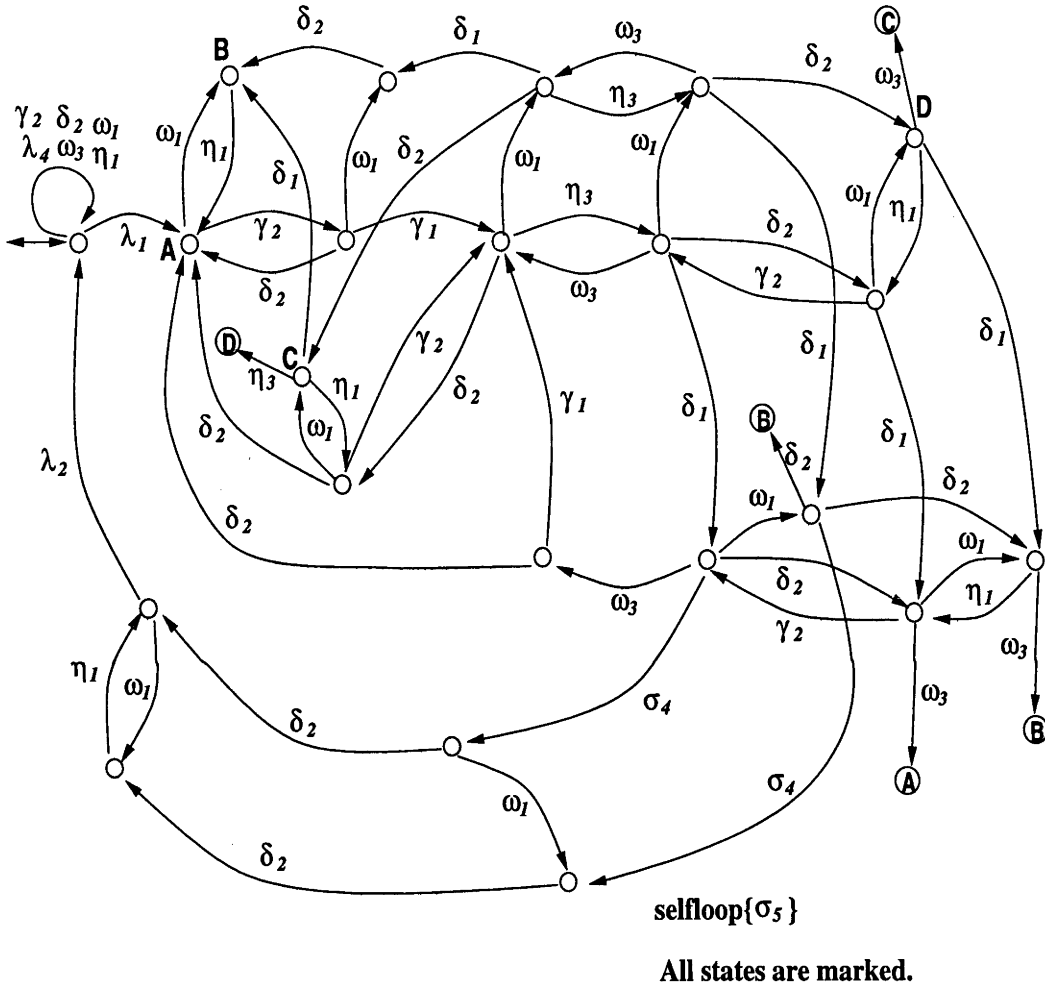


Figure 7.20: Specification for $G_{cs} (E_{cs})$

The supervisors for E_{cr} , E_{cs} and E_{he} are computed. The supervisor for E_{cr} has 44 states with 158 transitions and the supervisor for E_{cs} has 45 state with 109 transitions, while the supervisor for E_{he} has 54 states with 138 transitions.

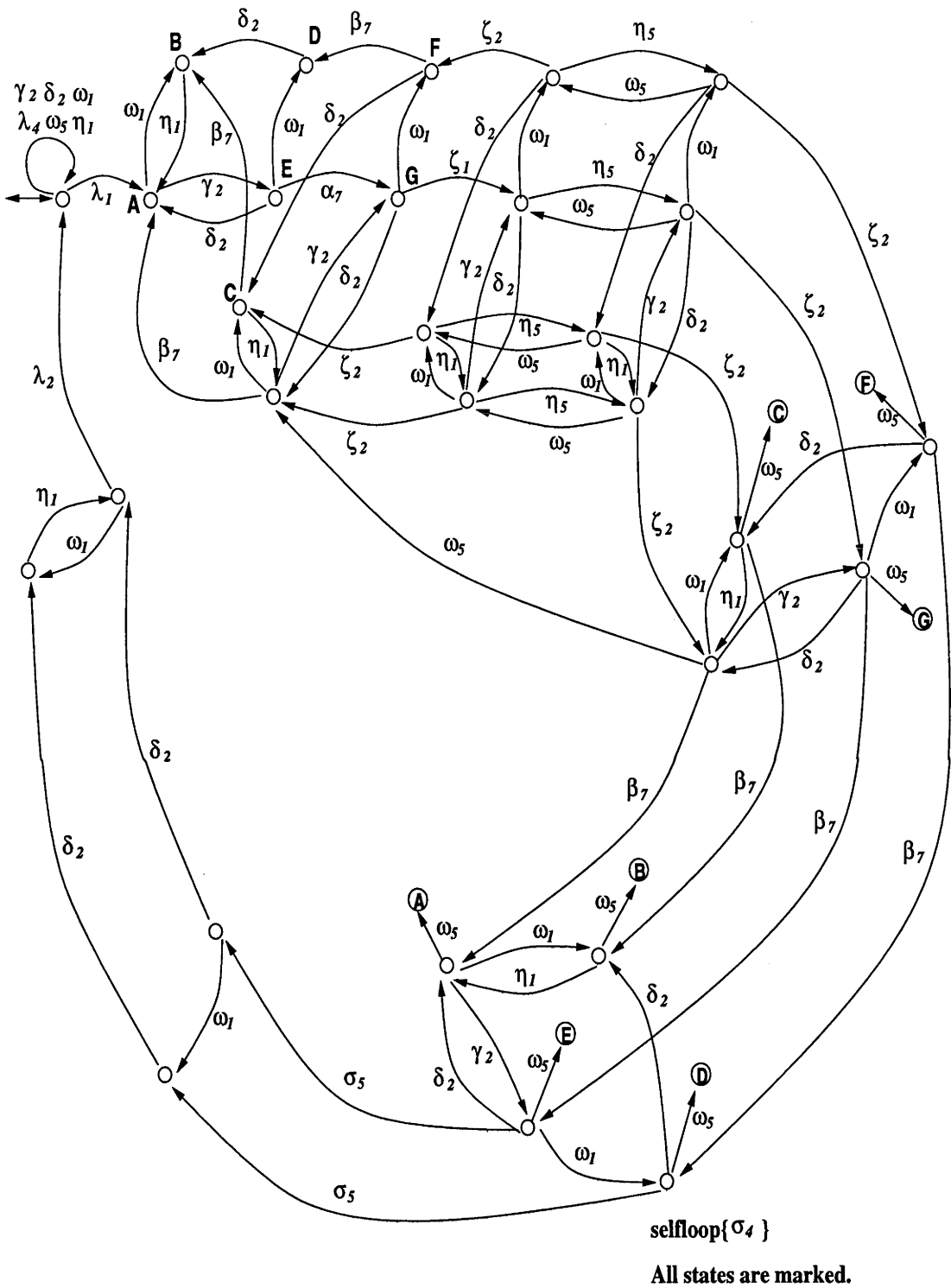


Figure 7.21: Specification for $G_{he} (E_{he})$

7.4.3 Modelling and Synthesis for G_{dnp}

The plant G_{dnp} consists of the elementary components that are required to remain at their initial states during the detergent preparation. The synchronisation events are included in the flags shown in Figure 7.22. The DES models for G_{dnp} are obtained by synchronous composition.

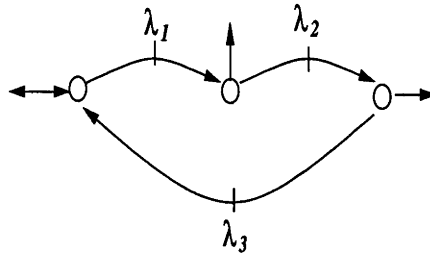


Figure 7.22: Synchronisation flag for G_{dnp}

The event set for G_{dnp} is

$$\Sigma_{dnp} = \{\alpha_2, \beta_2, \alpha_4, \beta_4, \alpha_6, \beta_6, \alpha_{18}, \beta_{18}, \lambda_1, \lambda_2, \lambda_4\}.$$

The plant G_{dnp} has 48 states with 240 transitions. Since the components of G_{dnp} are required to stay at their initial states during the detergent-preparation operation, the specification for G_{dnp} (E_{dnp}) is to disable all the events at their initial states. The DES model of E_{dnp} is presented in Figure 7.23. The supervisors for E_{dnp} is computed and its size is 48 state with 208 transitions.

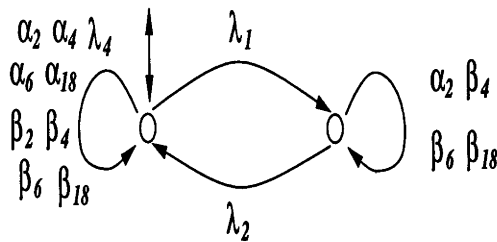


Figure 7.23: Specification for G_{dnp} (E_{dnp})

7.4.4 Verification for Detergent Preparation Operation

The verification of the structural conditions described in Chapter 3 for the detergent preparation operation can proceed similarly as the case of the water-rinse operation.

Firstly, we show that each local specification is $L_{i,m}$ -closed, i.e.,

$$E_i = \overline{E_i} \cap L_{i,m},$$

where E_i and $L_{i,m}$ are the specification and the marked behaviour of a local plant G_i , respectively. Here i is the index of the plant. The event sets are

$$\begin{aligned} \Sigma_{cr} &= \{\alpha_1, \beta_1, \alpha_3, \beta_3, \alpha_5, \beta_5, \gamma_2, \delta_2, \omega_1, \eta_1, \omega_2, \eta_2, \sigma_4, \sigma_5, \lambda_1, \lambda_2, \lambda_4\}, \\ \Sigma_{cs} &= \{\gamma_1, \delta_1, \gamma_2, \delta_2, \omega_1, \eta_1, \omega_3, \eta_3, \sigma_4, \sigma_5, \lambda_1, \lambda_2, \lambda_4\}, \\ \Sigma_{he} &= \{\alpha_7, \beta_7, \gamma_2, \delta_2, \omega_1, \eta_1, \omega_5, \eta_5, \zeta_1, \zeta_2, \sigma_4, \sigma_5, \lambda_1, \lambda_2, \lambda_4\}, \\ \Sigma_{dnp} &= \{\alpha_2, \beta_2, \alpha_4, \beta_4, \alpha_6, \beta_6, \alpha_{18}, \beta_{18}, \lambda_1, \lambda_2, \lambda_4\}. \end{aligned}$$

Then one has that the shared event sets of each system, $\Sigma_{s_i} = \Sigma_i \cap (\bigcup_{k \neq i} (\Sigma_k))$, are as follows:

$$\begin{aligned} \Sigma_{s_{cr}} &= \{\gamma_2, \delta_2, \omega_1, \eta_1, \sigma_4, \sigma_5, \lambda_1, \lambda_2, \lambda_4\}, \\ \Sigma_{s_{cs}} &= \{\gamma_2, \delta_2, \omega_1, \eta_1, \sigma_4, \sigma_5, \lambda_1, \lambda_2, \lambda_4\}, \\ \Sigma_{s_{he}} &= \{\gamma_2, \delta_2, \omega_1, \eta_1, \sigma_4, \sigma_5, \lambda_1, \lambda_2, \lambda_4\}, \\ \Sigma_{s_{dnp}} &= \{\lambda_1, \lambda_2, \lambda_4\}. \end{aligned}$$

Since all shared events are controllable, the mutual controllability conditions are verified trivially. For the shared-event-marking condition, we mark all the states before the shared events.

7.5 Detergent-Rinse Operation

The detergent-rinse operation is similar to the water-rinse operation except that here the detergent solution is sprayed instead of water. The control logic in water-rinse operation needs to be slightly changed to meet the requirements for this exception. If the level in the tank T.2 exceeds 20L, the flow is diverted to T.1 by the valve V.2. When the lid of T.2 is opened, the detergent-rinse operation will respond in the same way as the water-rinse operation. The effluent of the detergent solution in T.2 is returned to T.1. The supervisors for detergent-rinse operation are synthesised similarly.

7.6 Conclusions

In this chapter, we have shown an application of the theory of structural decentralised control of concurrent DES described in Chapter 3 to a CIP process for a small batch plant. The whole operation is firstly divided into four sequential operations and then in each operation further decomposition is carried out to reduce the number of states in the DES models of the plants and specifications. The modelling and the syntheses of supervisors for each operation are presented and the verifications of the conditions for each operation are also given.

Chapter 8

Conclusions and Future Works

In this thesis, within the framework of supervisory control theory, a structural decentralised control of DES and the related works are investigated. Firstly, we formulate a problem of structural decentralised control of concurrent DES. In particular, we consider the situation in which the global system is the synchronous composition of a number of subsystems with controllable or uncontrollable shared events. Local requirements are specified on the subsystems, and for these requirements local supervisors are designed. We have established two structural conditions such that, for a set of local requirements, local design and control achieve the same behaviour as that by a global optimal supervisor. Unlike the work in [LW88a], we allow non-prefix-closed local specifications and thus the question of blocking has been addressed. The first condition, the shared-event-marking condition, says that the states before the shared events in subsystems are required to be marked. The second condition, the mutual controllability condition, is interpreted as that a subsystem needs to track any uncontrollable shared event that could occur in the other subsystem. We have pointed out that the conditions are sufficient to achieve the distributivity of the control synthesis operator κ over synchronous composition. Also, since the conditions are dependent on the system structure not on each specification, once the conditions are established, decentralised control is achieved for a set of $L_{i,m}$ -closed local specifications. The main advantages of this approach are: by achieving these two structural conditions there may be an exponential savings of the computational efforts involved, while it still offers the same optimal behaviour as that would be obtained by a centralised control;

once a system structure has been verified, it can be used for a number of tasks without further checking. In Chapter 4, we have shown that a coordination scheme can be used to solve some rescheduling problems among local plants under structural conditions similar to the ones in Chapter 3. Again we have pointed out that since the conditions are structural, once the coordination architecture has been established, it can be used for different tasks. This approach may be useful for a plant which produces several products from a number of materials as shown in the example in Section 4.3. For the systems which do not satisfy the structural conditions, we have developed procedures to arrange the system structure (Chapter 5). Also, since we have found that in some situation, the shared-event-marking condition may be too stringent, we have investigated two possible ways to weaken the condition using the concepts of bisimulation and observer (Chapter 6). In Chapter 7, to illustrate our result, we consider a Cleaning-In-Process of a batch chemical process. Using our results, decentralised controllers of three suboperations, Water-Rinse, Detergent Preparation and Detergent-Rinse, are synthesised. Finally, in Appendix, as a result which is not related to the main subject of this thesis, we have developed a new non-linear approach to the design of model reference adaptive control scheme using a non-Euclidean gradient descent algorithm with respect to a Riemannian metric.

In summary, we have achieved the following in this thesis:

1. we have obtained structural properties to guarantee the distributivity of the control operator κ over synchronous composition (Chapter 3).
2. Using a coordination scheme, we have solved some rescheduling problems under the structural conditions (Chapter 4).
3. We have developed procedures to modify the system structure to arrange for the desired structural conditions (Chapter 5).
4. We have investigated ways to weaken the shared-event-marking condition (Chapter 6).

In the rest of this chapter, we give some areas for possible future research.

Extending to a more general structure

One possible direction of future research is to extend this result to more general structures. As mentioned in Chapter 3, the overall system G is the synchronous composition of subsystems as studied in [WH91]. Our conditions are obtained for this special structure. It would be useful to investigate whether our conditions can be extended to more general structures as in [LW88a], where the local systems are constructed by natural projections from a given global system, and this global system may not be a synchronous composition of the local systems. In addition, one may consider the situation in which the control status of shared events among subsystems is not the same.

Relaxation of the conditions

In Chapter 6, we have investigated methods to weaken the shared-event-marking condition. However, the new conditions are no longer structural. So it may be useful to investigate ways to weaken the condition while the structural property is maintained.

Building a better structure

Intuitively, the arrangement of a structural condition may mean building a better structure before one designs a controller. So, as an extension of our results in this thesis, it is useful to investigate this idea. For example, consider two systems as shown in Figure 8.1(a). The two systems are required to be synchronised at some stage. The states q_{12} , q_{13} , q_{22} and q_{23} are marked for the shared-event-marking condition. However, we have a situation that a string $\beta_1\beta_3$ in G_1 occurs while a string $\gamma_2\gamma_4$ in G_2 occurs. In this situation, since we mark the states q_{12} and q_{23} , we can consider that they are not blocking (i.e., both systems reach a marked states). However, strictly speaking it may be blocking since both systems do not have synchronisation. If one could use shared events σ_1 and σ_2 as communications between two plants (Figure 8.1(b)), this problem can be solved. How to formalise this idea mathematically requires more study.

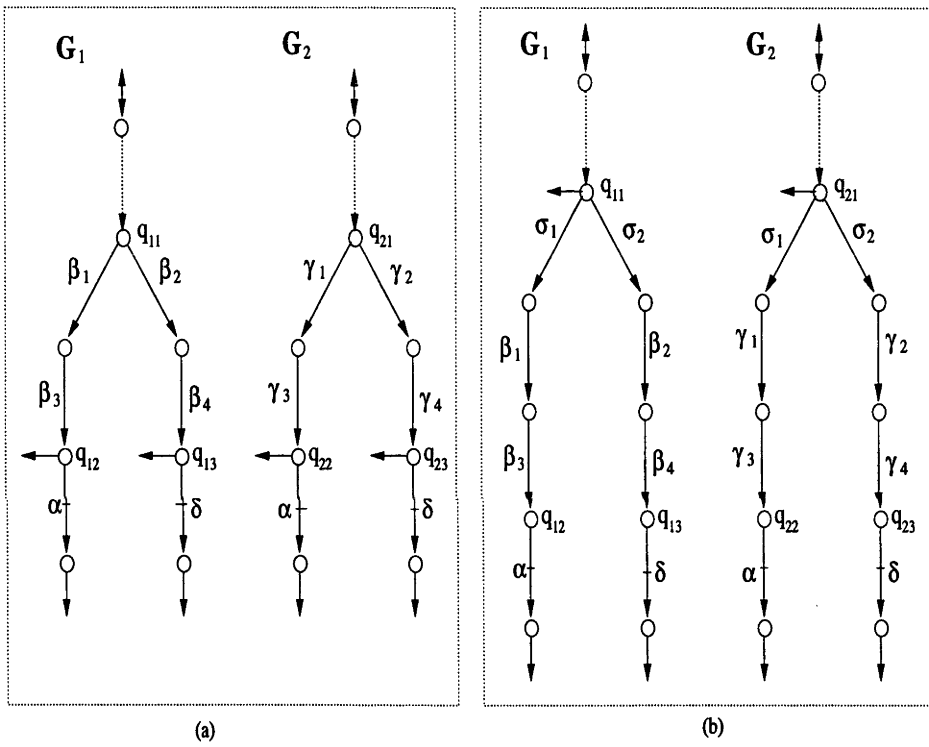


Figure 8.1: An example for building a better structure

Bibliography

- [ABJ⁺86] B. D. O. Anderson, R. R. Bitmead, C. R. Johnson, P. Kokotović, R. L. Kosut, I. M. Y. Mareels, L. Praly, and B. Riedle. *Stability of adaptive systems: Passivity and averaging analysis*. Cambridge, M.I.T. Press, Massachusetts, 1986.
- [Als96] N. J. Alsop. *Formal technique for the procedural control of industrial processes*. PhD thesis, The Imperial college, London, 1996.
- [Arn94] A. Arnold. *Finite Transition Systems: Semantics of communicating systems*. Prentice Hall, 1994.
- [BBW91] B. A. Brandin, B. Benhabib, and W. M. Wonham. Discrete-event system supervisory control applied to the management of manufacturing workcells. In *Proc. of 7th international conference on computer-aided prod. eng.*, pages 527–536, Cookeville, TN, August 1991.
- [BCOQ92] F. Baccelli, G. Cohen, G. J. Olsder, and J. P. Quadrat. *synchronisation and linearity; an algebra for discrete-event systems*. Wiley, 1992.
- [BHG⁺90] S. Balemi, G. J. Hoffmann, P. Gyugyi, H. Wong-Toi, and G. F. Franklin. Supervisory control of a rapid thermal multiprocessor. *IEEE Transactions on Automatic Control*, 38(7):1040–1059, 1990.
- [Bra93] B. A. Brandin. *Real-Time Supervisory Control of Automated Manufacturing Systems*. Ph.D. thesis, Department of Electrical Engineering, University of Toronto, 1993, Also appears as Technical Report 9302, Systems Control Group, Department of Electrical Engineering, University of Toronto, February, 1993.

- [BW94] B. A. Brandin and W. M. Wonham. Supervisory control of timed discrete-event systems. *IEEE Transactions on Automatic Control*, 39(2):329–342, 1994.
- [Cas93] C.G. Cassandras. *Discrete event systems-modelling and performance analysis*. Irwin, Boston, 1993.
- [CDFV88] R. Cieslak, C. Desclaux, A. S. Fawaz, and P. Varaiya. Supervisory control of discrete-event processes with partial observations. *IEEE Transactions on Automatic Control*, 33(3):249–260, March 1988.
- [CDQV85] G. Cohen, D. Dubois, J. P. Quadrat, and M. Viot. A linear-system-theoretic view of discrete-event processes and its use for performance evaluation in manufacturing. *IEEE Transactions on automatic control*, AC-30(3):210–220, 85.
- [CLL92] S.-L. Chung, S. Lafortune, and F. Lin. Limited lookahead policies in supervisory control of discrete event systems. *IEEE Transactions on Automatic Control*, 37(12):1921–1935, December 1992.
- [CLO95] C. G. Cassandras, S. Lafortune, and G. J. Olsder. Introduction to the modelling, control and optimisation of discrete-event systems. In A. Isidori, editor, *Trends in Control-European perspective*, pages 217–291. Springer-Verlag, 1995.
- [CMQV89] G. Cohen, P. Moller, J. P. Quadrat, and M. Viot. Algebraic tools for the performance evaluation of discrete-event systems. *Proceedings of IEEE*, 77:39–58, 1989.
- [DA94] R. David and H. Alla. Petri nets for modelling of dynamic systems: A survey. *Automatica*, 30(2):175–202, 1994.
- [dC92] M. P. do Carmo. *Riemannian geometry*. Birkhäuser, Boston, 1992.
- [Den88] M. J. Denham. A petri net approach to the control of discrete-event systems. *Advanced computing concepts and techniques in control engineering*, pages 191–214, 1988.
- [Eil74] S. Eilenberg. *Automata, language and machines, Vol. A*. Academic press, New York, 1974.

- [Fer89] Jean-Claude Fernandez. An implementation of an efficient algorithm for bisimulation equivalence. *Science of computer programming*, 13:219–236, 1989.
- [GG90] P. Glasserman and W. B. Gong. Smoothed perturbation analysis for a class of discrete event systems. *IEEE Transactions on Automatic Control*, 35:1218–1230, 1990.
- [GY88] M. P. Golden and B. E. Ydstie. Bifurcation in model reference adaptive control systems. *Systems and control letters*, 11:413–430, 1988.
- [Haj84] B. Hajek. Optimal control of two interacting service stations. *IEEE Transactions on automatic control*, AC-29(6):491–499, 84.
- [HC91] Y. C. Ho and X. R. Cao. *Perturbation analysis of discrete-event dynamic systems*. Kluwer academic publishers, London, 1991.
- [HC83] Y. C. Ho and C. Cassandras. A new approach to the analysis of discrete-event systems. *Automatica*, 19(2):149–167, 83.
- [HK90] L. E. Holloway and B. H. Krogh. Efficient synthesis of control logic for a class of discrete event systems. *IEEE Transactions on Automatic Control*, 35:2672–2677, 1990.
- [HL96] M. Heymann and F. Lin. Discrete event control of nondeterministic discrete event systems. In *In Proceedings of the 35th IEEE conference on Decision and Control*, pages 4445–4450, 1996.
- [HL98] M. Heymann and F. Lin. Discrete event control of nondeterministic systems. *IEEE Transactions on Automatic Control*, 43(1):3–17, 1998.
- [HM94] U. Helmke and J. B. Moore. *Optimisation and dynamical systems*. Springer-Verlag, London, 1994.
- [HP73] C.-C. Hang and P. C. Parks. Comparative studies of model reference adaptive control systems. *IEEE transaction on automatic control*, 18:419–428, 1973.

- [IH87] A. Ichikawa and K. Hiraishi. Analysis and control of discrete event systems represented by petri nets. *Lecture notes in control and information sciences 103 Discrete event systems: models and applications*, pages 115–134, 1987.
- [JP96] Z.-P. Jiang and L. Praly. A self-tuning robust nonlinear controller. *IFAC Congress 96*, 1996.
- [KAM87] R. L. Kosut, B. D. O. Anderson, and I. M. Y. Mareels. Stability theory for adaptive systems: Methods of averaging and persistency of excitation. *IEEE transaction on automatic control*, 32:26–34, 1987.
- [KB94] J. Kosecka and L. Bogoni. Application of discrete event systems for modelling and controlling robotic agents. *IEEE international conference on robotics and automation*, pages 2557–2562, 1994.
- [KKK92] P. Kokotović, I. Kanellakopoulos, and M. Krstić. On letting adaptive control be what it is: Nonlinear feedback. *IFAC adaptive systems in control and signal processing, Grenoble, France*, 1992.
- [KKK94] M. Krstić, I. Kanellakopoulos, and P. Kokotović. Nonlinear design of adaptive controller for linear systems. *IEEE transactions on automatic control*, 39:738–752, 1994.
- [KKK95] M. Krstić, I. Kanellakopoulos, and P. Kokotović. *Nonlinear and adaptive control design*. John Wiley & sons, New York, 1995.
- [KW95] P. Kozak and W. M. Wonham. Fully decentralised solutions of supervisory control problems. *IEEE Transactions on Automatic Control*, 40(12):2094–2097, December 1995.
- [Laf88] S. Lafortune. Modelling and analysis of transaction execution in database systems. *IEEE Transactions on Automatic Control*, 33(5):439–447, 1988.
- [Lan79] Y. D. Landau. *Adaptive control: The model reference approach*. Marcel Dekker, New York, 1979.

- [LC90] S. Lafortune and E. Chen. The infimal closed controllable superlanguage and its application in supervisory control. *IEEE Transactions on Automatic Control*, 35(4):398–405, April 1990.
- [Led96] R. Leduc. *PLC implementation of a DES supervisor for a manufacturing testbed: An implementation perspective*. Master's thesis, Dept. of electrical and computer engineering, University of Toronto, Toronto, 1996.
- [Lin91] F. Lin. Control of large scale discrete event systems: Task allocation and coordination. *Systems and Control Letters* 17, pages 169–175, 1991.
- [Lin93] F. Lin. Robust and adaptive supervisory control of discrete event systems. *IEEE Transactions of Automatic Control*, 38(12):1848–1852, 1993.
- [Lin96] Peter Linz. *An introduction to formal languages and automata, 2nd Ed.* D.C. Heath and Company, Lexington, MA, 1996.
- [LK84] W. Lin and P. R. Kumar. Optimal control of a queueing system with two heterogeneous servers. *IEEE Transactions on automatic control*, AC-29:696–703, 84.
- [LM93] Z. Liu and S. Macchietto. Cleaning in place policies for a food processing batch pilot plant. *Food and bioproduct processing*, 71, September 1993.
- [LMMB96] S. C. Lauzon, A. K. L. Ma, L. K. Mills, and B. Benhabib. Application of DES theory to flexible manufacturing. *IEEE control systems*, 16:41–48, 1996.
- [LW88a] F. Lin and W. M. Wonham. Decentralised supervisory control of discrete-event systems. *Information Sciences*, 44:199–224, 1988.
- [LW88b] F. Lin and W. M. Wonham. On observability of discrete-event systems. *Information Sciences*, 44:173–198, 1988.
- [LW90] F. Lin and W. M. Wonham. Decentralized control and coordination of discrete-event systems with partial observation. *IEEE Transactions of Automatic Control*,

- 35(12):1330–1337, December 1990. Also appears in shorter form as “Decentralized Control and Coordination of Discrete-Event Systems”, Proc. 27th IEEE Conf. on Decision and Control, Dec. 1988., pp. 1125-1130.
- [LW91] F. Lin and W. M. Wonham. Verification of nonblocking in decentralized control. *Control Theory and Advanced Technology*, 7(1):19–29, 1991.
- [LW93] Y. Li and W. M. Wonham. Control of vector discrete-event systems I - the base model. *IEEE Transactions on Automatic Control*, 38(8):1214–1227, August 1993.
- [LW94] Y. Li and W. M. Wonham. Control of vector discrete-event systems II - control synthesis. *IEEE Transactions on Automatic Control*, 39(3):521–530, March 1994.
- [MB86] I. M. Y. Mareels and R. R. Bitmead. Non-linear dynamics in adaptive control: Chaotic and periodic stabilisation. *Automatica*, 22:641–655, 1986.
- [MB88] I. M. Y. Mareels and R. R. Bitmead. Non-linear dynamics in adaptive control: Chaotic and periodic stabilisation - II analysis. *Automatica*, 24:485–497, 1988.
- [Mil80] R. Milner. *A Calculus of Communicating Systems*, volume 92 of *LNCS*. Springer-Verlag, New York, 1980.
- [NA89] K. S. Narendra and A. M. Annaswamy. *Stable adaptive systems*. Prentice Hall, Englewood Cliffs, NJ, 1989.
- [NN94] Y. Nesterov and A. Nemirovskii. Interior-point polynomial algorithms in convex programming. *Society for industrial and applied mathematics*, 1994.
- [Ove94] A. Overkamp. Supervisory control for nondeterministic systems. In *proceedings 11th Int’l Conf. on Analysis and Optimization of Systems*, pages 59–65, INRIA, École des Mines de Paris, 1994.
- [OW90] J. S. Ostroff and W. M. Wonham. A framework for real-time discrete event control. *IEEE Transactions on Automatic Control*, 35(4):386–397, April 1990.

- [OWK61] P. V. Osburn, H. P. Whitaker, and A. Kezer. New developments in the design of adaptive control systems. *IAS Paper*, 1961.
- [Par66] P. C. Parks. Lyapunov redesign of model reference adaptive control systems. *IEEE transaction on automatic control*, 11:362–367, 1966.
- [Par72] D. L. Parnas. On the criteria to be used in decomposing systems into modules. *Communications of the ACM*, 15(12):1053–1058, 1972.
- [PCW85] D. L. Parnas, P. C. Clements, and D. M. Weiss. The modular structure of complex systems. *IEEE Transactions on Software Engineering*, SE-11(3):259–266, 85.
- [Pet81] J. L. Peterson. *Petri Net Theory and the Modelling of Systems*. Prentice-Hall, Inc., 1981.
- [Ram83] P. J. Ramadge. *Control and Supervision of Discrete Event Processes*. Ph.D. thesis, Department of Electrical Engineering, University of Toronto, 1983.
- [Ram88] P. J. Ramadge. The complexity of some basic control problems for discrete-event systems. *Advanced computing concepts and techniques in control engineering*, pages 171–190, 1988.
- [RSR95] S. L. Ricker, N. Sarkar, and K. Rudie. A discrete-event systems approach to modelling dextrous manipulation. In *Proc. of 33rd Annual Allerton Conference on Communication, Control, and Computing*, pages 156–165, Monticello, Illinois, 1995.
- [Rud88] K. G. Rudie. *Software for the Control of Discrete Event Systems: A Complexity Study*. M.A.Sc thesis, Department of Electrical Engineering, University of Toronto, 1988.
- [RVW82] Z. Rosberg, P. P. Varaiya, and J. C. Walrand. Optimal control of service in random queues. *IEEE Transactions on automatic control*, AC-27(3):600–610, 82.
- [RW87a] P. J. Ramadge and W. M. Wonham. Modular feedback logic for discrete event systems. *SIAM J. Control and Optimization*, 25(5):1202–1218, 1987.

- [RW87b] P. J. Ramadge and W. M. Wonham. Supervisory control of a class of discrete-event processes. *SIAM J. Control and Optimization*, 25(1):206–230, 1987.
- [RW89] P. J. Ramadge and W. M. Wonham. The control of discrete event systems. *Proc. IEEE, Special Issue on Discrete Event Dynamic Systems*, 77(1):81–98, January 1989.
- [RW90] K. Rudie and W. M. Wonham. Supervisory control of communicating processes. In R.L. Probert L. Logrippo and H. Ural, editors, *protocol specification, testing and verification*, pages 243–257. North Holland: Elsevier, 1990.
- [RW92] K. Rudie and W. M. Wonham. Think globally, act locally: Decentralized supervisory control. *IEEE Transactions on Automatic Control*, 37(11):1692–1708, November 1992.
- [San96] A. Sanchez. *Formal specification and synthesis of procedural controllers for process systems: Lecture notes in control and information sciences 212*. Springer, 1996.
- [SB65] B. Shackcloth and R. L. Butchart. Synthesis of model reference adaptive systems by Lyapunov's second method. *Proceedings of IFAC symposium on the theory of self-adaptive control systems, Teddington, England*, pages 145–152, 1965.
- [str83] K. J. Åström. Theory and applications of adaptive control—a survey. *Automatica*, 19:471–486, 1983.
- [sW95] K. J. Åström and B. Wittenmark. *Adaptive control: 2nd edition*. Addison-Wesley, New York, 1995.
- [VB96] L. Vandenberghe and S. Boyd. Semidefinite programming. *SIAM review*, 38(1):45–95, 1996.
- [WBS96] R. A. Williams, B. Benhabib, and K. C. Smith. A DES-theory-based hybrid supervisory control system for manufacturing systems. *Journal of manufacturing systems*, 15(2):71–83, 1996.

- [WH91] Y. Willner and M. Heymann. Supervisory control of concurrent discrete-event systems. *International Journal of Control*, 54(5):1143–1169, 1991.
- [Won96] W. M. Wonham. Notes on control of discrete-event systems. *ELE 1636F/1637S, Department of Electrical Engineering, University of Toronto*, 1996.
- [WR87] W. M. Wonham and P. J. Ramadge. On the supremal controllable sublanguage of a given language. *SIAM J. Control and Optimization*, 25(3):637–659, 1987.
- [WR88] W. M. Wonham and P. J. Ramadge. Modular supervisory control of discrete event systems. *Mathematics of Control, Signal and Systems*, 1(1):13–30, 1988.
- [WTHM95] K. C. Wong, J. G. Thistle, H.-H. Hoang, and R. P. Malhamé. Conflict resolution in modular control with application to feature interaction. In *Proceedings of the 34th IEEE Conf. on Decision and Control*, December 1995.
- [WvS96] K. C. Wong and J. H. van Schuppen. Decentralized supervisory control of discrete-event systems with communication. In *Proceedings International Workshop on Discrete Event Systems 1996 (WODES96)*, pages 284–289, London, 1996. IEE.
- [WW96a] K. C. Wong and W. M. Wonham. Hierarchical control of discrete-event systems. *Discrete Event Dynamic Systems: Theory and Applications*, 6(3):241–273, July 1996.
- [WW96b] K. C. Wong and W. M. Wonham. Hierarchical control of timed discrete-event systems. *Discrete Event Dynamical Systems: Theory and Applications*, 6(3):275–306, July 1996.
- [WW96c] K. C. Wong and W. M. Wonham. Modular control and coordination of discrete-event systems. In *Proceedings of the 4th IEEE Mediterranean Symposium on New Direction in Control and Automation*, pages 595–599, Chania, Crete, Greece, June 1996.

- [YLA95] X. Yang, M. D. Lemmon, and P. J. Antsaklis. On the supremal controllable sub-language in the discrete-event model of nondeterministic hybrid control systems. *IEEE Transactions on Automatic Control*, 40(12):2098–2103, December 1995.
- [ZW90] H. Zhong and W. M. Wonham. On the consistency of hierarchical supervision in discrete-event systems. *IEEE Transactions on Automatic Control*, 35(10):1125–1134, October 1990.

List of Publications

The following is a list of papers leading to this thesis.

Journal papers

- Sang-Heon Lee and Robert Mahony, “Model Reference Adaptive Control using Non-Euclidean Gradient Descent”, To be submitted.
- Sang-Heon Lee and K.C.Wong, “Structural Decentralised Control of Concurrent Discrete Event Systems”, In preparation.
- Sang-Heon Lee and K.C.Wong, “An Application of Structural Decentralised Control of Discrete Event Systems to a CIP Process”, In preparation.

Conference papers

- Sang-Heon Lee and Robert Mahony, “Model Reference Adaptive Control using Non-Euclidean Gradient Descent”, *the 4th European Control Conference* Brussels, Belgium, 1997.
- Sang-Heon Lee and Robert Mahony, “Speeding up Adaptation using Non-Euclidean Gradient Descent in Model Reference Adaptive Control”, *2nd Asian Control Conference*, Seoul, Korea, 1997.
- Sang-Heon Lee and K.C.Wong, “Decentralised Control of Concurrent Discrete-Event Systems with Non-Prefix Closed Local Specifications”, *CDC97*, San Diego, USA, 1997.

Appendix A

Model Reference Adaptive Control Using Non-Euclidean Gradient Descent

Abstract

In this appendix, a non-linear approach to the design of model reference adaptive control is presented. The approach is demonstrated by a case study of a simple single-pole and no zero, linear, discrete-time plant. The essence of the idea is to generate a full non-linear model of the plant dynamics and the parameter adaptation dynamics as a gradient descent algorithm with respect to a Riemannian metric. It is shown how a Riemannian metric can be chosen so that the modelled plant dynamics do in fact match the true plant dynamics. The performance of the proposed scheme is compared to a traditional model reference adaptive control scheme using the classical sensitivity derivatives (Euclidean gradients) for the descent algorithm.

KEY WORDS: Adaptive control; Discrete-time system; Riemannian geometry

A.1 Introduction

Model reference adaptive control (MRAC) is one of the main approaches to the adaptive control of servo systems. Model reference adaptive control (MRAC) scheme has been attractive from the beginning of the adaptive control era because it is simple in practical implementation and does not require plant identification. In MRAC, the desired performance of a closed-loop system is specified in terms of a reference model and the controller parameters are adjusted to minimise a given error function.

Historically, the MIT rule [OWK61] was the parameter adaptation mechanism used for the first published application of MRAC. In this case, the control parameters are updated according to a continuous-time ordinary differential equation (ODE) generated by setting the time derivatives of the parameter equal to the negative gradient of a performance index. The per-

formance index used was the integral square of the response error, the difference between the actual closed-loop system output and the reference model output. This gradient is commonly known as the *sensitivity derivative* of the system. To balance between system stability and the adaptation speed, an adaptation gain is introduced into the control parameter ODE. This adaptation gain plays a crucial role in system stability, ensuring that the dynamics induced in the controller by the adaptation rule do not interfere with the system dynamics. Unfortunately, it is usually not possible *a priori* to choose a suitable value for the adaptation gain. Consequently, the MIT rule for the adaptation of control parameters suffers from a fundamental stability problem [Par66, HP73, str83].

Among many subsequent approaches to adaptive control, the schemes that retain the closest resemblance to the MIT rule are those based on using the Lyapunov stability method [Par66, SB65]. These designs have the advantage that they take into consideration the combined system and parameter adaptation dynamics and design a controller to guarantee stability of the full system [Lan79, NA89]. (A recent overview of design procedures is given in [sW95, Chapter 5].) However, it has proved difficult to generate valid Lyapunov functions for the full system dynamics, and the classical Lyapunov design relies on the assumption that the adaptation dynamics does not evolve quickly compared to the system dynamics. Once again the gain of the adaptation dynamics plays a crucial role in determining system stability. To provide an estimate of when MRAC systems designed using Lyapunov techniques are practical, several authors [KAM87, ABJ⁺86] have used the averaging theory to produce rigorous stability results.

Classical adaptive control designs were based around linear design techniques. However, works in the late eighties [MB86, MB88, GY88] showed that highly non-linear and even chaotic behaviour can result from relatively simple adaptive control schemes. This perspective has led some people to view the adaptive control design as a fully non-linear design problem. Authors [KKK92, KKK94, KKK95, JP96] have made some advances in explicit non-linear adaptive control design methodology. However, much of this design methodology is still based around Lyapunov concepts. A fundamental limitation of Lyapunov theory is the difficulty of finding a suitable Lyapunov function for a given system.

In this appendix, we present a non-linear approach to the design of model reference adaptive control schemes for linear systems. In our approach, we begin with a full non-linear system model, combining the non-linear parameter adaptation dynamics and the linear plant dynamics as a gradient descent algorithm with respect to a general Riemannian metric. At each step, the Riemannian metric is chosen so that the modelled plant dynamics does in fact match the true plant dynamics. Once the Riemannian metric is fully specified, the adaptation dynamics is uniquely defined. In this way, the adaptation dynamics induced in the adjustable parameters incorporates the knowledge of the true plant dynamics.

An advantage of the proposed design procedure is that the adaptation gain no longer plays a role in the adaptive mechanism. Adaptation gain requires to balance between the system stability and the adaptation speed. Since there is no knowledge about the size of this gain, we usually force to choose smaller gain than would be desired. As a consequence, classical adaptation schemes generally result in slow adaptation of the closed-loop systems. Conversely, the proposed scheme achieves fast convergence of the adaptive parameters by subsuming the adaptation gain into the Riemannian metric and incorporating knowledge of the plant dynamics in the adaptation rule.

The design procedure is demonstrated by a case study of a simple single-pole, strictly proper, discrete-time plant. Since our aim in this appendix is to study a new adaptation law in a simple situation, we make a number of strong assumptions on plant structure. We assume that the system is a deterministic model with no noise. The reference signal is taken to be a step function and the plant and the reference model are both strictly stable. In addition we assume that the high frequency gains of plant and reference model have the same sign.

This appendix consists of seven sections including the introduction. Section A.2 describes the explicit formulation of the MRAC scheme considered. In Section A.3, a classical Euclidean gradient adaptation scheme using the MIT rule is reviewed. Section A.4 shows how to form a non-linear adaptive system from the combination of parameter adaptation dynamics and the plant dynamics in the form of a non-Euclidean gradient descent algorithm. In Section A.5, the problem of finding a positive definite matrix which defines the Riemannian metric required for the adaptive law is presented as a semi-definite programming problem. In Section A.6, the

performance of the proposed MRAC scheme is compared to a classical MRAC scheme. Section A.7 reviews the contribution and outlines the advantages and limitations of the proposed scheme.

A.2 Problem Formulation

In this section, a MRAC system for a simple linear plant is presented. Controller design is done on the assumption that the plant to be controlled has a single pole and no zero, and is a linear, stable, discrete-time system. The classical approach to MRAC in the discrete-time domain is shown in Figure A.1. The performance specifications are given in terms of a reference model, \bar{G} , along with the reference input signal, $r(k)$. Based on an estimate of the plant parameters, the certainty equivalence principle is used to design a feedback controller, C . The parameters are updated at each time instance k , according to the mismatch error, $e(k)$, between the actual closed-loop system output, $y(k)$, and the reference model output, $\bar{y}(k)$.

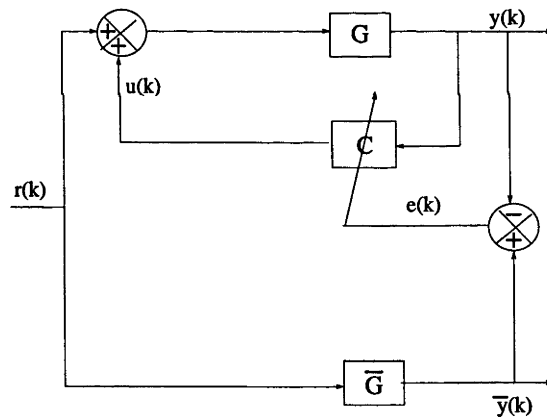


Figure A.1: System block diagram

In this work, the system considered is a deterministic model with no noise. The reference signal, $r(k)$, is taken to be a step function. As a result, the adaptation algorithm is not persistently excited and the consequence of this choice is discussed in later analysis. In addition, we assume that the plant and the reference model are both strictly-stable first order plants with relative degree one. It is assumed that the sign of the high frequency gain of the plant is known

and is the same as that of the reference model.

The discrete-time plant, G , to be controlled and the reference model, \bar{G} , representing the control objective are given by

$$G(z) = \frac{bz^{-1}}{1 + az^{-1}}, \quad \bar{G}(z) = \frac{\bar{b}z^{-1}}{1 + \bar{a}z^{-1}}, \quad (\text{A.1})$$

where, $a, b, \bar{a}, \bar{b} \in \mathbb{R}$ are the unknown plant parameters and the reference model parameters respectively.

Equivalently, the difference equations for the closed-loop system output and the reference model output are

$$y(k) = b[r(k-1) + u(k-1)] - ay(k-1), \quad (\text{A.2})$$

and

$$\bar{y}(k) = \bar{b}r(k-1) - \bar{a}\bar{y}(k-1). \quad (\text{A.3})$$

The controller design is based on a simple pole/zero placement technique using the *certainty equivalence principle*; the closed-loop transfer function of the true plant is

$$Y(z) = \frac{G(z)}{[I - G(z)C(z)]} R(z), \quad (\text{A.4})$$

where $C(z)$ is the controller. Using the certainty equivalence to replace the plant $G(z)$ in Eq. A.4 with an estimates, $\hat{G}(z)$, and solving for $C(z)$ yields

$$C(z) = \frac{[(\bar{b} - \hat{b}) + (\hat{a}\bar{b} - \bar{a}\hat{b})z^{-1}]}{\bar{b}\hat{b}z^{-1}}, \quad (\text{A.5})$$

where $\hat{a}, \hat{b} \in \mathbb{R}$ are the estimates of the unknown plant parameters.

A consequence of the simple design method used in here is that the controller $C(z)$ is non-causal. Since such a control strategy is impossible to apply in practice, it is necessary to modify Eq. A.5 to yield a causal controller. The option taken is to include a stable, low-pass filter of relative order 1 in the feedback loop

$$M(z) = \frac{z^{-1}(1 - \gamma)}{(1 - \gamma z^{-1})}, \quad (\text{A.6})$$

where $0 \leq \gamma < 1$. Thus, the difference equation of the final control action is

$$u(k-1) = \gamma u(k-2) + \frac{(1-\gamma)}{\bar{b}\hat{b}_{k-1}} \left[(\bar{b} - \hat{b}_{k-1})y(k-1) + (\hat{a}_{k-1}\bar{b} - \bar{a}\hat{b}_{k-1})y(k-2) \right]. \quad (\text{A.7})$$

Remark A.1 The low-pass filter, Eq. A.6, can be interpreted in two ways. Firstly, since the filter has relative order 1, the overall relative order of the controller is zero and the control law can be implemented as a causal operator. Secondly, and perhaps more importantly, the designed controller $C(z)$ of Eq. A.5 is a PD (Proportional Derivative) controller. This is evident by rewriting Eq. A.5 in the form

$$C(z) = \frac{1}{\bar{b}\hat{b}}(\bar{b} - \hat{b})(z-1) + \frac{(\bar{b} - \hat{b}) + (\hat{a}\bar{b} - \bar{a}\hat{b})}{\bar{b}\hat{b}}.$$

Such controllers are highly susceptible to high frequency noise due to the derivative operation. In practice, the derivative operator is usually combined with a low-pass filter to ensure good behaviour. This is exactly the form of the new control action $M(z)C(z)$. Even though the original control action generated by $C(z)$ is modified by the low-pass filter $M(z)$, the control action from $M(z)C(z)$ still results in the same steady-state behaviour as $C(z)$. \square

A.3 Classical Euclidean Gradient Descent Adaptation Scheme

In this section, a brief review of the MIT rule of the adaptive control is given in the context of the model considered. We refer to this adaptation rule as a *Euclidean gradient adaptation scheme*.

The key principle of MRAC design is to use an error, in this work the output mismatch error,

$$e(k) := \bar{y}(k) - y(k), \quad (\text{A.8})$$

to measure the performance of the adaptive algorithm. Consider taking a cost function,

$$\Phi(\hat{\theta}_{k-1}) := \frac{1}{2}e(k)^2, \quad (\text{A.9})$$

where $\hat{\theta}_{k-1} = (\hat{a}_{k-1}, \hat{b}_{k-1}) \in \mathbb{R}^2$ is the vector of parameter estimates at time $k - 1$. This cost function is used in the MIT rule of the original MRAC scheme [OWK61], [sW95, chapter 5].

The mismatch error $e(k)$ is obtained from Eq.'s A.2 and A.3. Explicitly, writing $e(k)$ to display its dependence on the parameters, $\hat{a}_{k-1}, \hat{b}_{k-1}$, yields

$$e(k) = \bar{b}r(k-1) - \bar{a}\bar{y}(k-1) - b \left[r(k-1) + \gamma u(k-2) + \frac{(1-\gamma)}{\bar{b}\hat{b}_{k-1}} \left[(\bar{b} - \hat{b}_{k-1})y(k-1) + (\hat{a}_{k-1}\bar{b} - \bar{a}\hat{b}_{k-1})y(k-2) \right] \right] + ay(k-1). \quad (\text{A.10})$$

Note that $u(k-2)$, $y(k-1)$ and $y(k-2)$ are independent from the $(k-1)$ 'th parameters, $\hat{a}_{k-1}, \hat{b}_{k-1}$.

The partial derivatives of $e(k)$ with respect to \hat{a}_{k-1} and \hat{b}_{k-1} are

$$\begin{aligned} \frac{\partial e(k)}{\partial \hat{a}_{k-1}} &= -\frac{b}{\hat{b}_{k-1}}(1-\gamma)y(k-2), \\ \frac{\partial e(k)}{\partial \hat{b}_{k-1}} &= \frac{b}{\hat{b}_{k-1}} \frac{(1-\gamma)}{\hat{b}_{k-1}} \left[y(k-1) + \hat{a}_{k-1}y(k-2) \right]. \end{aligned}$$

The adaptation mechanism for the parameter estimates vector is given by the discrete-time gradient descent algorithm;

$$\hat{\theta}_k = \hat{\theta}_{k-1} - s_k \frac{\partial \Phi(\hat{\theta}_{k-1})}{\partial \hat{\theta}_{k-1}} = \hat{\theta}_{k-1} - s_k e(k) \frac{\partial e(k)}{\partial \hat{\theta}_{k-1}}, \quad (\text{A.11})$$

where

$$\frac{\partial e(k)}{\partial \hat{\theta}_{k-1}} = \begin{pmatrix} \frac{\partial e(k)}{\partial \hat{a}_{k-1}} \\ \frac{\partial e(k)}{\partial \hat{b}_{k-1}} \end{pmatrix} = \begin{pmatrix} -(1-\gamma)y(k-2) \\ \frac{1}{\hat{b}_{k-1}}(1-\gamma)[y(k-1) + \hat{a}_{k-1}y(k-2)] \end{pmatrix} \frac{b}{\hat{b}_{k-1}}, \quad (\text{A.12})$$

and s_k is the adaptation gain. This algorithm is just a discrete-time version of the MIT rule [sW95, Chapter 5].

Remark A.2 Since the true plant parameters a and b are unknown, some approximations are required to compute the gradient in practice. In the adaptive scheme, the model estimates \hat{a}_{k-1} and \hat{b}_{k-1} are used to replace the unknown plant parameters a and b respectively. Also, to have the correct sign of the adaptation gain s_k , the sign of the high frequency gain of plant b is used and the term “ $\text{sign}(\frac{b}{\hat{b}_{k-1}})$ ” is substituted “ $\frac{b}{\hat{b}_{k-1}}$ ”. \square

Since an under exciting input reference signal is considered and the goal in MRAC systems is simply to force the error, $e(k) = \bar{y}(k) - y(k)$, to converge to zero, it is not necessary that the adjustable parameter values \hat{a} , \hat{b} , should converge to the true plant parameters a , b , respectively. Rather, it is expected there is a whole set of the possible parameters (\hat{a}, \hat{b}) for which the error, $e(k) \equiv 0$. By assuming asymptotic convergence of the adaptive parameters to this set and then analysing the closed-loop difference equations, one finds the explicit equation for this set to be

$$(a + 1)\hat{b}_{k-1} - \hat{a}_{k-1}b - b = 0.$$

The line defined by the above equation is termed as *I/O behaviour line*. It should be mentioned that since $e(k) = 0$ in the steady-state with (\hat{a}, \hat{b}) parameters on the *I/O behaviour line*, the adaptation dynamics are also zero. Thus, for a single step change in the reference input, the system should settle back into steady-state behaviour with constant controller coefficient after a short transient period.

Intuitively, it is expected that the adaptation gain s_k in Eq. A.11 has a significant effect on the parameter convergence rate, i.e. parameters will converge to *I/O behaviour line* slowly for small s_k and quickly for large s_k . In practice, however, for large s_k the adaptation behaviour becomes unpredictable and for small s_k convergence becomes sluggish. This observation is not conclusive evidence of the inadequacy of the scheme under normal operating conditions, but does tend to reduce confidence in the method.

Practically, MRAC schemes are used when there is a time-scale separation between the plant dynamics and the adaptive dynamics. Thus, the non-linearities introduced by the coupling of adaptation and plant dynamics are negligible, and the gradient, Eq. A.12, maintains the properties expected in gradient descent algorithms. This requirement tends to force a choice of adaptation gain, s_k , smaller than would be desired and results in closed-loop systems with very slow transient behaviour. To overcome this difficulty, it is necessary to design the adaptation dynamics to incorporate knowledge of the true plant dynamics.

A.4 Non-Euclidean Gradient Descent Adaptation Scheme

In this section, a non-linear approach to the design of the adaptation algorithm in MRAC scheme is presented. In the proposed approach, the state of the plant is combined with the parameter adaptation dynamics to form a state for the full non-linear system. Taking the gradient of the cost function Φ , Eq. A.9, with respect to a Riemannian metric, a gradient descent algorithm on the full non-linear state is induced. At each step, the Riemannian metric is chosen so that the modelled plant dynamics do in fact match the true plant dynamics. Once the Riemannian metric is fully specified, the adaptation dynamics are uniquely defined. In this way, the adaptation dynamics induced in the adjustable parameters incorporate the knowledge of the true plant dynamics. General background on Riemannian geometry can be found in [HM94, Appendix C.10]

The full state of the system given in Fig. 1 is defined to be

$$\xi_k := \begin{pmatrix} y(k) \\ \hat{a}_k \\ \hat{b}_k \end{pmatrix}. \quad (\text{A.13})$$

Note that the state of the adjustable parameter estimates vector $\hat{\theta}_k = (\hat{a}_k, \hat{b}_k)$ in the classical MRAC scheme is \mathbb{R}^2 while the state of parameter vector ξ_k in the proposed scheme is \mathbb{R}^3 . The $y(k)$ state added in the proposed scheme is just the state of the linear system dynamics.

The cost function Φ is the same as used in Eq. A.9. However, now it is considered as a cost on the full state space $\Phi : \mathbb{R}^3 \rightarrow \mathbb{R}$,

$$\Phi(\xi_{k-1}) := \frac{1}{2}e(k)^2, \quad (\text{A.14})$$

where $e(k)$ is given by Eq. A.8 and ξ_{k-1} is the full state of the system at time instance $k - 1$.

To define the non-Euclidean gradient vector of $\Phi(\xi_{k-1})$, a Riemannian metric is introduced in \mathbb{R}^3 . A Riemannian metric is a bilinear, positive definite map for each $\xi_{k-1} \in \mathbb{R}^3$,

$$\langle\langle \cdot, \cdot \rangle\rangle_{\xi_{k-1}} : \mathbb{R}^3 \times \mathbb{R}^3 \rightarrow \mathbb{R}$$

which varies smoothly with ξ_{k-1} . For $\eta, \nu \in \mathbb{R}^3$, tangent vectors of \mathbb{R}^3 at ξ_{k-1} , then

$$\langle\langle \eta, \nu \rangle\rangle_{\xi_{k-1}} := \eta^T Q_{\xi_{k-1}}^{-1} \nu, \quad Q_{\xi_{k-1}} \in \mathbb{R}^{3 \times 3}, \quad (\text{A.15})$$

where $Q_{\xi_{k-1}} = Q_{\xi_{k-1}}^T > 0$ (positive definite) and $Q_{\xi_{k-1}}^{-1}$ denotes the inverse of $Q_{\xi_{k-1}}$.

Let $D\Phi(\xi_{k-1})$ denote the vector of partial differentials of $\Phi(\xi_{k-1})$ with respect to the full state, i.e.

$$D\Phi(\xi_{k-1}) = \begin{pmatrix} \frac{\partial \Phi(\xi_{k-1})}{\partial y^{(k-1)}} \\ \frac{\partial \Phi(\xi_{k-1})}{\partial \hat{a}_{k-1}} \\ \frac{\partial \Phi(\xi_{k-1})}{\partial \hat{b}_{k-1}} \end{pmatrix}.$$

The gradient of $\Phi(\xi_{k-1})$, denoted $\text{grad}\Phi(\xi_{k-1})$, with respect to the choice of a Riemannian metric is uniquely generated as the solution of [dC92, page 83]

$$\langle\langle X, \text{grad}\Phi(\xi_{k-1}) \rangle\rangle_{\xi_{k-1}} = X^T D\Phi(\xi_{k-1}). \quad (\text{A.16})$$

Solving Eq. A.16 yields

$$\text{grad}\Phi(\xi_{k-1}) = Q_{\xi_{k-1}} D\Phi(\xi_{k-1}). \quad (\text{A.17})$$

Note that $Q_{\xi_{k-1}}$ is dependent on the states ξ_{k-1} .

Remark A.3 As long as $Q_{\xi_{k-1}}$ is positive definite, then instantaneously, the cost $\Phi(\xi_{k-1})$ is decreasing. This becomes clearer if a continuous-time adaptation law is studied. Consider the continuous-time gradient descent adaptation law

$$\dot{\xi}(t) = -\text{grad}\Phi(\xi(t)),$$

then, the directional derivative of the cost Φ in the direction of flow of $\xi(t)$ is

$$\dot{\Phi} = \frac{d}{dt} \Phi(\xi(t)) = -D\Phi(\xi(t))^T \text{grad}\Phi = -\langle \text{grad}\Phi, \text{grad}\Phi \rangle = -\|\text{grad}\Phi\|^2 < 0. \quad (\text{A.18})$$

Thus, the control parameters $\hat{a}(t), \hat{b}(t)$, must evolve such that the cost $\Phi(\xi(t))$ is decreased. Of course, it is necessary to ensure that $y(t)$ evolves to match the true dynamics of the plant by utilising the freedom of choice in $Q_{\xi(t)}$. This is a difficult problem in itself and in this work we consider discrete-time plants and present a method of determining a suitable positive definite matrix $Q_{\xi_{k-1}}$. \square

The explicit equations for the entries of $D\Phi(\xi_{k-1})$ are computed as partial derivatives of the difference equation for $e(k)$, Eq. A.10, with respect to the $(k-1)$ 'th parameters $y(k-1)$, \hat{a}_{k-1} , \hat{b}_{k-1} , respectively. The second and the third entries of $D\Phi(\xi_{k-1})$ are obtained using Eq. A.12. For the additional state, the partial differentiation of the cost function $\Phi(\xi_{k-1})$ with respect to $y(k-1)$, $\frac{\partial\Phi(\xi_{k-1})}{\partial y(k-1)}$, is required. One has,

$$\frac{\partial\Phi(\xi_{k-1})}{\partial y(k-1)} = e(k) \frac{\partial e(k)}{\partial y(k-1)},$$

where

$$\frac{\partial e(k)}{\partial y(k-1)} = \left[\frac{a\hat{b}_{k-1}}{b} - \left(1 - \frac{\hat{b}_{k-1}}{b}\right)(1 - \gamma) \right] \frac{b}{\hat{b}_{k-1}}.$$

As described in Section A.2, the model estimates \hat{a}_{k-1} and \hat{b}_{k-1} can be substituted for the unknown plant parameters a and b respectively. Also, the sign of the high frequency gain of plant b is known, yielding

$$\frac{\partial\Phi(\xi_{k-1})}{\partial y(k-1)} = e(k) \left[\text{sign}\left(\frac{\hat{b}_{k-1}}{b}\right) \hat{a}_{k-1} - \left(1 - \frac{\hat{b}_{k-1}}{b}\right)(1 - \gamma) \right] \text{sign}\left(\frac{b}{\hat{b}_{k-1}}\right). \quad (\text{A.19})$$

The gradient descent algorithm induced by Eq. A.17 is simply

$$\xi_k = \xi_{k-1} - s_k \text{grad}\Phi(\xi_{k-1}) = \xi_{k-1} - s_k Q_{\xi_{k-1}} D\Phi(\xi_{k-1}), \quad (\text{A.20})$$

where s_k is the adaptation gain.

Note that since the only constraint on $Q_{\xi_{k-1}}$ at this stage is that $Q_{\xi_{k-1}}$ is positive definite, then, without loss of generality one can write

$$s_k Q_{\xi_{k-1}} \equiv Q_{\xi_{k-1}}.$$

The gradient descent algorithm becomes

$$\xi_k = \xi_{k-1} - Q_{\xi_{k-1}} D\Phi(\xi_{k-1}), \quad (\text{A.21})$$

with unit step size.

Remark A.4 One of the advantages of the proposed scheme is the fact that the adaptation gain size no longer plays a role in the scheme. This is significant because, as mentioned in

Section A.3, the adaptation gain s_k affects the convergence rate of the adjustable parameters and in turn, the stability of the system. Instead of guessing the size of s_k to keep the system stable, the adaptation speed is automatically considered in the calculation procedure of the positive definite matrix $Q_{\xi_{k-1}}$ by incorporating the true plant dynamics. This guarantees fast convergence. \square

Remark A.5 A disadvantage of the proposed scheme (cf. Eq. A.21) is that there is no *a priori* guarantee that the cost $\Phi(\xi_{k+1}) < \Phi(\xi_k)$. Recalling that for the continuous-time adaptation rule, Eq. A.18, one has $\dot{\Phi} < 0$. For s_k sufficiently small in Eq. A.20, an equivalent result should hold. However, the necessity of subsuming s_k into $Q_{\xi_{k-1}}$ leads to potential stability problems. Understanding this issue is an area of ongoing research. \square

Consider the dynamics induced in the state $y(k)$ by Eq. A.21. These dynamics can be written

$$y(k) = y(k-1) - E_1^T Q_{\xi_{k-1}} D\Phi(\xi_{k-1}), \quad (\text{A.22})$$

where $E_1 \in \mathbb{R}^3$ is the unit vector with a one in the first entry. Given that $y(k)$ and $y(k-1)$ are measured directly from the plant output, then Eq. A.22 generates a linear constraint on $Q_{\xi_{k-1}}$

$$E_1^T Q_{\xi_{k-1}} D\Phi(\xi_{k-1}) = y(k-1) - y(k). \quad (\text{A.23})$$

As long as $Q_{\xi_{k-1}}$ satisfies Eq. A.23, the first entry of the induced gradient dynamics in Eq. A.21 exactly replicate the true plant dynamics.

This leads to an optimisation problem that lies at the heart of the proposed scheme.

Problem A.1 At each time instance k , find a matrix $Q_{\xi_{k-1}}$ which depends smoothly on ξ_{k-1} , satisfying

1. $Q_{\xi_{k-1}} > 0$ and $Q_{\xi_{k-1}} = Q_{\xi_{k-1}}^T$ (Positive definite).
2. $E_1^T Q_{\xi_{k-1}} D\Phi(\xi_{k-1}) = y(k-1) - y(k)$ (Linear constraint)

and such that the closed-loop system shows desirable behaviour. \square

Remark A.6 Requirements 1 and 2 of Problem A.1 are the practical requirements that ensure the gradient descent algorithm replicates the true plant dynamics and displays gradient characteristics. These constraints, however, leave a great deal of leeway in choosing $Q_{\xi_{k-1}}$ to ensure the closed-loop system shows desirable behaviour. \square

A.5 Determination of an Optimal Positive Definite Matrix

In this section, a specific approach to solve the optimisation problem, Problem A.1, is presented. The approach relies on choosing $Q_{\xi_{k-1}}$ to minimise a one-step-ahead estimate of the cost Φ of Eq. A.14.

A natural approach to finding $Q_{\xi_{k-1}}$ is to generate a one-step-ahead estimate of the output, $\hat{y}^e(k+1)$, based on a particular $Q_{\xi_{k-1}}$ and then minimise the cost $\|\hat{y}^e(k+1) - \bar{y}(k+1)\|^2$ subject to the requirements of Problem A.1. Ensuring that $Q_{\xi_{k-1}}$ satisfies Eq. A.23 should not be difficult as this is simply a linear constraint on symmetric matrix space. Dealing with the positive definite constraint forces one into the realm of semi-definite programming. Following the lead of recent development of semi-definite programming [VB96], we introduce the cost function

$$\Psi_{\xi_{k-1}}^\epsilon(Q_{\xi_{k-1}}^e) = \|\hat{y}^e(k+1) - \bar{y}(k+1)\|^2 - \epsilon \ln(\det(Q_{\xi_{k-1}}^e)), \quad (\text{A.24})$$

where $\hat{y}^e(k+1)$ is one-step-ahead estimation of output $y(k+1)$ and $Q_{\xi_{k-1}}^e$ is the estimate of $Q_{\xi_{k-1}}$. Here, $\det(Q_{\xi_{k-1}}^e)$ is the determinant of the matrix $Q_{\xi_{k-1}}^e$ and $\bar{y}(k+1)$ is the output of the reference model which can be easily obtained from Eq. A.3.

Remark A.7 The first term in Eq. A.24 is the desired quadratic cost term while the second term is a self-concordant barrier function [NN94][VB96] added to ensure the minimum of $\Psi_{\xi_{k-1}}^\epsilon(Q_{\xi_{k-1}}^e)$ always lies in the set of positive definite matrices. By choosing ϵ sufficiently small, the influence of the barrier function on the quadratic cost function is negligible except in the neighbourhood of the boundary of positive definite matrices. \square

When $Q_{\xi_{k-1}}$ needs to be determined, the current and the previous value of output $y(k)$, $y(k-1)$, reference model output $\bar{y}(k)$, $\bar{y}(k-1)$, input $r(k)$, $r(k-1)$, and error $e(k)$ as well as

the previous values of control action $u(k-1)$ and the adjustable parameter vector $\hat{a}_{k-1}, \hat{b}_{k-1}$ are known. In addition, the derivative of the cost, $D\Phi(\xi_{k-1})$, is also available.

The one-step-ahead output estimation $\hat{y}^e(k+1)$ is obtained using the one-step-ahead estimation of control action to the system, $\hat{u}^e(k)$,

$$\hat{y}^e(k+1) = \hat{b}_k^e(\hat{u}^e(k) + r(k)) - \hat{a}_k^e y(k). \quad (\text{A.25})$$

The control action $\hat{u}^e(k)$ is generated from the difference equation for the control, Eq. A.7,

$$\hat{u}^e(k) = \gamma u(k-1) + \frac{(1-\gamma)}{\bar{b}\hat{b}_k^e} \left[(\bar{b} - \hat{b}_k^e)y(k) + (\hat{a}_k^e \bar{b} - \bar{a}\hat{b}_k^e)y(k-1) \right], \quad (\text{A.26})$$

where \hat{a}_k^e and \hat{b}_k^e are the one-step-ahead estimates of model parameters based on the gradient descent algorithm generated from the Riemannian metric given by the matrix $Q_{\xi_{k-1}}^e > 0$. Equation A.21 yields

$$\hat{a}_k^e = \hat{a}_{k-1} - E_2^T Q_{\xi_{k-1}}^e D\Phi(\xi_{k-1}), \quad (\text{A.27})$$

and

$$\hat{b}_k^e = \hat{b}_{k-1} - E_3^T Q_{\xi_{k-1}}^e D\Phi(\xi_{k-1}), \quad (\text{A.28})$$

where E_2 and E_3 are the unit vectors with a one in the second and third entries respectively.

To solve Problem A.1, we proceed by deriving a gradient descent algorithm on the set of positive definite matrices satisfying the constraints of the Problem A.1. To find the gradient of $\Psi_{\xi_{k-1}}^e(Q_{\xi_{k-1}}^e)$, consider the Euclidean metric on the positive definite matrices,

$$\langle U, V \rangle = \text{tr}(U^T V),$$

where $U = U^T, V = V^T, U, V \in \mathbb{R}^{n \times n}$. This is just the metric on the positive definite matrices inherit as an open set of symmetric matrix space.

Using this metric along with the definition of gradient, Eq. A.16, yields

$$\langle X, \text{grad}\Psi_{\xi_{k-1}}^e(Q_{\xi_{k-1}}^e) \rangle = \text{tr}(X^T \text{grad}\Psi_{\xi_{k-1}}^e(Q_{\xi_{k-1}}^e)) = D\Psi_{\xi_{k-1}}^e \Big|_{Q_{\xi_{k-1}}^e} [X], \quad (\text{A.29})$$

where $D\Psi_{\xi_{k-1}}^e|_{Q_{\xi_{k-1}}^e} [X]$ is the directional derivatives of $\Psi_{\xi_{k-1}}^e$ in direction X evaluated at the point $Q_{\xi_{k-1}}^e$. This is given by¹

$$D\Psi_{\xi_{k-1}}^e|_{Q_{\xi_{k-1}}^e} [X] = 2 (\hat{y}^e(k+1) - \bar{y}(k+1)) D(\hat{y}^e(k+1) - \bar{y}(k+1))|_{Q_{\xi_{k-1}}^e} [X] \\ - \epsilon \text{tr} \left(X^T (Q_{\xi_{k-1}}^e)^{-1} \right),$$

or as only $\hat{y}^e(k+1)$ contains $Q_{\xi_{k-1}}^e$,

$$D\Psi_{\xi_{k-1}}^e|_{Q_{\xi_{k-1}}^e} [X] = 2 (\hat{y}^e(k+1) - \bar{y}(k+1)) D\hat{y}^e(k+1)|_{Q_{\xi_{k-1}}^e} [X] \\ - \epsilon \text{tr} \left(X^T (Q_{\xi_{k-1}}^e)^{-1} \right). \quad (\text{A.30})$$

Rewriting $\hat{y}^e(k+1)$ in terms of \hat{a}_k^e and \hat{b}_k^e , one has an explicit form for

$D\hat{y}^e(k+1)|_{Q_{\xi_{k-1}}^e} [X]$ as follows

$$D\hat{y}^e(k+1)|_{Q_{\xi_{k-1}}^e} [X] = D \left[\left\{ \hat{b}_k^e (\gamma u(k-1) + r(k)) + \frac{(1-\gamma)}{\bar{b}} \{ (\bar{b} - \hat{b}_k^e) y(k) \right. \right. \\ \left. \left. + (\hat{a}_k^e \bar{b} - \bar{a} \hat{b}_k^e) y(k-1) \right\} - \hat{a}_k^e y(k) \right] \Big|_{Q_{\xi_{k-1}}^e} [X].$$

As shown in Eq.'s A.27 and A.28, \hat{a}_k^e and \hat{b}_k^e are the only terms dependent on $Q_{\xi_{k-1}}^e$. Thus,

$$D\hat{y}^e(k+1)|_{Q_{\xi_{k-1}}^e} [X] = -E_3^T X D\Phi(\xi_{k-1}) \{ \gamma u(k-1) + r(k) \} \\ + \frac{(1-\gamma)}{\bar{b}} \left\{ E_3^T X D\Phi(\xi_{k-1}) y(k) + \{ -\bar{b} E_2^T X D\Phi(\xi_{k-1}) \right. \\ \left. + \bar{a} E_3^T X D\Phi(\xi_{k-1}) \} y(k-1) \right\} + E_2^T X D\Phi(\xi_{k-1}) y(k),$$

or,

$$D\hat{y}^e(k+1)|_{Q_{\xi_{k-1}}^e} [X] = \frac{1}{2} \text{tr} \left\{ X^T \{ D\Phi(\xi_{k-1}) E_3^T + E_3 D\Phi(\xi_{k-1})^T \} \right\} \left\{ -\{ \gamma u(k-1) \right. \\ \left. + r(k) \} + \frac{(1-\gamma)}{\bar{b}} \{ y(k) + \bar{a} y(k-1) \} \right\} + \frac{1}{2} \text{tr} \left\{ X^T \{ D\Phi(\xi_{k-1}) E_2^T \right. \\ \left. + E_2 D\Phi(\xi_{k-1})^T \} \right\} \left\{ y(k) - (1-\gamma) y(k-1) \right\},$$

¹ $D \ln(\det(Q_{\xi_{k-1}}^e)) |_{Q_{\xi_{k-1}}^e} [X] = \text{tr}(X^T (Q_{\xi_{k-1}}^e)^{-1})$ [VB96, Page 70].

where $D\Phi(\xi_{k-1})^T$ is the transpose of $D\Phi(\xi_{k-1})$.

By substituting the above equation into Eq. A.30 and comparing with Eq. A.29, the gradient of the cost function is obtained explicitly,

$$\begin{aligned} \text{grad}\Psi_{\xi_{k-1}}^\epsilon(Q_{\xi_{k-1}}^\epsilon) &= \left\{ \hat{y}^e(k+1) - \bar{y}(k+1) \right\} \left\{ \left[D\Phi(\xi_{k-1})E_3^T + E_3D\Phi(\xi_{k-1})^T \right] \right. \\ &\quad \left[-\{\gamma u(k-1) + r(k)\} + \frac{(1-\gamma)}{\bar{b}}\{y(k) + \bar{a}y(k-1)\} \right] + \left[D\Phi(\xi_{k-1})E_2^T \right. \\ &\quad \left. \left. + E_2D\Phi(\xi_{k-1})^T \right] \left[y(k) - (1-\gamma)y(k-1) \right] \right\} - \epsilon(Q_{\xi_{k-1}}^\epsilon)^{-1}. \end{aligned} \quad (\text{A.31})$$

The gradient, $\text{grad}\Psi_{\xi_{k-1}}^\epsilon(Q_{\xi_{k-1}}^\epsilon)$, gives the optimum descent direction of the cost $\Psi_{\xi_{k-1}}^\epsilon(Q_{\xi_{k-1}}^\epsilon)$ on the set of symmetric matrices. Due to the barrier function, a steepest descent algorithm based on a descent direction, $\text{grad}\Psi_{\xi_{k-1}}^\epsilon(Q_{\xi_{k-1}}^\epsilon)$, and initialised with a positive definite matrix will never evolve outside the set of positive definite matrices. It is now necessary to ensure that the gradient descent direction also satisfies the linear constraint, Eq. A.23. This is achieved by projecting $\text{grad}\Psi_{\xi_{k-1}}^\epsilon(Q_{\xi_{k-1}}^\epsilon)$ orthogonally onto the tangent space of the linear space, $\mathcal{L}(\xi_{k-1}) = \{Q_{\xi_{k-1}}^\epsilon \mid E_1^T Q_{\xi_{k-1}}^\epsilon D\Phi(\xi_{k-1}) = y(k-1) - y(k)\}$, to generate a constrained descent direction satisfying Eq. A.23.

The tangent space of the linear space $\mathcal{L}(\xi_{k-1})$ is

$$\begin{aligned} \mathcal{L}_0(\xi_{k-1}) &= \mathbf{T}_{Q_{\xi_{k-1}}^\epsilon} \mathcal{L}(\xi_{k-1}) \\ &= \{X \mid E_1^T X D\Phi(\xi_{k-1}) = 0\}. \end{aligned} \quad (\text{A.32})$$

Taking the projection of $\text{grad}\Psi_{\xi_{k-1}}^\epsilon(Q_{\xi_{k-1}}^\epsilon)$ onto $\mathbf{T}_{Q_{\xi_{k-1}}^\epsilon} \mathcal{L}(\xi_{k-1})$, one generates a constrained descent direction satisfying Eq. A.23. The projection operator is denoted

$$\mathbb{P}_{\mathcal{L}_0(\xi_{k-1})} : \mathbb{R}^{3 \times 3} \longrightarrow \mathbf{T}_{Q_{\xi_{k-1}}^\epsilon} \mathcal{L}(\xi_{k-1}),$$

and an explicit form for $\mathbb{P}_{\mathcal{L}_0(\xi_{k-1})}$ can be computed.

The final result required for an effective optimisation procedure to solve Problem A.1 is an initial condition satisfying the constraints. An explicit method for calculating such a matrix is given in Appendix B.

To conclude this section, we give the optimisation procedure used to generate $Q_{\xi_{k-1}}$.

Algorithm A.1 *Optimisation algorithm to determine $Q_{\xi_{k-1}}$.*

- 0 :** *Input the known output $y(k)$, $y(k-1)$, reference model output $\bar{y}(k)$, $\bar{y}(k-1)$, input $r(k)$, $r(k-1)$, error $e(k)$, control $u(k-1)$, and parameter estimates $\hat{\theta}_{k-1} = (\hat{a}_{k-1}, \hat{b}_{k-1})$ as well as the derivative of the cost $D\Phi(\xi_{k-1})$ to the algorithm.*
- 1 :** *Generate the initial positive definite matrix $Q_{\xi_{k-1}}^e(0)$ satisfying the linear constraint Eq. A.23 (See Appendix B.)*
- 2 :** *Let $\epsilon = 1$ in Eq. A.24 and set $j = 0$.*
- 3 :** *Compute $\text{grad}\Psi_{\xi_{k-1}}^\epsilon$ using Eq. A.31 and the projection $\mathbb{P}_{\mathcal{L}_0(\xi_{k-1})}(\text{grad}\Psi_{\xi_{k-1}}^\epsilon)$. (The projection is taken to ensure the linear constraint (Eq. A.23) is satisfied.)*
- 4 :** *Compute $\alpha_j := \min_{\alpha>0} \Psi_{\xi_{k-1}}^\epsilon \{Q_{\xi_{k-1}}^e(j) - \alpha \mathbb{P}_{\mathcal{L}_0(\xi_{k-1})}(\text{grad}\Psi_{\xi_{k-1}}^\epsilon)\}$. (Use the MATLAB optimisation toolbox.)*
- 5 :** *Set $Q_{\xi_{k-1}}^e(j+1) = Q_{\xi_{k-1}}^e(j) - \alpha_j \mathbb{P}_{\mathcal{L}_0(\xi_{k-1})}(\text{grad}\Psi_{\xi_{k-1}}^\epsilon)$.*
- 6 :** *Compute the projection $Q_{\xi_{k-1}}^e(j+1) = \mathbb{P}_{\mathcal{L}(\xi_{k-1})} Q_{\xi_{k-1}}^e(j+1)$, the orthogonal projection onto $\mathcal{L}(\xi_{k-1})$ to compensate numerical error.*
- 7 :** *If j is divisible by 5, let $\epsilon = 10^{-1}\epsilon$.*
- 8 :** *If $\epsilon = 10^{-8}$ goto 9. Else $j = j + 1$, goto 3.*
- 9 :** *$Q_{\xi_{k-1}} = Q_{\xi_{k-1}}^e(j+1)$; This matrix always satisfies the requirements of Problem A.1.*

□

Remark A.8 Note that the computational cost of calculating $Q_{\xi_{k-1}}$ is significant. Simulations indicate that this calculation can be achieved in 2-3 seconds for the plant considered². It is expected that by improving the computational efficiency of the optimisation algorithm, this computational cost can be significantly reduced. As a consequence, the authors expect that the proposed method should be applicable to most process control problems. In contrast, the computational complexity is likely to rule out applications in telecommunications. □

²Calculations were done using the MATLAB optimisation toolbox on a Sun UltraSPARC I machine, clock speed 167 MHz.

A.6 Simulation Results and Discussion

In this section, the results of simulations of both the classical and proposed model reference adaptive control systems are presented. The simulation indicates several desirable features of the proposed algorithm, though there are still unanswered questions in the application of this method, such as developing a full understanding of its stability properties. Simulations of numerous examples have shown, however, that the proposed scheme displays good convergence properties.

For this simulation, the plant and reference model transfer functions were chosen to be

$$G(z) = \frac{-0.8z^{-1}}{1-0.3z^{-1}}, \quad \bar{G}(z) = \frac{-0.8z^{-1}}{1+0.6z^{-1}}.$$

The initial condition for the model parameter estimates was chosen to be $(\hat{a}, \hat{b}) = (0.4, -0.2)$, leading to an estimated transfer function;

$$\hat{G}(z) = \frac{-0.2z^{-1}}{1+0.4z^{-1}}.$$

As mentioned before, the choice of adaptation gain for the classical MRAC system is important. The tradeoff between the system stability and the parameter convergence speed was examined for various gain values. After several trials, a fixed adaptation gain $s_k = 0.2$ for all time instance k has been chosen for this example. In both schemes, the cost function $\Phi = \frac{1}{2}e(k)^2$ is used. The value of γ in the low-pass filter, Eq. A.6, was chosen to be 0.9.

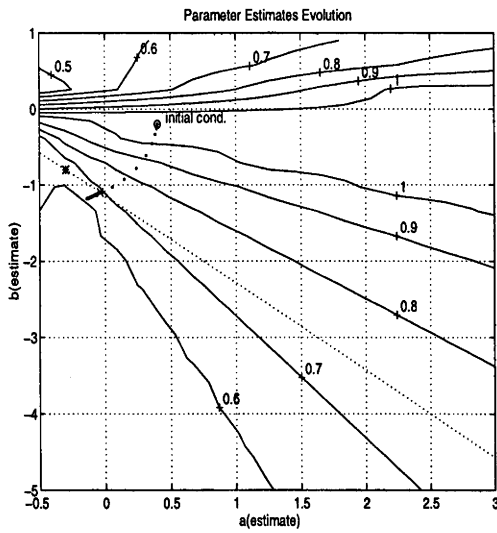
A step input was used as the reference command input. Because of this, the parameter estimates need not to approach their true plant values, but should converge to the *I/O behaviour line*.

The simulation results are presented in Figures A.2 and A.3. In the figures of parameter estimates evolution, Figures A.2, the dotted line is the *I/O behaviour line* and the contour lines are closed-loop system stability measure lines. These contour lines enclose regions in parameter space where the largest absolute value of a pole of the closed-loop system is less than or equal to a marked contour value. In both the classical and proposed schemes, the convergence to the *I/O behaviour line* seems acceptable. However, the relative difference in performance is clearly shown in Figures A.3. Here, the log plot of error versus time is given and the extremely rapid asymptotic convergence of the proposed scheme is displayed. In the both schemes, the

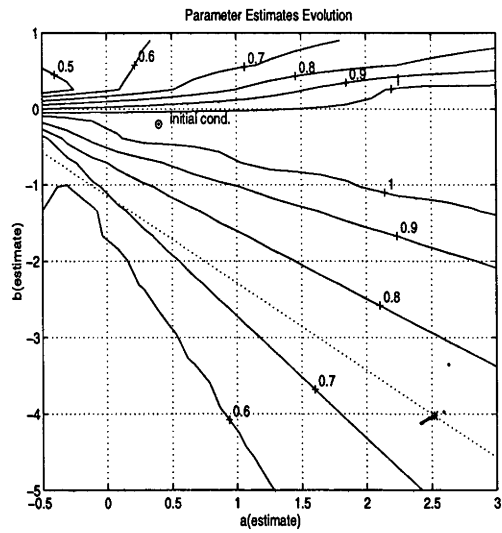
adaptation scheme is only initiated at time $k=4$ to avoid initialisation difficulties. The initial condition is chosen in an unstable region of parameter space and consequently, a short transient is observed in both schemes. Observe in Figure A.3(b) that immediately the parameter adjustment is initiated, the error is stabilised and quickly decreased. Conversely, in the classical MRAC scheme the small adaptation gain restricts the rate of adaptation, leading to a larger transient before convergence. Moreover, the adaptation gain also limits the asymptotic rate of the convergence. Note that increasing the magnitude of the adaptation gain for the classical scheme leads to stability problems.

A.7 Conclusion

In this appendix, we have developed a new non-linear approach to the design of adaptive control schemes based around the use of a non-Euclidean gradient descent algorithm with respect to a Riemannian metric. It is shown that how a Riemannian metric can be chosen so that the modelled plant dynamics do in fact match the true plant dynamics. Simulations show that the proposed scheme offers faster asymptotic convergence of parameters and more flexibility in the transient response of the closed-loop system. The key contributions are; the formulation of the gradient descent algorithm in such a way as to incorporate the true plant dynamics and the development of a criterion and method to determine suitable positive definite matrices for the adaptation mechanism. Simulations have shown that the parameter convergence of the proposed scheme is much faster than the classical MRAC scheme. Further work is required to investigate stability issues as well as the generalisation to continuous-time plants and more general system models.

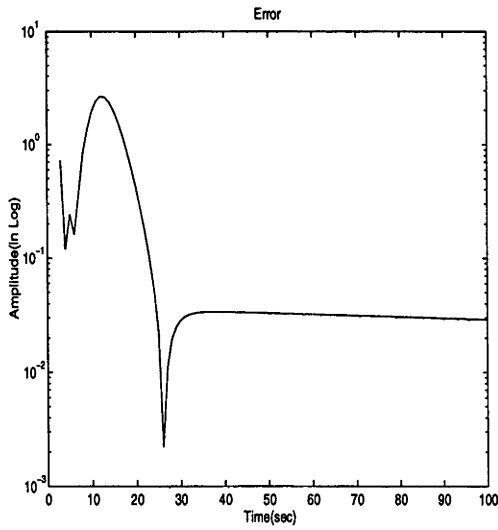


(a) the classical MRAC scheme

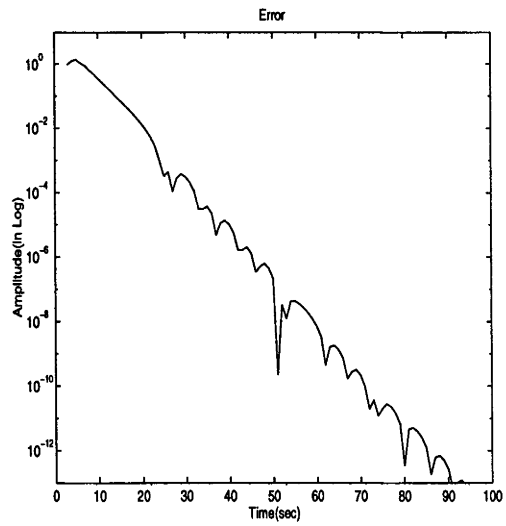


(b) the proposed MRAC scheme

Figure A.2: Parameter estimates evolution plot



(a) the classical MRAC scheme



(b) the proposed MRAC scheme

Figure A.3: Error($\bar{y} - y$) plot

Appendix B

Finding a feasible initial positive definite matrix

In this appendix, a systematic method for a calculating an initial positive definite matrix satisfying the constraints in Problem A.1 is presented. This matrix is then used as the input for the Algorithm A.1 in Section A.5.

A positive definite matrix can always be factored into the form

$$Q_{\xi_{k-1}}(0) = P_{\xi_{k-1}} P_{\xi_{k-1}}^T, \quad (\text{B.1})$$

where $Q_{\xi_{k-1}}(0) = (Q_{\xi_{k-1}}(0))^T > 0$ is an initial positive matrix required for Algorithm A.1 and $P_{\xi_{k-1}} \in GL(n)$ is a square root of $Q_{\xi_{k-1}}(0)$.

Substituting Eq. B.1 into the linear constraint Eq. A.23, one has

$$E_1^T P_{\xi_{k-1}} P_{\xi_{k-1}}^T D\Phi(\xi_{k-1}) = y(k-1) - y(k).$$

Assume that $D\Phi(\xi_{k-1}) \neq 0$ (that is $e(k) \neq 0$), and define

$$g(k) = \frac{D\Phi(\xi_{k-1})}{\|D\Phi(\xi_{k-1})\|} \quad \text{and} \quad t(k) = \frac{y(k-1) - y(k)}{\|D\Phi(\xi_{k-1})\|}.$$

Then, one has

$$E_1^T P_{\xi_{k-1}} P_{\xi_{k-1}}^T g(k) = t(k), \quad (\text{B.2})$$

where $g(k)$ and E_1 are both the unit length vectors, and all the scaling information is contained in $t(k)$.

The development proceeds by thinking of $P_{\xi_{k-1}}^T$ as a transformation on $\mathbb{R}^{3 \times 3}$. Using the vector inner product $\langle u, v \rangle = u^T v$, where $u, v \in \mathbb{R}^n$, one has from Eq. B.2

$$\langle P_{\xi_{k-1}}^T E_1, P_{\xi_{k-1}}^T g(k) \rangle = t(k). \quad (\text{B.3})$$

Consider the 2-dimensional subspace in \mathbb{R}^3 given by $\text{sp}\{E_1, g(k)\}$. An orthonormal basis for this subspace is provided by the vectors

$$W_1 = \frac{E_1 + g(k)}{\|E_1 + g(k)\|} \quad \text{and} \quad W_2 = \frac{E_1 - g(k)}{\|E_1 - g(k)\|} \in \mathbb{R}^3. \quad (\text{B.4})$$

Expressing E_1 and $g(k)$ as 2-dimensional vectors in this subspace, written in terms of the coordinates induced by the orthonormal vectors, W_1 and W_2 , one has

$$\tilde{E}_1 = \begin{pmatrix} W_1^T \\ W_2^T \end{pmatrix} E_1 \text{ and } \tilde{g}(k) = \begin{pmatrix} W_1^T \\ W_2^T \end{pmatrix} g(k) \in \mathbb{R}^2. \quad (\text{B.5})$$

Here $\|\tilde{E}_1\| = 1$ and $\|\tilde{g}(k)\| = 1$ since E_1 and $g(k)$ lie in the span of W_1 and W_2 .

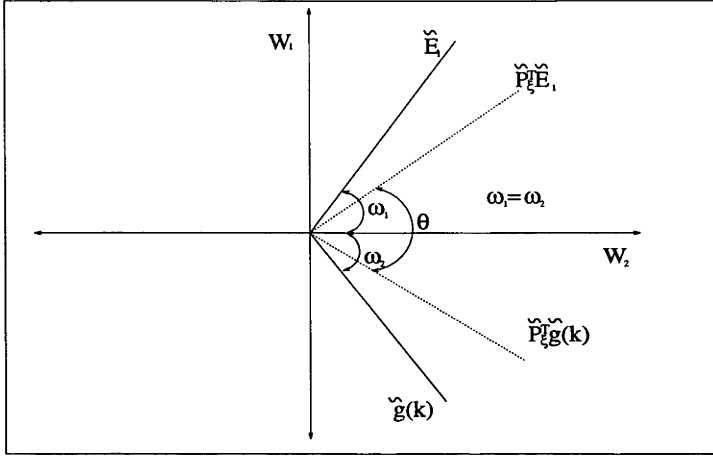
Note that the two coordinate vectors W_1 and W_2 are chosen such that the vectors \tilde{E}_1 and $\tilde{g}(k)$ always lie symmetrically about the W_2 axis in the right half plane of the 2-D subspace, spanned by W_1 and W_2 (see Figure B.1). To preserve the intuition provided by this construction, it is necessary to be careful about the sign of $t(k)$ in Eq. B.3.

1. **case of $t(k) \geq 0$:** In this case, the two vectors \tilde{E}_1 and $\tilde{g}(k)$ are in the right half plane. If the angle between two vectors \tilde{E}_1 and $\tilde{g}(k)$ is larger than a set value (in this case $\pi/3$) and less than π , a scaling matrix $\tilde{P}_{\xi_{k-1}}$ is chosen to premultiply \tilde{E}_1 and $\tilde{g}(k)$ which acts to reduce the angle. In the case that the angle is less than the set value, it is sufficient to choose $\tilde{P}_{\xi_{k-1}} = \mathbb{I}_2 \in \mathbb{R}^{2 \times 2}$. This is the case that Eq.B.2 can be satisfied by adjusting a scaling parameter δ_k . (cf. Eq. B.16).
2. **case of $t(k) < 0$:** In this case, the intuition of reducing the angle between \tilde{E}_1 and $\tilde{g}(k)$ is not valid. Take $E_1 = -E_1$ and $t(k) = -t(k)$. This leaves Eq. B.3 valid, and returns the analysis to that discussed in case 1.

Remark B.1 The choice of $\tilde{P}_{\xi_{k-1}}$ to premultiply \tilde{E}_1 and $\tilde{g}(k)$ in case 1 can be thought of as tweaking the direction of gradient flow slightly by adjusting the metric to make sure that the sign of $\langle P_{\xi_{k-1}}^T E_1, P_{\xi_{k-1}}^T g(k) \rangle$ is the same as the sign of $t(k)$. Once this is the case, then ensuring that Eq. B.3 is satisfied can be achieved by scaling $P_{\xi_{k-1}}$. \square

Remark B.2 To complete the construction indicated above, it is necessary that

$\tilde{E}_1 \neq \pm \tilde{g}(k)$, or equivalently the angle between \tilde{E}_1 and $\tilde{g}(k)$ is neither 0 nor π . In practice, this seems never to occur. In the case that such a situation occur, the adaptation would be frozen for that time instance. \square

Figure B.1: Subspace given by $\text{sp}\{E_1, g(k)\}$

The matrix $\tilde{P}_{\tilde{\xi}_{k-1}} \in \mathbb{R}^{2 \times 2}$ applied case 1 above is simply

$$\tilde{P}_{\tilde{\xi}_{k-1}} = \begin{pmatrix} 1 + \beta & 0 \\ 0 & 1 \end{pmatrix}, \text{ where } \beta > -1. \quad (\text{B.6})$$

To determine the unknown β , consider computing the angle between $\tilde{P}_{\tilde{\xi}_{k-1}}^T \tilde{E}_1$ and $\tilde{P}_{\tilde{\xi}_{k-1}}^T \tilde{g}(k)$ via the equation

$$\langle \tilde{P}_{\tilde{\xi}_{k-1}}^T \tilde{E}_1, \tilde{P}_{\tilde{\xi}_{k-1}}^T \tilde{g}(k) \rangle = \cos(\theta) \|\tilde{P}_{\tilde{\xi}_{k-1}}^T \tilde{E}_1\| \|\tilde{P}_{\tilde{\xi}_{k-1}}^T \tilde{g}(k)\|. \quad (\text{B.7})$$

Remark B.3 In this development, we do not explicitly fix θ , the desired angle between \tilde{E}_1 and $\tilde{g}(k)$. In practice, we have been choosing $\theta = \pi/3$. \square

Observe that the symmetry of the construction (cf. Figure B.1) yields

$$\|\tilde{P}_{\tilde{\xi}_{k-1}}^T \tilde{E}_1\| = \|\tilde{P}_{\tilde{\xi}_{k-1}}^T \tilde{g}(k)\|. \quad (\text{B.8})$$

Using this along with the symmetry of $P_{\tilde{\xi}_{k-1}}$, one has from Eq. B.7

$$\tilde{E}_1^T \tilde{P}_{\tilde{\xi}_{k-1}}^2 \tilde{g}(k) = \cos(\theta) \tilde{E}_1^T \tilde{P}_{\tilde{\xi}_{k-1}}^2 \tilde{E}_1, \quad (\text{B.9})$$

or

$$\tilde{E}_1^T \tilde{P}_{\tilde{\xi}_{k-1}}^2 [\tilde{g}(k) - \tilde{E}_1 \cos(\theta)] = 0. \quad (\text{B.10})$$

To simplify the computation, let

$$\tilde{E}_1 = \begin{pmatrix} s_1 \\ s_2 \end{pmatrix} \text{ and } \tilde{g}(k) = \begin{pmatrix} s_1 \\ -s_2 \end{pmatrix}. \quad (\text{B.11})$$

Substituting Eq. B.6 and Eq. B.11 into Eq. B.10 yields

$$\begin{pmatrix} s_1 & s_2 \end{pmatrix} \begin{pmatrix} (1+\beta)^2 & 0 \\ 0 & 1 \end{pmatrix} \left[\begin{pmatrix} s_1 \\ -s_2 \end{pmatrix} - \begin{pmatrix} s_1 \\ s_2 \end{pmatrix} \cos(\theta) \right] = 0,$$

or

$$s_1[1 + 2\beta + \beta^2][s_1 - s_1 \cos(\theta)] + s_2[-s_2 - s_2 \cos(\theta)] = 0.$$

Solving for β yields

$$\beta = -1 \pm \left\| \frac{s_2}{s_1} \right\| \sqrt{\frac{1 + \cos(\theta)}{1 - \cos(\theta)}}, \quad (\text{B.12})$$

where the argument inside the square root can not be negative because of triangular relationships.

Observe that s_1 and s_2 are the orthogonal projections of E_1 onto the axes W_1 and W_2 .

Thus, one has from Eq. B.5 and Eq. B.11,

$$s_1 = W_1^T E_1 = \langle W_1, E_1 \rangle = \frac{[E_1^T + g^T(k)]E_1}{\|E_1 + g(k)\|} = \frac{1 + \langle E_1, g(k) \rangle}{\|E_1 + g(k)\|}$$

and

$$s_2 = W_2^T E_1 = \langle W_2, E_1 \rangle = \frac{[E_1^T - g^T(k)]E_1}{\|E_1 - g(k)\|} = \frac{1 - \langle E_1, g(k) \rangle}{\|E_1 - g(k)\|}.$$

Also, it is easily verified that

$$\|E_1 + g(k)\|^2 = \|E_1\|^2 + \|g(k)\|^2 + 2E_1^T g(k) = 2(1 + \langle E_1, g(k) \rangle),$$

and

$$\|E_1 - g(k)\|^2 = \|E_1\|^2 + \|g(k)\|^2 - 2E_1^T g(k) = 2(1 - \langle E_1, g(k) \rangle).$$

Therefore, from the above equations, the explicit expression for s_1 and s_2 are

$$s_1 = \frac{1}{2}\|E_1 + g(k)\| \text{ and } s_2 = \frac{1}{2}\|E_1 - g(k)\|. \quad (\text{B.13})$$

Choosing the positive square root of Eq. B.12 (so that $\beta > -1$) yields an explicit value for β

$$\beta = -1 + \frac{\|E_1 - g(k)\|}{\|E_1 + g(k)\|} \sqrt{\left[\frac{1 + \cos(\theta)}{1 - \cos(\theta)} \right]}. \quad (\text{B.14})$$

Thus, $\tilde{P}_{\xi_{k-1}} \in \mathbb{R}^{2 \times 2}$ of Eq. B.6 is uniquely defined.

The matrix $P_{\xi_{k-1}} \in \mathbb{R}^{3 \times 3}$ is now defined by

$$P_{\xi_{k-1}} = \delta_k \left[\mathbb{I}_3 - (W_1 \ W_2) \begin{pmatrix} W_1^T \\ W_2^T \end{pmatrix} + (W_1 \ W_2) \tilde{P}_{\xi_{k-1}} \begin{pmatrix} W_1^T \\ W_2^T \end{pmatrix} \right], \quad (\text{B.15})$$

where $\mathbb{I}_3 \in \mathbb{R}^{3 \times 3}$ is the identity matrix and δ_k is the scaling factor, $\delta_k > 0 \in \mathbb{R}$.

By construction, the matrix $P_{\xi_{k-1}}$ given by Eq. B.15 should satisfy Eq. B.2. Observe that when $P_{\xi_{k-1}}$ is substituted into Eq. B.2 only the last term contributes to the inner product. Thus, Eq. B.2 becomes

$$\delta_k^2 E_1^T (W_1 \ W_2) \tilde{P}_{\xi_{k-1}} \begin{pmatrix} W_1^T \\ W_2^T \end{pmatrix} g(k) = t(k),$$

or,

$$\delta_k^2 \langle P'_{\xi_{k-1}} E_1, P'_{\xi_{k-1}} g(k) \rangle = t(k),$$

where

$$P'_{\xi_{k-1}} = (W_1 \ W_2) \tilde{P}_{\xi_{k-1}} \begin{pmatrix} W_1^T \\ W_2^T \end{pmatrix}.$$

Finally, solving for the scaling factor $\delta(k)$, one has

$$\delta_k = \sqrt{\frac{t(k)}{\langle P'_{\xi_{k-1}} E_1, P'_{\xi_{k-1}} g(k) \rangle}}, \quad (\text{B.16})$$

where the argument of the square root is strictly positive since $\langle P'_{\xi_{k-1}} E_1, P'_{\xi_{k-1}} g(k) \rangle > 0$ by construction and $t(k) > 0$ by choice (cf. case 2).

Thus, an initial positive definite matrix $Q_{\xi_{k-1}}(0)$ for input into the optimisation procedure, Algorithm A.1, is given by

$$Q_{\xi_{k-1}}(0) = P_{\xi_{k-1}} P_{\xi_{k-1}}^T = \delta_k^2 P_{\xi_{k-1}}^*. \quad (\text{B.17})$$

where

$$P_{\xi_{k-1}}^* = \mathbb{I}_3 - (W_1 \ W_2) \begin{pmatrix} W_1^T \\ W_2^T \end{pmatrix} + (W_1 \ W_2) \tilde{P}_{\xi_{k-1}} \begin{pmatrix} W_1^T \\ W_2^T \end{pmatrix}.$$