2007

Supervisory control of discrete event systems for bisimulation or simulation equivalence

Changyan Zhou
Iowa State University

Supervisory control of discrete event systems for bisimulation or simulation equivalence

by

Changyan Zhou

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Electrical Engineering

Program of Study Committee:
Ratnesh Kumar, Major Professor
Murti V. Salapaka
Nicola Elia
Andrew S. Miner
Simanta Mitra

Iowa State University

Ames, Iowa

2007

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

My deepest thanks and gratitude are due to my advisor Dr. Kumar for his guidance, inspiration and patience. Dr. Kumar's unlimited enthusiasm for research has made it a pleasure to work with him. Working with him on this thesis has been a very rewarding experience that has greatly benefited my education. I thank him for his positive attitude, kind patience and trust during my entire Ph.D. study. Also, I would like to thank him for teaching me the art of writing scientific articles and this dissertation. His constant persistence to reach the highest standards in all aspects of his work has inspired me to work very hard. I specially thank him for providing me with financial support throughout my graduate studies at Iowa State University.

I also wish to express my cordial thanks to Dr. Salapaka, Dr. Elia, Dr. Miner and Dr. Mitra for dedicating their time to participate in my committee. Their feedback and comments were very helpful in improving the work presented in this dissertation.

There are many friends who through their generosity and friendship have made my Ph.D. study an unforgettable experience. I owe my warm gratitude to my parents and brothers for their affection and support. Finally, I wish to thank my husband, Yuchi Liu, who has been a constant source of encouragement.

ABSTRACT

The supervisory control of discrete event systems provides a framework for control of event-driven systems. Applications of supervisory control theory include protocol design for communication processes, control logic synthesis in manufacturing systems, and collision avoidance in human-computer interaction systems.

When designing a system at a certain level of abstraction, lower level details of the system and its specification are normally omitted to obtain higher level models that may be (nondeterministic) event-driven systems. Nondeterministic systems exhibit both branching and sequential behaviors and are captured using bisimulation equivalence (the traditional language equivalence only captures sequential behaviors). Simulation equivalence is more expressive than language equivalence but captures only the universal fragment of branching behaviors.

This dissertation presents supervisory control of discrete event systems for enforcing *bisimulation equivalence* or *simulation equivalence* with respect to given specifications. We show that in the general setting of nondeterministic systems and specifications, the complexity for bisimilarity enforcing control is doubly exponential and for similarity enforcing control remains polynomial solvable. So the choice of behavioral equivalence used depends on the application at hand and there is a trade-off between the expressivity and the complexity. We further show that the bisimilarity enforcing control problem becomes polynomially solvable when the system model is deterministic and there is complete observability of events. When the complete observability requirement is relaxed, the control existence problem remains polynomially solvable and the control synthesis problem becomes singly exponential. These complexities are similar to the ones for control under partial observation in completely deterministic setting Tsitsiklis (1989).

We introduce various notions of state-controllability (SC), state-recognizability (SR), state-achievability (SA), state-controllable-similar (SCS), state-controllability-bisimilar (SCB), and state-achievability-bisimilar (SAB) for deterministic system model. SC is a property of a controlled system under complete observation. Under partial observation, an additional property of a controlled system due to the partial observation is SR. The combined property of SC and SR is called SA. We show that properties of SC, SR and SA are not preserved under bisimulation equivalence and therefore cannot be served as a necessary condition for the existence of a bisimilarity enforcing supervisor. We introduce the notions of SCB and SAB, which are preserved under bisimulation, as part of the necessary and sufficient condition for the existence of a supervisor under complete and partial observation, respectively. We show that SC is not preserved under simulation equivalence and introduce SCS as a necessary and sufficient condition for the existence of a similarity enforcing supervisor under complete observation.

The aforementioned results use strict synchronous composition (SSC) of the system and supervisor as a mechanism of control. In SSC, it is required that individual systems synchronously execute all events. Prioritized synchronous composition (PSC) relaxed such synchronization requirements and this has been shown to enrich the control capability when the plant is nondeterministic. (The presence of nondeterminism in a plant model may cause the current state to be known with ambiguity, and allowing the flexibility of not synchronizing an event at all the candidate states that plant may have reached provides for additional benefits.) This dissertation introduces a notion of prioritized synchronous composition under mask (PSCM) to account for partial observation. We study the supervisory control when PSCM is adopted as a mechanism of interaction for both language and bisimulation equivalences. We show that the control & observation-compatibility requirements are removed of a supervisor. For control to achieve a language equivalence, the existence condition is given by achievability that is weaker than controllability and observability combined. (The weaker condition is required since we allow supervisors to be nondeterministic.) This suggests that the notion of PSCM is an appropriate generalization of PSC to account for partial observation.

CHAPTER 1. INTRODUCTION

Discrete event systems (DESs) are systems that evolve according to the occurrence of irregularly occurred events Cassandras and Lafortune (1995); Kumar and Garg (1995). Examples of events include pressing a button, failure of a device, sending a message though a communication channel, or water level reaching a limit, etc. A DES has a discrete state-space which may take symbolic values rather than real values, such as (a software is) running, waiting or terminated. Examples of DESs include manufacturing systems, communication networks and reactive software.

1.1 Supervisory Control of DESs

The supervisory control theory of DESs was introduced by Ramadge-Wonham Ramadge and Wonham (1989) for designing controllers so that the controlled system satisfies a specification. The event set of a DES, also called a pant, is finite and is partitioned into sets of controllable events and uncontrollable events. A supervisory controller, also called supervisor, disables certain controllable events to ensure that the controlled system behavior agrees with the desired behavior. Prior work in deterministic setting (plant/specification/supervisor all are deterministic) include control under full Ramadge and Wonham (1989, 1987) or partial observation Cieslak et al. (1988); Yoo and Lafortune (2001); Prosser et al. (1998), non-blocking control Kumar et al. (2005c); Fabian and Kumar (2000), modular control Rohloff and Lafortune (2002); Chen et al. (2000), decentralized control Jiang and Kumar (2000); Tripakis (2001); Kozak and Wonham (1995), and hierarchical control Brave and Heymann (1993); Wong and Wonham (1996); Zhong and Wonham (1990).

Nondeterminism in plant model can arise from unmodeled dynamics or abstraction. A

nondeterministic plant can have a language or a finer specification such as: failures Hoare (1985), refusal-trace (same as trajectory) Phillips (1987); Heymann and Meyer (1991), ready-trace Baeten et al. (1987), simulation and bisimulation equivalences Milner (1989). Also, the supervisors can be deterministic as well as nondeterministic.

The control of nondeterministic plant subject to language specification is studied in Shayman and Kumar (1995); Kumar and Shayman (1996a, 1997), where plant is modeled using the trajectory model. In Overkamp (1994) and Heymann and Lin (1998), both plant and specification are nondeterministic and are represented using failures and trajectory models, respectively. Authors in Heymann and Lin (1998) show how to transform their control problem of nondeterministic setting to one of deterministic setting with added partial observability. Control of plants modeled using nondeterministic state machines for language specification is also studied in Kumar and Heymann (2000a); Jiang and Kumar (2002). All these work used deterministic supervisors.

The use of nondeterministic supervisors for specification represented using language model was explored in Inan (1994); Shayman and Kumar (1999). The notion of nondeterministic control was formalized in Kumar et al. (2005b) and used for control under partial observation for language specification, and the notion of achievability (a property weaker than controllability and observability combined) was introduced. The problem of finding a nondeterministic supervisor so that it's parallel-composition with plant conforms to a deterministic specification (via language containment) is studied in Yevtushenko et al. (2001). Nondeterministic supervisors were also used in Jiang and Kumar (2006) where nondeterministic specification was specified in the temporal logic of CTL*, generalizing the work reported in Antoniotti (1995) which used CTL to express specification. Other work related to control subject to temporal logic based specification include Kupferman et al. (2001, 2000); Kupferman and Vardi (1999); Arnold et al. (2003a); Riedweg and Pinchinat (2003).

Bisimulation equivalence is the finest known notion of behavioral equivalence and it was first introduced in communicating systems by Milner Milner (1980). Bisimulation equivalence specification is equivalent to a specification in the temporal logic of μ-calculus that subsumes

the complete branching-time logic CTL* Hennessey and Milner (1985). So if a supervisor is designed to ensure that the controlled system is bisimilar to a specification system, then this is equivalent to ensuring that the controlled system satisfies the same μ-calculus or CTL* specification that is satisfied by the specification system. On the other hand, a language equivalence based control only guarantees the satisfaction of a LTL (Linear Temporal Logic) specification which is a strict subclass of μ-calculus and CTL*. Simulation equivalence is finer than language equivalence but coarser than bisimulation equivalence. While language equivalence poses no constraints on the nondeterministic (branching) behavior, and bisimulation equivalence specifies the exact branching behavior, simulation equivalence specifies also allows any "smaller" branching behavior. Simulation equivalence preserves universal fragment of μ-calculus specificationsHenzinger et al. (2003), which is more general than LTL (preserved under language equivalence) but less general than μ-calculus (preserved under bisimulation equivalence).

Control for achieving CTL* specification was studied by Jiang and Kumar in Jiang and Kumar (2001), under the assumption that plant model is deterministic. Arnold et al. (2003b) studied the synthesis of controllers for deterministic plants subject to μ-calculus based specifications under partial observation, where the observation mask is restricted to be projection type. The control problem is solved by reduction to a discrete-event game problem, and explicit conditions for the existence of a supervisor are not provided. In this paper we allow both plant and specification models to be nondeterministic. Furthermore, our approach is quite different: In Jiang and Kumar (2001), the control problem was reduced to a decision problem of CTL*, whereas our results are based on the properties of the automata models of the plant and the specification. Given nondeterministic models of plant and its specification, we study the design of a supervisor (possibly nondeterministic) such that the controlled system is bisimilar to the specification system. Issues regarding implementation of nondeterministic supervisors are discussed in Kumar et al. (2005b).

The input-output model matching control studied in Benedetto et al. (2001) also uses the notion of simulation, and as shown in Barrett and Lafortune (1998) it can be cast as an instance of standard supervisory control problem of deterministic setting. Khatri et al. (1996)

studied the problem of finding a controller such that the controlled system is simulated by the specification, where the plant is assumed to be pseudo-nondeterministic and specification is allowed to be nondeterministic. Bisimulation relation has been used as a technique for supervisory control of deterministic systems subject to language equivalence in Rutten (2000); Komenda (2002b); Marchand and Pinchinat (2000); Komenda (2002a). In Rutten (2000); Komenda (2002a) the controllability and observability is characterized as a bisimulation type relation. Qin and Lewis (1990) studied the problem of synthesizing a supervisor so that the controlled system is bisimilar to a deterministic specification. The event set of the system and specification need not be same, and all events are treated controllable. Madhusudan and Thiagarajan (2002) studied control for simulation and bisimulation equivalence for a partial specification (defined over an "external event set"). The plant is taken to be deterministic and all events are treated controllable. Further it is required that all indistinguishable events be either all enabled or all disabled at a state. Such a requirement does not make sense in supervisory control context. Tabuada (2004) studied the controller synthesis problem for deterministic plants subject to a possibly nondeterministic partial specification such that the controlled system is bisimulation equivalent to the specification. This is the same problem as that studied in Madhusudan and Thiagarajan (2002) except the aforementioned control requirement is removed.

Most work on supervisory control of discrete event systems (DESs) use strict synchronous composition (SSC) of the plant and supervisor as a mechanism of control. In SSC, it is required that the common events occur synchronously, which is a restriction. For example, there is no a priori reason for a supervisor to synchronously execute all the uncontrollable events that a plant executes.

Heymann Heymann (1990) proposed a type of interaction, called prioritized synchronous composition (PSC), which relaxed such synchronization requirements. PSC delegates the effects of control limitations of a supervisor from its logic part (implemented as an automaton) to its interface part (implemented as PSC), and thereby, removes the requirement of "completeness" Kumar et al. (1991) or "Σ_u-compatibility" Kumar et al. (1991) of a supervisor. In

PSC, each system possesses an event priority set specifying a set of events whose execution require its participation. Thus, an event can occur if and only if all the systems having priority over the event can participate in its execution. In this case, the event occurs synchronously in all such systems, and otherwise the event gets blocked from occurring. The systems which do not have priority over the event also participate in its execution if they can, and otherwise the event takes place without the participation of such systems. Thus, a system with no priority over an event cannot block its execution. In other words, at a given state of a system, all executable and all non-priority events are enabled by that system at that state. Supervisory control of DESs using PSC has been studied in Heymann and Meyer (1991); Balemi (1994); Fabian (1995); Heymann and Lin (1998); Shayman and Kumar (1995); Kumar and Shayman (1996a, 1997, 1996b).

PSC models the interaction among systems when all the events are completely observable. When systems interact under partial observation (modeled as non-identity observation masks), their interaction through PSC requires that the systems be observation compatible with respect to their masks Kumar and Shayman (1997). So it is meaningful to generalize the notion of PSC to allow interaction of systems possessing non-identity observation mask so that the systems can interact without needing to ensure that they are control or observation compatible.

An effort to generalize PSC in such a direction was presented in Shayman and Kumar (1999), and the generalization was called masked composition (MC). The notion of process-objects was introduced and their MC was defined. In MC, each system is associated with two types of mask function: A control mask that identifies events from the control perspective, and an observation mask that identifies events from the observation perspective. One difficulty with that work is the underlying modeling formalism of process objects that contains "virtual transitions" besides "real" ones, and modeling of practical systems as such process objects is not quite clear.

Another generalization of PSC to describe prioritized synchronization of systems via interfaces, *masked prioritized synchronous composition (MPSC)*, was introduced by Kumar-Heymann in Kumar and Heymann (2000b). MPSC was latter used for control with "driven"

events in Jiang and Kumar (2002). MPSC retains the basic concept of PSC in that each system has its own event priority set, i.e., the set of events in which it must participate in order for them to occur in the composition. In MPSC, each system is allowed to interact with its environment via interfaces that are modeled as event mask functions. When two or more systems interact at a common interface, they can synchronize on events that are mapped to common interface events.

MPSC is appropriate for systems interacting via common interfaces. When MPSC is employed for control the condition for existence of a supervisor is *normality together with controllability*, as opposed to the usual weaker condition of *observability together with controllability*. This suggests that MPSC imposes certain stringent interface constraints. In MPSC, not only the observations but also the controls are filtered through the interface mask - An event is enabled as long as an indistinguishable event is enabled. This makes the control more restrictive than that of the usual supervisory control setting where control and observation of events are not inter-dependent. This is the reason for the stronger condition of normality required of a control specification.

1.2 Proposed Problems

In general, plant, specification, and supervisor all can be nondeterministic. Nondeterministic plant and specification are useful when designing a system at a higher level of abstraction so that lower level details of system and its specification are omitted to obtain higher level models that are nondeterministic. Nondeterministic specifications are also meaningful when the system to be controlled has a nondeterministic model due to lack of information (caused for example by partial observation or unmodeled dynamics). In this dissertation we study the control of (nondeterministic) DESs so that the controlled system is bisimulation or simulation equivalence to (nondeterministic) specifications.

We study a more general bisimulation equivalence control problem, namely, in which both the plant as well as the specification are nondeterministic. No prior work addresses this problem in this generality—they impose determinism either on the plant or on the specification.

To understand the nature of the problem when both the plant and the specification are nondeterministic, note that even when the specification is the same as the plant (and so trivially bisimilar to the plant), the specification itself may not be work as a supervisor, for the composition of two of the same system need not be bisimilar to the system itself. Note this complication does not arise when either the plant or the specification is deterministic, since in this case if the plant and the specification are the same (or less generally, the plant can simulate the specification), the specification itself can be chosen as a supervisor. This is because the composition of plant and specification will be bisimilar to the specification (when plant is deterministic), or language equivalent to the specification (when specification is deterministic). In the more general case, the option of choosing the specification itself as a supervisor is not necessarily available, which introduces a "new dimension" (added complexity) to the nature of the control problem.

We show that in the general setting of nondeterministic systems and specifications, the complexity for bisimilarity enforcing control is doubly exponential and for similarity enforcing control remains polynomial solvable. So the choice of behavioral equivalence used depends on the application at hand and there is a trade-off between the expressivity and the complexity.

The high expressivity of bisimulation equivalence and high computational complexity for bisimilarity enforcing control motivate us seeking specializations which possess more practical complexity bounds. We identify a specialization when the system model is deterministic. We show that both existence and synthesis of a bisimilarity enforcing supervisor remains polynomially solvable under complete observation. Further, where there is a partial observation, the existence condition is polynomially verifiable, whereas the complexity of synthesizing a supervisor (when one exists) is singly exponential. These complexities are similar to the ones for control under partial observation in completely deterministic setting Tsitsiklis (1989). An elaborate computation is required to construct a supervisor. Such a construction (see Algorithm 2) is quite novel and to the best of our knowledge the first in the literature dealing with supervisory control theory.

We introduce a new interaction/control mechanism for partially observed DESs, namely

prioritized synchronous composition under mask (PSCM). When MPSC Kumar and Heymann (2000b) is employed for control the condition for existence of a supervisor is *normality together with controllability*, as opposed to the usual weaker condition of *observability together with controllability*. This suggests that MPSC imposes certain stringent interface constraints. This serves as a motivation to introduce PSCM. We show that when PSCM is adopted as a mechanism of interaction, not only the control & observation-compatibility requirements are removed of a supervisor, the existence condition for a supervisor such that the controlled system is language equivalent to a specification is given by *achievability* that is weaker than controllability and observability combined. (The weaker condition is required since we allow supervisors to be nondeterministic.) This suggests that the notion of PSCM introduced in the dissertation is an appropriate generalization of PSC to account for partial observation. We study control of (nondeterministic) DESs for achieving bisimulation equivalence specifications using PSCM as a control mechanism, and establish an equivalence between a PSCM-based bisimilarity enforcing controller and a SSC-based bisimilarity enforcing controller by showing that if one of them exists and the other one also exists.

1.3 Dissertation Organization

The dissertation is organized as follows.

In Chapter 2, we present notation and preliminaries. The concepts of automata, simulation relation, bisimulation equivalence, simulation equivalence, synchronous composition, and prioritized synchronous composition are introduced.

In Chapter 3, we establish that the set of all automata having the same event set endowed with the simulation relation is a prelattice, and show that synchronization of two automata gives an infimal element for the two automata, whereas the union of the two automata gives a supremal element.

In Chapter 4, we present supervisory control for bisimulation equivalence. For control of nondeterministic plant for bisimulation equivalence, we obtain a small model theorem showing that a supervisor for enforcing bisimulation equivalence between the specification and the

controlled system exists if and only if it exists over a certain finite state space. For the special case of deterministic plants we obtain a necessary and sufficient condition for the existence of a bisimilarity enforcing supervisor which can be verified polynomially in both plant and specification states. A stronger notion of controllability, called state-controllability, is introduced as part of the necessary and sufficient condition for the existence of such a supervisor.

In Chapter 5, we extend the small model theorem by showing that a control and observation compatible supervisor for enforcing bisimulation equivalence between the specification and the controlled system exists if and only if it exists over a certain finite state space. For the special case of deterministic plants, we introduce new concepts of state-achievability-bisimilar. We provide polynomial algorithms for verifying state-achievability-bisimilar. While the existence of a bisimilarity enforcing supervisor can be determined polynomially in both plant and specification states for deterministic plants, the upper bound for the synthesis we provide is exponential.

In Chapter 6, we study supervisory control for simulation equivalence. We establish the polynomial solvability of the simulation equivalence control. The results are further generalized to the "range" control problem, where the controlled behavior must lie in a range specified by a lower and an upper bound behavior (ordering is defined by the simulation relation). We show that our necessary and sufficient condition for the existence of a similarity enforcing supervisor for deterministic plant specializes to the condition of "state-controllable-similar", which is another new concept introduced in this paper.

In Chapter 7, we introduce the concept of prioritized synchronous composition under mask and studies its properties. We introduce augmentation in the setting of PSCM, and show how in certain cases, PSCM of systems is equivalent to PSC of their augmentations. We study PSCM-based supervisory control with no "driven" events for both language equivalence and bisimulation equivalence.

In Chapter 8, we summarize the results and conclude with suggestions for further research.

CHAPTER 2. NOTATION AND PRELIMINARIES

We use finite state machine (or finite automata) to model discrete event systems at the logical level, and bisimulation or simulation equivalence to describe the behavioral equivalence. Typically a controller operates under limited control and observation capabilities. So an admissible controller state machine must be control & observation-compatible. This chapter reviews concepts of finite automata, their compositions, bisimulation and simulation equivalence, and control & observation-compatibility. The interested reader may consult Hopcroft and Ullman (1979); Ramadge and Wonham (1989); Milner (1980); Kumar and Garg (1995) for more details.

2.1 Finite Automata

A nondeterministic automaton is a 5-tuple,

$$G = (X, \Sigma, \alpha, X_0, X_m),$$

where

- X is the set of states,
- Σ is the alphabet of events,
- $\alpha : X \times \overline{\Sigma} \to 2^X$ is the state transition function, where $\overline{\Sigma} := \Sigma \cup \{\epsilon\}$ with ϵ being a label for "silent" transitions,
- $X_0 \subseteq X$ is the set of initial states,
- $X_m \subseteq X$ is the set of final states.

A triple $(x, \sigma, x') \in X \times \overline{\Sigma} \times X$ such that $x' \in \alpha(x, \sigma)$ is called a transition. The automata is said to be deterministic if its transition function is a partial map $\alpha : X \times \Sigma \to X$, i.e., if the

transition function uniquely determines the resulting next state. By entering a marked state, we record that the system has completed some operation or task. Masked states can be used to study the issue of blocking in DESs. It is possible for the controlled system to execute a sequence of events, which cannot be extended to reach a marked state. Thus, the controlled system may get blocked.

Σ^* denotes the set of all finite sequences of events in Σ, called event-traces, and includes the zero length trace, denoted ϵ. The ϵ-closure (denoted as $\epsilon^*(\cdot)$) of $x \in X$ is the set of states reached by the execution of a sequences of ϵ-transitions from state x. By using ϵ-closure map, we can extend the definition of transition function from events to traces, $\alpha^* : X \times \Sigma^* \to 2^X$, which is defined inductively as:

$$\forall x \in X, \alpha^*(x, \epsilon) := \epsilon^*(x); \quad \forall s \in \Sigma^*, \sigma \in \Sigma : \alpha^*(x, s\sigma) := \epsilon^*(\alpha(\alpha^*(x, s), \sigma)).$$

The language generated (resp., marked) by G, is denoted as $L(G)$ (resp., $L_m(G)$). $L(G)$ is the sequence of events generated starting from the initial state, i.e., $L(G) = \{s \in \Sigma^* \mid \alpha^*(X_0, s) \neq \emptyset\}$, and $L_m(G)$ is the set of generated sequences that end in a marked state, i.e., $L_m(G) = \{s \in L(G) \mid \alpha^*(X_0, s) \cap X_m \neq \emptyset\}$. For $x \in X$, we define

$$\Sigma(x) := \{\sigma \in \Sigma \mid \alpha(x, \sigma) \neq \emptyset\} \text{ and } \overline{\Sigma}(x) := \Sigma(x) \cup \{\epsilon\}$$

to denote the set of all event and labels on which transitions are defined at state x, respectively.

The *strict synchronous composition* of two automata G_1 and G_2, where G_1 =(X_1, Σ, α_1,X_{01},X_{m1}) and $G_2 = (X_2, \Sigma, \alpha_2, X_{02}, X_{m2})$, is the automaton

$$G_1 \| G_2 = (X_1 \times X_2, \Sigma, \alpha, X_{01} \times X_{02}, X_{m1} \times X_{m2}),$$

where for $x_1 \in X_1, x_2 \in X_2, \sigma \in \overline{\Sigma}$,

$$\alpha((x_1, x_2), \sigma) = \begin{cases} \alpha_1(x_1, \sigma) \times \alpha_2(x_2, \sigma) & \text{if } \sigma \neq \epsilon \\ (\alpha_1(x_1, \epsilon) \times \{x_2\}) \cup (\{x_1\} \times \alpha_2(x_2, \epsilon)) & \text{if } \sigma = \epsilon \end{cases}$$

We also define the *union* of G_1 and G_2 as the automaton

$$G_1 \cup G_2 = (X_1 \cup X_2, \Sigma, \alpha_\cup, X_{01} \cup X_{02}, X_{m1} \cup X_{m2}),$$

where for $x \in X_1 \cup X_2$, and $\sigma \in \overline{\Sigma}$,

$$\alpha_{\cup}(x,\sigma) = \begin{cases} \alpha_1(x,\sigma) \cup \alpha_2(x,\sigma) & \text{if } x \in X_1 \cap X_2 \\ \alpha_1(x,\sigma) & \text{if } x \in X_1 - X_2 \\ \alpha_2(x,\sigma) & \text{if } x \in X_2 - X_1 \end{cases}$$

In SSC, it requires that the common events occur synchronously, which is a restriction. Heymann Heymann (1990) proposed *prioritized synchronous composition (PSC)*, which relaxed such synchronization requirements. In PSC, each system has an event priority set. An event can occur as long as all systems having the priority over the event can participate. For $i = 1, 2$, consider NSM $G_i = (X_i, \Sigma, \alpha_i, X_{0i})$ with its event priority set $A_i \subseteq \Sigma$. Then the prioritized synchronous composition of G_1 and G_2 is given by

$$G_{1A_1} \|_{A_2} G_2 = (X_1 \times X_2, \Sigma, \alpha, X_{01} \times X_{02}),$$

where for $x_1 \in X_1, x_2 \in X_2$ and $\sigma \in \Sigma$,

$$\begin{aligned} \alpha((x_1,x_2),\sigma) &:= \begin{cases} \alpha_1(x_1,\sigma) \times \alpha_2(x_2,\sigma), & \text{if } \alpha_1(x_1,\sigma) \neq \emptyset, \alpha_2(x_2,\sigma) \neq \emptyset \\ \alpha_1(x_1,\sigma) \times \{x_2\}, & \text{if } \alpha_1(x_1,\sigma) \neq \emptyset, \alpha_2(x_2,\sigma) = \emptyset, \sigma \notin A_2 \\ \{x_1\} \times \alpha_2(x_2,\sigma), & \text{if } \alpha_2(x_2,\sigma) \neq \emptyset, \alpha_1(x_1,\sigma) = \emptyset, \sigma \notin A_1 \\ \emptyset, & \text{otherwise} \end{cases} \\ \alpha((x_1,x_2),\epsilon) &:= (\alpha_1(x_1,\epsilon) \times \{x_2\}) \cup (\{x_1\} \times \alpha_2(x_2,\epsilon)). \end{aligned}$$

The event priority set of $G_{1A_1} \|_{A_2} G_2$ is taken to be $A_1 \cup A_2$. In the special case when the event priority sets of the two systems exhaust the entire event set Σ, PSC can be transformed to SSC using the method of augmentation introduced in Heymann (1990).

2.2 Bisimulation and Simulation Equivalence

We next introduce the concept of a simulation relation.

Definition 1 Given $G_1 = (X_1, \Sigma, \alpha_1, X_{01}, X_{m1})$, $G_2 = (X_2, \Sigma, \alpha_2, X_{02}, X_{m2})$, a *simulation relation* is a binary relation $\Phi \subseteq X_1 \times X_2 \subseteq (X_1 \cup X_2)^2$ such that $(x_1, x_2) \in \Phi$ implies

1. $\sigma \in \overline{\Sigma}$, $x_1' \in \alpha_{\cup}^*(x_1, \sigma) \Rightarrow \exists x_2' \in \alpha_{\cup}^*(x_2, \sigma)$ such that $(x_1', x_2') \in \Phi$.

2. $x_1 \in X_{m1} \cup X_{m2} \Rightarrow x_2 \in X_{m1} \cup X_{m2}$.

G_1 is said to be simulated by G_2, denoted as $G_1 \sqsubseteq_\Phi G_2$, if there exists a simulation relation $\Phi \subseteq (X_1 \cup X_2)^2$ such that for all $x_{01} \in X_{01}$, exists $x_{02} \in X_{02}$ with $(x_{01}, x_{02}) \in \Phi$. This last fact is concisely written as $X_{01} \sqsubseteq_\Phi X_{02}$.

We write $x_1 \sqsubseteq_\Phi x_2$ to denote that there exists a simulation relation Φ with $(x_1, x_2) \in \Phi$, read as x_1 is simulated by x_2. We sometimes omit the subscript Φ from $\sqsubseteq_\Phi$ when it is clear from the context.

A bisimulation relation is a symmetric simulation relation which is captured by the following definition.

Definition 2 Given two automata G_1 and G_2 as defined above, a *bisimulation relation* is a binary relation $\Phi \subseteq (X_1 \cup X_2)^2$ such that for $x_1 \in X_1, x_2 \in X_2$, $(x_1, x_2) \in \Phi$ implies

1. $\sigma \in \overline{\Sigma}$, $x_1' \in \alpha_\cup^*(x_1, \sigma) \Rightarrow \exists x_2' \in \alpha_\cup^*(x_2, \sigma)$ such that $(x_1', x_2') \in \Phi$.

2. $\sigma \in \overline{\Sigma}$, $x_2' \in \alpha_\cup^*(x_2, \sigma) \Rightarrow \exists x_1' \in \alpha_\cup^*(x_1, \sigma)$ such that $(x_1', x_2') \in \Phi$.

3. $x_1 \in X_{m1} \cup X_{m2} \Rightarrow x_2 \in X_{m1} \cup X_{m2}$.

Further G_1 and G_2 are said to be *bisimulation equivalent* (or *bisimilar*), denoted as $G_1 \simeq_\Phi G_2$, if Φ is symmetric so that $X_{01} \simeq_\Phi X_{02}$.

We write $x_1 \simeq_\Phi x_2$ to denote that there exists a bisimulation relation Φ with $(x_1, x_2) \in \Phi$, read as x_1 is bisimilar to x_2. We sometimes omit the subscript Φ from $\simeq_\Phi$ when it is clear from the context. From the definition of bisimulation relation and simulation relation, we easily observe that $x_1 \simeq_\Phi x_2$ if and only if $x_1 \sqsubseteq_\Phi x_2$, $x_2 \sqsubseteq_\Phi x_1$ and Φ is symmetric.

Definition 3 Given two automata G_1 and G_2, $\Phi \subseteq (X_1 \cup X_2)^2$ is a *similarity relation* if exist simulation relations $\Phi_1, \Phi_2 \subseteq (X_1 \cup X_2)^2$ such that $\Phi = \Phi_1 \cup \Phi_2$, and

$$\forall x : [\exists x' : (x, x') \in \Phi_i \Rightarrow \exists x'' : (x'', x) \in \Phi_j], \forall i, j \in \{1, 2\}, i \neq j.$$

G_1 and G_2 are *simulation equivalent* (or *similar)*, denoted $G_1 \sim_\Phi G_2$, if exist simulation relations Φ_1 and Φ_2, such that $\Phi = \Phi_1 \cup \Phi_2$ and $G_1 \sqsubseteq_{\Phi_1} G_2$ and $G_2 \sqsubseteq_{\Phi_2} G_1$.

We write $x_1 \sim_\Phi x_2$ to denote that there exists a similarity relation Φ such that $(x_1, x_2) \in \Phi$, read as x_1 and x_2 are simulation equivalent or similar. Note that a similarity relation Φ need not be an equivalence relation (as it need not be symmetric), however the similarity of automata is an equivalence relation. Also note that $G_1 \sqsubseteq G_2$ implies $L(G_1) \subseteq L(G_2)$, and $G_1 \simeq G_2$ implies $G_1 \sim G_2$ which in turn implies $L(G_1) = L(G_2)$.

Remark 1 Existence of simulation relation or simulation or bisimulation equivalence between a pair of automata G_1 and G_2 can be checked linearly in the sizes of G_1 and G_2 Henzinger et al. (1995).

Next we give examples to illustrate these concepts.

Example 1 Consider two automata G_1 and G_2 shown in Figure 2.1. Their synchronous composition $G_1 \| G_2$ and union $G = G_1 \cup G_2$ are shown in Figure 2.2.

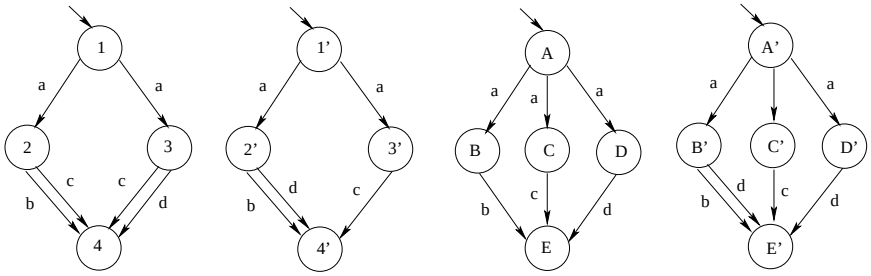

Figure 2.1 G_1 (first), G_2 (second), G_3 (third), and G_4 (fourth)

Example 2 Consider automata G_2 and G_4 shown in Figure 2.1. There exists a simulation relation $\Phi_1 \subseteq X_2 \times X_4$: $\Phi_1 = \{(1', A'), (2', B'), (3', C'), (4', E')$. Thus $G_2 \sqsubseteq_{\Phi_1} G_4$. Also, there exists a simulation relation $\Phi_2 \subseteq X_4 \times X_2$: $\Phi_2 = \{(A', 1'), (B', 2'), (C', 3'), (D', 2'), (E', 4')\}$. Thus $G_4 \sqsubseteq_{\Phi_2} G_2$. Therefore, there exists a similarity relation $\Phi = \Phi_1 \cup \Phi_2$ such that $G_2 \sim_\Phi G_4$. However, Φ is not symmetric. So $G_2 \not\simeq_\Phi G_4$.

Consider G_3 in Figure 2.1 and $G' = G_1 || G_2$ in Figure 2.2. There exists a symmetric simulation relation $\Phi' \subseteq (X_3 \cup X')^2$ given by,

$$\begin{aligned} \Phi' \quad = \quad & \{(A, 11'), (B, 22'), (C, 23'), (C, 33'), (D, 32'), (E, 44'), \\ & (11', A), (22', B), (23', C), (32', D), (33', C), (44', E)\}. \end{aligned}$$

Therefore, $G_3 \simeq_{\Phi'} G'$.

Note that $L(G_1) = L(G_2) = L(G_3) = L(G_4)$. However, $G_1 \not\sqsubseteq G_2$, $G_2 \not\sqsubseteq G_1$, $G_1 \not\sqsubseteq G_4$, and $G_4 \not\sqsubseteq G_1$.

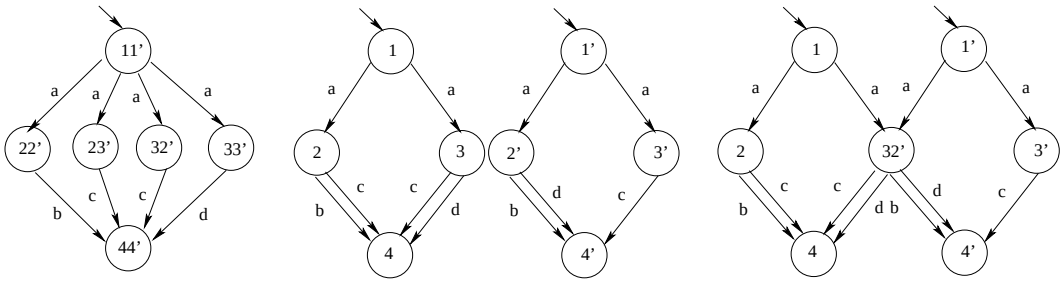

Figure 2.2 $G_1 \| G_2$ (left), $G = G_1 \cup G_2$ (middle), and $\overline{G_{\langle 3,2' \rangle}}$ (right)

Our construction of a bisimilarity enforcing control requires merger of states of a certain automaton as defined below.

Definition 4 We use $\overline{G_{\langle x,x' \rangle}} = (\overline{X}, \Sigma, \overline{\alpha}, \overline{X}_0, \overline{X}_m)$ to denote $G = (X, \Sigma, \alpha, X_0, X_m)$ in which two states $x, x' \in X$ are merged. Use $\langle x, x' \rangle$ to denote the merger of two states x and x'. Then,

$$\overline{X} = (X - \{x, x'\}) \cup \{\langle x, x' \rangle\},$$

and $\forall \hat{x}, \tilde{x} \in X - \{x, x'\}$, $\forall \sigma \in \overline{\Sigma}$:

$$x \in \alpha(\hat{x}, \sigma) \text{ in } G \Rightarrow \langle x, x' \rangle \in \overline{\alpha}(\hat{x}, \sigma) \text{ in } \overline{G_{\langle x,x' \rangle}}$$

$$x' \in \alpha(\tilde{x}, \sigma) \text{ in } G \Rightarrow \langle x, x' \rangle \in \overline{\alpha}(\tilde{x}, \sigma) \text{ in } \overline{G_{\langle x,x' \rangle}}$$

$$\hat{x} \in \alpha(x, \sigma) \text{ in } G \Rightarrow \hat{x} \in \overline{\alpha}(\langle x, x' \rangle, \sigma) \text{ in } \overline{G_{\langle x,x' \rangle}}$$

$$\tilde{x} \in \alpha(x', \sigma) \text{ in } G \Rightarrow \tilde{x} \in \overline{\alpha}(\langle x, x' \rangle, \sigma) \text{ in } \overline{G_{\langle x,x' \rangle}}$$

$$\tilde{x} \in \alpha(\hat{x}, \sigma) \text{ in } G \Rightarrow \tilde{x} \in \overline{\alpha}(\hat{x}, \sigma) \text{ in } \overline{G_{\langle x,x' \rangle}}.$$

$$x \in \alpha(x', \sigma) \text{ in } G \Rightarrow \langle x, x' \rangle \in \overline{\alpha}(\langle x, x' \rangle, \sigma) \text{ in } \overline{G_{\langle x,x' \rangle}}$$

$$x' \in \alpha(x, \sigma) \text{ in } G \Rightarrow \langle x, x' \rangle \in \overline{\alpha}(\langle x, x' \rangle, \sigma) \text{ in } \overline{G_{\langle x,x' \rangle}}$$

Example 3 Consider automaton G in Figure 2.2, the automaton $\overline{G_{\langle 3,2' \rangle}}$ is drawn in Figure 2.2.

It is known (see Rutten (1995)) that merger of bisimilar states in an automaton yields a bisimilar automaton.

Theorem 1 Rutten (1995) Given an automaton G, if $x, x' \in X$ are such that $x \simeq x'$, then $G \simeq \overline{G_{\langle x,x' \rangle}}$.

2.3 Control of Discrete Event Systems

A DES, called a plant, is controlled to restrict it's behavior so as to prevent any undesirable behavior by dynamically disabling certain controllable events Ramadge and Wonham (1989). Such a controller is called a supervisor. The supervisor can be modeled as another automaton operating in synchronous composition with the plant.

Typically a supervisor operates under limited control and observation capabilities. A limitation in control arises due the presence of uncontrollable events, denoted $\Sigma_u \subseteq \Sigma$, that cannot be prevented from occurring. For this reason, an admissible supervisor state machine must enable all uncontrollable events at all its states, a property known as control-compatibility or Σ_u-compatibility. A limitation in observation arises due to the type event-sensors used which can provide only a partial or no information about the occurrence of an event they monitor. The sensing capabilities of such event-sensors can be represented as an observation mask function $M : \overline{\Sigma} \rightarrow \overline{\Delta}$ that maps events into observations (Δ is the set of observed symbols) and satisfies $M(\epsilon) = \epsilon$. M is said to be projection type if $\Delta \subseteq \Sigma$ and for $\sigma \in \Delta$, $M(\sigma) = \sigma$. $\sigma \in \Sigma$ is said to be an unobservable event if $M(\sigma) = \epsilon$, and otherwise it is said to be an observable event. Two events $\sigma_1, \sigma_2 \in \overline{\Sigma}$ are said to be indistinguishable if $M(\sigma_1) = M(\sigma_2)$. The observation mask M is extended to be defined over traces in Σ^* as follows: $M(\epsilon) := \epsilon$; $\forall s \in \Sigma^*, \sigma \in \Sigma : M(s\sigma) := M(s)M(\sigma)$. Further the inverse of the mask function is defined as: $\forall \tau \in \Delta^* : M^{-1}(\tau) := \{s \in \Sigma^* \mid M(s) = \tau\}$.

The notion of control & observation-compatibility or (Σ_u, M)-compatibility is formally defined as follows.

Definition 5 Let $\Sigma_u \subseteq \Sigma$ be the set of uncontrollable events and $M : \overline{\Sigma} \rightarrow \overline{\Delta}$ be the observation mask, then

- $S = (Y, \Sigma, \beta, Y_0, Y_m)$ is called Σ_u-*compatible* if $\forall y \in Y$ and $\forall a \in \Sigma_u$, $\beta(y, a) \neq \emptyset$.
- S is called M-*compatible* if $\forall y \in Y$ and $\forall a, b \in \overline{\Sigma}(y)$, if $M(a) = M(b)$, then $\beta(y, a) = \beta(y, b)$, where it is assumed that a silent transition is implicitly defined as a self-loop.
- S is called (Σ_u, M)-*compatible* if S is Σ_u-compatible and M-compatible.

Unless otherwise stated, we will use $G = (X, \Sigma, \alpha, X_0, X_m)$, $R = (Q, \Sigma, \delta, Q_0, Q_m)$, and $S = (Y, \Sigma, \beta, Y_0, Y_m)$ to denote the (deterministic) plant, specification, and supervisor, respectively. The controlled system will be denoted by $G \| S = (X \times Y, \Sigma, \gamma, X_0 \times Y_0, X_m \times Y_m)$.

CHAPTER 3. PRELATTICE AUTOMATA UNDER SIMULATION RELATION

In this chapter we show that the simulation relation serves as a preorder for the set of all automata defined over a common event set, and also that the set of automata defined over a common event set together with the simulation relation preorder constitutes a prelattice.

Definition 6 Kumar and Garg (1995) Given a set X, a preorder over X, denoted $\leq \subseteq X^2$, is a transitive and reflexive relation, in which case the pair $(X, \leq)$ is called a *preordered set*. Given $Y \subseteq X$, $x \in X$ is said to be a *supremal of* Y if

- (upper bound): $\forall y \in Y : y \leq x$, and
- (least upper bound): $\forall z \in X : [\forall y \in Y : y \leq z] \Rightarrow [x \leq z]$.

Similarly, $x \in X$ is called an *infimal of* $Y \subseteq X$ if

- (lower bound): $\forall y \in Y : x \leq y$, and
- (greatest lower bound): $\forall z \in X : [\forall y \in Y : z \leq y] \Rightarrow [z \leq x]$.

Note that supremal and infimal when defined with respect to a preordered are not unique. However if x_1 and x_2 are two supremal or infimal elements of Y, then it holds that $x_1 \leq x_2$ and $x_2 \leq x_1$. Since a preorder is not antisymmetric we cannot claim that $x_1 = x_2$, and so the uniqueness of supremal/infimal does not hold. We denote the set of all supremals and infimals of Y by $SUP(Y)$ and $INF(Y)$, respectively.

Definition 7 Kumar and Garg (1995) A preordered set $(X, \leq)$ is said to be a *prelattice* if $SUP(Y) \cap X \neq \emptyset$, and $INF(Y) \cap X \neq \emptyset$ for any finite $Y \subseteq X$. It is said to be a complete prelattice if the same holds for any $Y \subseteq X$.

3.1 Prelattice Automata under Simulation Relation

We next consider the set of all automata $\mathcal{A}$ over a fixed alphabet Σ and the simulation relation over this set. It is known that the simulation relation is transitive, i.e., given automata G_1, G_2 and G_3, if $G_1 \sqsubseteq G_2$ and $G_2 \sqsubseteq G_3$, then $G_1 \sqsubseteq G_3$. Also, for any automaton G, it holds that $G \sqsubseteq G$, implying the reflexivity of the simulation relation. However $G_1 \sqsubseteq G_2$ and $G_2 \sqsubseteq G_1$ only implies $G_1 \sim G_2$ but not $G_1 = G_2$, i.e., antisymmetry does not hold. Therefore, the pair $(\mathcal{A}, \sqsubseteq)$ is a preordered set.

In the following we establish that the automata-union (resp., automata-synchronization) yields a supremal (resp., an infimal) element.

Theorem 2 Given G_1 and G_2, $G_1 \cup G_2 \in SUP\{G_1, G_2\}$.

Proof: From the definition of automata union, G_1 and G_2 are "subautomata" of $G_1 \cup G_2$ and so it is easy to see that $G_1 \sqsubseteq G_1 \cup G_2$ and $G_2 \sqsubseteq G_1 \cup G_2$, i.e., $G_1 \cup G_2$ is an upper bound for $\{G_1, G_2\}$. Next, we show that it is a least upper bound, i.e., $G_1 \sqsubseteq G_3$ and $G_2 \sqsubseteq G_3$ implies $(G_1 \cup G_2) \sqsubseteq G_3$.

Notice that $G_1 \sqsubseteq G_3$ implies $X_{01} \sqsubseteq X_{03}$ and $G_2 \sqsubseteq G_3$ implies $X_{02} \sqsubseteq X_{03}$. This implies for $i = 1, 2$, for each $x_{0i} \in X_{0i}$ exists $x_{03} \in X_{03}$ such that $x_{0i} \sqsubseteq x_{03}$. Since the set of transitions of $G_1 \cup G_2$ is the union of the set of transitions of the two automata, this implies that for each $x \in X_{01} \cup X_{02}$, there exists $x_{03} \in X_{03}$, such that $x \sqsubseteq x_{03}$. Since the initial state set of $G_1 \cup G_2$ is $X_{01} \cup X_{02}$, it follows that $(G_1 \cup G_2) \sqsubseteq G$. ■

Theorem 3 Given G_1 and G_2, $G_1 \| G_2 \in INF\{G_1, G_2\}$.

Proof: We first prove that $G_1 \| G_2$ is a lower bound, i.e., $G_1 \| G_2 \sqsubseteq_{\Phi_1} G_1$ and $G_1 \| G_2 \sqsubseteq_{\Phi_2} G_2$. By the reflexivity property of simulation relation, there exists Φ such that $G_1 \| G_2 \sqsubseteq_{\Phi} G_1 \| G_2$. Define a simulation relation Φ_1 by

$$\Phi_1 = \{((x_1, x_2), x_1') \mid ((x_1, x_2), (x_1', x_2')) \in \Phi\}.$$

Then it can be seen that Φ_1 is a simulation relation, and so $G_1 \| G_2 \sqsubseteq_{\Phi_1} G_1$. Similarly, we can show $G_1 \| G_2 \sqsubseteq_{\Phi_2} G_2$.

Next, we prove that $G_1 \| G_2$ is a greatest lower bound, i.e., $G_3 \sqsubseteq_{\Phi_1} G_1$ and $G_3 \sqsubseteq_{\Phi_2} G_2$ implies $G_3 \sqsubseteq_{\Phi} G_1 \| G_2$. In order to show $G_3 \sqsubseteq_{\Phi} (G_1 \| G_2)$, define

$$\Phi := \{(x_3, (x_1, x_2)) \mid (x_3, x_1) \in \Phi_1, (x_3, x_2) \in \Phi_2, \exists s \in \Sigma^*, \text{ s.t. } x_i \in \alpha_i^*(X_{0i}, s), \forall i = 1, 2, 3\}.$$

$G_3 \sqsubseteq_{\Phi_i} G_i$ implies $X_{03} \sqsubseteq_{\Phi_i} X_{0i}$ for $i = 1, 2$. Since $G_3 \sqsubseteq_{\Phi_i} G_i$, it follows that $L(G_3) \subseteq L(G_i)$, this further implies $L(G_3) \subseteq L(G_1) \cap L(G_2) = L(G_1 \| G_2)$. This means for every $x_3 \in X_3$ such that there exists $s \in \Sigma^*$ with $x_3 \in \alpha_3^*(X_{03}, s)$, exists $x_i \in X_i$, such that $x_i \in \alpha_i^*(X_{0i}, s)$ for $i = 1, 2$. So Φ above is well defined and serves as a simulation relation for establishing $G_3 \sqsubseteq_{\Phi} G_1 \| G_2$. ■

The following corollary follows from Theorem 3 and provides a property of simulation order.

Corollary 1 Given automata G_1, G_2, G_3, $G_3 \sqsubseteq G_1 \| G_2$ implies $G_3 \sqsubseteq_{\Phi_1} G_1$ and $G_3 \sqsubseteq_{\Phi_2} G_2$.

3.2 Conclusion

We established that the set of all automata having the same event set endowed with the simulation relation is a prelattice, and showed that synchronization of two automata gives an infimal element for the two automata, whereas the union of the two automata gives a supremal element.

CHAPTER 4. SUPERVISORY CONTROL FOR BISIMULATION EQUIVALENCE

In this chapter, we study the control of DESs to ensure bisimilarity of the controlled system and a given specification. For the control of nondeterministic plant for bisimulation equivalence, we obtain a small model theorem showing that a supervisor for enforcing bisimulation equivalence between the specification and the controlled system exists if and only if it exists over a certain finite state space. For the special case of deterministic plants we obtain a necessary and sufficient condition for the existence of a bisimilarity enforcing supervisor which can be verified polynomially in both plant and specification states. A stronger notion of controllability, called state-controllability, is introduced as part of the necessary and sufficient condition for the existence of such a supervisor.

4.1 A Motivating Example

We give an example to illustrate some of the issues that are prevalent when controlling a nondeterministic system.

Example 4 Consider an automatic check-out scanner in a shopping center, a state machine model G of which is shown in Figure 4.1. Initially, a customer presses the start button to start the check-out process. Then it scans an item, upon which, owing to a malfunction, the scanner nondeterministically transitions to one of two states. In the first state, the scanner allows the customer to either put the item in a bag, or cancel; whereas in the second state the only option offered is to put the item in the bag. Not giving an option to cancel in the second state is unacceptable. A reset button may be pressed in either of the states to return to the first state. After this, the scanner waits for either a request for a next item, or if there is no more items

then a request to pay. In the latter case, scanner returns to its initial state, and in the former case it goes back to the state from where check-out process resumes. Since a customer must pay at the end of the check-out process, the event "pay" is deemed uncontrollable. All other events are controllable.

The partial specification R, also shown in Figure 4.1, is given in order to restrict the plant to exhibit only an acceptable behavior. According to the specification, after start and scan, two possible states may be reached nondeterministically. In both states, cancel is an available option which is what we desire of the scanner, while put is an additional option at the first state. The rest of the behavior is the same as the one feasible in the scanner. Note that the "reset" event does not appear in the specification state machine since an occurrence or non-occurrence of it is immaterial to the specification. This implies that the specification R is for the plant G projected on to the event set $\hat{\Sigma} := \Sigma - \{reset\}$, denoted $G{\uparrow}\hat{\Sigma}$. (Projecting an automaton onto $\hat{\Sigma}$ replaces each event label outside $\hat{\Sigma}$ of the automaton by ϵ.)

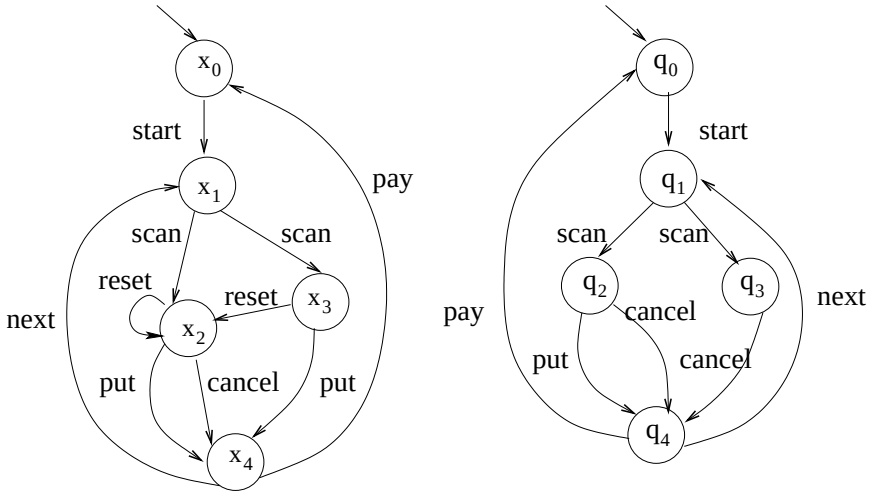

Figure 4.1 Plant G (left) and specification R (right)

Note that $L(R) = L(G{\uparrow}\hat{\Sigma})$, i.e., $G{\uparrow}\hat{\Sigma}$ is language equivalent to R. Thus if we use language equivalence as a notion of behavioral equivalence, then there is no need to control. However, as mentioned above, G can exhibit some behavior that is not acceptable (i.e., not always giving the option to cancel after scan). We develop a theory in this paper that lets us design a supervisor S such that $(G\|S){\uparrow}\hat{\Sigma}$ is bisimilar to R.

4.2 Control of Nondeterministic Plant for Bisimilarity

Before we give the main result of this section, we first give some preliminary results.

Lemma 1 For G, S, and R defined as above, consider $x \in X, y \in Y, q \in Q$.

1. If $q \sqsubseteq_{\Phi} (x, y)$, then $q \sqsubseteq_{\Phi'} x$ and $q \sqsubseteq_{\Phi''} y$.

2. If $q \sqsubseteq_{\Phi'} x$ and $q \sqsubseteq_{\Phi''} y$, then $q \sqsubseteq_{\Phi} (x, y)$.

Proof:

1. $q \sqsubseteq_{\Phi} (x, y)$ implies there exists a simulation relation Φ such that $(q, (x, y)) \in \Phi$. Also any pair $(q', (x', y')) \in \Phi$ implies

$$\sigma \in \overline{\Sigma}, q'_{\sigma} \in \delta^*(q', \sigma) \Rightarrow \exists (x'_{\sigma}, y'_{\sigma}) \in \gamma^*((x', y'), \sigma) \text{ and } (q'_{\sigma}, (x'_{\sigma}, y'_{\sigma})) \in \Phi.$$

$$q' \in Q_m \Rightarrow (x', y') \in X_m \times Y_m.$$

   By definition of SSC,

$$\sigma \in \overline{\Sigma}, (x'_{\sigma}, y'_{\sigma}) \in \gamma^*((x', y'), \sigma) \Rightarrow x'_{\sigma} \in \alpha^*(x', \sigma),$$

$$(x', y') \in X_m \times Y_m \Rightarrow x' \in X_m.$$

   Define a relation $\Phi' = \{(q', x') \mid (q', (x', y')) \in \Phi\}$. Then $(q', x') \in \Phi'$ implies,

$$\sigma \in \overline{\Sigma}, q'_{\sigma} \in \delta^*(q', \sigma) \Rightarrow \exists x'_{\sigma} \in \alpha^*(x', \sigma) \text{ such that } (q'_{\sigma}, x'_{\sigma}) \in \Phi'.$$

$$q' \in Q_m \Rightarrow x' \in X_m.$$

   Clearly, Φ' is simulation relation. By Definition 1, $q \sqsubseteq_{\Phi'} x$. Similarly, we can prove $q \sqsubseteq_{\Phi''} y$.

2. $q \sqsubseteq_{\Phi'} x$ implies there exists a simulation relation Φ' with $(q, x) \in \Phi'$. Also for any pair $(q', x') \in \Phi'$,

$$\sigma \in \overline{\Sigma}, q'_{\sigma} \in \delta^*(q', \sigma) \Rightarrow \exists x'_{\sigma} \in \alpha^*(x', \sigma) \text{ such that } (q'_{\sigma}, x'_{\sigma}) \in \Phi'.$$

$$q' \in Q_m \Rightarrow x' \in X_m.$$

Similarly, $q \sqsubseteq_{\Phi''} y$ implies there exists a simulation relation Φ'' with $(q, y) \in \Phi''$. Also for any pair $(q', y') \in \Phi''$,

$$\sigma \in \overline{\Sigma}, q'_\sigma \in \delta^*(q', \sigma) \Rightarrow \exists y'_\sigma \in \beta^*(y', \sigma) \text{ such that } (q'_\sigma, y'_\sigma) \in \Phi''.$$

$$q' \in Q_m \Rightarrow y' \in Y_m.$$

Define a relation $\Phi = \{(q', (x', y')) \mid (q', x') \in \Phi' \text{ and } (q', y') \in \Phi''\}$. Then $(q', (x', y')) \in \Phi$ implies $\forall \sigma \in \overline{\Sigma}, \forall q'_\sigma \in \delta^*(q', \sigma)$,

$$\exists (x'_\sigma, y'_\sigma) \in \alpha^*(x', \sigma) \times \beta^*(y', \sigma) \text{ such that } (q'_\sigma, (x'_\sigma, y'_\sigma)) \in \Phi.$$

$$q' \in Q_m \Rightarrow (x', y') \in X_m \times Y_m.$$

Thus, $q \sqsubseteq_\Phi (x, y)$. ■

The following corollary follows from Lemma 1 and serves as a necessary condition for the existence of a supervisor for enforcing bisimulation equivalence.

Corollary 2 Given G, R and S, if $G \| S \simeq R$, then $R \sqsubseteq G$.

Proof: $G \| S \simeq R$ implies $R \sqsubseteq G \| S$. By Definition 1, $Q_0 \sqsubseteq (X_0, Y_0)$. By Lemma 1, $Q_0 \sqsubseteq X_0$. Then by Definition 1, $R \sqsubseteq G$. ■

Remark 2 For a deterministic G and any R, it can be verified that $R \sqsubseteq G$ is equivalent to $L(R) \subseteq L(G)$ and $L_m(R) \subseteq L_m(G)$.

Next, we present our main result on existence of supervisor S for plant G such that $G \| S$ is bisimilar to specification R.

Theorem 4 Given nondeterministic G and R, there exists a Σ_u-compatible supervisor S such that $G \| S \simeq R$ if and only if there exists a Σ_u-compatible automaton T with state space $2^{X \times Q}$ such that $G \| T \simeq R$.

Proof: (*Only if*) Given G, R, and S such that $G \| S \simeq R$, we construct a Σ_u-compatible T such that $G \| T \simeq R$ and state space of T is $2^{X \times Q}$. We assume without loss of generality that all transitions of S participate in the composition with G. If a transition of S never participates in

$G\|S$, then we can remove this transition from S, and call the result as $\langle S\rangle$. Then $G\|\langle S\rangle = G\|S$ and every transition of $\langle S\rangle$ participates in the composed automaton $G\|\langle S\rangle$. We compute T from G, R and S as follows:

1. For $y \in Y$, we define

$$X_{syn}(y) := \{x \in X \mid \exists s \in \Sigma^*, x \in \alpha^*(X_0, s) \text{ and } y \in \beta^*(Y_0, s)\},$$

to be the states in G that are reachable by a same trace as is the state y of S. In other words, $x \in X_{syn}(y)$ if and only if (x, y) is a reachable state in $G\|S$. Since $G\|S \simeq R$, each such state (x, y) is bisimilar to some state $q \in Q$. Collection of all such states is denoted as $Q_{sim}(y)$, i.e.,

$$Q_{sim}(y) := \{q \in Q \mid \exists x \in X_{syn}(y), (x, y) \simeq q\}.$$

2. Label each state y of S by $lbl(y) \subseteq X \times Q$, such that $(x, q) \in lbl(y)$ if and only if $x \in X_{syn}(y), q \in Q_{sim}(y)$ and $(x, y) \simeq q$.

3. Define $[S]_0 := S$. For $k \geq 0$, $[S]_{k+1}$ is obtained by merging two states of $[S]_k$ carrying the same label, stop when $[S]_k = [S]_{k+1} =: [S]$.

Define T to be $[S]$. Then each state of T carries a unique label that is an element of $2^{X\times Q}$, and so the state space of T can be thought to be $2^{X\times Q}$. Next we prove that $G\|[S] \simeq R$ by induction on the number of mergers.

Base case : $[S]_0 = S$. So $G\|[S]_0 = G\|S \simeq R$.

Induction step: Suppose at step k, $G\|[S]_k \simeq R$. Denote the state of $[S]_k$ as $y^{(k)}$ with label $lbl(y^{(k)})$. At step $k+1$ of merging, suppose we merge $y^{(k)}, y'^{(k)}$, where $lbl(y^{(k)}) = lbl(y'^{(k)})$. Now we prove $G\|[S]_{k+1} \simeq G\|[S]_k$. Merging $y^{(k)}$ and $y'^{(k)}$ causes the merger of states $(x, y^{(k)})$ and $(x, y'^{(k)})$ of $G\|[S]_k$ for all $x \in X_{syn}(y^{(k)})$. Define automata $P_1, ...P_{|X_{syn}(y^{(k)})|}$ as follows:

1. $n := 0$, $P_n := G\|[S]_k$, $X_n := X_{syn}(y^{(k)})$.

2. If $X_n \neq \emptyset$, then $P_{n+1} := \overline{P_{n_{\langle (x,y^{(k)}),(x,y'^{(k)})\rangle}}}$, where $x \in X_n$, $X_{n+1} := X_n - \{x\}$, $n := n+1$; else stop, and $G\|[S]_{k+1} = P_n$.

Since $lbl(y^{(k)}) = lbl(y'^{(k)})$, it follows that both $(x, y^{(k)})$ and $(x, y'^{(k)})$ are bisimilar to same state of Q, i.e., $(x, y^{(k)}) \simeq (x, y'^{(k)})$. From the repeated application of Theorem 1 it follows that $G\|[S]_k = P_0 \simeq P_1 \simeq ... \simeq P_n = G\|[S]_{k+1}$. This proves the induction step. It remains to show that $[S]$ is Σ_u-compatible. Since S is Σ_u-compatible, and since Σ_u-compatibility is preserved under state mergers, $[S] = T$ is Σ_u-compatible.

(*If:*) The result follows by letting $S := T$. ■

Remark 3 Note that $G_1 \simeq G_2$ implies G_1 is trim (a marked state can be reached from every reachable state) if and only if G_2 is trim. So bisimilarity of controlled system and specification implies that the supervisor is nonblocking if and only if the specification automaton is trim. In other words, requiring a bisimilarity enforcing supervisor to be nonblocking is equivalent to requiring that the supervisor be bisimilarity enforcing and specification be trim. Thus there is no need to separately study nonblocking control in context of bisimulation equivalence specification (all that is needed is the specification automaton be trim).

Remark 4 From Theorem 4, an exhaustive search can be performed to determine the existence of a supervisor S over the state space $2^{X\times Q}$, the complexity of which is $O(2^{2^{|X|\times|Q|}})$. Since there may exist more systematic ways of searching for a desired S, the tightness of the upper bound complexity remains open.

In the following example, we illustrate the computation of T through labeling states of S by $lbl(\cdot) \in 2^{X\times Q}$ as in the proof of Theorem 4. Next we also show that labeling states of S just by $Q_{sim}(\cdot) \in 2^Q$ and using such labels to perform mergers of the states of S can yield a $[S]$ for which $G\|[S] \simeq R$ need not hold. This example illustrates that it may not be possible to replace the labeling function $lbl(\cdot)$ used in the proof of Theorem 4 by something simpler such as $Q_{sim}(\cdot)$.

Example 5 Consider nondeterministic G and S as shown in Figure 4.2.

$G\|S$ and R are shown in Figure 4.3, and we can easily see that $G\|S \simeq R$. By step 1, we compute $X_{syn}(y)$:

$$X_{syn}(y_0) = \{x_0\}, X_{syn}(y_1) = \{x_2, x_3\}, X_{syn}(y_2) = \{x_3, x_4\},$$

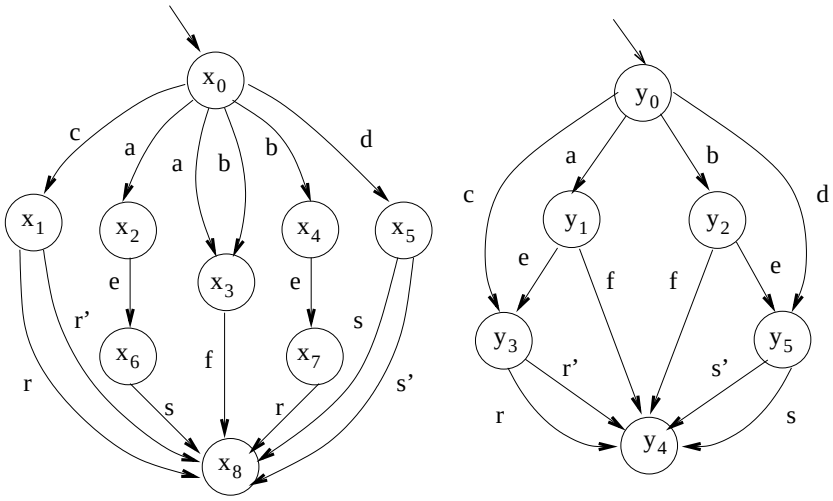

Figure 4.2 Plant G (left) and supervisor S (right)

$$X_{syn}(y_3) = \{x_1, x_6\}, X_{syn}(y_5) = \{x_5, x_7\}, X_{syn}(y_4) = \{x_8\}.$$

Since

$$(x_0, y_0) \simeq q_0, (x_2, y_1) \simeq q_2, (x_3, y_1) \simeq q_3, (x_3, y_2) \simeq q_3, (x_4, y_2) \simeq q_2,$$
$$(x_1, y_3) \simeq q_1, (x_6, y_3) \simeq q_5, (x_8, y_4) \simeq q_5, (x_5, y_5) \simeq q_4, (x_7, y_5) \simeq q_5$$

$Q_{sim}(y)$ is computed as:

$$Q_{sim}(y_0) = \{q_0\}, Q_{sim}(y_1) = \{q_2, q_3\}, Q_{sim}(y_2) = \{q_2, q_3\},$$
$$Q_{sim}(y_3) = \{q_1, q_5\}, Q_{sim}(y_4) = \{q_5\}, Q_{sim}(y_5) = \{q_4, q_5\}.$$

Then by step 2, label of each y is given by:

$$lbl(y_0) = \{(x_0, q_0)\}, lbl(y_1) = \{(x_2, q_2), (x_3, q_3)\}, lbl(y_2) = \{(x_3, q_3), (x_4, q_2)\},$$
$$lbl(y_3) = \{(x_1, q_1), (x_6, q_5)\}, lbl(y_4) = \{(x_8, q_5)\}, lbl(y_5) = \{(x_5, q_4), (x_7, q_5)\}.$$

No states can be merged since no states have the same label, so step 3 simply yields $T = S$.

In contrast, if we label each y of S by $Q_{sim}(y)$, and merge two states y and y' having the same label, then since $Q_{sim}(y_1) = Q_{sim}(y_2)$, we merge y_1 and y_2. The resulting state machine $[S]$ is shown in Figure 4.4. The synchronous composition $G||[S]$ is shown in Figure 4.4. It can

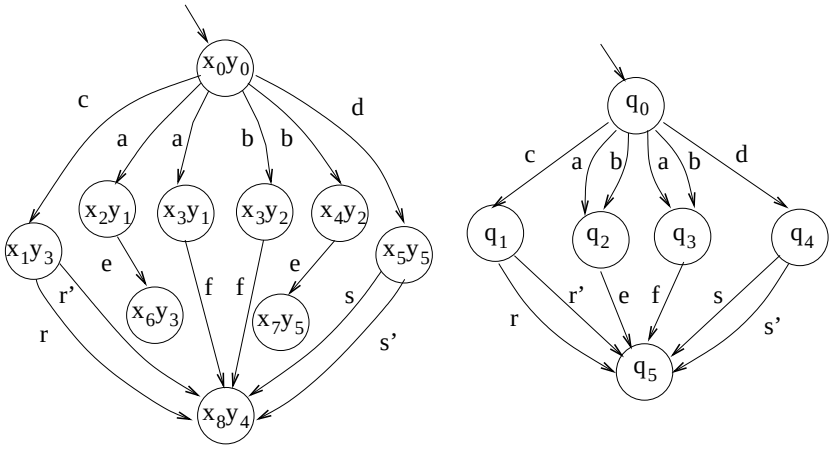

Figure 4.3 Controlled system $G\|S$ (left) and specification R (right)

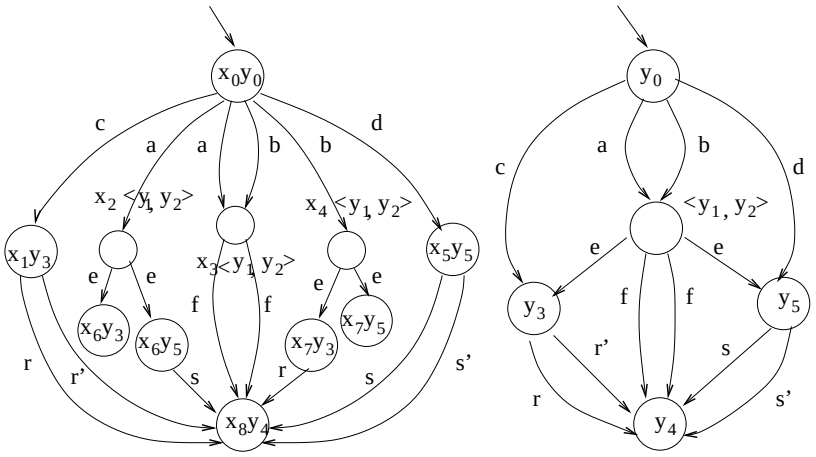

Figure 4.4 $G\|[S]$ (left) and $[S]$ (right)

be seen that $G\|[S] \not\simeq R$, since the states $(x_2, \langle y_1, y_2\rangle)$ and $(x_4, \langle y_1, y_2\rangle)$ of $G\|[S]$ are bisimilar to no state of R.

Now we revisit the motivating example.

Example 6 We need to find a Σ_u-compatible supervisor S such that $(G\|S)\uparrow\hat{\Sigma} \simeq R$, where $\hat{\Sigma} = \Sigma - \{reset\}$. Such a supervisor S is shown in Figure 4.5. The synchronous composition of G and S is drawn in Figure 4.6. The following bisimulation relation Φ exists between

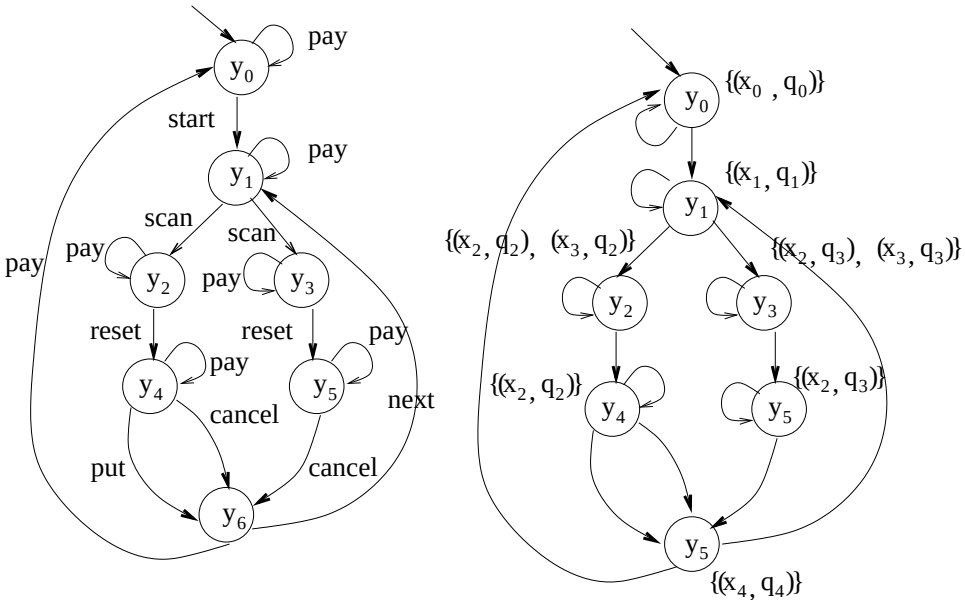

Figure 4.5 Supervisor S (left) and labeling of its states (right)

$(G\|S)\uparrow\hat{\Sigma}$ and R:

$$\begin{aligned}\Phi \;=\; &\{(x_0y_0, q_0), (x_1y_1, q_1), (x_2y_2, q_2), (x_2y_3, q_3), (x_3y_2, q_2), (x_3y_3, q_3),\\ &(x_2y_4, q_2), (x_2y_5, q_3), (x_4y_6, q_4), (q_0, x_0y_0), (q_1, x_1y_1), (q_2, x_2y_2),\\ &(q_2, x_3y_2), (q_3, x_2y_3), (q_3, x_3y_3), (q_2, x_2y_4), (q_3, x_2y_5), (q_4, x_4y_6)\}.\end{aligned}$$

Thus, the controlled system is bisimilar to the specification with respect to $\hat{\Sigma}$. States in S can be labeled by elements of $2^{X\times Q}$ as guaranteed by Theorem 4 (shown in Figure 4.5). A state $(x, q) \in X \times Q$ belongs to the label of a state $y \in Y$ of S if (x, y) appears in $G\|S$ (i.e., exists a common trace from x_0 to x in G and from y_0 to y in S so (x, y) is a reachable state

of $G\|S$), and (x, y) is bisimilar to state q of R. All states of S with identical labels may be merged to obtain the state machine T stated in Theorem 4. T is same as S in this case, and so $G\|T = G\|S$.

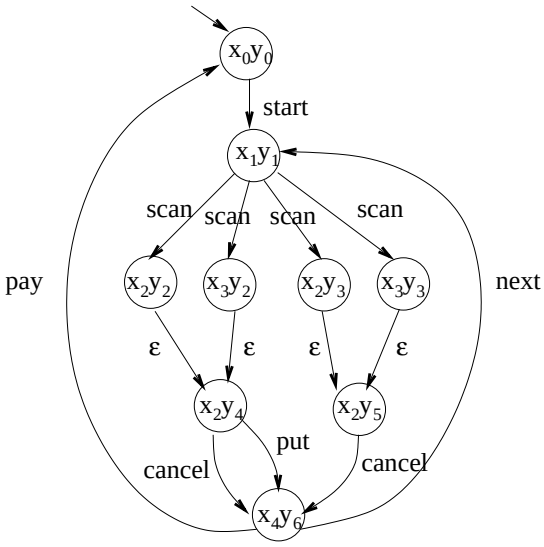

Figure 4.6 Controlled system $(G\|S)\uparrow\hat{\Sigma}$

4.3 Specialization to Deterministic Plant

In the earlier sections, we studied the supervisory control problem for enforcing bisimilarity in the setting of nondeterministic plants. In this section, we study the specialized case when plant is deterministic and specification is (possibly) nondeterministic and show that now the problem can be polynomially solved.

4.3.1 State-Controllability

In the deterministic setting, the controllability of specification language $L(R)$ with respect to plant language $L(G)$ and uncontrollable event set Σ_u is defined as:

$$L(R)\Sigma_u \cap L(G) \subseteq L(R).$$

This definition of "language-controllability" requires the following extension to the nondeterministic setting where instead of language models, automata models are used for plant and

specification.

Definition 8 Given plant G and an automaton $G' = (X', \Sigma, \alpha', X'_0)$ with $L(G') \subseteq L(G)$, we say G' is *state-controllable with respect to* G *and* Σ_u if

$$s \in L(G'), \sigma \in \Sigma_u \text{ such that } s\sigma \in L(G) \Rightarrow \forall x' \in \alpha'^*(X'_0, s), \sigma \in \Sigma(x').$$

G' is state-controllable with respect to G if for trace s in $L(G')$ and uncontrollable event σ defined at *some* state reachable by s in G, σ is defined at *all* states reachable by s in G'. Clearly, state-controllability implies language-controllability; the converse need not hold.

Remark 5 In contrast, the language-controllability requires the following:

$$s \in L(G'), \sigma \in \Sigma_u \text{ such that } s\sigma \in L(G) \Rightarrow \exists x' \in \alpha'^*(X'_0, s), \sigma \in \Sigma(x').$$

It is clear that when G' is deterministic, state-controllability of G' with respect to G and Σ_u reduces to language-controllability of $L(G')$ with respect to $L(G)$ and Σ_u. So when G' is deterministic, we can simply use the term controllability, dropping the prefix "state" or "language".

It is clear that Σ_u-compatibility implies state-controllability, i.e., the latter is a weaker notion. On the other hand, a state-controllable automaton can be converted to a Σ_u-compatible one without altering the controlled behavior. The following lemma establishes a type of equivalence between Σ_u-compatibility and state-controllability.

Lemma 2 Suppose S is state-controllable with respect to G and Σ_u. Define S' as S augmented with self-loops at each state on undefined uncontrollable events at the state. Then S' is Σ_u-compatible and $G\|S \simeq G\|S'$.

Proof: Since S is state-controllable, for any state (x, y) of $G\|S$ such that x has uncontrollable events defined, S also has those events defined at y. Therefore, adding self-loops at each state on undefined uncontrollable events in S does not change the result of synchronous composition. It follows that $G\|S = G\|S'$. Thus $G\|S \simeq G\|S'$. ■

It is known that language-controllability is closed under intersection for prefix-closed languages. We prove that state-controllability is also preserved under synchronous composition of automata.

Lemma 3 Suppose G_1 and G_2 are state-controllable with respect to G and Σ_u. Then so is $G_1 \| G_2$.

Proof: Pick $s \in L(G_1 \| G_2) = L(G_1) \cap L(G_2)$, and $\sigma \in \Sigma_u$ such that $s\sigma \in L(G)$. Let (x_1, x_2) be a state reached by execution of s in $G_1 \| G_2$. Then for $i = 1, 2$, from state-controllability of G_i, σ is defined at x_i in G_i. So σ is defined at (x_1, x_2) of $G_1 \| G_2$. This completes the proof. ■

4.3.2 Test for State-Controllability

We present below an algorithm of polynomial complexity for verifying state-controllability of an automaton G' with respect to a plant G.

Algorithm 1 Algorithm for testing state-controllability of $G' = (X', \Sigma, \alpha', X_0')$ with respect to $G = (X, \Sigma, \alpha, X_0)$.

1. Construct $\overline{G}'$ by augmenting G' with a new state called *dump*, and by adding transitions at each state of G' on each undefined uncontrollable event at that state to the *dump* state, i.e., $\overline{G}' = (X' \cup \{dump\}, \Sigma, \overline{\alpha}', X_0')$, where

$$\forall x' \in X', \sigma \in \overline{\Sigma} : \overline{\alpha}'(x', \sigma) = \begin{cases} \alpha'(x', \sigma) & \text{if } \sigma \in \Sigma(x') \\ dump & \text{if } \sigma \in \Sigma_u - \Sigma(x') \end{cases}$$

2. Obtain $G \| \overline{G}'$.

3. G' is state-controllable with respect to G if and only if there does not exist $\overline{x} \in X$ such that $(\overline{x}, dump)$ is reachable in $G \| \overline{G}'$.

The following lemma is needed in order to prove Algorithm 1.

Lemma 4 G' is state-controllable with respect to G if and only if

$$\forall x' \in X', \forall x \in X_{syn}(x') : \Sigma(x) \cap \Sigma_u \subseteq \Sigma(x') \cap \Sigma_u. \tag{4.1}$$

Proof: (*If*) For each state $q \in Q$ of R define $X_{syn}(q) \subseteq X$ to be the set of states of G that are reachable by a common trace, i.e.,

$$X_{syn}(q) := \{x \in X \mid \exists s \in L(R) \cap L(G) \text{ s.t. } q \in \delta^*(Q_0, s), \text{ and } x \in \alpha^*(X_0, s)\}.$$

Pick $s \in L(R)$, $x \in \alpha^*(X_0, s)$ such that $\sigma \in \Sigma(x) \cap \Sigma_u$. Then for any $q \in \delta^*(Q_0, s)$, $x \in X_{syn}(q)$ and so from hypothesis, $\sigma \in \Sigma(q) \cap \Sigma_u$. It follows that R is state-controllable with respect to G.

(*Only If*) Pick $q \in Q, x \in X_{syn}(q), \sigma \in \Sigma(x) \cap \Sigma_u$. It suffices to show that $\sigma \in \Sigma(q)$. Since $x \in X_{syn}(q)$, exists $s \in L(R)$ such that $q \in \delta^*(Q_0, s)$ and $x \in \alpha^*(X_0, s)$. Then from state-controllability, $\sigma \in \Sigma(q)$. ■

The following theorem establishes the correctness of Algorithm 1.

Theorem 5 Algorithm 1 is correct.

Proof: Let $(x, x') \in X \times X'$ be a state reachable in $G \| G'$. Then it is obvious that $(x, x') \in X_{syn}(x') \times \{x'\}$, and so for state-controllability to hold, $\Sigma(x) \cap \Sigma_u \subseteq \Sigma(x') \cap \Sigma_u$ must hold. On the other hand, if this condition is violated, i.e., if exists $\sigma \in \Sigma(x) \cap \Sigma_u - \Sigma(x')$, then a transition $(x, \sigma, \overline{x})$ for some $\overline{x} \in X$ is defined in G and the transition $(x', \sigma, dump)$ is defined in $\overline{G}'$. So the transition $((x, x'), \sigma, (\overline{x}, dump))$ is defined in $G \| \overline{G}'$. It follows that a state $(\overline{x}, dump)$ is reachable in $G \| \overline{G}'$ if and only if G' is not state-controllable with respect to G. ■

Remark 6 Since G and G' are nondeterministic, their number of transitions is $O(|X|^2)$ and $O(|X'|^2)$ respectively. So the complexity of constructing $G \| \overline{G}'$ is $O(|X|^2 \times |X'|^2)$, and the complexity of checking the reachability of $(\cdot, dump)$ in $G \| \overline{G}'$ is also $O(|X|^2 \times |X'|^2)$. So the complexity of the algorithm for testing state-controllability of G' with respect to G is $O(|X|^2 \times |X'|^2)$, i.e., it is quadratic in the number of states of both G and G' (equivalently, linear in size of G and G').

Example 7 Consider the automata G_1 and G_2 shown in Figure 4.7. Then following the definition of the state-controllability, it is obvious that G_2 is not state-controllable with respect to G_1 ($a \in L(G_1) \cap L(G_2), ac \in L(G_1)$, but c is not defined at y_1, one of the states reached by the execution of a in G_2).

Now we verify the state-controllability of G_2 with respect to G_1 by our algorithm. The constructed $\overline{G}_2$ and $G_1 \| \overline{G}_2$ are depicted in Figure 4.7. It can be seen that the state $(4, dump)$ is reachable in $G_1 \| \overline{G_2}$. Thus, from Algorithm 1, G_2 is not state-controllable with respect to G_1. It should be noted that $L(G_1) = L(G_2) = pr(ab + ac + ad)$, and so $L(G_2)$ is language-controllable with respect to $L(G_1)$.

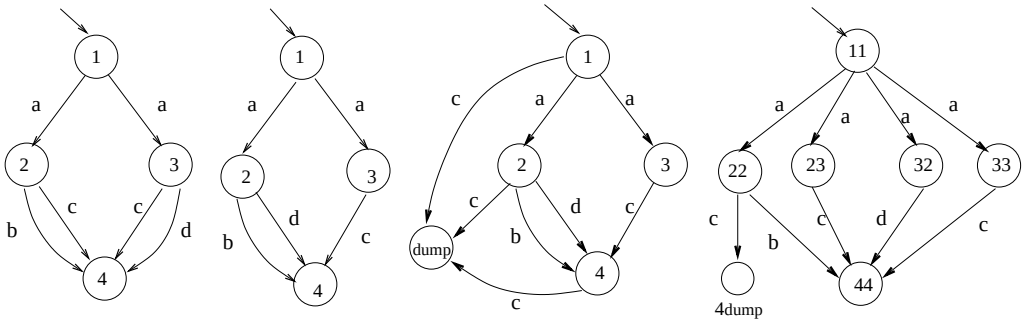

Figure 4.7 G_1 (first), G_2 (second), $\overline{G}_2$ (third), and $G_1 \| \overline{G}_2$ (forth)

4.3.3 Control of Deterministic Plant for Bisimilarity

Suppose exists Σ_u-compatible S such that $G \| S \simeq R$. Due to Σ_u-compatibility of S, we find that $G \| S$ is state-controllable. The bisimilarity of R with $G \| S$ however does not guarantee the state-controllability of R since as shown by the following example the state-controllability may not be preserved under the bisimilarity.

Example 8 Consider the plant G and two specifications R_1 and R_2 shown in Figure 4.8. Let $\Sigma_u = \{b\}$. It can be verified that $R_1 \simeq R_2$, and R_2 is state-controllable with respect to G. However, R_1 is not state-controllable since in G, $b \in \Sigma_u$ is defined after $\epsilon \in L(R)$, but in R, b is not defined at state $1 \in \delta^*(1, \epsilon)$.

The reason for the non-preservation of the state-controllability under bisimulation is the non-preservation of the set of events defined at a pair of bisimilar states. In the following, we introduce the notion of R^* which is the NSM R with its states renamed and transition function δ replaced by δ^*. We show that R^* possesses the properties that $R^* \simeq R$ and further, if R is state-controllable then R^* remains state-controllable.

Definition 9 Given $R = (Q, \Sigma, \delta, Q_0, Q_m)$, we define $R^* = (Q^*, \Sigma, \delta^*, Q_0^*, Q_m^*)$, where

- $Q^* := \{q^* \mid q \in Q\}$,
- $\forall q^* \in Q^*$:
  $\delta^*(q^*, \epsilon) := \epsilon^*(q), \;\; \delta^*(q^*, \sigma) := \epsilon^*(\delta(\epsilon^*(q), \sigma))$,
- $Q_0^* := \{q^* \mid q \in Q_0\}$, and
- $Q_m^* := \{q^* \mid q \in Q_m\}$.

Note that in the absence of ϵ-transitions, $R^* = R$.

Figure 4.8 shows an example illustrating Definition 9.

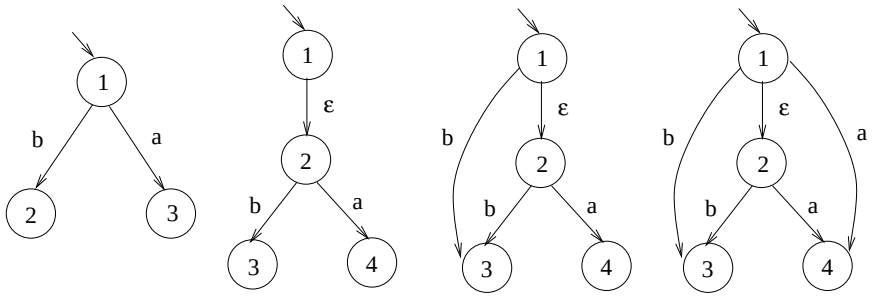

Figure 4.8 G (first), R_1 (second), R_2 (third), and R_2^* (fourth)

Since the definition of bisimilarity between two NSMs $G_i = (X_i, \Sigma, \alpha_i, X_{0i}, X_{mi})$ $(i = 1, 2)$ depends on the transition function α_i^* (and not α_i), it is obvious that any two NSMs R and R^* are bisimilar. This and another property is summarized in the following lemma.

Lemma 5 Consider NSMs R and R^*. Then

1. $R \simeq R^*$.
2. $\forall q \in Q$: $\Sigma(q) \subseteq \Sigma(q^*)$.

Proof:

1. It is easy to see that we can pick $\Phi := \{(q, q^*), (q^*, q) \mid q \in Q\}$ to establish $R \simeq R^*$.
2. Follows from the fact that $\delta(q, \sigma) \subseteq \delta^*(q^*, \sigma)$ for all $q \in Q$ and $\sigma \in \Sigma$. ■

Prior to obtaining the main result of this section, we need to prove the following lemmas.

Lemma 6 Given a deterministic plant G, a possibly nondeterministic R, if $R \sqsubseteq G$, then $G \| R \simeq R$.

Proof: Choose

$$\Phi := \{((x,q),q),(q,(x,q))|(x,q) \text{ state in } G||R \text{ s.t. } q \sqsubseteq x\}.$$

Since $R \sqsubseteq G$, $L(R) \subseteq L(G)$, and so for each $s \in L(R)$ and $q \in \delta^*(Q_0, s)$, exists singleton $\{x\} = \alpha^*(X_0, s)$ such that $q \sqsubseteq x$. So for each $q \in Q$, exists a unique $x \in X$ such that $((x,q),q) \in \Phi$. Since Φ is symmetric, it suffices to show that it is a simulation relation. Pick $(q,(x,q)) \in \Phi$ and σ-successor $q' \in \delta^*(q,\sigma)$ for some $\sigma \in \overline{\Sigma}$. Since $q \sqsubseteq x$, there exists x' such that $\{x'\} = \alpha(x,\sigma)$, $q' \sqsubseteq x'$ and (x',q') is a state in $G||R$. It follows that $(q',(x',q')) \in \Phi$. Similarly, pick $((x,q),q) \in \Phi$ and σ-successor $(x',q') \in \alpha^*(x,\sigma) \times \delta^*(q,\sigma)$ for some $\sigma \in \overline{\Sigma}$. Then (x',q') is a state in $G\|R$. Also since $x' \in \alpha^*(x,\sigma)$ is unique and $q \sqsubseteq x$, $q' \sqsubseteq x'$. It follows that $((x',q'),q') \in \Phi$ as desired.

Further if q is marked, $q \sqsubseteq x$ implies that x is marked. I.e., if q is marked, then (x,q) is marked. Moreover, if q is not marked, then (x,q) is not marked. So, for $x \in X$, $q \in Q$ such that $q \sqsubseteq x$, $(x,q) \in X_m \times Q_m \Leftrightarrow q \in Q_m$.

Finally, $R \sqsubseteq G$ implies $Q_0 \sqsubseteq X_0$, which further implies $((x_0,q_0),q_0),(q_0,(x_0,q_0)) \in \Phi$ for $q_0 \in Q_0$ and $x_0 \in X_0$. Thus, by definition of bisimulation equivalence, $G\|R \simeq R$. ■

Lemma 7 For two bisimilar automata $R_1 \simeq R_2$, if R_1 is state-controllable with respect to G, then R_2^* is state-controllable with respect to G.

Proof: Since R_1 is state-controllable, from Lemma 4, $q_1 \in \delta_1^*(Q_{01}, s)$ and $x \in X_{syn}(q_1)$ implies

$$\Sigma(x) \cap \Sigma_u \subseteq \Sigma(q_1) \cap \Sigma_u. \tag{4.2}$$

To prove that R_2^* is state-controllable with respect to G, it suffices to show that for each state q_2^* of R_2^* and for each $x \in X_{syn}(q_2^*)$ it holds that

$$\Sigma(x) \cap \Sigma_u \subseteq \Sigma(q_2^*) \cap \Sigma_u.$$

Since (4.2) holds, it suffices to show that $\Sigma(q_1) \subseteq \Sigma(q_2^*)$. Since $x \in X_{syn}(q_2^*)$, exists a trace $s \in L(R_2^*)$ such that $q_2^* \in \delta_2^*(Q_{02}, s)$ and $x \in \alpha^*(X_0, s)$. Bisimilarity of R_1 and R_2 implies $R_1 \simeq R_2^*$ (from Lemma 5). From the fact that $R_1 \simeq R_2^*$ (which is equivalent to $Q_{01} \simeq Q_{02}^*$), we know $s \in L(R_1) = L(R_2^*)$ and exists a state $q_1 \in \delta_1^*(Q_{01}, s)$ such that $q_1 \simeq q_2^*$ (obtained by inductively extending the definition of bisimulation equivalence from one step to multiple steps).

From Lemma 5, $R_1 \simeq R_2$ implies $R_1^* \simeq R_2^*$. Since

$$\Sigma(q_i^*) = \{\sigma \in \Sigma \mid \delta_i^*(q_i^*, \sigma) \neq \emptyset\}(i = 1, 2),$$

$R_1^* \simeq R_2^*$ implies $\Sigma(q_1^*) = \Sigma(q_2^*)$. From Lemma 5, $\Sigma(q_1) \subseteq \Sigma(q_1^*)$ and so we have $\Sigma(q_1) \subseteq \Sigma(q_2^*)$ as desired. ■

The following lemma holds for a deterministic plant but not in general.

Lemma 8 Let G be a deterministic plant, and S be a Σ_u-compatible supervisor. Then $G\|S$ is state-controllable.

Proof: To prove state-controllability of $G\|S$ with respect to G, we apply the test of Algorithm 1 to $G\|\overline{(G\|S)}$. Since G is deterministic if a state $(x, (x', y))$ is reached in $G\|(G\|S)$, then $x = x'$. We claim that there does not exist a state $(x, (x, y))$ reachable in $G\|\overline{(G\|S)}$ from where a state $(\cdot, dump)$ in $G\|\overline{(G\|S)}$ can be reached. For this to hold, every uncontrollable event defined at x in G must also be defined at (x, y) in $G\|S$, i.e., we need to show that $\Sigma(x) \cap \Sigma_u$ is a subset of $\Sigma(x, y) \cap \Sigma_u$. This follows since

$$\Sigma(x) \cap \Sigma_u = \Sigma(x) \cap \Sigma(y) \cap \Sigma_u = \Sigma(x, y) \cap \Sigma_u,$$

where in the first equality we have used the Σ_u-compatibility of S, which implies $\Sigma_u \subseteq \Sigma(y)$. This completes the proof. ■

Now we are ready to present a necessary and sufficient condition for the existence of a bisimilarity enforcing supervisor for a deterministic plant.

Theorem 6 Given a deterministic plant G and a possibly nondeterministic specification R, there exists a Σ_u-compatible supervisor S such that $G\|S \sim R$ if and only if $R \sqsubseteq G$ and R^* is state-controllable.

Proof: (*If*) Choose $S = R_u^*$ (where R_u^* is obtained by adding in R^* self-loops at each state on undefined uncontrollable events). Then S is Σ_u-compatible. Also from Lemma 2, $G||S = G||R_u^* = G||R^*$. Further since $R^* \sqsubseteq G$ (since $R^* \simeq R$ and $R \sqsubseteq G$), and G is deterministic, from Lemma 6, $G||R^* \simeq R^*$. And so $G||S = G||R^* \simeq R^* \simeq R$.

(*Only if*) Since S is Σ_u-compatible, from Lemma 8, $G||S$ is state-controllable. Also since $R \simeq G||S$, it follows from Lemma 7 that R^* is state-controllable. Also since $G||S \simeq R$, $R \sqsubseteq G||S$, which implies that $R \sqsubseteq G$. This completes the proof. ■

Remark 7 The statement of Theorem 6 above is a slight refinement of the one appearing in the conference version of the paper (Zhou et al., 2004, Theorem 4). The two statements are identical when there are no ϵ-transitions (so that $R^* = R$).

Remark 8 From Theorem 6, the complexity of checking the existence of a supervisor for enforcing bisimilarity for a deterministic plant is $O(|X| \times |Q|^2)$, which is linear in the sizes of the plant and the specification. Moreover, when the existence conditions are satisfied, R_u^* serves as a supervisor, i.e., the complexity of synthesizing a supervisor is linear in the size of the specification. It is interesting to note that when specification is deterministic but the plant is nondeterministic, the complexity of existence as well as synthesis is again polynomial Kumar et al. (2005b). In contrast, the situation seems to be different when both the plant and the specification are nondeterministic.

4.4 Conclusion

For control of nondeterministic plant for bisimulation equivalence, we obtained a small model theorem showing that a supervisor for enforcing bisimulation equivalence between the specification and the controlled system exists if and only if it exists over a certain finite state space, namely the power set of Cartesian product of the plant and the specification state spaces.

For the special case of deterministic plants we obtained a necessary and sufficient condition for the existence of a bisimilarity enforcing supervisor which can be verified polynomially in

both plant and specification states. This happens to be the same as the complexity of verifying the existence of a supervisor when both plant and specification are deterministic. A stronger notion of controllability, called state-controllability, is introduced as part of the necessary and sufficient condition for the existence of such a supervisor. State-controllability is stronger than the "language-controllability", where the latter is a property of language models, and the former is a property of the automata models. We presented an algorithm of linear complexity for testing state-controllability matching the complexity of testing the language-controllability.

CHAPTER 5. SUPERVISORY CONTROL FOR BISIMULATION EQUIVALENCE UNDER PARTIAL OBSERVATION

In this chapter, we study the control of DESs to ensure bisimilarity of the controlled system and a given specification under partial observation. we extend the small model theorem by showing that a control and observation compatible supervisor for enforcing bisimulation equivalence between the specification and the controlled system exists if and only if it exists over a certain finite state space. For the special case of deterministic plants, we introduce the notions of *state-achievability* and *state-achievability-bisimilar* as part of the existence condition, and develop effective algorithms for verify the existence conditions as well as for synthesizing a supervisor when the existence condition holds. We show that the complexity of verifying the existence of a controller is polynomial, whereas that of computing a controller (when one exists) is singly exponential. The proposed approach can be applied to enforce any property that depends on branching and sequential behavior.

5.1 A Motivating Example

To motivate bisimilarity control under partial observation, we introduce the following manufacturing example, a solution to which is discussed latter.

Example 9 Consider a manufacturing system (shown in Figure 5.1) consisting of two workstations, one robot and three storage-stations. The robot moves among the workstations and storage-stations on guide rails. Initially, the robot departs from workstation 1 and nondeterministically travels on one of the rails (event a). On rail 1, the robot picks up a part from storage-station 1 (event b_1) and then delivers this part to workstation 2 for processing (event c). After the processing, robot returns the part to storage-station 1 (event b_1). On rail 2,

the robot either picks up a part from storage-station 2 (event b_2) or from storage-station 3 (event b_3), and then delivers the part to workstation 2 for processing (event c). After the processing, the robot returns the part to either storage-station 2 or 3 (event b_2 or b_3). Not returning the part to its original storage-station is undesirable. After returning the part to the storage-station, the robot goes back to workstation 1 (event a) from where the entire process may be repeated. The state machine model G of the system is drawn in Figure 5.2.

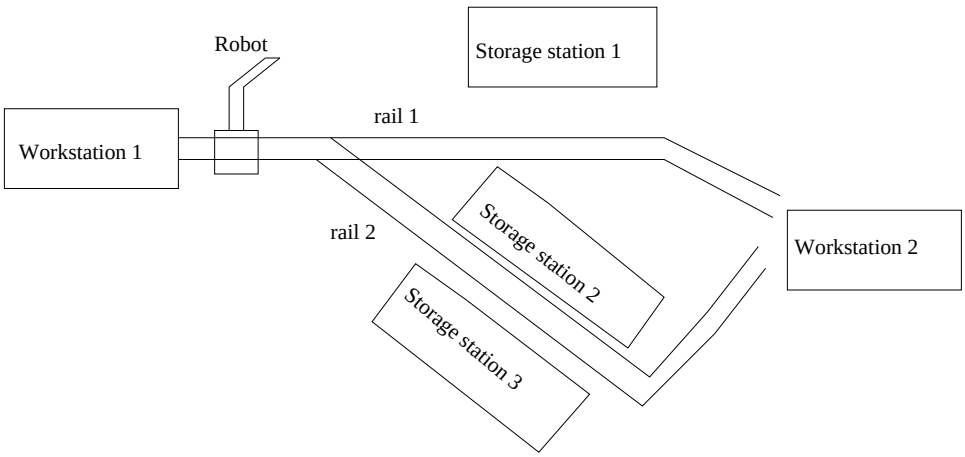

Figure 5.1 A manufacturing system

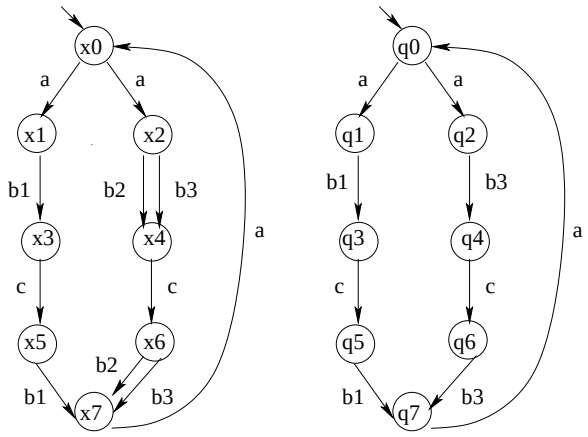

Figure 5.2 Model G (left) and Specification R (right)

The specification R, also drawn in Figure 5.2, shows the acceptable behavior. According to the specification, the robot returns any processed part to the same work-station from where it picked that part.

A part once picked must be delivered to workstation 2 for processing, i.e., the event c is uncontrollable. Only the events a and c are completely observable. Events b_1, b_2 and b_3 are observationally indistinguishable. Thus, we have $\Sigma = \{a, b_1, b_2, b_3, c\}$, $\Sigma_u = \{c\}$, and the observation mask M is given by, $M(a) = a$, $M(b_1) = M(b_2) = M(b_3) \neq \epsilon$ and $M(c) = c$. The control goal is to find a (Σ_u, M)-compatible supervisor S such that the controlled system $G\|S$ is bisimilar to the specification R.

5.2 Control for Nondeterministic Plants under Partial Observation

Definition 10 Let $\Sigma_u \subseteq \Sigma$ be the set of uncontrollable events and $M : \overline{\Sigma} \rightarrow \overline{\Delta}$ be the observation mask, then

- S is called Σ_u-*compatible* if $\forall y \in Y$ and $\forall a \in \Sigma_u$, $\beta(y, a) \neq \emptyset$.
- S is called M-*compatible* if $\forall y \in Y$ and $\forall a, b \in \Sigma(y)$, if $M(a) = M(b)$, then $\beta(y, a) = \beta(y, b)$, where it is assumed that an ϵ-transition is implicitly defined as a self-loop.
- S is called (Σ_u, M)-*compatible* if S is Σ_u-compatible and M-compatible.

We establish the main result that proves the decidability of bisimilarity enforcing control under partial observation by extending the "small model theorem" from the setting of complete observation (Chapter 4) to the setting of partial observation. The small model theorem states that a bisimilarity enforcing Σ_u-compatible supervisor exists if and only if it exists over the state space $2^{X \times Q}$, where X is the state space of plant G and Q is the state space of specification R. The sufficiency is clearly obvious, while the key idea behind necessity is that given a Σ_u-compatible bisimilarity enforcing supervisor S (i.e., $G\|S \simeq R$), each state $y \in Y$ of S can be labeled by $lbl(y) \in 2^{X \times Q}$, and then states carrying identical labels can be merged to obtain state machine T with state space $2^{X \times Q}$ such that T is Σ_u-compatible and $G\|T \simeq R$. We recall that $(x, q) \in X \times Q$ belongs to $lbl(y)$ if and only if (x, y) is a state in $G\|S$ that is bisimilar to state q of R. We use the same labeling function for extending the small model theorem to the setting of partial observation.

The following example illustrates how the labels are computed.

Example 10 Consider G, S and R shown in Figure 5.3. It can be verified that $G \| S \simeq R$ since the following bisimulation relation exists between $G \| S$ and R (for the sake of simplicity we write a state (x, y) simply as xy):

$$\Phi := \{(x_0y_0, q_0), (x_1y_1, q_1), (x_1y_2, q_2), (x_2y_3, q_3), (x_3y_4, q_4), (x_4y_5, q_5), (x_1y_6, q_1), (x_1y_7, q_2), (x_2y_8, q_3), (x_3y_8, q_4), (x_4y_8, q_5), (q_0, x_0y_0), (q_1, x_1y_1), (q_2, x_1y_2), (q_3, x_2y_3), (q_4, x_3y_4), (q_5, x_4y_5), (q_1, x_1y_6), (q_2, x_1y_7), (q_3, x_2y_8), (q_4, x_3y_8), (q_5, x_4y_8)\}.$$

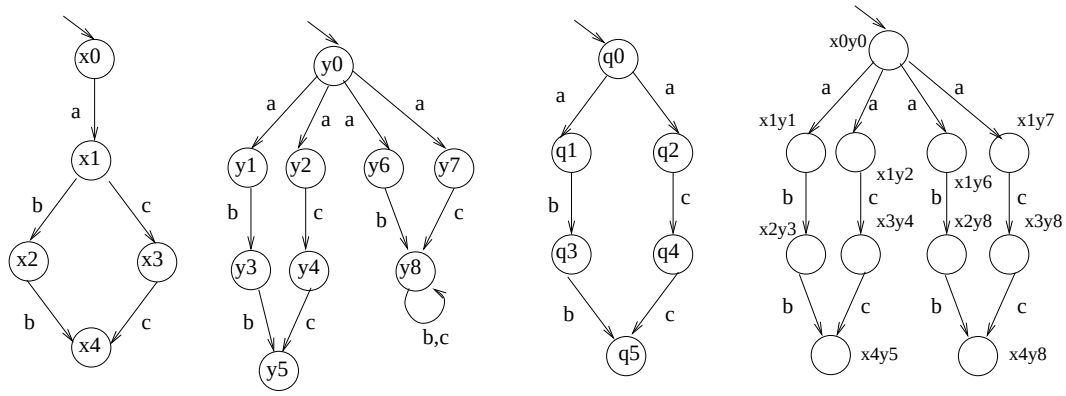

Figure 5.3 G (first), S (second), R (third), and $G \| S$ (fourth)

	x_0	x_1	x_2	x_3	x_4	$lbl(\cdot)$
y_0	q_0	-	-	-	-	$\{(x_0, q_0)\}$
y_1	-	q_1	-	-	-	$\{(x_1, q_1)\}$
y_2	-	q_2	-	-	-	$\{(x_1, q_2)\}$
y_3	-	-	q_3	-	-	$\{(x_2, q_3)\}$
y_4	-	-	-	q_4	-	$\{(x_3, q_4)\}$
y_5	-	-	-	-	q_5	$\{(x_4, q_5)\}$
y_6	-	q_1	-	-	-	$\{(x_1, q_1)\}$
y_7	-	q_2	-	-	-	$\{(x_1, q_2)\}$
y_8	-	-	q_3	q_4	q_5	$\{(x_2, q_3), (x_3, q_4), (x_4, q_5)\}$

Table 5.1 Computation of labeling function for Example 10

Using this bisimulation relation, we construct Table 5.1 in order to compute the labeling function of each state in S. Entries in the top-most row (resp., left-most column) represent a state x in G (resp., y in S), and the corresponding cell entry is a state q in R such that

$q \simeq_\Phi (x, y)$. $lbl(y) \subseteq X \times Q$ contains all (x, q) pairs such that $q \simeq_\Phi (x, y)$. This labeling is depicted in Figure 5.4.

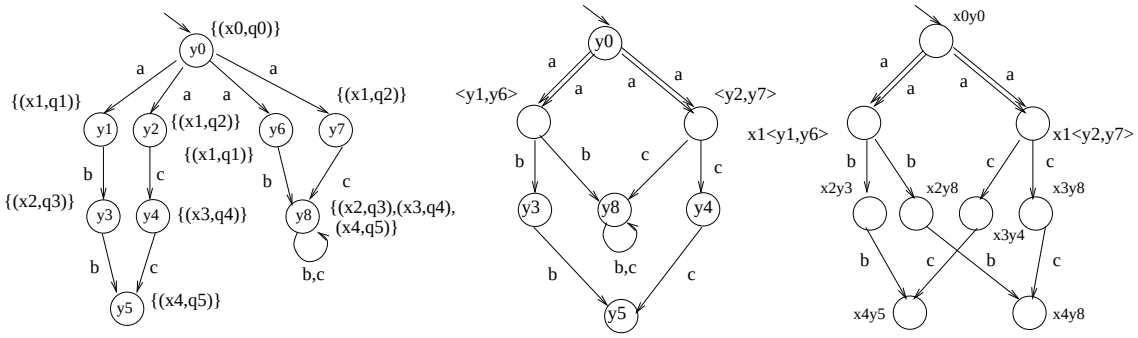

Figure 5.4 The labeling of states in S (left), T (middle), and $G\|T$ (right)

Since $lbl(y_1) = lbl(y_6)$ and $lbl(y_2) = lbl(y_7)$, we merge y_1 and y_6, and y_2 and y_7, respectively. The state machine T obtained by merging states in S carrying the merged state label is drawn in Figure 5.4. It can be seen that $G\|T \simeq R$.

Theorem 7 Given G and R, and a mask M, there exists a (Σ_u, M)-compatible supervisor S such that $G\|S \simeq R$ if and only if there exists a (Σ_u, M)-compatible state machine T with state space $2^{X\times Q}$ such that $G\|T \simeq R$.

Proof: (*Only If*) For necessity, suppose exists (Σ_u, M)-compatible S such that $G\|S \simeq_\Phi R$. Without loss of generality, all transitions of S participate in $G\|S$ (otherwise we can simply omit such transitions from S). Label each $y \in Y$ of S by $lbl(y) \subseteq X \times Q$ where $(x, q) \in lbl(y)$ if and only if (x, y) reachable in $G\|S$ and $q \in Q$ is such that $(x, y) \simeq_\Phi q$. Merge all states carrying the same label, and call the resulting state machine T. Then from the proof of Theorem 4, $G\|T \simeq R$, T is Σ_u-compatible, and state space of T is $2^{X\times Q}$. We claim that T is also M-compatible, which will prove the necessity. Suppose $y_1, y_2 \in Y$ are such that $lbl(y_1) = lbl(y_2)$. Then y_1 and y_2 are merged to obtaining $\langle y_1, y_2\rangle$. Also since there are no redundant transitions in S, it was shown in proof of Theorem 4 that,

$$\Sigma(y_1) = \cup_{(x,q)\in lbl(y_1)}\Sigma(q) = \cup_{(x,q)\in lbl(y_2)}\Sigma(q) = \Sigma(y_2).$$

So after merger, $\Sigma(\langle y_1, y_2\rangle) = \Sigma(y_1) = \Sigma(y_2)$. Since S is M-compatible, for any pair of indistinguishable events $a_1, a_2 \in \Sigma(y_1) = \Sigma(y_2)$, $\beta(y_1, a_1) = \beta(y_1, a_2)$ and $\beta(y_2, a_1) = \beta(y_2, a_2)$. So

$$\beta(\langle y_1, y_2\rangle, a_1) = \beta(y_1, a_1) \cup \beta(y_2, a_1) = \beta(y_1, a_2) \cup \beta(y_2, a_2) = \beta(\langle y_1, y_2\rangle, a_2).$$

Thus merger of states carrying the same label preserves M-compatibility, and so T is M-compatible.

(*If*) Set $S := T$, then S is (Σ_u, M)-compatible and $G\|S = G\|T \simeq R$.

This completes the proof. ■

Remark 9 From Theorem 7, an exhaustive search can be performed to determine the existence of a supervisor S over the state space $2^{X\times Q}$, the upper bound complexity of which is $O(2^{2^{|X|\times|Q|}})$. From this, the upper bound complexity of checking the existence of a supervisor under partial observation is same as the one under full observation. Better upper bounds may exist for the two cases, but are not known at this time.

Next we revisit the manufacturing example.

Example 11 Our goal is to find a (Σ_u, M)-compatible supervisor S such that $G\|S \simeq R$ (provided one exists). Such a supervisor is drawn in Figure 5.5. Since $\Sigma_u = \{c\}$, and c is defined at each state of S, S is Σ_u-compatible. Also state updates on indistinguishable pair of events b_1 and b_3 at states y_1, y_3, y_5, y_6, where they are both defined, are identical, implying that S is also M-compatible.

The controlled system $G\|S$ is also drawn in Figure 5.5, and it can be verified that $G\|S \simeq R$.

From Theorem 7, exists a (Σ_u, M)-compatible T with state space $2^{X\times Q}$ such that $G\|T \simeq R$. To obtain such a T, the labeling of each state in S is shown in Figure 5.6, and is computed using Table 5.2. State machine T is obtained by merging states in S carrying the same label. Since $lbl(y_1) = lbl(y_5)$, we merge y_1 and y_5. The resulting state machine T is drawn in Figure 5.6. T is (Σ_u, M)-compatible as expected. Moreover, $G\|T \simeq G\|S \simeq R$ (Figure 5.7).

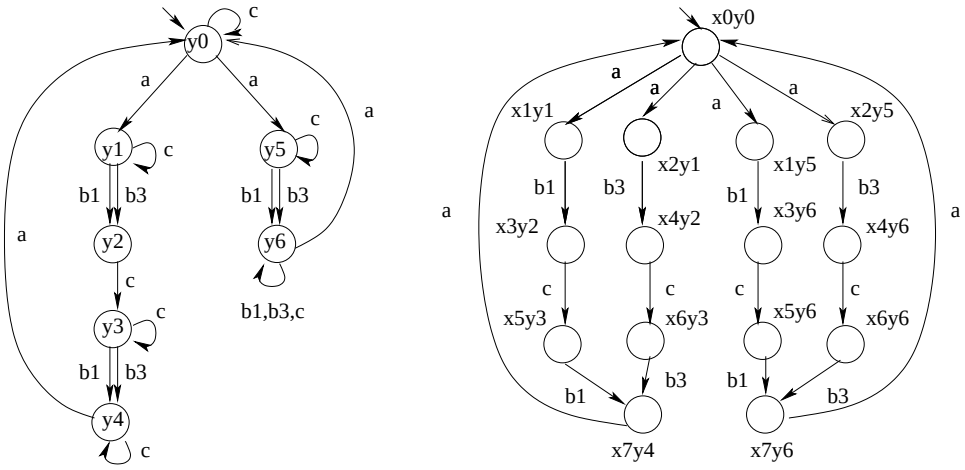

Figure 5.5 S (left) and $G \| S$ (right)

	x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	$lbl(\cdot)$
y_0	q_0	-	-	-	-	-	-	-	$\{(x_0, q_0)\}$
y_1	-	q_1	q_2	-	-	-	-	-	$\{(x_1, q_1), (x_2, q_2)\}$
y_2	-	-	-	q_3	q_4	-	-	-	$\{(x_3, q_3), (x_4, q_4)\}$
y_3	-	-	-	-	-	q_5	q_6	-	$\{(x_5, q_5), (x_6, q_6)\}$
y_4	-	-	-	-	-	-	-	q_7	$\{(x_7, q_7)\}$
y_5	-	q_1	q_2	-	-	-	-	-	$\{(x_1, q_1), (x_2, q_2)\}$
y_6	-	-	-	q_3	q_4	q_5	q_6	q_7	$\{(x_3, q_3), (x_4, q_4), (x_5, q_5), (x_6, q_6), (x_7, q_7)\}$

Table 5.2 Computation of labeling function for Example 11

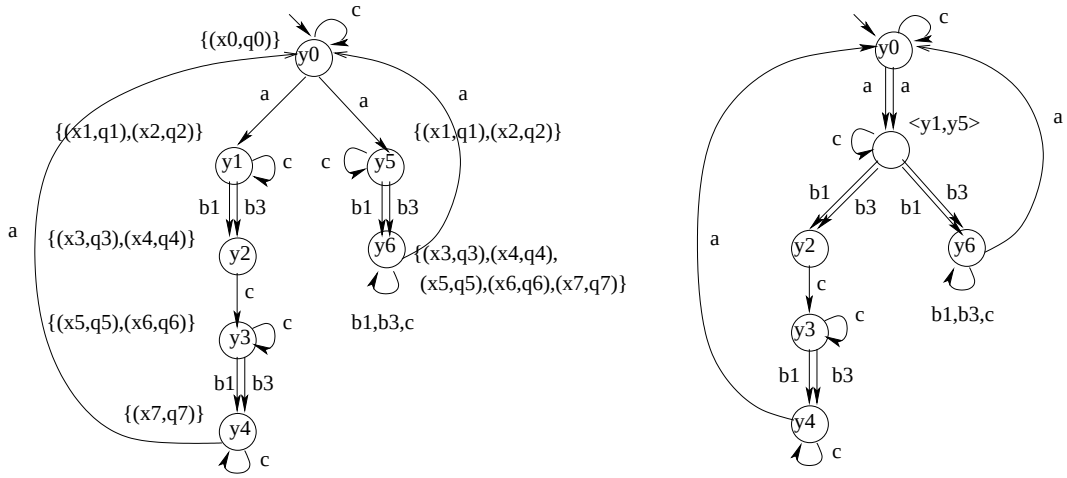

Figure 5.6 The labeling of states in S (left) and T (right)

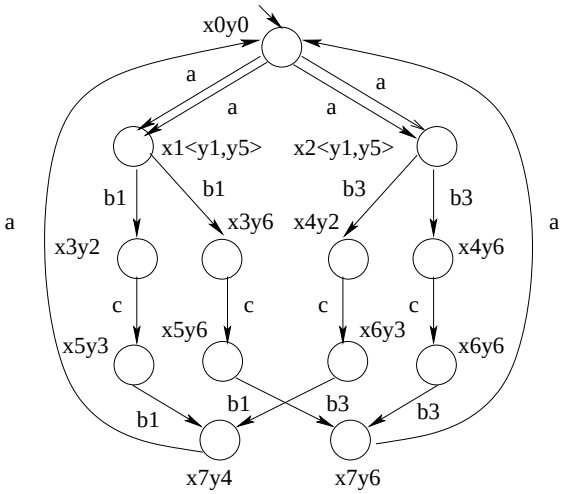

Figure 5.7 $G \| T$

5.3 Specialization to Deterministic Plants

To motivate the case of deterministic plant, we introduce the following manufacturing example, a solution of which is discussed later.

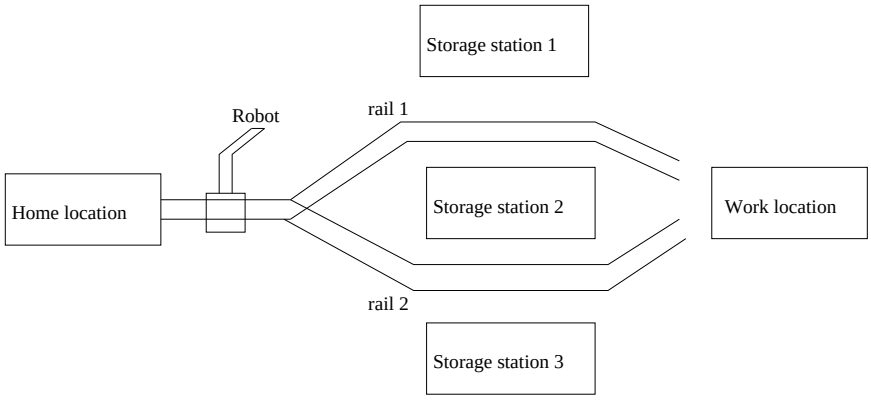

Figure 5.8 A manufacturing system

Example 12 A robot is available at its home location to traverse on one of the two available rails (Figure 5.8). Traversal on rail-i ($i = 1, 2$) is denoted by event a_i, and while on rail-i, the robot can pick a part from storage-i (event b_i) or storage-$(i+1)$ (event b_{i+1}). The robot then takes the part to work location (event c). Upon completion of processing, the robot returns the part to either storage-i (event b_i) or storage-$(i+1)$ (event b_{i+1}) and returns to its home location

(event d). Not returning the part to its original pick-up location is undesirable. To avoid this undesirable behavior, the specification requires that while the robot is in its home location, it nondeterministically decides whether to use storage-i or storage-$(i+1)$ while traversing on rail-i. It is also required that the robot always be able to return to its home location (which means that the state representing the home location is the only marked state).

Models G and R^{det} of the manufacturing system and its deterministic specification, respectively, are given in Figure 5.9. We assume that all events are controllable. However, only events c and d are completely observable. Events a_1 and a_2 are indistinguishable, and so are the events b_1, b_2 and b_3. I.e., $\Sigma_u = \emptyset$, $M(a_1) = M(a_2)$, and $M(b_1) = M(b_2) = M(b_3)$. Controllability clearly holds since all events are controllable. However, the observability is violated. This is because $a_1b_1c, a_1b_2c \in L(R)$, $M(a_1b_1c) = M(a_1b_2c)$, $a_1b_1cb_1 \in L(R^{det})$, yet $a_1b_2cb_1 \in L(G) - L(R^{det})$. It follows that there does not exists a deterministic nonblocking supervisor S such that $L_m(G\|S) = L_m(R^{det})$.

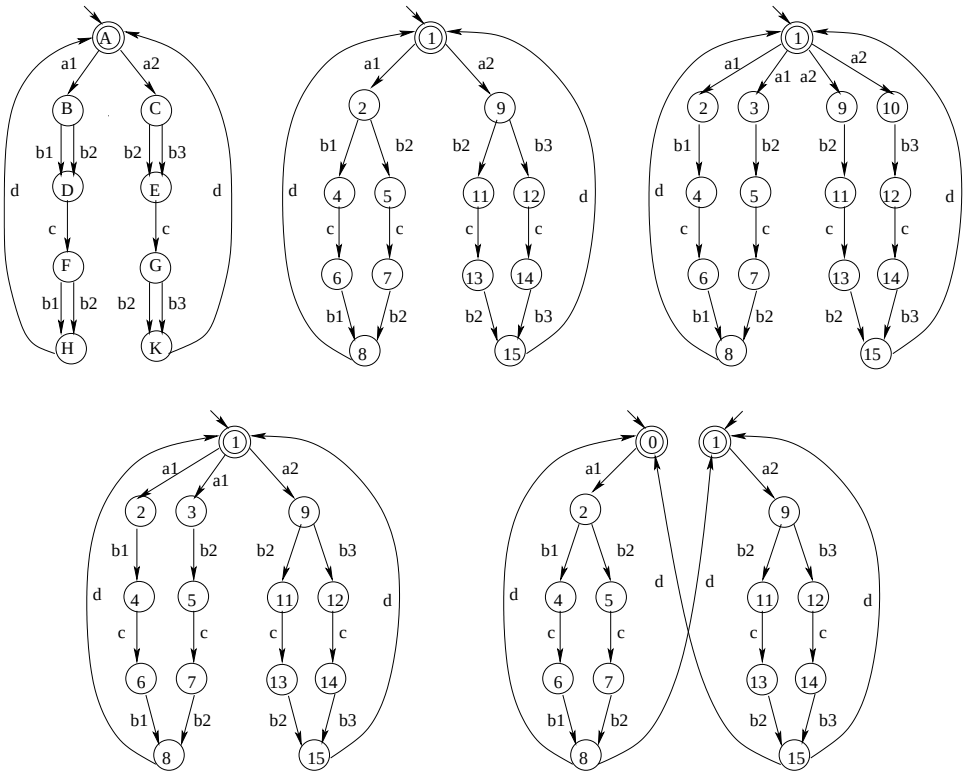

Figure 5.9 G (top left), R^{det} (top middle), R (top right), R' (bottom left), R'' (bottom right)

Figure 5.9 also shows some of the nondeterministic specifications R, R', R'' that are "trim" and language equivalent to R^{det}. Due to the finiteness of R^{det} the number of such "basic" nondeterministic specifications is bounded (the boundedness follows from the small model result of the decidability of CTL* or μ-calculus specifications), and as long as one of them possesses a bisimilarity enforcing supervisor (possibly nondeterministic), the nonblocking control problem with language specification we set out to solve possesses a supervisor. We show later that there exists a *nondeterministic* supervisor S such that $G\|S \simeq R$. Since bisimilarity preserves language equivalence as well as branching behavior (such as nonblockingness), $L_m(G\|S) = L_m(R) = L_m(R^{det})$ and further S is nonblocking as desired.

5.3.1 State-Recognizability and State-Achievability

Since R is bisimilar to $G\|S$, we first ask what property must $G\|S$ satisfy given that G is deterministic and S is (Σ_u, M)-compatible. In case of complete observation, we characterized the property that $G\|S$ possesses in form of state-controllability (SC). In the presence of partial observation, S must also be M-compatible and therefore $G\|S$ must possess some additional property. In this section, we characterize this additional property in form of state-recognizability (SR). We call the combined property of SC and SR as state-achievability (SA).

5.3.1.1 State-Recognizability and its Properties

Recall that M-compatibility of S implies indistinguishable events when defined at a state must have identical successors. Clearly, when S is composed with G such a property may no longer hold. However, it has the property that if we merge the successors of indistinguishable events defined at certain states of $G\|S$ in a certain way, we can construct a M-compatible state machine that also enforces R when used as a supervisor for G. This property is called SR and is defined in terms of a "state-recognizability relation". The idea behind defining such a relation is that state-pairs in this relation can be merged to satisfy M-compatibility, while preserving the bisimilarity of control exercised. Let $X_{syn}(y) := \{x \in X \mid \exists s \in \Sigma^*, x \in \alpha^*(X_0, s) \text{ and } y \in \beta^*(Y_0, s)\}$ denote the set of states in G that synchronize with a state y in S, i.e., are reached

by a trace $s \in L(G) \cap L(S)$.

Definition 11 Given NSMs $G = (X, \Sigma, \alpha, X_0, X_m)$ and $S = (Y, \Sigma, \beta, Y_0, Y_m)$, a symmetric relation Φ over states of S is said to be a *state-recognizability relation* if $(y_1, y_2) \in \Phi$ implies

1. For $i, j = 1, 2, i \neq j$, $x_i \in X_{syn}(y_i)$ implies $\Sigma(x_i) \cap \Sigma(y_j) \subseteq \Sigma(x_i) \cap \Sigma(y_i)$,
   (If x_i can synchronize with y_i in $G \| S$ and is made to synchronize with y_j $(i, j = 1, 2)$, then events enabled at x_i remain unchanged.)

2. For $i, j = 1, 2, i \neq j$, $a_i \in \overline{\Sigma}(y_i)$, $M(a_i) = M(a_j)$ implies $\forall y_i' \in \beta(y_i, a_i)$, $\exists y_j' \in \beta(y_j, a_j)$ s.t. $(y_i', y_j') \in \Phi$,
   (If indistinguishable events a_i are defined at y_i, then for each a_i-successor y_i' of y_i, there exists a a_j-successor y_j' of y_j such that $(y_i', y_j') \in \Phi$.)

3. For $i, j = 1, 2, i \neq j$, $y_i \in Y_m$ implies $[y_j \in Y_m] \vee [X_{syn}(y_j) \cap X_m = \emptyset]$. (If y_i is marked then either y_j is also marked or y_j synchronizes with only unmarked states.)

We use $\langle y_1, y_2 \rangle$ to denote the state obtained by merger of y_1 and y_2. The set of events defined at a state $\langle y_1, y_2 \rangle$ is the union of the set of events defined at y_1 and y_2, i.e., $\Sigma(\langle y_1, y_2 \rangle) = \cup_i \Sigma(y_i)$. For (i) preserving control exercised, and (ii) satisfying M-compatibility when states y_1 and y_2 are merged, the following should hold:

- Enabled events should not change:

  $$\Sigma(x_i) \cap [\Sigma(y_1) \cup \Sigma(y_2)] = \Sigma(x_i) \cap \Sigma(y_i), \text{ for } i = 1, 2,$$

  where left/right hand side is set of events enabled at x_i after/before the merger. Suppose we let $i = 1$, then the above simplifies to

  $$[\Sigma(x_1) \cap \Sigma(y_1)] \cup [\Sigma(x_1) \cap \Sigma(y_2)] = \Sigma(x_1) \cap \Sigma(y_1).$$

  For this to hold, we need

  $$\Sigma(x_1) \cap \Sigma(y_2) \subseteq \Sigma(x_1) \cap \Sigma(y_1).$$

  Similarly, if we let $i = 2$, then $\Sigma(x_2) \cap \Sigma(y_1) \subseteq \Sigma(x_2) \cap \Sigma(y_2)$ should hold. This justifies the first requirement in Definition 11.

- Any pair of indistinguishable events defined at the merged state $\langle y_1, y_2 \rangle$ should have successors that can be paired and combined (for M-compatibility). For this, pairings of successors should exist that can be merged. This justifies the second requirement in Definition 11.

- The combined state should not change the marking status of the controlled system. Thus, for a state-pair being combined, if one of them is marked, then either the other one is also marked or, all states of G that can synchronize with this unmarked state are themselves unmarked. This justifies the third requirement in Definition 11.

If an automaton S possesses a state-recognizability relation which includes all pairs $(y_0, y_0) \in Y_0^2$ of the initial states, it is called a state-recognizable automaton:

Definition 12 Given G and an observation mask M, S is *state-recognizable* with respect to G and M if there exists a state-recognizability relation Φ over states of S such that $(y_0, y_0) \in \Phi$ for all $y_0 \in Y_0$.

Remark 10 The name state-recognizability is chosen since it is a generalization of "language-recognizability" introduced in Kumar et al. (2005b): Language-recognizability is a property of $L(G\|S)$, whereas as we show below the state-recognizability is a property of $G\|S$.

Note that by definition Φ is *symmetric*. Using the definition of state-recognizability it can then be concluded that $(y, y) \in \Phi$ for each $y \in Y$, i.e., Φ is also *reflexive*. However Φ *need not be transitive*.

State-recognizability relation is closed under intersection. To see this, consider two state-recognizability relations Φ_1 and Φ_2 over states of S. Pick any state-pair $(y_1, y_2) \in \Phi_1 \cap \Phi_2$. By Definition 11, condition 1, 2 and 3 hold for (y_1, y_2). Thus, $\Phi_3 = \Phi_1 \cap \Phi_2$ is also a state-recognizability relation. Further if both Φ_1 and Φ_2 contain (y_0, y_0) for each $y_0 \in Y_0$, then so does $\Phi_3 = \Phi_1 \cap \Phi_2$. Therefore, whenever S is state-recognizable, there always exists a certain state-recognizability relation that is unique in the sense that it is the minimal one; which is what we work with by default.

The following example illustrates the above concepts.

Example 13 Consider G and S drawn in Figure 5.10, $M(a) = \epsilon$, $M(b_1) = M(b_2)$, $M(c) = c$,

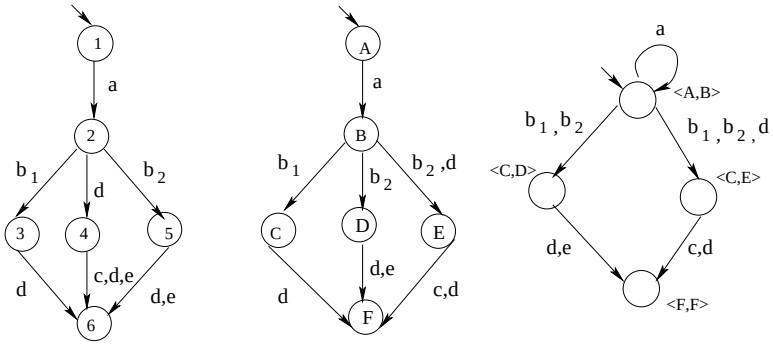

Figure 5.10 G (left), S (middle), and S^{Φ} (right)

$M(d) = d$ and $M(e) = e$. We examine whether S is state-recognizable. I.e., we need to check whether there exists a state-recognizability relation Φ with $(A, A) \in \Phi$. Since all states are assumed marked, condition 3 need not be checked.

The first condition obviously holds for $(y_1, y_2) = (A, A) \in \Phi$ since both states in the pair are the same. Next, since we have the transitions (A, ϵ, A), (A, a, B) and $M(a) = \epsilon$, we check condition 1 for the state pair (A, B). Indeed we have, $X_{syn}(A) = \{1\}$, $[\Sigma(1) \cap \Sigma(B) = \emptyset] \subseteq [\Sigma(1) \cap \Sigma(A) = \{a\}]$, and $X_{syn}(B) = \{2\}$, $[\Sigma(2) \cap \Sigma(A) = \emptyset] \subseteq [\Sigma(2) \cap \Sigma(B) = \{b_1, b_2, d\}]$.

The only pair of indistinguishable transitions defined at the pair of states (A, B) are (A, a, B) and (B, ϵ, B). So we need to check the condition 1 for pair (B, B), which is obviously satisfied since the two states in the pair are the same. Next, since $b_1, b_2 \in \Sigma(B)$ and $M(b_1) = M(b_2)$, condition 1 needs to be checked for each b_1-successor and each b_2-successor of B. Consider first the b_1-successor state C. We find that there exists b_2-successor state D such that condition 1 holds, namely, $X_{syn}(C) = \{3\}$; $[\Sigma(3) \cap \Sigma(D) = \{d\}] \subseteq [\Sigma(3) \cap \Sigma(C) = \{d\}]$, and $X_{syn}(D) = \{5\}$; $[\Sigma(5) \cap \Sigma(C) = \{d\}] \subseteq [\Sigma(5) \cap \Sigma(D) = \{d, e\}]$.

Thus, for b_1-successor state C (resp. b_2-successor state D) of B, there exists b_2-successor state D (resp. b_1-successor state C) such that condition 1 holds. Next consider b_2-successor state E of B. Then there exists b_1-successor state C such that $X_{syn}(E) = \{4, 5\}$; $[\Sigma(4) \cap \Sigma(C) = \{d\}] \subseteq [\Sigma(4) \cap \Sigma(E) = \{c, d\}]$, and $[\Sigma(5) \cap \Sigma(C) = \{d\}] \subseteq [\Sigma(5) \cap \Sigma(E) = \{d\}]$, and $X_{syn}(C) = \{3\}$; $[\Sigma(3) \cap \Sigma(E) = \{d\}] \subseteq [\Sigma(3) \cap \Sigma(C) = \{d\}]$.

Therefore, there exists a symmetric state-recognizability relation Φ:
$\Phi = \{(A,A),(A,B),(B,A),(B,B),(C,C),(C,D),(D,C),$
$(D,D),(C,E),(E,C),(E,E),(F,F)\}$. Since $(A,A) \in \Phi$, we conclude that S is state-recognizable.

The following lemma proves our earlier conjecture that M-compatibility of S and determinism of G imply the state-recognizability of $G\|S$. Since $R \simeq G\|S$, this establishes the property of "state-recognizable-bisimilar" as a necessary property for the existence of a bisimilarity enforcing M-compatible supervisor.

Lemma 9 Given deterministic G and M-compatible S, $G\|S$ is state-recognizable.

Proof: Define a relation $\Phi \subseteq (X \times Y)^2$ as follows:

1. $((x_0,y_0),(x_0,y_0)) \in \Phi$ for all $y_0 \in Y_0$,
2. For $((x_1,y),(x_2,y)) \in \Phi$, if $a_i \in \overline{\Sigma}((x_i,y))$ and $M(a_1) = M(a_2)$, then for all $\overline{y} \in \beta(y,a_1) \stackrel{(a)}{=} \beta(y,a_2)$, $((x_{1a_1},\overline{y}),(x_{2a_2},\overline{y})) \in \Phi$, where $x_{ia_i} = \alpha(x_i,a_i)$.

Equality (a) holds because S is M-compatible. From the second part of the definition of Φ, the condition 2 of state-recognizability automatically holds. So we only need to show the conditions 1 and 3. Consider a state-pair $((x_1,y),(x_2,y)) \in \Phi$. Then due to the determinism of G, $X_{syn}((x_i,y)) = \{x_i\}$, i.e., x_i in G is the unique state that synchronizes with state (x_i,y) in $G\|S$. For $((x_1,y),(x_2,y)) \in \Phi$, we have

$$\begin{aligned}
\Sigma(x_1) \cap \Sigma((x_2,y)) &= \Sigma(x_1) \cap [\Sigma(x_2) \cap \Sigma(y)] \\
&= [\Sigma(x_1) \cap \Sigma(y)] \cap \Sigma(x_2) \\
&\subseteq \Sigma(x_1) \cap \Sigma(y) \\
&= \Sigma(x_1) \cap [\Sigma(x_1) \cap \Sigma(y)] \\
&= \Sigma(x_1) \cap \Sigma((x_1,y)).
\end{aligned}$$

Similarly, $\Sigma(x_2) \cap \Sigma((x_1,y)) \subseteq \Sigma(x_2) \cap \Sigma((x_2,y))$.

It remains to show that the condition 3 of Definition 11 also holds. Suppose $(x_1,y) \in X_m \times Y_m$, then $y \in Y_m$. If $x_2 \in X_m$, then $(x_2,y) \in X_m \times Y_m$. On the other hand, if $x_2 \notin X_m$,

then since $X_{syn}((x_2, y)) = \{x_2\}$ (follows from determinism of G), $X_{syn}((x_2, y)) \cap X_m = \emptyset$, establishing the condition 3. It follows that Φ is a state-recognizability relation. Further since $((x_0, y_0), (x_0, y_0)) \in \Phi$ for each $y_0 \in Y_0$, $G\|S$ is state-recognizable as desired. ■

Next we establish the following main property of state-recognizability: Given a state-recognizable S possessing an underlying state-recognizability relation Φ, it is possible to compute S^Φ such that S^Φ is M-compatible and $G\|S \simeq G\|S^\Phi$. The following algorithm computes S^Φ by combining state-pairs in a state-recognizability relation.

Definition 13 Suppose S is state-recognizable, possessing an underlying state-recognizability relation Φ. We say that $\hat{Y} \subseteq Y$ is a Φ-*compatible set* if $y_1, y_2 \in \hat{Y}$ implies $(y_1, y_2) \in \Phi$. A Φ-compatible set is *maximal* if no larger Φ-compatible set exists.

Note that a state-recognizability relation Φ *need not be transitive*, and as a result, the Φ-compatible sets do not form a partition of Y, rather only a cover. The states of S^Φ are then chosen to be the maximal Φ-compatible sets.

Algorithm 2 Given a state-recognizability relation Φ under which S is state-recognizable, an algorithm for the computation of a M-compatible S^Φ is given below.

$$S^\Phi := (\mathcal{Y}, \Sigma, \beta^\Phi, \mathcal{Y}_0, \mathcal{Y}_m),$$

where

- $\mathcal{Y} \subseteq 2^Y$ is set of states of S^Φ, and

$$\mathcal{Y} = \{\hat{Y} \subseteq Y \mid \hat{Y} \text{ is a maximal } \Phi\text{-compatible set}\}.$$

- β^Φ is its transition function such that for all $\hat{Y}, \tilde{Y} \in \mathcal{Y}$ and for all

$$\sigma \in \overline{\Sigma}(\hat{Y}, \tilde{Y}): \quad \{\sigma \in \overline{\Sigma}(\hat{Y}) = \cup_{y \in \hat{Y}} \overline{\Sigma}(y) \mid \beta(\hat{Y}, \sigma) \cap \tilde{Y} \neq \emptyset\},$$

$\tilde{Y} \in \beta^\Phi(\hat{Y}, \sigma) \Leftrightarrow M^{-1}M(\sigma) \cap \overline{\Sigma}(\hat{Y}) \subseteq \overline{\Sigma}(\hat{Y}, \tilde{Y})$.

- $\mathcal{Y}_0$ is its set of initial states, and $\mathcal{Y}_0 = \{\hat{Y} \in \mathcal{Y} \mid \hat{Y} \cap Y_0 \neq \emptyset\}$.
- $\mathcal{Y}_m$ is its set of marked states, and $\mathcal{Y}_m = \{\hat{Y} \in \mathcal{Y} \mid \hat{Y} \cap Y_m \neq \emptyset\}$.

The following proposition proves the correctness of Algorithm 2.

Proposition 1 Algorithm 2 is correct. I.e., consider G, S and an observation mask M. Suppose S is state-recognizable so that there exists a state-recognizability relation Φ such that $(y_0, y_0) \in \Phi$ for all $y_0 \in Y_0$. Then S^Φ is M-compatible, where S^Φ is computed by Algorithm 2.

Proof: Pick $\hat{Y}, \tilde{Y} \in \mathcal{Y}$ and $\sigma, \sigma' \in \overline{\Sigma}(\hat{Y})$ such that $M(\sigma) = M(\sigma')$. Suppose $\sigma \in \overline{\Sigma}(\hat{Y}, \tilde{Y})$. We first show that $\sigma' \in \overline{\Sigma}(\hat{Y}, \tilde{Y})$. Since $\hat{Y}, \tilde{Y} \in \mathcal{Y}$ are maximal Φ-compatible sets, by Definition 11, we have

$$\begin{aligned}
& \sigma \in \overline{\Sigma}(\hat{Y}, \tilde{Y}), \sigma' \in \overline{\Sigma}(\hat{Y}), M(\sigma) = M(\sigma') \\
\Rightarrow\ & \exists y \in \beta(\hat{Y}, \sigma) \cap \tilde{Y}, \text{ and } \exists \tilde{y} \in \beta(\hat{Y}, \sigma') \text{ s.t. } (y, \tilde{y}) \in \Phi \\
\Rightarrow\ & \tilde{y} \in \tilde{Y} \\
\Rightarrow\ & \beta(\hat{Y}, \sigma') \cap \tilde{Y} \neq \emptyset \\
\Rightarrow\ & \sigma' \in \overline{\Sigma}(\hat{Y}, \tilde{Y}).
\end{aligned}$$

Next from the definition of β^Φ, for $\sigma, \sigma' \in \overline{\Sigma}(\tilde{Y}, \hat{Y})$,

$$\begin{aligned}
\tilde{Y} \in \beta^\Phi(\hat{Y}, \sigma) \quad & \Leftrightarrow \quad M^{-1}M(\sigma) \cap \overline{\Sigma}(\hat{Y}) \subseteq \overline{\Sigma}(\hat{Y}, \tilde{Y}) \\
& \Leftrightarrow \quad M^{-1}M(\sigma') \cap \overline{\Sigma}(\hat{Y}) \subseteq \overline{\Sigma}(\hat{Y}, \tilde{Y}) \\
& \Leftrightarrow \quad \tilde{Y} \in \beta^\Phi(\hat{Y}, \sigma'),
\end{aligned}$$

as desired. ■

Remark 11 Given a state-recognizability relation $\Phi \subseteq Y^2$, checking the Φ-compatibility of a set $\hat{Y} \subseteq Y$ is quadratic in its size. By examining all subsets of Y (which are exponentially many in number), we can identify all those that are ϕ-compatible, and then the ones that are maximal. It follows that the complexity of Algorithm 2 is $O(2^{|Y|^2}) = O(2^{|S|})$.

The following example illustrates the construction of S^Φ.

Example 14 Consider state-recognizable state machine S shown in Example 13. We first compute $\mathcal{Y}$. By the state-recognizability relation Φ computed in Example 13, we have

$$\mathcal{Y} = \{\{A, B\}, \{C, D\}, \{C, E\}, \{F, F\}\},$$

and $\mathcal{Y}_0 = \{\{A, B\}\}$.

Next we compute the set of transitions. Let

$$\hat{Y}_0 = \{A, B\}, \hat{Y}_1 = \{C, D\}, \hat{Y}_2 = \{C, E\}, \hat{Y}_3 = \{F, F\}.$$

Note that $\overline{\Sigma}(\hat{Y}_0) = \{a, b_1, b_2, d, \epsilon\}$.

- Since $\overline{\Sigma}(\hat{Y}_0, \hat{Y}_0) = \{a, \epsilon\}$ and $M^{-1}M(a) \cap \overline{\Sigma}(\hat{Y}_0) = \{a, \epsilon\} \subseteq \overline{\Sigma}(\hat{Y}_0, \hat{Y}_0)$, transition $(\hat{Y}_0, a, \hat{Y}_0)$ is in S^{Φ}.

- Since $\overline{\Sigma}(\hat{Y}_0, \hat{Y}_1) = \{b_1, b_2, \epsilon\}$ and $M^{-1}M(b_1) \cap \overline{\Sigma}(\hat{Y}_0) = \{b_1, b_2\} = M^{-1}M(b_2) \cap \overline{\Sigma}(\hat{Y}_0) \subseteq \overline{\Sigma}(\hat{Y}_0, \hat{Y}_1)$, transitions $(\hat{Y}_0, b_1, \hat{Y}_1)$ and $(\hat{Y}_0, b_2, \hat{Y}_1)$ are in S^{Φ}.

- Since $\overline{\Sigma}(\hat{Y}_0, \hat{Y}_2) = \{b_1, b_2, d, \epsilon\}$, and $M^{-1}M(b_1) \cap \overline{\Sigma}(\hat{Y}_0) = \{b_1, b_2\} = M^{-1}M(b_2) \cap \overline{\Sigma}(\hat{Y}_0) \subseteq \overline{\Sigma}(\hat{Y}_0, \hat{Y}_2)$, and $M^{-1}M(d) \cap \overline{\Sigma}(\hat{Y}_0) = \{d\} \subseteq \overline{\Sigma}(\hat{Y}_0, \hat{Y}_2)$, transitions $(\hat{Y}_0, b_1, \hat{Y}_2)$, $(\hat{Y}_0, b_2, \hat{Y}_2)$ and $(\hat{Y}_0, d, \hat{Y}_2)$ are in S^{Φ}.

Similarly one can compute the other transitions; the details are omitted here. S^{Φ} is drawn in Figure 5.10 (self-loop transitions on ϵ are omitted). It can be verified by inspection that S^{Φ} is M-compatible.

Having showed that S^{Φ} is M-compatible whenever S is state-recognizable, we next show that if in addition S is bisimilarity enforcing, then so is S^{Φ}.

Proposition 2 Given G and S, if S is state-recognizable possessing an underlying state-recognizability relation Φ, then $G\|S \simeq G\|S^{\Phi}$, where S^{Φ} is computed by Algorithm 2.

Proof: Define a relation $\Psi \subseteq ((X \times \mathcal{Y}) \cup (X \times Y))^2$ as:

$$\begin{aligned} \Psi \quad := \quad & \{((x, \hat{Y}), (x, y)), ((x, y), (x, \hat{Y})) \mid (x, \hat{Y}) \text{ in } G\|S^{\Phi}, \\ & (x, y) \text{ in } G\|S, \text{ and } y \in \hat{Y}\}. \end{aligned}$$

Note that from construction in Algorithm 2, $X_{syn}(\hat{Y}) = \cup_{y' \in \hat{Y}} X_{syn}(y')$. So, $x \in X_{syn}(y)$ implies $x \in X_{syn}(\hat{Y})$ when $y \in \hat{Y}$, i.e., a pair $((x,y),(x,\hat{Y}))$ with $y \in \hat{Y}$ in Ψ is indeed feasible.

We next prove that Ψ is a bisimulation relation. For a state-pair $((x,\hat{Y}),(x,y)) \in \Psi$, it is clear that $(x,y) \in (X \times Y)_{syn}(x,\hat{Y})$. Further,

$$\begin{aligned}
\overline{\Sigma}((x,\hat{Y})) &= \overline{\Sigma}(x) \cap \overline{\Sigma}(\hat{Y}) \\
&= \overline{\Sigma}(x) \cap [\cup_{y' \in \hat{Y}} \overline{\Sigma}(y')] \\
&= [\overline{\Sigma}(x) \cap \overline{\Sigma}(y)] \cup [\overline{\Sigma}(x) \cap [\cup_{y' \in \hat{Y} - \{y\}} \overline{\Sigma}(y')]] \\
&= \overline{\Sigma}((x,y)) \cup [\cup_{y' \in \hat{Y} - \{y\}} (\overline{\Sigma}(x) \cap \overline{\Sigma}(y'))] \\
&\overset{(a)}{=} \overline{\Sigma}((x,y)).
\end{aligned}$$

Equality (a) holds because $y' \in \hat{Y} - \{y\}$, and $y \in \hat{Y}$ implies $(y,y') \in \Phi$. So by Definition 11, $\overline{\Sigma}(x) \cap \overline{\Sigma}(y') \subseteq \overline{\Sigma}(x) \cap \overline{\Sigma}(y) = \overline{\Sigma}((x,y))$ for $x \in X_{syn}(y)$ and $y' \in \hat{Y} - \{y\}$.

Similar event-set equality as established above for $((x,\hat{Y}),(x,y))$ pair needs to hold for their respective σ-successors, which will be the case whenever the respective σ-successors form a pair that lies in Ψ. For a σ-successor state (x_σ, y_σ) in $G\|S$, there exists some $\hat{Y}_\sigma \in \beta^\Phi(\hat{Y},\sigma)$ such that $y_\sigma \in \hat{Y}_\sigma$ since $\overline{\Sigma}(\hat{Y}) = \cup_{y' \in \hat{y}} \overline{\Sigma}(y')$. Then obviously, $((x_\sigma,\hat{Y}_\sigma),(x_\sigma,y_\sigma)) \in \Psi$, as desired.

Further by Algorithm 2, $y \in \hat{Y}$ and $y \in Y_m$ implies $\hat{Y} \cap Y_m \neq \emptyset$, i.e., $\hat{Y} \in \mathcal{Y}_m$. Thus $(x,y) \in X_m \times Y_m$ implies $(x,\hat{Y}) \in X_m \times \mathcal{Y}_m$. On the other hand, if $(x,\hat{Y}) \in X_m \times \mathcal{Y}_m$, then for any $y \in \hat{Y}$, we claim that $y \in Y_m$. Suppose for contradiction that $y \notin Y_m$. Since $\hat{Y} \in \mathcal{Y}_m$, then there exists $y' \in \hat{Y} \cap Y_m$. Also since $y, y' \in \hat{Y}$, $(y,y') \in \Phi$. Thus by condition 3 of Definition 11, $X_{syn}(y) \cap X_m = \emptyset$. Then for $x \in X_{syn}(y)$, $x \notin X_m$, a contradiction to the fact that $(x,\hat{Y}) \in X_m \times \mathcal{Y}_m$.

By Definition 1, Ψ is a simulation relation. Further, each state (x,y) of $G\|S$ is paired with a state $(x,\hat{Y})$ of $G\|S^\Phi$ such that $y \in \hat{Y}$, and vice-versa. So Ψ is symmetric. I.e., Ψ is a bisimulation relation. Finally, since for each $\hat{Y}_0 \in \mathcal{Y}_0$ and $y_0 \in Y_0$, $\{((x_0,\hat{Y}_0),(x_0,y_0)) \mid y_0 \in \hat{Y}_0\} \subseteq \Psi$, $G\|S^\Phi \simeq_\Psi G\|S$, as desired. ∎

5.3.1.2 State-Achievability and its Properties

State-achievability is the combined property of state-controllability and state-recognizability introduced earlier. It is a generalization of language-achievability which is a property of a controlled plant behavior $L(G\|S)$, whereas state-achievability is a property of the controlled deterministic-plant $G\|S$.

Definition 14 Given G, uncontrollable event set Σ_u, and observation mask M, S is *state-achievable (SA)* with respect to G, Σ_u and M if

- S is state-controllable with respect to G and Σ_u, i.e., $s \in L(S)$, $\sigma \in \Sigma_u$ such that $s\sigma \in L(G) \Rightarrow \forall y \in \beta^*(Y_0, s), \sigma \in \Sigma(y)$, and
- S is state-recognizable with respect to G and M.

Next we establish the following main property of state-achievability: Given a state-achievable S possessing an underlying state-recognizability relation Φ, it is possible to compute S^{Φ,Σ_u} such that S^{Φ,Σ_u} is (Σ_u, M)-compatible and $G\|S \simeq G\|S^{\Phi,\Sigma_u}$. We first show that the SC property is preserved under $(\cdot)^{\Phi}$ computation. We need the following alternative characterization of the SC property.

Lemma 10 Given G, S is state-controllable if and only if for each y in S, $[\cup_{x \in X_{syn}(y)} \Sigma(x)] \cap \Sigma_u \subseteq \Sigma(y) \cap \Sigma_u$.

Proof: Note that we have the following equivalence:

$$a \in [\cup_{x \in X_{syn}(y)} \Sigma(x)] \cap \Sigma_u \Leftrightarrow$$
$$a \in \Sigma_u, \exists s \in L(G) : y \in \beta^*(Y_0, s) \text{ and } sa \in L(G).$$

Thus, the proof follows from Definition 14. ■

Lemma 11 Consider G and S such that S is state-achievable. Let Φ be the state-recognizability relation under which S is state-recognizable and S^{Φ} be computed by Algorithm 2. Then S^{Φ} is state-controllable.

Proof: Consider a state $\hat{Y}$ in S^{Φ}, and the set of states $X_{syn}(\hat{Y})$ in G that synchronize with $\hat{Y}$. Then from construction in Algorithm 2, $X_{syn}(\hat{Y}) = \cup_{y\in\hat{Y}} X_{syn}(y)$ and $\Sigma(\hat{Y}) = \cup_{y\in\hat{Y}}\Sigma(y)$. By Lemma 10, for S^{Φ} to be SC, it suffices to show that

$$[\cup_{y\in\hat{Y}}[\cup_{x\in X_{syn}(y)}\Sigma(x)]] \cap \Sigma_u \subseteq [\cup_{y\in\hat{Y}}\Sigma(y)] \cap \Sigma_u.$$

Since S is SC, for $y \in Y$, $[\cup_{x\in X_{syn}(y)}\Sigma(x)] \cap \Sigma_u \subseteq \Sigma(y) \cap \Sigma_u$. Thus, the following holds:

$$\cup_{y\in\hat{Y}}[\cup_{x\in X_{syn}(y)}\Sigma(x) \cap \Sigma_u] \subseteq \cup_{y\in\hat{Y}}[\Sigma(y) \cap \Sigma_u] \Leftrightarrow$$
$$\cup_{y\in\hat{Y}}[\cup_{x\in X_{syn}(y)}\Sigma(x)] \cap \Sigma_u \subseteq \cup_{y\in\hat{Y}}[\Sigma(y)] \cap \Sigma_u.$$

■

Using the fact that S^{Φ} is SC, we can obtain S^{Φ,Σ_u} that is Σ_u-compatible and preserves the M-compatibility property as well as the bisimilarity of control exercised.

Algorithm 3 Given a state-achievable state machine S, consider S^{Φ} where Φ is the state-recognizability relation under which S is state-recognizable, and S^{Φ} is as computed by Algorithm 2. The computation of a (Σ_u, M)-compatible S^{Φ,Σ_u} is given as below.

For each state $\hat{Y}$ of S^{Φ} and $\sigma \in \Sigma_u - \Sigma(\hat{Y})$, add the following transition(s) on σ:

- If $\nexists a \in \overline{\Sigma}(\hat{Y})$ such that $M(a) = M(\sigma)$, then add σ as self-loop at $\hat{Y}$;
- Else, $\forall a \in \overline{\Sigma}(\hat{Y})$ such that $M(a) = M(\sigma)$, add transition on σ from $\hat{Y}$ to each state in $\beta^{\Phi}(\hat{Y}, a)$.

Theorem 8 Algorithm 3 is correct. I.e., given a state-achievable state machine S, S^{Φ,Σ_u} is (Σ_u, M)-compatible and $G\|S \simeq G\|S^{\Phi,\Sigma_u}$, where S^{Φ,Σ_u} is computed by Algorithm 3.

Proof: Since $\sigma \in \Sigma_u$ is defined at every state of S^{Φ,Σ_u}, S^{Φ,Σ_u} is Σ_u-compatible. By Proposition 1, S^{Φ} is M-compatible. Also, transitions on undefined uncontrollable events are added in S^{Φ} so that all indistinguishable transitions at a state have the same successors in S^{Φ,Σ_u}. It follows that S^{Φ,Σ_u} is M-compatible, and so S^{Φ,Σ_u} is also (Σ_u, M)-compatible.

By Lemma 11, S is state-controllable implies S^{Φ} is state-controllable. Then for any state $(x, \hat{Y})$ in $G\|S^{\Phi}$ such that x has uncontrollable events defined, S^{Φ} also has those events defined

at $\hat{Y}$. Therefore, adding transitions at each state $\hat{Y}$ on undefined uncontrollable events in S^{Φ} does not change the result of synchronous composition. It follows that $G\|S^{\Phi} = G\|S^{\Phi,\Sigma_u}$. Since by Proposition 2, $G\|S \simeq G\|S^{\Phi}$, $G\|S \simeq G\|S^{\Phi,\Sigma_u}$, as desired. ■

5.3.2 Existence Condition and Verification

We established the state-achievability of $G\|S$. Since $G\|S$ needs to be bisimilar to R, R needs to be "state-achievable-bisimilar", which we define next.

Definition 15 Given G, uncontrollable event set Σ_u and observation mask M), R is said to be *state-achievable-bisimilar (SAB)* (with respect to G, Σ_u, and M) if there exists a state-achievable S (with respect to G, Σ_u, and M) such that $S \simeq R$.

It is evident from the above definition that SAB is preserved under bisimilarity and also it is implied by SA. However SAB is strictly weaker than SA. This is established by the following example which shows that the SA property is not preserved under bisimulation.

Example 15 Consider an example shown in Figure 5.11, where $\Sigma_u = \emptyset$, $M(a_1) = M(a_2)$ and identity mask for other events.

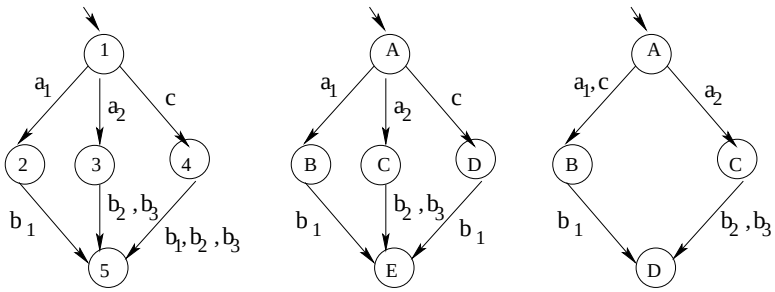

Figure 5.11 G (left), R_1 (middle), and R_2 (right)

It can be easily verified that $R_1 \simeq R_2$, R_1 is state-achievable, but R_2 is not. To see this, consider state A in R_2. Since $M(a_1) = M(a_2)$, for R_2 to be state-recognizable, the condition 1 in Definition 11 should hold for the corresponding successors B and C. Since $X_{syn}(B) = \{2, 4\}$ and $[\Sigma(4) \cap \Sigma(C) = \{b_2, b_3\}] \not\subseteq [\Sigma(4) \cap \Sigma(B) = \{b_1\}]$, the condition 1 does not hold. Thus, R_2 is not state-recognizable, and therefore not state-achievable.

The following theorem establishes a necessary and sufficient condition for the existence of a bisimilarity enforcing supervisor.

Theorem 9 Consider a deterministic G, a possibly nondeterministic R, and an observation mask M. Then there exists a (Σ_u, M)-compatible supervisor S such that $G\|S \simeq R$ if and only if $R \sqsubseteq G$ and R is state-achievable-bisimilar.

Proof: We first prove the necessity. From Corollary 2, $R \sqsubseteq G$ and from Lemma 8, $G\|S$ is state-controllable. Since S is M-compatible and G is deterministic, from Lemma 9, $G\|S$ is state-recognizable. So $G\|S$ is state-achievable (SA). Since $R \simeq G\|S$, it follows that R is state-achievable-bisimilar (SAB).

To see the sufficiency, since R is SAB, there exists $S \simeq R$ such that S is SA. Since S is SA, there exists a state-recognizability relation Φ over the states of S. We choose S^{Φ,Σ_u} to be the desired supervisor. Then from Theorem 8, S^{Φ,Σ_u} is (Σ_u, M)-compatible and $G\|S^{\Phi,\Sigma_u} \simeq G\|S$. Further since $S \simeq R$, it follows that $G\|S \simeq G\|R$. Further since $R \sqsubseteq G$ and G is deterministic, it follows from Lemma 6 that $G\|R \simeq R$. ■

Remark 12 When the events can be completely observed (i.e., when M is the identify function), the state-achievability is equivalent to state-controllability, and as a result the condition of Theorem 9 becomes "state-controllable-bisimilar" (SCB). This is equivalent to the condition of Theorem 6 since R is SCB if and only if $R^* := (Q^*, \Sigma, \delta^*, Q_0^*, Q_m^*)$ is SC. (We recall that R^* is obtained from $R = (Q, \Sigma, \delta, Q_0, Q_m)$ by replacing the transition function δ with its "ϵ-closure" δ^* and renaming the states as: $Q = Q^*, Q_0 = Q_0^*, Q_m = Q_m^*$.)

In order to verify the existence of a bisimilarity enforcing supervisor for a deterministic-plant using Theorem 9, we need a method to verify the SAB property of R. (Methods to verify $R \sqsubseteq G$ are well-known.) We show below that when R is simulated by G and G is deterministic, R is SAB if and only if $G\|R^*$ is SA. With such a result in hand all we need is a way to test the SA property, which we present in the next section. We need the results of the following two lemmas.

Lemma 12 Given deterministic G, S is state-achievable implies that $G\|S^*$ is state-achievable.

Proof: We need to show that $G\|S^*$ is SC and SR. SC of $G\|S^*$ follows from Lemma 7 and Definition 14. Next we show that if S is SR then so is $G\|S^*$.

S is SR implies that there exists a state-recognizability relation $\Phi \subseteq Y^2$ with $(y_0, y_0) \in \Phi$ for $y_0 \in Y_0$. Define a relation $\Phi \subseteq (X \times Y^*)^2$ as

$$\Phi' := \Phi_{(y_1,y_2)\to((x_1,y_1^*),(x_2,y_2^*))},$$

i.e., Φ with $(y_1, y_2) \in \Phi$ replaced by $((x_1, y_1^*), (x_2, y_2^*))$, where $(y_1, y_2) \in \Phi$ and $x_i \in X_{syn}(y_i) = X_{syn}(y_i^*)$. We need to show that all the conditions appearing in Definition 11 are satisfied. Due to determinism of G, $X_{syn}((x_i, y_i^*)) = \{x_i\}$. Thus, for $((x_1, y_1^*), (x_2, y_2^*)) \in \Phi'$,

$$\begin{aligned}
& \Sigma(x_1) \cap \Sigma((x_2, y_2^*)) \\
= \; & [\Sigma(x_1) \cap \Sigma(y_2^*)] \cap \Sigma(x_2) \\
= \; & \cup_{y_2' \in \epsilon^*(y_2)} [\Sigma(x_1) \cap \Sigma(y_2')] \cap \Sigma(x_2) \\
\overset{(a)}{\subseteq} \; & \cup_{y_1' \in \epsilon^*(y_1)} [\Sigma(x_1) \cap \Sigma(y_1')] \cap \Sigma(x_2) \\
\subseteq \; & \Sigma(x_1) \cap \Sigma(y_1^*) \\
= \; & \Sigma(x_1) \cap \Sigma((x_1, y_1^*)).
\end{aligned}$$

Containment (a) holds since SR of S implies $(y_1', y_2') \in \Phi$, where $y_i' \in \epsilon^*(y_i)$ and $(y_1, y_2) \in \Phi$. Further, $(y_1', y_2') \in \Phi$ implies $\Sigma(x_1) \cap \Sigma(y_2') \subseteq \Sigma(x_1) \cap \Sigma(y_1')$ for $x_1 \in X_{syn}(y_1)$. Similarly, $\Sigma(x_2) \cap \Sigma((x_1, y_1^*)) \subseteq \Sigma(x_2) \cap \Sigma((x_2, y_2^*))$. Thus, the condition 1 of Definition 11 holds.

Also, $a_i \in \overline{\Sigma}((x_i, y_i^*))$ implies $\exists \hat{y}_i \in \epsilon^*(y_i)$ such that $a_i \in \overline{\Sigma}(\hat{y}_i)$, and $(y_1, y_2) \in \Phi$ implies $(\hat{y}_1, \hat{y}_2) \in \Phi$ and $(\hat{y}_{1a_1}, \hat{y}_{2a_2}) \in \Phi$. It follows that $((x_{1a_1}, \hat{y}_{1a_1}^*), (x_{2a_2}, \hat{y}_{2a_2}^*)) \in \Phi'$. Thus the condition 2 of Definition 11 also holds.

Finally, if $(x_1, y_1^*) \in X_m \times Y_m^*$ and $(x_2, y_2^*) \notin X_m \times Y_m^*$, then either $x_2 \notin X_m$ or $y_2 \notin Y_m$. If $y_2 \notin Y_m$, then since $(y_1, y_2) \in \Phi$, we have $X_{syn}(y_2) \cap X_m = \emptyset$. So since $x_2 \in X_{syn}(y_2)$, $x_2 \notin X_m$. Under determinism of G, this further implies $X_{syn}((x_2, y_2)) \cap X_m = \{x_2\} \cap X_m = \emptyset$. It follows that the condition 3 of Definition 11 holds. Thus, Φ' is a state-recognizability relation. Further since $((x_0, y_0^*), (x_0, y_0^*)) \in \Phi'$ for each $y_0^* \in Y_0^*$, $G\|S^*$ is SR as desired. ■

Lemma 13 Given deterministic G, (possibly nondeterministic) $S = (Y, \Sigma, \beta, Y_0, Y_m)$ and $\hat{S} = (\hat{Y}, \Sigma, \hat{\beta}, \hat{Y}_0, \hat{Y}_m)$, if $S^* \simeq_\Psi \hat{S}^*$ and $G\|S^*$ is state-achievable, then $G\|\hat{S}^*$ is also state-achievable.

Proof: Since $G\|S^*(\simeq G\|S) \simeq G\|\hat{S}$, SC of $G\|\hat{S}^* = (G\|\hat{S})^*$ follows from Lemma 7. $G\|S^*$ is SR implies there exists a state-recognizability relation $\Phi \subseteq (X \times Y^*)^2$ with $((x_0, y_0^*), (x_0, y_0^*)) \in \Phi$ for all $y_0^* \in Y_0^*$. Define a relation $\Phi' \subseteq (X \times \hat{Y}^*)^2$ as

$$\Phi' := \Phi_{((x_1,y_1^*),(x_2,y_2^*))\to((x_1,\hat{y}_1^*),(x_2,\hat{y}_2^*))},$$

i.e., Φ with $((x_1, y_1^*), (x_2, y_2^*))$ replaced by $((x_1, \hat{y}_1^*), (x_2, \hat{y}_2^*))$, where $(y_1^*, \hat{y}_1^*), (y_2^*, \hat{y}_2^*) \in \Psi$. Since G is deterministic, $X_{syn}((x_i, y_i^*)) = X_{syn}((x_i, \hat{y}_i^*)) = \{x_i\}$. Thus, for $i, j = 1, 2$, $\forall x \in X_{syn}((x_i, y_i^*)) = \{x_i\}$:

$$\begin{aligned}
& \Sigma(x) \cap \Sigma((x_j, y_j^*)) \subseteq \Sigma(x) \cap \Sigma((x_i, y_i^*)) \\
\Rightarrow\ & \Sigma(x_i) \cap \Sigma((x_j, y_j^*)) \subseteq \Sigma(x_i) \cap \Sigma((x_i, y_i^*)) \\
\overset{(a)}{\Rightarrow}\ & \Sigma(x_i) \cap \Sigma((x_j, \hat{y}_j^*)) \subseteq \Sigma(x_i) \cap \Sigma((x_i, \hat{y}_i^*)) \\
\Rightarrow\ & \forall x \in X_{syn}((x_i, \hat{y}_i^*)), \\
& \Sigma(x) \cap \Sigma((x_j, \hat{y}_j^*)) \subseteq \Sigma(x) \cap \Sigma((x_i, \hat{y}_i^*))
\end{aligned}$$

Implication (a) holds since $(y_i^*, \hat{y}_i^*) \in \Psi$ implies $\Sigma(y_i^*) = \Sigma(\hat{y}_i^*)$. By Definition 11, Φ' is a state-recognizability relation with $((x_0, \hat{y}_0^*), (x_0, \hat{y}_0^*)) \in \Phi'$ for all $\hat{y}_0^* \in \hat{Y}_0^*$. Therefore, $G\|\hat{S}^*$ is SR. ■

The following theorem establishes a way to verify state-recognizability of a specification R.

Theorem 10 Given a deterministic G and a nondeterministic R such that $R \sqsubseteq G$, R is state-achievable-bisimilar if and only if $G\|R^*$ is state-achievable.

Proof: For the necessity, there exists S such that $S \simeq R$ (which implies $S^* \simeq R^*$) and S is SA. By Lemma 12, $G\|S^*$ is SA. This together with the fact that $S^* \simeq R^*$, implies from Lemma 13 that $G\|R^*$ is SA.

To see the sufficiency, we need to show the existence of $S \simeq R$ such that S is SA. Choose $S = G\|R$. Then by SA of $G\|R^*$, S is SA. So it remains to show that $S(= G\|R^*) \simeq R$. Since G is deterministic and $R^* \simeq R \sqsubseteq G$, it follows that $G\|R^* \simeq R$. ■

Remark 13 According to Theorem 9, S^{Φ,Σ_u} can be used as a supervisor, where S is any state machine such that $S \simeq R$ and S is state-achievable. According to Theorem 10, S can be chosen to be $G\|R^*$. Thus a possible (Σ_u, M)-compatible bisimilarity enforcing supervisor for a deterministic-plant is given by $(G\|R^*)^{\Phi,\Sigma_u}$. The complexity of its computation is the same as that of $(G\|R^*)^{\Phi}$ since the complexity of computing $(\cdot)^{\Phi,\Sigma_u}$ is linear in size of $(\cdot)^{\Phi}$. Recall that a state in $(G\|R^*)^{\Phi}$ is a subset of $X \times Q^* = X \times Q$ implying that the complexity of synthesizing a bisimilarity enforcing supervisor for a deterministic-plant is of order $O(2^{|G|\times|R|})$.

Next we illustrate Theorem 9 by revisiting Example 12.

Example 16 Applying Theorem 9, we first check whether $R \sqsubseteq G$ (R and G are shown in Figure 5.9). This can be easily verified. Also, since $\Sigma_u = \emptyset$, $G\|R^*$ is trivially SC.

Next, we verify whether $G\|R^*(= G\|R$ since no ϵ-transitions exist)(shown in Figure 5.12) is state-recognizable. We find the following state-recognizability relation $\Phi \subseteq (X \times Q^*)^2 = (X \times Q)^2$ exists (the details are omitted):

$$\begin{aligned}\Phi \;=\; & \{((A,1),(A,1)),((B,2),(C,10)),((B,3),(C,9)),\\ & ((D,4),(E,12)),((D,5),(E,11)),((F,6),(G,14)),\\ & ((F,7),(G,13)),((H,8),(K,15)),((C,10),(B,2)),\\ & ((C,9),(B,3)),((E,12),(D,4)),((E,11),(D,5)),\\ & ((G,14),(F,6)),((G,13),(F,7)),((K,15),(H,8))\}.\end{aligned}$$

Thus, R is SAB. Therefore, there exists a (Σ_u, M)-compatible supervisor S such that $G\|S \simeq R$. Since $\Sigma_u = \emptyset$, S is same as $(G\|R^*)^{\Phi}$, which is drawn in Figure 5.12. The controlled system $G\|S$ turns out to be the same as R and is also shown in Figure 5.12. Obviously $G\|S \simeq R$ and so S is nonblocking and $L_m(G\|S) = L_m(R) = L_m(R^{det})$, where R^{det} as shown in Figure 5.9 is a deterministic automaton representing the desired language specification.

Of course if $G\|R^*$ was not SA, one would verify whether other nondeterministic, trim, and language equivalent specifications (such as R' or R'' shown in Figure 5.9) satisfy such property. Note that due to the finiteness of R^{det} the number of "basic" nondeterministic, trim,

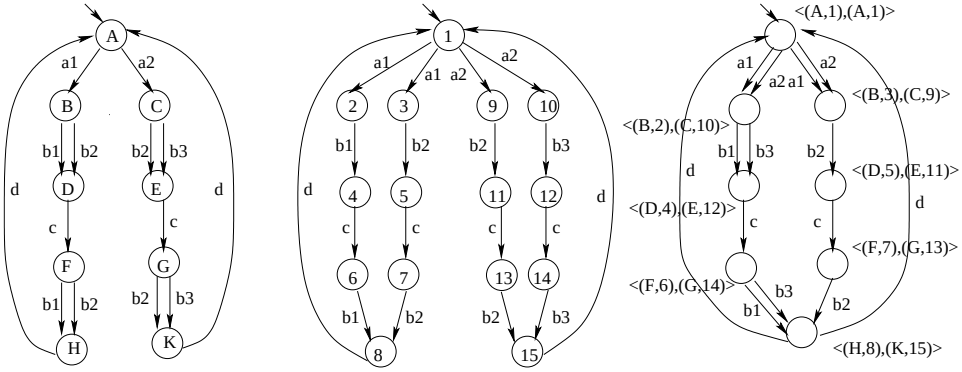

Figure 5.12 $G\|R^*$ (left), $S = (G\|R^*)^{\Phi}$ (middle), and $G\|S$ (right)

and language equivalent specifications is bounded. (The boundedness follows from the small model result of the decidability of CTL* or μ-calculus specifications.)

5.3.3 Test for State-Achievability

Theorem 10 reduces the verification of state-achievability-bisimilar (SAB) to that of state-achievability (SA), i.e., state-controllability (SC) and state-recognizability (SR). The verification of SC can be performed by Algorithm 1. We next develop an algorithm for verifying state-recognizability.

Algorithm 4 The algorithm for verifying state-recognizability of S is given as below:

1. Consider two copies of S and perform their masked-composition, denoted $MC(S,S) := (Y^2, \Sigma^2, \beta^2, Y_0^2, Y_m^2)$, where $\forall (y_1, y_2) \in Y^2$, $(\sigma_1, \sigma_2) \in \overline{\Sigma}^2$,

$$\beta^2((y_1,y_2),(\sigma_1,\sigma_2)) := \begin{cases} \beta(y_1,\sigma_1) \times \beta(y_2,\sigma_2), & \text{if } M(\sigma_1) = M(\sigma_2), \\ & \sigma_1 \neq \epsilon \neq \sigma_2, \\ \beta(y_1,\sigma_1) \times \{y_2\}, & \text{if } M(\sigma_1) = \epsilon = \sigma_2, \\ \{y_1\} \times \beta(y_2,\sigma_2), & \text{if } M(\sigma_2) = \epsilon = \sigma_1, \\ \emptyset, & \text{if } \sigma_1 = \sigma_2 = \epsilon. \end{cases}$$

$$\beta^2((y_1,y_2),\epsilon) := (\beta(y_1,\epsilon) \times \{y_2\}) \cup (\{y_1\} \times \beta(y_2,\epsilon)).$$

2. Mark a state (y_1, y_2) of $MC(S,S)$ "good" if and only if $\forall x_i \in X_{syn}(y_i)$, $\Sigma(x_i) \cap \Sigma(y_j) \subseteq \Sigma(x_i) \cap \Sigma(y_i), i,j = 1,2,$ and $y_i \in Y_m, y_j \notin Y_m \Rightarrow X_{syn}(y_j) \cap X_m = \emptyset, i,j = 1,2$.

3. (a) $n := 0$, $\Phi_n := \{(y_1, y_2) \mid (y_1, y_2)$ is a "good" state in $MC(S,S)\}$.

   (b) For $(y_1, y_2) \in \Phi_n$, if $a_i \in \overline{\Sigma}(y_i)$ for $i = 1,2$, and $M(a_1) = M(a_2)$, then for each $y_i' \in \beta(y_i, a_i)$, check whether there exists $y_j' \in \beta(y_j, a_j)$, where $i,j = 1,2$, such that $(y_1', y_2') \in \Phi_n$. If not, $\Phi_{n+1} := \Phi_n - \{(y_1, y_2)\}$.

   (c) If $\Phi_{n+1} = \Phi_n$ or $\Phi_{n+1} = \emptyset$, then $\Phi := \Phi_{n+1}$ and stop; Otherwise, $n := n + 1$, go to step (b).

4. S is state-recognizable if and only if $(y_0, y_0) \in \Phi$ for all $y_0 \in Y_0$.

The following theorem proves the correctness of Algorithm 4.

Theorem 11 Algorithm 4 is correct, i.e, S is state-recognizable if and only if $(y_0, y_0) \in \Phi$ for all $y_0 \in Y_0$, where Φ is as computed in Algorithm 4.

Proof: We first prove that Φ is a state-recognizability relation. By step 1, every state in $MC(S,S)$ is a pair of states reached by indistinguishable traces in S and vice-versa, i.e.,

$$(y_1, y_2) \text{ in } MC(S,S) \quad \Leftrightarrow \quad \exists s_1, s_2, M(s_1) = M(s_2) \text{ s.t.}$$
$$y_i \in \beta^*(Y_0, s_i), \text{ for } i = 1,2.$$

Note that by Definition 11, an element of a state-recognizability relation is a state-pair that is reached by indistinguishable traces. Thus, the set of state-pairs in a state-recognizability relation is a subset of state-pairs in $MC(S,S)$ computed in step 1. By step 2, for each $(y_i, y_j) \in \Phi$, conditions 1 and 3 of Definition 11 is satisfied. By step 3, for each $(y_i, y_j) \in \Phi$, conditions 1, 2, and 3 of Definition 11 is satisfied. Thus, Φ is a state-recognizability relation as desired. Further if by step 4, $(y_0, y_0) \in \Phi$ for all $y_0 \in Y_0$, then by Definition 12, S is state-recognizable.

Next we prove that if S is state-recognizable, then Algorithm 4 computes one state-recognizability relation. Assume Φ_1 is a state-recognizability relation under which S is state-recognizable. Then we need to show that $\Phi_1 \subseteq \Phi$. As argued above, by conditions 1 and 3

of Definition 11 and steps 1 and 2, $\Phi_1 \subseteq \Phi_0$. Also by condition 2 in Definition 11 and step 3, $\Phi_1 \subseteq \Phi$. Finally since $(y_0, y_0) \in \Phi_1$ for each $y_0 \in Y_0$ and by step 4, $\Phi_1 \subseteq \Phi$, proving the correctness of the algorithm. ■

Remark 14 The complexity of masked-composition of S and S is $O(|S|^2)$. The complexity of marking "good" states is linear in the size of $G\|S$ and quadratic in the size of S, i.e., $O(|G| \times |S|) + O(|S|^2)$. This is because for each $(y_1, y_2) \in MC(S, S)$, we need to check $|X_{syn}(y_1)| + |X_{syn}(y_2)|$ number of event containment relations and $|X_{syn}(y_1)| + |X_{syn}(y_2)|$ number of state marking status. So the total number of checks needed is,

$2\sum_{(y_1,y_2)\in Y^2}(|X_{syn}(y_1)| + |X_{syn}(y_2)|)$

$$\begin{aligned}
&= 2\sum_{y_2}(\sum_{y_1}|X_{syn}(y_1)|) + \sum_{y_1}(\sum_{y_2}|X_{syn}(y_2)|) \\
&\leq 2\sum_{y_2}|X| \times |Y| + \sum_{y_1}|X| \times |Y| \\
&= 2|X| \times |Y|[\sum_{y_1}1 + \sum_{y_2}1] \\
&= 4|X| \times |Y|^2 = O(|G| \times |S|).
\end{aligned}$$

The complexity of computing $X_{syn}(\cdot)$ is $O(|G| \times |S|)$ and that of $MC(S, S)$ is $O(|S|^2)$, and so the complexity of step 1 is $O(|G| \times |S|) + O(|G| \times |S|) + O(|S|^2) = O(|G| \times |S|) + O(|S|^2)$. The complexity of step 3 is linear in the size of $MC(S, S)$, i.e., $O(|S|^2)$. Thus, the complexity of Algorithm 4 of verifying state-recognizability of S is linear in the size of $G\|S$ and quadratic in the size of S, i.e., $O(|G| \times |S|) + O(|S|^2)$.

It follows that the complexity of checking state-recognizability of $G\|R^*$ is $O(|G| \times (|G| \times |R^*|)) + O((|G| \times |R^*|)^2) = O(|G|^2 \times |R|^2)$ (since $|R^*| = |R|$). Complexity of checking $G\|R^*$ is SC is $O(|G|^2 \times |R|)$. Complexity for checking whether $R \sqsubseteq G$ is $O(|G| \times |R|)$. Thus, the complexity of checking the existence of a (Σ_u, M)-compatible bisimilarity enforcing supervisor for deterministic-plants is $O(|G|^2 \times |R|^2)$.

5.4 Comparison with Control for Language Specification

In a prior section, we showed that a necessary and sufficient condition for the existence of a (possibly) nondeterministic supervisor such that the controlled system is bisimilar to a

(possibly) nondeterministic specification for deterministic-plants is state-achievable-bisimilar (SAB). In Kumar et al. (2005b), it was shown that a necessary and sufficient condition for the existence of a (possibly) nondeterministic supervisor for a language specification is (Σ_u, M)-achievability (called language-achievability or LA for short). A necessary and sufficient condition for the existence of a nonblocking deterministic supervisor such that the language of the controlled system equals a specification language is controllability together with observability (called language-controllability and observability or LO & LC for short). All the above conditions also apply to deterministic-plants. Thus, it is interesting to compare these conditions and explore how they are related to each other.

It was shown in Kumar et al. (2005b) that LO & LC is strictly stronger than LA. We show that SAB is a strictly stronger notion than LA, and the notions of SAB and LO & LC are not comparable in general. However, it turns out that they are equivalent requirements when the specification is deterministic and trim.

We first show that the notion of SAB is stronger than LA. Recall the definition of language-achievability. Let $L(R) \subseteq L(G)$,

> (i) $L(R)$ is said to be language-controllable with respect to $L(G)$ and Σ_u if $\forall s \in L(R)$ and $\forall a \in \Sigma_u$, $sa \in L(G) \Rightarrow sa \in L(R)$. (ii) $L(R)$ is said to be language-recognizable with respect to $L(G)$ and M if $\forall s, t \in \Sigma^*$ and $\forall a \in \Sigma$ with $M(a) = \epsilon$, $sat \in L(R) \Rightarrow sa^*t \cap L(G) \subseteq L(R)$. (iii) $L(R)$ is said to be language-achievable with respect to $L(G)$ if $L(R)$ is language-controllable and language-recognizable, and $\forall s, t \in \Sigma^*$, $\forall a \in \overline{\Sigma}, b \in \Sigma_u$ with $M(a) = M(b)$, $sat \in L(R) \Rightarrow \{sbt\} \cap L(G) \subseteq L(R)$.

Proposition 3 If R is state-achievable-bisimilar, then R is language-achievable.

Proof: Note R is LA if and only if R' is LA for some $R' \simeq R$ (since bisimilarity preserves languages). We pick R' to be such that it is SA. (Since R is SAB, such a R' exists.) So it suffices to show $L(R')$ is LA given that R' is SA. Renaming R' as R, it suffices to show that $L(R)$ is LA given that R is SA.

For notational convenience, let x_s (resp., q_s) and x_{sa} (resp., q_{sa}) be a state reached by a trace s and sa in G (resp., R), respectively. We first show that if R is SA then $L(R)$ is Σ_u-controllable.

Suppose $s \in L(R)$ and $a \in \Sigma_u$. Since R is SC, by Lemma 10, $\cup_{x \in X_{syn}(q_s)} \Sigma(x) \cap \Sigma_u \subseteq \Sigma(q_s) \cap \Sigma_u$. Note that $sa \in L(G)$ implies that $a \in \cup_{x \in X_{syn}(q_s)} \Sigma(x) \cap \Sigma_u$. Then $a \in \Sigma(q_s) \cap \Sigma_u$. I.e., $sa \in L(R)$.

Next we show that if R is SA then $L(R)$ is M-recognizable. We first prove that if $sat \in L(R)$ for $M(a) = \epsilon$, then $sa^n \cap L(G) \subseteq L(R)$ for $n \geq 0$. Since $\epsilon \in \overline{\Sigma}(q_s)$, $a \in \overline{\Sigma}(q_s)$ and $M(a) = \epsilon$, $((x_s, q_s), (x_{sa}, q_{sa})) \in \Phi$, where Φ is a state-recognizability relation under which $G \| R$ is state-recognizable (which can be proved similarly as Lemma 12). Since G is deterministic, $X_{syn}((x_{sa}, q_{sa})) = \{x_{sa}\}$. Thus, $\Sigma(x_{sa}) \cap \Sigma((x_s, q_s)) \subseteq \Sigma(x_{sa}) \cap \Sigma((x_{sa}, q_{sa}))$. If $saa \in L(G)$, i.e., $a \in \Sigma(x_{sa})$, then $a \in \Sigma(q_{sa})$, establishing base step. Note that since $\epsilon \in \overline{\Sigma}(q_s)$, $a \in \overline{\Sigma}(q_{sa})$ and $M(a) = \epsilon$, for R being SR, $((x_s, q_s), (x_{saa}, q_{saa})) \in \Phi$. Suppose $sa^n \in L(R)$, i.e., $((x_s, q_s), (x_{sa^n}, q_{sa^n})) \in \Phi$, then $\Sigma(x_{sa^n}) \cap \Sigma((x_s, q_s)) \subseteq \Sigma(x_{sa^n}) \cap \Sigma((x_{sa^n}, q_{sa^n}))$. If $sa^{n+1} \in L(G)$, i.e., $a \in \Sigma(x_{sa^n})$, then $a \in \Sigma(q_{sa^n})$, i.e., $sa^{n+1} \in L(R)$, which proves the induction step.

Knowing $sa^n \cap L(G) \subseteq L(R)$, next we prove $sa^n t \cap L(G) \subseteq L(R)$. When $|t| = 0$, then $sa^n \cap L(G) \subseteq L(R)$, establishing the base step. Supposing $sa^n t' \cap L(G) \subseteq L(R)$ for $t' \leq t$ (i.e., t' is a prefix of t). Then $((x_{sat'}, q_{sat'}), (x_{sa^n t'}, q_{sa^n t'})) \in \Phi$. Thus, $\Sigma(x_{sa^n t'}) \cap \Sigma((x_{sat'}, q_{sat'})) \subseteq \Sigma(x_{sa^n t'}) \cap \Sigma((x_{sa^n t'}, q_{sa^n t'}))$. If $sa^n t' \sigma \in L(G)$, i.e., $\sigma \in \Sigma(x_{sa^n t'})$, then $\sigma \in \Sigma(q_{sa^n t'})$. I.e., $sa^n t' \sigma \cap L(G) \subseteq L(R)$, which proves the induction step.

Next, we show that if R is SA then R is (Σ_u, M)-achievable. We first notice that $\{sb\} \cap L(G) \subseteq L(R)$ for $b \in \Sigma_u$ because R is SC. We next prove that $\{sbt\} \cap L(G) \subseteq L(R)$. When $|t| = 0$, then $\{sb\} \cap L(G) \subseteq L(R)$, establishing the base step. Supposing $\{sbt'\} \cap L(G) \subseteq L(R)$ for $t' \leq t$ (i.e., t' is a prefix of t). Note that $((x_s, q_s), (x_s, q_s)) \in \Phi$. Also $M(b) = M(a)$ and R is SR, then $((x_{sa}, q_{sa}), (x_{sb}, q_{sb})) \in \Phi$. Then $((x_{sat'}, q_{sat'}), (x_{sbt'}, q_{sbt'})) \in \Phi$. Thus, $\Sigma(x_{sbt'}) \cap \Sigma((x_{sat'}, q_{sat'})) \subseteq \Sigma(x_{sbt'}) \cap \Sigma((x_{sbt'}, q_{sbt'}))$. If $\{sbt'\sigma\} \subseteq L(G)$, i.e., $\sigma \in \Sigma(x_{sbt'})$, then $\sigma \in \Sigma(q_{sbt'})$. I.e., $\{sbt'\sigma\} \cap L(G) \subseteq L(R)$, which proves the induction step. ■

The following example shows that SAB is strictly stronger than LA.

Example 17 Consider $L(G) = a(c+d)+bc$ and a deterministic state machine R with $L(R) = ad + bc$, where $M(a) = M(b) \neq \epsilon$ and $\Sigma_u = \emptyset$. Let $x_a = \alpha(x_0, a)$ and $x_b = \alpha(x_0, b)$, then

$\Sigma(x_a) = \{c, d\}$ and $\Sigma(x_b) = \{c\}$. Suppose R is SAB. Then $G\|R^* = G\|R$ is SA. Let $q_a \in \delta(q_0, a)$ with $\Sigma(q_a) = \{d\}$ and $q_b \in \delta(q_0, b)$ with $\Sigma(q_b) = \{c\}$. Then there should exist a state-recognizability relation Φ such that $((x_a, q_a), (x_b, q_b)) \in \Phi$. For this to hold, $\Sigma(x_a) \cap \Sigma((x_b, q_b)) \subseteq \Sigma(x_a) \cap \Sigma((x_a, q_a))$ should hold, which is not true since $\Sigma(x_a) \cap \Sigma((x_b, q_b)) = \{c\}$, $\Sigma(x_a) \cap \Sigma((x_a, q_a)) = \{d\}$. Thus, R is not SAB, a contradiction to the hypothesis. It is easy to verify that $L(R)$ is language-recognizable.

Thus, SAB is strictly stronger than LA. Next we show that SAB and LC & LO are not comparable in general.

Example 18 Consider an example shown in Figure 5.13. The observation mask is given by: $M(b_1) = M(b_2)$ and identity mask for the other events. Assume all events are controllable. Then state-achievable-bisimilar (SAB) reduces to state-recognizable-bisimilar (SRB). We show that R is not LO but it is SRB, whereas R' is LO but it is not SRB.

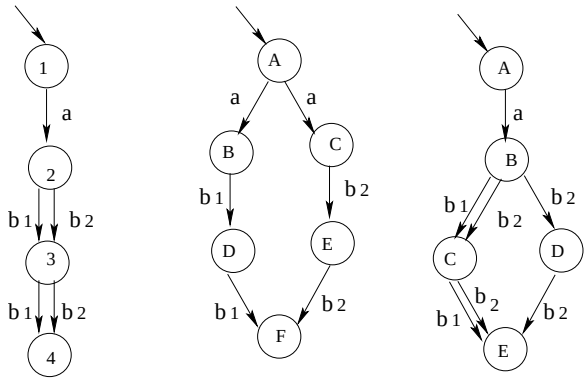

Figure 5.13 G (left), R (middle), and R' (right)

R is not LO since $ab_1, ab_2 \in L(R)$ with $M(ab_1) = M(ab_2)$, $ab_1b_1 \in L(R)$ and $ab_2b_1 \in L(G) - L(R)$. However, R is SRB. A desired state-recognizability relation $\Phi \subseteq (X \times Q^*)^2 = (X \times Q)^2$ is given by:

$$\begin{aligned} \Phi \;=\; & \{((1, A), (1, A)), ((2, B), (2, B)), ((2, C), (2, C)), \\ & ((3, D), (3, D)), ((3, E), (3, E)), ((4, F), (4, F))\}. \end{aligned}$$

Next, R' is LO since $L(R') = L(G)$. However, R' is not SRB. At state $(2, B)$ of $G\|(R')^* = G\|R'$, $M(b_1) = M(b_2)$. For b_2-successor $(3, D)$, there should exist a b_1-successor such that

the condition 1 of Definition 11 holds. Note that $(3, C)$ is the only b_1-successor of $(2, B)$, and $X_{syn}((3, D)) = \{3\}$,

$$[\Sigma(3) \cap \Sigma((3, C)) = \{b_1, b_2\}] \not\subseteq [\Sigma(3) \cap \Sigma((3, D)) = \{b_2\}],$$

implying R' is not SRB.

Although SAB and LC & LO are not comparable in general, they are equivalent when R is deterministic and trim as shown next.

Proposition 4 Given deterministic G, deterministic and trim R with $R \sqsubseteq G$, R is state-achievable-bisimilar if and only if R is language-controllable & language-observable.

Proof: We first show the necessity. Since G and R are deterministic, $G||R^* = G\|R$ is deterministic, and so $(G||R)^{\Phi}$ and $(G||R)^{\Phi,\Sigma_u} = S$ are deterministic. I.e., supervisor S such that $G||S \simeq R$ can be chosen to be deterministic. $G||S \simeq R$ implies $L(G||S) = L(R)$ and $L_m(G||S) = L_m(R)$, and so since S is deterministic, R is LC & LO.

For the sufficiency, when $L(R) = pr(L_m(R))$ is LC & LO, there exists a deterministic S such that $L(G||S) = L(R)$ and $L_m(G||S) = L_m(R)$. Since $G||S$ and R are both deterministic, their language model equivalence implies their bisimulation equivalence, i.e., $G||S \simeq R$. So, it follows that R is SAB. ■

5.5 Conclusion

We extended the small model theorem by showing that a control and observation compatible supervisor for enforcing bisimulation equivalence between the specification and the controlled system exists if and only if it exists over a certain finite state space, namely the power set of Cartesian product of the plant and the specification state spaces. This proves the decidability of bisimilarity enforcing control under partial observation for general nondeterministic systems and nondeterministic specifications. The results are illustrated through a simple example.

The existence condition we find for the special case of deterministic-plants is polynomially verifiable, whereas the complexity of synthesizing a supervisor (when one exists) is singly

exponential. These complexities are similar to the ones for control under partial observation in the completely deterministic setting Tsitsiklis (1989).

We obtained a necessary and sufficient condition for the existence of a supervisor in terms of the notion of state-achievable-bisimilar introduced in this paper, and presented an algorithm of polynomial complexity for verifying it. The presence of partial observation poses a new challenge in the setting of nondeterministic specifications. It turns out certain properties that are relevant in the setting of partial observation such as observation-compatibility are not preserved under bisimilarity. So unlike the deterministic setting where when a supervisor exists, the specification itself can be chosen as a supervisor, we cannot use the specification itself as a supervisor. An elaborate computation is required to show that the composition of the plant and the specification when suitably transformed through certain state-mergers can be used as a supervisor. Further the state-mergers are unique: Unlike the usual situations where states to be merged belong to certain equivalence class, the states to be merged in our case don't form a partition, rather only a cover (see Algorithm 2). Such a construction (i.e., one involving state-mergers based on a cover rather than a partition) is quite novel, and to the best of our knowledge the first in the literature dealing with supervisory control theory. This particular idea of state-mergers over a cover (rather a partition) may be useful in future in other problems of relevance to supervisory control.

We have identified the condition of state-achievable-bisimilar for the existence of a supervisor. When this condition is not satisfied, the specification (or the plant) must be altered. An interesting future direction to consider is developing a method for "minimally altering" the specification so that the above required condition is satisfied.

CHAPTER 6. SUPERVISORY CONTROL FOR SIMULATION EQUIVALENCE

In this chapter, we study the control of DESs to ensure simulation equivalence of the controlled system and a given (nondeterministic) specification. The simulation equivalence can express specifications in the temporal logic of ACTL* (the universal fragment of CTL*) Bensalem et al. (1992); Maidi (2000). The expressivity of simulation equivalence may suffice for certain applications and in which case the more general requirement of bisimulation equivalence need not be imposed. Such a choice results in a complexity gain since as we demonstrate in the chapter, the simulation equivalence control problem remains polynomially solvable. The results are further generalized to the "range" control problem, where the controlled behavior must lie in a range specified by a lower and an upper bound behavior (ordering is defined by the simulation relation). We show that our necessary and sufficient condition for the existence of a similarity enforcing supervisor for deterministic plant specializes to the condition of "state-controllable-similar", which is another new concept introduced in this chapter.

6.1 Supervisory Control for Simulation Equivalence

In this section, we study the control of a (nondeterministic) plant to ensure simulation equivalence of controlled plant and given (nondeterministic) specification. In order to find a Σ_u-compatible similarity enforcing supervisor we examine the class of all S such that $G\|S \sim R$. It turns out that this class possesses an infimal elements and we provide an algorithm for the computation of such an element.

The following lemma is needed before we proceed.

Lemma 14 Given $G_1 \sqsubseteq G_2$ and $G_1' \sqsubseteq G_2'$, it holds that $G_1\|G_1' \sqsubseteq G_2\|G_2'$.

Proof: By Theorem 3, $G_1 \| G_1' \sqsubseteq G_1$ and $G_1 \| G_1' \sqsubseteq G_1'$. Also since $G_1 \sqsubseteq G_2$ and $G_1' \sqsubseteq G_2'$, it follows that $G_1 \| G_1' \sqsubseteq G_2$ and $G_1 \| G_1' \sqsubseteq G_2'$. Therefore, by Theorem 3, we have $G_1 \| G_1' \sqsubseteq G_2 \| G_2'$. ■

Let $\mathcal{S} := \{S \mid S \ \Sigma_u\text{-compatible}, G \| S \sim R\}$. The following lemma shows that $\mathcal{S}$ possess an infimal element.

Lemma 15 $S_1, S_2 \in \mathcal{S}$ implies $S_1 \| S_2 \in \mathcal{S}$.

Proof: We need to prove $S_1 \| S_2$ is Σ_u-compatible and $G \| (S_1 \| S_2) \sim R$. Since S_1 and S_2 are Σ_u-compatible, from Definition 10, it is obvious that $S_1 \| S_2$ is Σ_u-compatible. Next we show $G \| (S_1 \| S_2) \sim R$, for which we need to show $G \| (S_1 \| S_2) \sqsubseteq R$ and $R \sqsubseteq G \| (S_1 \| S_2)$. For $i = 1, 2$, $S_i \in \mathcal{S}$ implies $G \| S_i \sim R$, which further implies $G \| S_i \sqsubseteq R$. From Theorem 3, we have $(G \| S_1) \| (G \| S_2) \sqsubseteq G \| S_i \sqsubseteq R$. Next note that $(G \| S_1) \| (G \| S_2) = (G \| G) \| (S_1 \| S_2)$ and $G \sqsubseteq G \| G$ (follows from Lemma 14). So from Lemma 14, we have $G \| (S_1 \| S_2) \sqsubseteq (G \| G) \| (S_1 \| S_2)$, i.e., $G \| (S_1 \| S_2) \sqsubseteq (G \| S_1) \| (G \| S_2) \sqsubseteq R$. Similarly, one can show $R \sqsubseteq G \| (S_1 \| S_2)$. ■

Next we present an algorithm for computing an element of $INF(\mathcal{S})$ when $\mathcal{S}$ is nonempty.

Algorithm 5 Suppose G, R and Σ_u are such that $\mathcal{S} \neq \emptyset$. Then the following algorithm computes an automaton $R_u \in INF(\mathcal{S})$. $R_u = (Q \cup \{dump\}, \Sigma, \delta_u, Q_0, Q_m)$, where

$$\forall q \in Q \cup \{dump\}, \sigma \in \overline{\Sigma} : \delta_u(q, \sigma) := \begin{cases} \delta(q, \sigma) & \text{if } \sigma \in \Sigma(q) - \{\epsilon\} \\ dump & \text{if } \sigma \in \Sigma_u - \Sigma(q) \\ \delta(q, \epsilon) & \text{if } \sigma = \epsilon \end{cases}$$

In other words, R_u is obtained by adding in R an extra dump state and adding the "missing" uncontrollable transitions from each state to the dump state.

The following theorem proves the correctness of the algorithm.

Theorem 12 Algorithm 5 is correct. I.e., R_u is Σ_u-compatible, and $\mathcal{S} \neq \emptyset$ implies $R_u \in INF(\mathcal{S})$.

Proof: From the construction of R_u, we know that each $\sigma \in \Sigma_u$ is defined at each state of R_u. So R_u is Σ_u-compatible.

To prove the infimality of R_u under nonemptiness of $\mathcal{S}$, we need to prove that if there exists a Σ_u-compatible S such that $G\|S \sim R$, then $G\|R_u \sim R$ and $R_u \sqsubseteq S$. We first prove $R_u \sqsubseteq S$. Note that $G\|S \sim R$ implies $R \sqsubseteq_{\Phi_1} S$. Using the fact that S is Σ_u-compatible, it can be show that $R_u \sqsubseteq_{\Phi_2} S$, where

$$\Phi_2 := \Phi_1 \cup \{(dump, y) | \exists (q, y') \in \Phi_1, \sigma \in \Sigma_u : \delta(q, \sigma) = dump, y \in \beta(y', \sigma)\}.$$

Next, since $R_u \sqsubseteq S$, from Lemma 14, $G\|R_u \sqsubseteq G\|S$. This together with the fact $G\|S \sqsubseteq R$ implies $G\|R_u \sqsubseteq R$. It remains to show that $R \sqsubseteq G\|R_u$. Since R_u is obtained by adding an extra state and extra transitions, it is obvious that $R \sqsubseteq R_u$. The fact $R \sqsubseteq G$ follows from the fact that $R \sqsubseteq G\|S \sqsubseteq G$. This completes the proof. ■

The following result follows from Theorem 12 and provides a necessary and sufficient condition for the existence of a similarity enforcing supervisor.

Theorem 13 Given G and R, there exists a Σ_u-compatible supervisor S such that $G\|S \sim R$ if and only if $G\|R_u \sim R$ (or equivalently, $G\|R_u \sqsubseteq R \sqsubseteq G$), where R_u is as computed in Algorithm 5. Further when the existence condition holds, R_u can be chosen as a supervisor.

Proof: Sufficiency is obvious since S can be chosen as R_u. For necessity, suppose the desired S exists. Then $\mathcal{S} \neq \emptyset$ and so from Theorem 12, $R_u \in INF(\mathcal{S}) \subseteq \mathcal{S}$. Since $R_u \in \mathcal{S}$, the necessity follows. ■

Remark 15 The complexity of checking $G\|R_u \sqsubseteq R$ is linear in the size of the plant and quadratic in the size of the specification, whereas $R \sqsubseteq G$ can be checked linearly in the size of G and R. Also, R_u can be used as supervisor, whose can be computed linearly in the size of R. (R_u has just an extra added state and compared to R.)

So far we have studied the "target" control problem when the controlled system $G\|S$ and specification R are simulation equivalent, i.e., $G\|S \sim R$. This is equivalent to saying $R \sqsubseteq G\|S \sqsubseteq R$, which is a special case of a more general "range" control problem $A \sqsubseteq G\|S \sqsubseteq E$. Here the automaton A specifies a *minimally adequate* behavior, whereas the automaton E specifies a *maximally acceptable* behavior. Note that in a "target" control problem, $A = E = R$. In the remainder of the section we extend our results to the range control problem.

Given an automaton R and the set of uncontrollable events, we have shown the computation of R_u in Algorithm 5. We show in the next lemma that the simulation relation is preserved under such a computation.

Lemma 16 Given $R_1 \sqsubseteq R_2$, it holds that $R_{1u} \sqsubseteq R_{2u}$.

Proof: $R_1 \sqsubseteq R_2$ implies exists a simulation relation $\Phi_1 \subseteq Q_1 \times Q_2$ such that $Q_{01} \times Q_{02} \subseteq \Phi_1$. Also,

$$\Phi_1 = \{(q_1, q_2) \mid q_1 \sqsubseteq q_2\}.$$

Define a relation $\Phi_2 \subseteq (Q_1 \cup \{dump_1\}) \times (Q_2 \cup \{dump_2\})$ as:

$$\Phi_2 = \Phi_1 \cup \{(dump_1, dump_2)\}.$$

Then it is easy to see that Φ_2 is a simulation relation and $R_{1u} \sqsubseteq_{\Phi_2} R_{2u}$. ■

We next present a necessary and sufficient condition for the "range" control problem.

Theorem 14 Given plant G and lower and upper bound specifications $A \sqsubseteq E$, there exists a Σ_u-compatible supervisor S such that $A \sqsubseteq G\|S \sqsubseteq E$ if and only if $A \sqsubseteq G$ and $G\|A_u \sqsubseteq E$. Further when the existence condition holds, A_u can be chosen as a desired supervisor.

Proof: (*If*) Let $S = A_u$, where A_u is constructed by Algorithm 5 and is Σ_u-compatible. Then $A \sqsubseteq G$ and $A \sqsubseteq A_u$ implies $A \sqsubseteq G\|A_u$, which together with $G\|A_u \sqsubseteq E$, yields $A \sqsubseteq G\|A_u \sqsubseteq E$. This proves the sufficiency.

(*Only If*) By Corollary 1, $A \sqsubseteq G\|S$ implies $A \sqsubseteq G$. It remains to show that $G\|A_u \sqsubseteq E$. Suppose $G\|S := R'$, then $A \sqsubseteq G\|S \sqsubseteq E$ implies $A \sqsubseteq R' \sqsubseteq E$. By Lemma 16, $A \sqsubseteq R'$ implies $A_u \sqsubseteq R'_u$. By Lemma 14, we have $G\|A_u \sqsubseteq G\|R'_u$. Also, $G\|S = R'$ implies $G\|S \sim R'$, then by Theorem 13, $G\|R'_u \sqsubseteq R'$. Combining previous inequations yields $G\|A_u \sqsubseteq G\|R'_u \sqsubseteq R' \sqsubseteq E$. This completes the proof. ■

Remark 16 The complexity for checking $G\|A_u \sqsubseteq E$ is linear in the sizes of A, G and E. Also, from the proof of Theorem 14, A_u can serve as a supervisor for the "range" control problem, where A_u can be computed linearly in size of the lower bound specification A.

The following example serves to illustrate the simulation equivalence enforcing control.

Example 19 Consider a simple vending machine that delivers a cookie or a candy in exchange for a coin, whose state machine model is shown in Figure 6.1. Upon getting a coin, the vending machine nondeterministically transits to one of two states. At each state, user can wait for a delivery or push a button. In the first state if the user chooses to wait, the machine times out delivering a cookie; whereas if the user chooses to push the button the machine transits to the second state and remains there with additional pushes of the button. In the second state, the pushing of the button does not cause a state change but when the user opts to wait, the machine times out and delivers either a candy or a cookie. Once a delivery is completed, the machine returns to its initial state. The timeout event is deemed uncontrollable, whereas the other events are controllable.

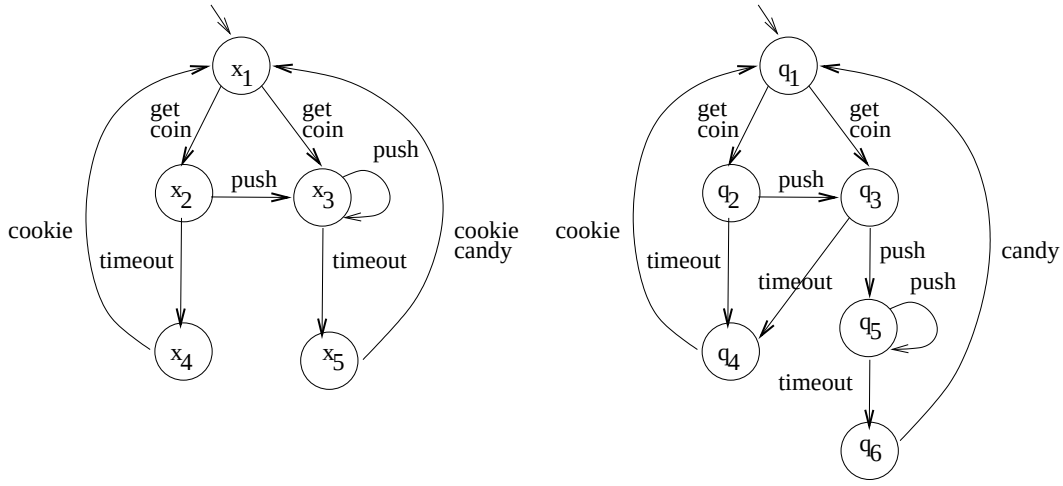

Figure 6.1 Model G of vending machine (left) and its specification R (right)

Note that in the above vending machine example it is not possible for a user to receive a candy with certainty, which is an undesirable behavior. To rectify this situation, a desired specification is shown in Figure 6.1. According to the specification, after a user inserts a coin, the vending machine nondeterministically transits to one of two states. However, regardless of the state reached, if the user chooses to wait, the machine delivers a cookie; whereas if the user chooses to push the button at least twice (before timeout), the machine delivers a candy. If the user opts to push the button once and then to wait, then the machine delivers a cookie or a

candy depending on the initial nondeterministic transition made. Note since the specification allows such a nondeterministic choice, it is not adequate to use a language to capture the behavior of the specification.

The above specification can be expressed in a temporal logic syntax as follows: After receiving a coin, for all paths, deliver a cookie if the button is not pushed (before timeout); deliver a candy if the button is pushed at least twice (before timeout); deliver a cookie or a candy if the button is pushed only once (before timeout). This is an instance of a "universal" temporal logic specification (no existential quantifier is needed to express the specification). Since a universal temporal logic specification is preserved under simulation equivalence, it suffices to require that the controlled plant and specification be simulation equivalent.

Our goal is to find a Σ_u-compatible supervisor S for the vending machine such that $G \| S \sim R$. We first check whether $R \sqsubseteq G$. We find the following simulation relation Φ_1 exists between R and G:

$$\Phi_1 = \{(q_1, x_1), (q_2, x_2), (q_3, x_3), (q_4, x_4), (q_5, x_3), (q_6, x_5)\},$$

implying $R \sqsubseteq_{\Phi_1} G$. Next, we check whether $G \| R_u \sqsubseteq R$. We construct R_u using Algorithm 5, and the result is depicted in Figure 6.2. The synchronous composition of R_u with plant, namely $G \| R_u$, is shown in Figure 6.2.

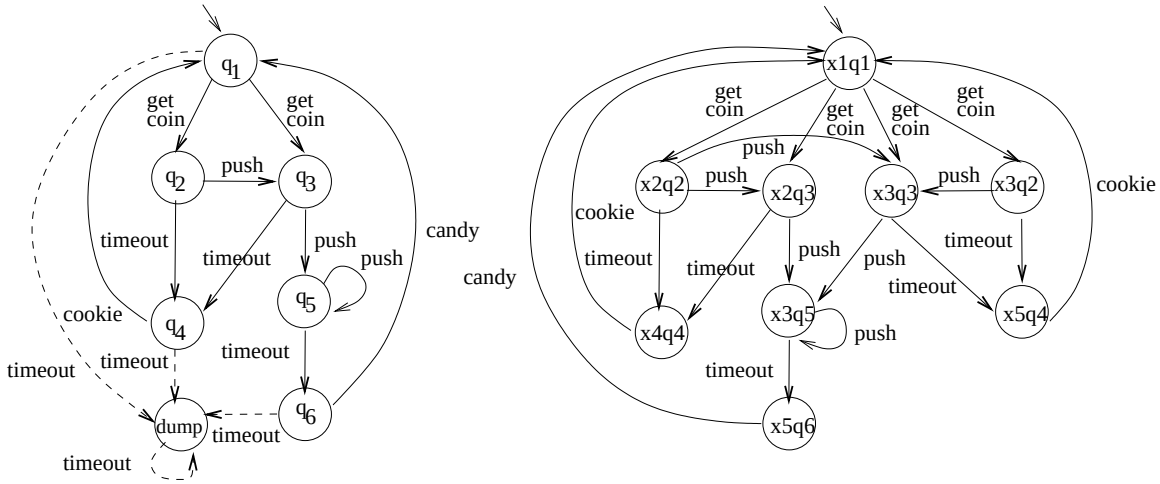

Figure 6.2 R_u (left) and $G \| R_u$ (right)

The following simulation relation Φ_2 exists between $G\|R_u$ and R:

$$\begin{aligned}\Phi_2 \quad := \quad & \{((x_1,q_1),q_1),((x_2,q_2),q_2),((x_3,q_3),q_3),((x_4,q_4),q_4),((x_5,q_4),q_4),\\ & ((x_2,q_3),q_3),((x_3,q_2),q_2),((x_3,q_5),q_5),((x_5,q_6),q_6)\},\end{aligned}$$

implying $G\|R_u \sqsubseteq_{\Phi_2} R$. It follows from Theorem 13 that there exists a Σ_u-compatible supervisor to enforce simulation equivalence between the controlled system and the specification, and R_u serves as such a supervisor. ■

6.2 Specialization to Deterministic Case

The results obtained in section 6.1 are applicable to deterministic plants. However the special case of deterministic plants is of separate interest since a weaker condition may be required for the existence of a supervisor. In fact this happens to be the case when the specification is of bisimulation equivalence, where it was shown that the bisimulation equivalence control problem can be solved polynomially when plant model is deterministic. (No polynomial algorithm is known when the plant is nondeterministic.) A necessary and sufficient condition for the existence of a bisimilarity enforcing supervisor for a deterministic G and a possibly nondeterministic R is that R be simulated by G and R^* be *state-controllable* with respect to G and Σ_u (Recall R^* is the NSM obtained by replacing the transition function δ of R by the transition function δ^* and $R^* = R$ in the absence of ϵ-transitions.) In this section we show that if we only require the simulation equivalence of the controlled plant and the specification, then a weaker condition than state-controllability is required (as expected). In this section we introduce that weaker condition, called state-controllable-similar, prove its necessity and sufficiency, and present a way to test it.

Definition 16 Given automata G and R with $L(R) \subseteq L(G)$, we say R is a *state-controllable-similar (SCS)* with respect to G and Σ_u if it is simulation equivalent to a system R' that is state-controllable with respect to G and Σ_u.

We recall that the language-controllability requires the following:

$$s \in L(R), \sigma \in \Sigma_u \text{ such that } s\sigma \in L(G) \Rightarrow \exists q \in \delta^*(Q_0, s), \sigma \in \Sigma(q).$$

It is clear that when R is deterministic, state-controllability of R with respect to G and Σ_u reduces to language-controllability of $L(R)$ with respect to $L(G)$ and Σ_u. Also it can be easily deduced that Σ_u-compatibility implies state-controllability, i.e., the latter is a weaker notion.

The notion of SCS is stronger than language-controllable (LC) and weaker than state-controllable (SC). Recall that for deterministic G and possibly nondeterministic R with $L(R) \subseteq L(G)$, LC serves as a necessary and sufficient condition for a language equivalence control, whereas SC serves as a necessary and sufficient condition for a bisimulation equivalence control. We show that the "intermediate" condition of SCS serves as a necessary and sufficient condition for a simulation equivalence control. The next example illustrates the above various concepts of controllability.

Example 20 Consider automata $G = R$, R', and R'' shown in Figure 6.3, and suppose $\Sigma_u = \{b\}$. Notice that in G, uncontrollable event b is defined after trace a. R'' is SC since at

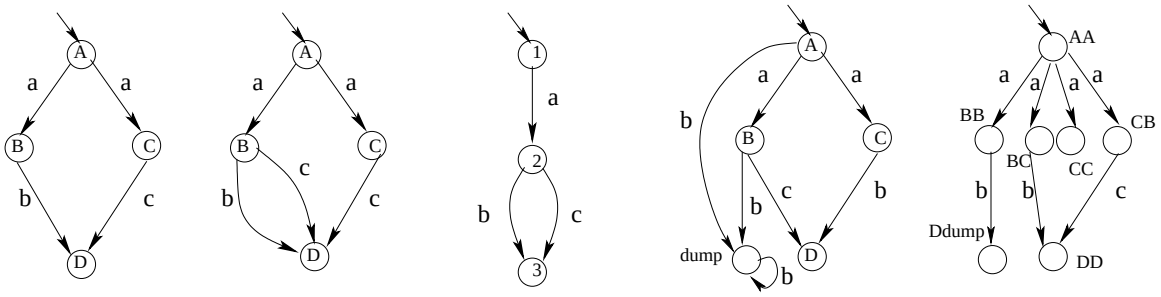

Figure 6.3 R (First), R' (second), R'' (third), R_u (fourth) and $R \| R_u$ (fifth)

state 2 reached by trace a, event b is defined. It follows that R'' is also SCS and LC. On the other hand, R' is not SC since at state C reached by trace a, event b is undefined. However R' is SCS, since $R' \sim R''$, where R'' is SC. Also, similarity of R' and R'' implies their language equivalence. So since R'' is LC, so is R'. Finally, R is LC since $L(R) = L(G)$. However R is not SC since at state C reached by trace a, event b is undefined. Also R is not SCS since it can be argued that one cannot find a SC $\overline{R}$ such that $\overline{R} \sim R$: For the reason that $\overline{R}$ simulates R, event c must be defined at some state reached by trace a in $\overline{R}$. Further for the reason that $\overline{R}$ is SC, event b must be defined at this state. I.e., there must exist a state in $\overline{R}$ reached by

trace a where events b and c are both defined. Since none of the states of R reachable by trace a have both events b and c defined, R cannot simulate $\overline{R}$.

Next, we establish a necessary and sufficient condition for the existence of a similarity enforcing control for deterministic plants.

Theorem 15 Given deterministic plant G and possibly nondeterministic specification R, exists a Σ_u-compatible supervisor S such that $G\|S \sim R$ if and only if R is simulated by G and state-controllable-similar with respect to G and Σ_u.

Proof: (*Only If*) From Theorem 13, $G\|S \sim R$ implies $G\|R_u \sqsubseteq R \sqsubseteq G$. Since $R \sqsubseteq G$ and $R \sqsubseteq R_u$ (from construction of R_u), it follows that $R \sqsubseteq G\|R_u$. So, $G\|R_u \sim R$. By construction, R_u is Σ_u-compatible and since G is deterministic, from Lemma 8, $G\|R_u$ is state-controllable. Since R is simulation equivalent to $G\|R_u$, it follows that R is state-controllable-similar.

(*If*) Let R be similar to R' that is SC. Define S to be R' with each state augmented with self-loops on all the undefined uncontrollable events of that state. Then $G\|S \simeq G\|R'$ (by Lemma 2). Since $R' \sim R \sqsubseteq G$, it follows that $R' \sqsubseteq G$. Also, $R' \sqsubseteq R'$, and so $R' \sqsubseteq G\|R'$. On the other hand, $G\|R' \sqsubseteq R'$ by Theorem 3, and so we have $G\|R' \sim R'$. It follows that $G\|S \simeq G\|R' \sim R$, which completes the proof. ■

The next theorem presents a method to verify the property of SCS of R with respect to G and Σ_u.

Theorem 16 Given deterministic G and $R \sqsubseteq G$, R is SCS with respect to G if and only if $G\|R_u \sqsubseteq R$.

Proof: For deterministic G, by Theorems 13 and 15 we have the equivalence:

$$[R \text{ is } G\text{-simulated and } G\|R_u \sqsubseteq R] \Leftrightarrow [R \text{ is } G\text{-simulated and } R \text{ is SCS}].$$

This can be rewritten as,

$$[R \text{ is } G\text{-simulated}] \Rightarrow [G\|R_u \sqsubseteq R \Leftrightarrow R \text{ is } SCS].$$

■

Remark 17 From Theorem 16 we can test whether R is SCS by testing whether $G\|R_u \sqsubseteq R$. It follows that the complexity of checking SCS is linear in the size of G and quadratic in the size of R. The complexity of checking R is simulated by G is linear in sizes of G and R. Thus the complexity of checking the existence of a supervisor for a similarity enforcing control of deterministic plants is $O(|G| \times |R|^2)$.

In the following corollary we show that when the existence condition of Theorem 15 holds, R_u can be chosen as a desired supervisor.

Corollary 3 Given deterministic plant G and possibly nondeterministic specification R, if R is simulated by G and state-controllable-similar with respect to G, then R_u as computed in Algorithm 5 can serve as a similarity enforcing supervisor, i.e., $G\|R_u \sim R$.

Proof: By Algorithm 5, R_u is Σ_u-compatible. Since $R \sqsubseteq G\|R_u$ (see the proof of Theorem 15) and $G\|R_u \sqsubseteq R$ (see Theorem 16), we obtain $G\|R_u \sim R$ as desired. ■

Remark 18 We showed in Theorem 16 that when G is deterministic and R is simulated by G, R being SCS is equivalent to $G\|R_u \sqsubseteq R$. In general, however (when G is nondeterministic), R being SCS is stronger than $G\|R_u \sqsubseteq R$. This can be illustrated by considering automaton $G = R$ drawn in Figure 6.3. R_u and $G\|R_u = R\|R_u$ are also drawn in Figure 6.3. It can be verified $G\|R_u \sqsubseteq R$. However, as explained in Example 20, R is not SCS.

We illustrate the results of this section through an example.

Example 21 Consider a message transmission system, shown in Figure 6.4, that sends messages from a sender to a receiver.

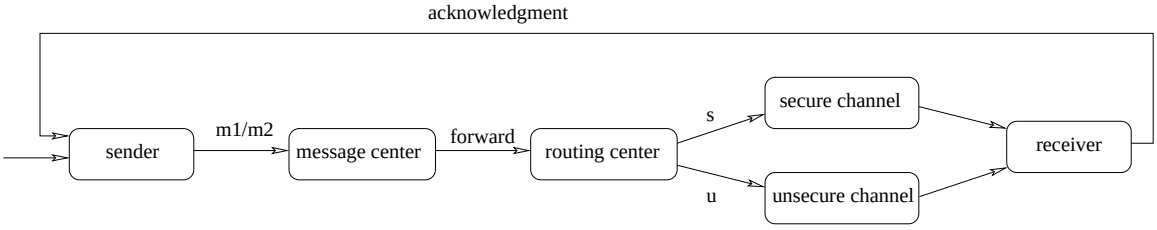

Figure 6.4 Block diagram of a message transmission system

Two types of messages are generated by the sender, m_1 and m_2, which are first received by a message center. The messages are then forwarded (event f) to a routing center which decides along which channels the messages be routed. Two types of channels, secure (s) and unsecure (u), are available for routing. Upon a successful reception, an acknowledgment (a) is sent by the receiver to the sender, allowing transmission of another message. The acknowledgment is generated automatically, and is treated as an uncontrollable event. The deterministic automaton G, drawn in Figure 6.5, models the above behavior.

A specification for the legal behavior of the system is also drawn in Figure 6.5. It requires that messages of type 1 (m_1) be transmitted over the secure channel, while no such restriction is imposed on the type of channel to be used for the transmission of the messages of the second type (m_2). However, once a message of type 2 gets forwarded, it (nondeterministically) finds the routing center to be in one of its two states: In the first state, the transmission occurs on the secure channel, whereas in the second state, on the unsecure channel.

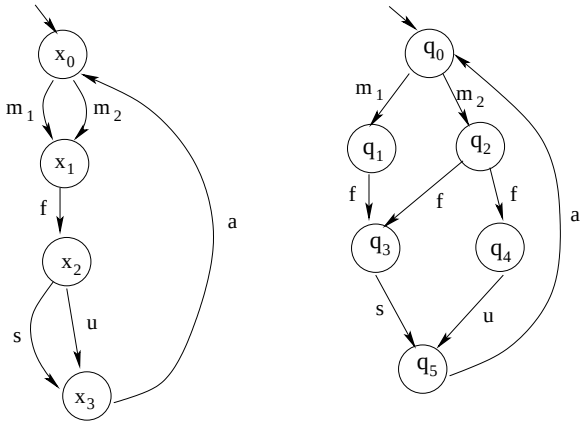

Figure 6.5 Model G of message transmission system (left) and its specification R (right)

It is easy to verify that the specification language is a controllable sublanguage of the plant language. If we apply the supervisory control results from the deterministic setting and use a deterministic generator of the specification language as a supervisor, the controlled system will be a deterministic generator of the specification language (since the plant is given to be deterministic, whereas supervisor is constructed to be deterministic, and plant language is a

superlanguage of the specification language). A deterministic generator of the specification language however will allow both the choices (secure as well as unsecure channel) for the routing of *all* messages of type 2 after they have arrived at the routing center. This situation is not permitted by the desired specification, and so the specification will be violated.

Our goal is to find a Σ_u-compatible supervisor S for the message transmission system such that $G\|S \sim R$. To do this, we first check whether R is simulated by G (i.e., $R \sqsubseteq G$). We find the following simulation relation exists between R and G:

$$\Phi_1 = \{(q_0, x_0), (q_1, x_1), (q_2, x_1), (q_3, x_2), (q_4, x_2), (q_5, x_3)\},$$

implying $R \sqsubseteq_{\Phi_1} G$.

Next, we need to check whether R is SCS. By Theorem 16, we need to check whether $G\|R_u \sqsubseteq R$. For this we need to construct R_u using Step 1 of Algorithm 5. The constructed R_u is depicted in Figure 6.6. The synchronous composition of G and R_u is shown in Figure 6.6. We find the following simulation relation Φ_2 exists between $G\|R_u$ and R:

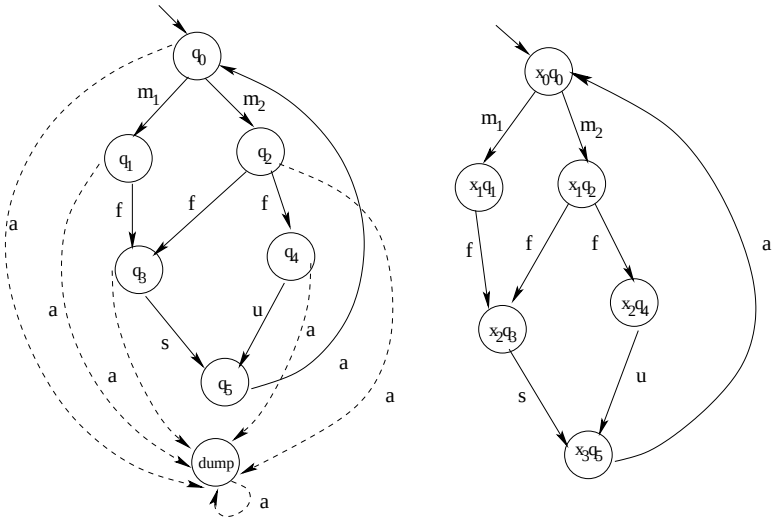

Figure 6.6 R_u (left) for R and $G\|R_u$ (right) for G and R of Figure 6.5

$$\Phi_2 = \{(x_0, q_0), q_0), ((x_1, q_1), q_1), ((x_1, q_2), q_2), (((x_2, q_3), q_3), (x_2, q_4), q_4), ((x_3, q_5), q_5)\},$$

i.e., $G\|R_u \sqsubseteq_{\Phi_2} R$. Thus we conclude that there exists a Σ_u-compatible supervisor to enforce simulation equivalence between the controlled system and the specification, and R_u can serve

as a supervisor.

To verify whether using R_u as a supervisor yields $G\|R_u \sim R$, we search for a similarity relation between the controlled system $G\|R_u$ and the specification R. A similarity relation Φ_3 between $G\|R_u$ and R is given by:

$$\begin{aligned}\Phi_3 \quad = \quad & \{(x_0,q_0),q_0),((x_1,q_1),q_1),((x_1,q_2),q_2),((x_2,q_3),q_3),\\ & ((x_2,q_4),q_4),((x_3,q_5),q_5)(q_0,(x_0,q_0)),(q_1,(x_1,q_1)),\\ & (q_2,(x_1,q_2)),(q_3,(x_2,q_3)),(q_4,(x_2,q_4)),(q_5,(x_3,q_5))\}.\end{aligned}$$

A meaning for the control being exercised is as follows. In the plant model, the routing center can be thought to have a single queue for all arrived messages. When the routing center is ready to put a message on a channel it picks one of the messages from the queue (say from the head of the queue) and places it on either of the two channels. The controller restricts this behavior of the routing center by essentially implementing two queues, *one for each channel* (and not one for each message). Upon arrival, messages of type 1 are always placed in the queue for the secure channel, whereas the messages of the type 2 can be placed in either of the two queues. The exact channel selection for a message of type 2 can be done for example based on the lengths of two queues. However since the lengths of the two queues at any given time are not known in advance, the selection of a channel essentially occurs nondeterministically for each message of type 2.

6.3 Simulation Equivalence via Deterministic Control

The condition $G\|R_u \sqsubseteq R \sqsubseteq G$ (or equivalently, $G\|R_u \sim R \sqsubseteq G$), is necessary and sufficient for the existence of a similarity enforcing supervisor. When a supervisor exists, R_u can be chosen to be one. Clearly, R_u is deterministic if and only if R is deterministic. But a deterministic supervisor may exist even when R is not deterministic, and we present a necessary and sufficient condition for the same. The point of this exercise is to show two things, (i) existence condition for deterministic supervisor is stronger than that for nondeterministic one (this is to be expected), and (ii) the time complexity of verifying existence of a determin-

istic supervisor is exponential. Thus we can draw the conclusion that it is preferable to opt for a nondeterministic supervisor. The following theorem presents a necessary and sufficient condition for its existence. We use $det(R)$ to denote the deterministic generator of $L(R)$.

Theorem 17 Given G and R, there exists a Σ_u-compatible deterministic supervisor S such that $G\|S \sim R$ if and only if

- $R \sqsubseteq G$,
- $L(R)$ is controllable with respect to $L(G)$ and Σ_u, and
- $G\|det(R) \sqsubseteq R$.

Proof: (*Only if*) Since existence of similarity enforcing supervisor ($G\|S \sim R$) implies existence of a language enforcing supervisor ($L(G\|S) = L(R)$), $L(R)$ must be controllable with respect to $L(G)$ whenever a similarity enforcing supervisor exists. In other words, a necessary condition is that $L(R)$ is controllable. Next, $G\|S \sim R$ implies $R \sqsubseteq S$ and $R \sqsubseteq G$. Since $R \sqsubseteq S$ implies $L(R) \subseteq L(S)$ and $L_m(R) \subseteq L_m(S)$, which further implies $det(R) \sqsubseteq det(S)$, and further since $det(S) = S$, we have $det(R) \sqsubseteq S$. This implies, $G\|det(R) \sqsubseteq G\|S$. Combining this with $G\|S \sqsubseteq R$ (since $G\|S \sim R$), we obtain $G\|det(R) \sqsubseteq G\|S \sqsubseteq R$. This proves the necessity.

(*If*) For the sufficiency, choose S' to be $det(R)$. Then S' is deterministic. Since $det(R)$ is controllable, S' is language-controllable (as well as state-controllable). Further, $G\|det(R) \sqsubseteq R$ implies $G\|S' \sqsubseteq R$. For the reverse, since $R \sqsubseteq G$ and since $R \sqsubseteq det(R)$, we have $R \sqsubseteq G\|det(R) = G\|S'$. Thus we have shown $G\|S' \sim R$. We define S to be S' with each state of S' augmented with self-loops on all the undefined uncontrollable events of that state. Then S is Σ_u-compatible. Further since S' is state-controllable, $G\|S' \simeq G\|S$ (by Lemma 2). Further since $G\|S' \sim R$, we also have $G\|S \sim R$. ■

Remark 19 The complexity of checking the existence of a similarity enforcing deterministic supervisor using the condition of Theorem 17 is linear in the size of plant and exponential in the size of specification (due to the need for the computation of a deterministic automaton that accepts the same language as that accepted by the specification automaton). Requiring supervisor to be deterministic, makes the problem computationally more expensive. This situation

is similar to control under partial observation for language specification. The deterministic supervisor synthesis problem in that setting is known to be NP-complete Tsitsiklis (1989), but it is shown to be of polynomial complexity when the supervisor is allowed to be nondeterministic Kumar et al. (2005b).

Let us revisit the message transmission system example studied in Section 6.2.

Example 22 Our goal it to find a deterministic Σ_u-compatible supervisor S for the message transmission system such that $G\|S \sim R$. Condition 1 in Theorem 17 is verified in Example 6.7. In Example 6.7 it was also shown that R is SCS, which implies R is LC, which establishes the second condition. So next we check condition 3. $det(R)$ as well as $G\|det(R)$ are drawn in Figure 6.7. Since no state of R can simulate the state x_2q_4 of $G\|det(R)$, it follows that $G\|det(R)$ cannot be simulated by R. Therefore, there does not exist a Σ_u-compatible deterministic supervisor S, such that $G\|S \sim R$. Note that we showed earlier that a nondeterministic supervisor exists for this system.

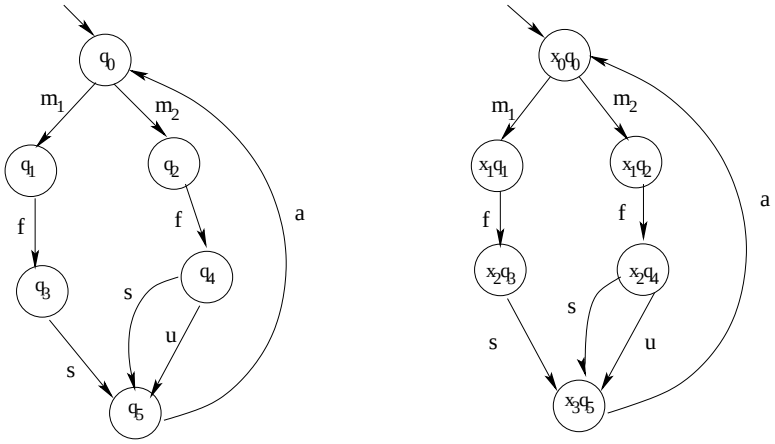

Figure 6.7 $det(R)$ (left) and $G\|det(R)$ (right) for G and R of Figure 6.5

6.4 Conclusion

The chapter studied the problem of supervisory control for enforcing simulation equivalence between the controlled plant and the specification. Through our work we showed that the

simulation equivalence represents a nice compromise between the complexity of control specification vs. its expressiveness. While the bisimilarity enforcing control is the most expressive, the best known complexity for such a control is doubly exponential in the sizes of the plant and the specification. On the other hand, while a language equivalence control is polynomially solvable, it is the least expressive. A simulation equivalence specification is more expressive than a language equivalence specification, yet it remains polynomially solvable. (The complexity turns to be an order higher in the specification size when compared to a language equivalence specification.)

We presented necessary and sufficient conditions for the existence of similarity enforcing supervisors for nondeterministic and also for deterministic plants. Our results are constructive and find a supervisor when one exists. Both the target and range control problems are studied. We also presented a condition for the existence of a similarity enforcing deterministic supervisor.

CHAPTER 7. INTERACTION AND CONTROL VIA PRIORITIZED SYNCHRONOUS COMPOSITION UNDER MASK

In this chapter, we introduce the notion of prioritized synchronous composition under mask (PSCM) to model interaction/control of partially observed discrete event systems, and study the control of DESs via PSCM for both language equivalence and bisimulation equivalence.

In PSCM, each system has its own event priority set. At any given state a system enables all priority events that it can execute at that state, together with all non-priority events. An event is enabled in the composition (i.e., globally enabled) if and only if it is enabled by all interacting systems (i.e., locally enabled by all). A globally enabled event that is executable by one of the systems, can occur in the composed system upon "initiation" by a system that can execute it. Then other systems track it by executing an observation indistinguishable event. If the event is unobservable to or if no observationally indistinguishable events are defined in one of the systems, then that system does not participate in tracking. The transition in the composed system is labeled by the initiating event (and not by the tracking event).

We established a link between PSCM and PSC (and thereby SSC) under certain constraints on the priority sets and the mask functions: PSCM of two systems can be alternatively obtained by first augmenting individual systems, and next computing the PSC of the augmented systems. For this to work, the priority sets of the two systems must exhaust the entire event set, and each event must be observable to a system having priority over that event. These constraints are naturally met in supervisory control where a plant has priority over each event and can also "observe" each event. Note while MPSC is always convertible to PSC (through unmasking of the PSC of masked systems), this is not the case with PSCM, showing its generality over PSC. Similarly, while PSC and MPSC are known to be associative, this is not the case with PSCM

as is shown by an example.

Through the introduction of PSCM we are able to relax the restriction on control that MPSC imposes. We show that when PSCM is adopted as a mechanism of interaction, not only the control & observation-compatibility requirements are removed of a supervisor, the existence condition is given by *achievability* Kumar et al. (2005a) that is weaker than controllability and observability combined. (The weaker condition is required since we allow supervisors to be nondeterministic, whereas the conditions of controllability and observability combined are required for the existence of a deterministic supervisor.) This suggests that the notion of PSCM, presented in the paper, is an appropriate generalization of PSC to account for partial observations. Also, both the existence verification and the synthesis of a PSCM-based supervisor can be performed polynomially: The existence verification is linear in the plant size and quadratic in the specification size, whereas the synthesis is linear in the specification size. The results on PSCM-based control presented in the paper have benefited from the past work of our group Kumar et al. (2005a) that laid the foundation of nondeterministic control and introduced the notion of *achievability* as a condition for existence of a nondeterministic supervisor under partial observation. We also show that when control is exercised via PSCM, the the small model theorem remains valid by showing that a PSCM-based supervisor for enforcing bisimulation equivalence between the specification and the controlled system exists if and only if it a SSC-based supervisor exists over a certain finite state space, namely the power set of Cartesian product of the plant and the specification state spaces.

Earlier we have defined an observation mask to be a function, $M : \overline{\Sigma} \to \overline{\Delta}$. This induces the function $M^{-1}M : \overline{\Sigma} \to 2^{\overline{\Sigma}}$ that maps each $\sigma \in \overline{\Sigma}$ to a set of observationally equivalent events $M^{-1}M(\sigma) \subseteq \overline{\Sigma}$, where such sets form a partition of the set $\overline{\Sigma}$. It is sometimes convenient to represent an observation mask in terms of such a partition it induces. In this chapter, with a slight abuse of notation, we use $M(\sigma)$ to denote the set of events that are observationally equivalent to σ, i.e., the observation mask is now viewed as a map, $M : \overline{\Sigma} \to 2^{\Sigma}$. Since this alternative representation of an observation mask is limited to this last chapter, it will not cause any confusion with the earlier representation. For $\sigma \in \overline{\Sigma}$, we use $M(\sigma) \subseteq \overline{\Sigma}$ to denote

the set of *σ-indistinguishable events.* $\sigma \in \Sigma$ is said to be *unobservable* if $\sigma \in M(\epsilon)$; σ is said to be *completely observable* if $M(\sigma) = \{\sigma\}$. The set of unobservable events is $M(\epsilon)$, and the set of completely observable events is denoted Σ_o.

7.1 Prioritized Synchronization under Mask

In this section, we formalize the notion of prioritized synchronous composition under mask (PSCM) and study its properties. PSCM generalizes the prioritized synchronization of systems to incorporate partial observation. In PSCM, each system possesses an event priority set and an observation mask. The observation mask is not required to be priority-consistent as in MPSC. In Kumar and Heymann (2000b), it was shown that MPSC can alternatively be computed by "unmasking" the PSC of "masked" systems, thereby established a link between MPSC and PSC. In this section, we show that when certain constraints are imposed on the priority sets and observation masks, the PSCM of two systems can alternatively be obtained by first "augmenting" each of the systems, and next computing the PSC of augmented systems. Note that although MPSC can *always* be transformed to PSC via "pre-masking" and "post-unmasking", PSCM can *only* be transformed to PSC under certain restrictions, showing the generality of PSCM over PSC.

In PSCM, an event is "locally" enabled at a certain state of a system if it is executable at that state or is a non-priority event. An event is enabled in the composition (i.e., "globally" enabled) if and only if it is enabled by all interacting systems (i.e., locally enabled by all). A globally enabled event that is executable by one of the systems, can occur in the composed system upon "initiation" by a system that can execute it. Then other systems track it by executing an observation indistinguishable event. If the event is unobservable to or if no observationally indistinguishable events are defined in one of the systems, then that system does not participate in tracking. The transition in the composed system is labeled by the initiating event (and not by the tracking event).

Since any executable event is automatically enabled, and since a system cannot block events outside its priority set (meaning they always remain enabled) or an ϵ-transition, the set

of enabled "events" at a state x_i of G_i is $\Sigma(x_i) \cup A_i^c \cup \{\epsilon\}$, the set of executable events at x_i together with the non-priority events and all ϵ-transitions. We denote this as,

$$\Sigma_e(x_i) := \Sigma(x_i) \cup A_i^c \cup \{\epsilon\}.$$

An event is enabled at a state $(x_1, \cdots, x_n)$ of PSCM composed systems $\{G_i, i \leq n\}$ if and only if it is enabled at state x_i of G_i. In other words, the set of enabled events at state $(x_1, \cdots, x_n)$ of the composition is given by,

$$\Sigma_e((x_1, \cdots, x_n)) := \bigcap_{i=1}^{n} \Sigma_e(x_i)(\bigcap_{i=1}^{n}[\Sigma(x_i) \cup A_i^c]) \cup \{\epsilon\}.$$

Next we formally define the notion of PSCM.

Definition 17 For $i = 1, 2$, consider system $G_i = (X_i, \Sigma, \alpha_i, X_{0i})$, possessing event priority set A_i, and observation mask M_i. Then the *prioritized synchronous composition under mask (PSCM)* of G_1 and G_2 is given by

$$G_1 {}^{M_1}_{A_1}\|^{M_2}_{A_2} G_2 = (X_1 \times X_2, \Sigma, \alpha, X_{01} \times X_{02}),$$

where for $x_1 \in X_1, x_2 \in X_2$ and $\sigma \in \Sigma$,

$$\alpha((x_1,x_2),\sigma) := \begin{cases} \alpha_1(x_1,\sigma)\times\alpha_2(x_2,\sigma'), \text{ if } \begin{cases} \sigma\in\Sigma_e((x_1,x_2)), \\ \alpha_1(x_1,\sigma)\neq\emptyset, \\ \alpha_2(x_2,\sigma')\neq\emptyset, \\ \sigma'\in M_2(\sigma)\neq M_2(\epsilon) \end{cases} \\ \alpha_1(x_1,\sigma')\times\alpha_2(x_2,\sigma), \text{ if } \begin{cases} \sigma\in\Sigma_e((x_1,x_2)), \\ \alpha_1(x_1,\sigma')\neq\emptyset, \\ \alpha_2(x_2,\sigma)\neq\emptyset, \\ \sigma'\in M_1(\sigma)\neq M_1(\epsilon) \end{cases} \\ \alpha_1(x_1,\sigma)\times\{x_2\}, \text{ if } \begin{cases} \sigma\in\Sigma_e((x_1,x_2)), \\ \alpha_1(x_1,\sigma)\neq\emptyset, \\ [\epsilon\in M_2(\sigma)\bigvee\alpha_2(x_2,M_2(\sigma))=\emptyset] \end{cases} \\ \{x_1\}\times\alpha_2(x_2,\sigma), \text{ if } \begin{cases} \sigma\in\Sigma_e((x_1,x_2)), \\ \alpha_2(x_2,\sigma)\neq\emptyset, \\ [\epsilon\in M_1(\sigma)\bigvee\alpha_1(x_1,M_1(\sigma))=\emptyset] \end{cases} \\ \emptyset, \text{ otherwise} \end{cases}$$

$$\alpha((x_1,x_2),\epsilon) := (\alpha_1(x_1,M_1(\epsilon))\times\{x_2\})\cup(\{x_1\}\times\alpha_2(x_2,M_2(\epsilon))).$$

Note in all clauses, the executable event σ is also enabled in the composition ($\sigma\in\Sigma_e((x_1,x_2))$). In the first clause, σ is executable at x_1 ($\alpha_1(x_1,\sigma)\neq\emptyset$). G_1 initiates σ by transitioning to a state in $\alpha_1(x_1,\sigma)$, and G_2 tracks by executing an M_2-indistinguishable event $\sigma'\in M_2(\sigma)\neq M_2(\epsilon)$ that is defined at state x_2 ($\alpha_2(x_2,\sigma')\neq\emptyset$). The second clause is similar to the first clause, except here G_2 initiates σ and G_1 tracks by executing $\sigma'\in M_1(\sigma)\neq M_1(\epsilon)$. In the third clause, σ is executable in G_1 and either it is unobservable to G_2 ($\sigma\in M_2(\epsilon)$) or there is no M_2-indistinguishable event defined at state x_2 ($\alpha_2(x_2,M_2(\sigma))=\emptyset$). So, σ occurs asynchronously in G_1. (Note that G_2 does not block it since σ is enabled by both G_1 and G_2 by virtue of its membership in $\Sigma_e((x_1,x_2))=\Sigma_e(x_1)\cap\Sigma_e(x_2)$.) The fourth clause can be understood in a similar way as the third clause. Finally, an ϵ-transition in the composition corresponds to an asynchronous execution of a label in $M_i(\epsilon)$, $i=1,2$, in which case only G_i participates.

The event priority set of $G_1{}^{M_1}_{A_1}\|^{M_2}_{A_2}G_2$ can be taken to be $A := A_1\cup A_2$, whereas the

class of observationally indistinguishable events for an event σ in $G_1{}_{A_1}^{M_1}\|_{A_2}^{M_2}G_2$ is given by $M(\sigma) := M_1(\sigma) \cap M_2(\sigma)$.

Remark 20 In the special case when both systems have identity mask (Id) functions, the PSCM reduces to the PSC. I.e., $G_1{}_{A_1}^{Id}\|_{A_2}^{Id}G_2 = G_1{}_{A_1}\|_{A_2}G_2$. To see this, when $M_1 = M_2 = Id$, $M_1(\sigma) = M_2(\sigma) = \{\sigma\}$ for all $\sigma \in \overline{\Sigma}$, the transition function of Definition 17 simplifies to:

$$\alpha((x_1,x_2),\sigma) := \begin{cases} \alpha_1(x_1,\sigma) \times \alpha_2(x_2,\sigma), \text{ if } \begin{cases} \sigma \in \Sigma_e((x_1,x_2)), \\ \alpha_1(x_1,\sigma) \neq \emptyset, \\ \alpha_2(x_2,\sigma) \neq \emptyset, \end{cases} \\ \alpha_1(x_1,\sigma) \times \{x_2\}, \text{ if } \begin{cases} \sigma \in \Sigma_e((x_1,x_2)), \\ \alpha_1(x_1,\sigma) \neq \emptyset, \\ \alpha_2(x_2,\sigma) = \emptyset, \end{cases} \\ \{x_1\} \times \alpha_2(x_2,\sigma), \text{ if } \begin{cases} \sigma \in \Sigma_e((x_1,x_2)), \\ \alpha_2(x_2,\sigma) \neq \emptyset, \\ \alpha_1(x_1,\sigma) = \emptyset, \end{cases} \\ \emptyset, \text{ otherwise} \end{cases}$$

$\alpha(x,\epsilon) := (\alpha_1(x_1,\epsilon) \times \{x_2\}) \cup (\{x_1\} \times \alpha_2(x_2,\epsilon))$.

Consider clause 1. Then $\alpha_i(x_i,\sigma) \neq \emptyset \Leftrightarrow \sigma \in \Sigma(x_i) \Leftrightarrow \sigma \in \Sigma(x_1) \cap \Sigma(x_2) \subseteq \Sigma_e((x_1,x_2))$. So the condition of clause 1 is equivalent to $\alpha_i(x_i,\sigma) \neq \emptyset$ as in clause 1 of definition of PSC. Next consider clause 2. Then $\alpha_1(x_1,\sigma) \neq \emptyset$, $\alpha_2(x_2,\sigma) = \emptyset \Rightarrow \alpha \in \Sigma(x_1) - \Sigma(x_2)$. So for $\sigma \in \Sigma_e((x_1,x_2))$ to hold, $\sigma \in A_2^c$. So the condition of clause 2 is equivalent to $\alpha_1(x_1,\sigma) \neq \emptyset$, $\alpha_2(x_2,\sigma) = \emptyset$, $\sigma \notin A_2$ as in clause 2 of definition of PSC. Clause 3 can be analyzed similar to clause 2.

The following example illustrates the concept of PSCM.

Example 23 Consider G_1 and G_2 shown in Figure 7.1, with

$$A_1 = \{a\}, M_1(a) = M_1(b) = \{a,b\}, M_1(c) = \{\epsilon,c\}, M_1(d) = \{d\};$$

$$A_2 = \{b\}, M_2(a) = M_2(d) = \{a,d\}, M_2(b) = \{b\}, M_2(c) = \{c\}.$$

$G_1{}_{A_1}^{M_1}\|_{A_2}^{M_2}G_2$ is drawn in Figure 8, where for simplicity a state (x_1,x_2) of the composition is

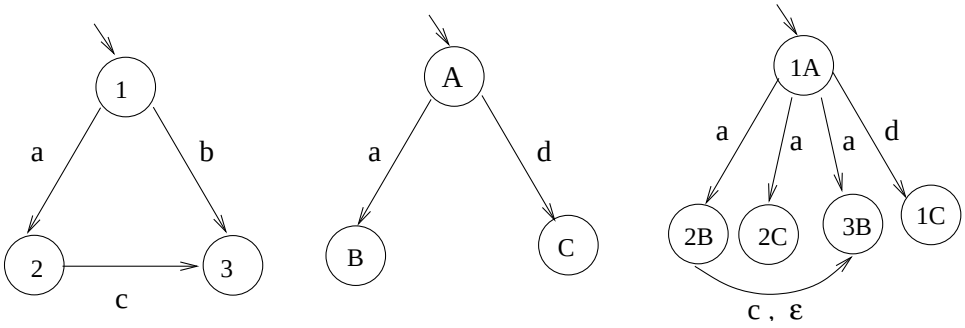

Figure 7.1 G_1 (left), G_2 (middle), and $G_1 {}^{M_1}_{A_1}\|^{M_2}_{A_2} G_2$ (right)

written as x_1x_2. At state $1A$,

$$\Sigma_e(1A) \quad = \quad [\{a,b\} \cup \{b,c,d\}] \cap [\{a,d\} \cup \{a,c,d\}] = \{a,c,d\}.$$

Since for $a \in \Sigma_e(1A)$, $\alpha_1(1,a) = \{2\}$, $a,d \in M_2(a)$, $\alpha_2(A,a) = \{B\}$, and $\alpha_2(A,d) = \{C\}$, by clause 1, we have the transitions $(1A,a,2B)$ and $(1A,a,2C)$ in $G_1 {}^{M_1}_{A_1}\|^{M_2}_{A_2} G_2$. Similarly, for $a \in \Sigma_e(1A)$, $\alpha_2(A,a) = \{B\}$, $a,b \in M_1(a)$, $\alpha_1(1,a) = \{2\}$, and $\alpha_1(1,b) = \{3\}$. By clause 2, we have the transitions $(1A,a,2B)$ and $(1A,a,3B)$ in $G_1 {}^{M_1}_{A_1}\|^{M_2}_{A_2} G_2$. Similarly, for $d \in \Sigma_e(1A)$, $\alpha_2(A,d) = \{C\}$, $d \in M_1(d)$ and $\alpha_1(1,d) = \emptyset$. By clause 4, we have the transition $(1A,d,1C)$ in $G_1 {}^{M_1}_{A_1}\|^{M_2}_{A_2} G_2$. Note that for $c \in \Sigma_e(1A)$, $\alpha_1(1,c) = \emptyset$ and $\alpha_2(A,c) = \emptyset$. Thus, transition on c at state $1A$ in $G_1 {}^{M_1}_{A_1}\|^{M_2}_{A_2} G_2$ does not exist.

At state $2B$,

$$\Sigma_e(2B) = [\{c\} \cup \{b,c,d\}] \cap [\emptyset \cup \{a,c,d\}] = \{c,d\}.$$

Since for $c \in \Sigma_e(2B)$, $\alpha_1(2,c) = \{3\}$, $c \in M_2(c)$ and $\alpha_2(B,c) = \emptyset$. By clause 3, we have the transition $(2B,c,3B)$ in $G_1 {}^{M_1}_{A_1}\|^{M_2}_{A_2} G_2$. Also, since $c \in M_1(\epsilon)$, by clause 5, transition $(2B,\epsilon,3B)$ is defined in $G_1 {}^{M_1}_{A_1}\|^{M_2}_{A_2} G_2$.

When the priority set of the composition is taken to be the union of the two priority sets, and the event-indistinguishability partition is taken to be the intersection of the two event-indistinguishability partitions, then the property of associativity may not be preserved under PSCM. This is in contrast to the operations of PSC and MPSC (both are known to be associative), showing again the generality of PSCM.

Remark 21 Consider G_1, G_2, and G_3 drawn in Figure 7.2, with

$$A_1 = \{a\}, M_1(a) = M_1(b) = \{a, b\}, M_1(c) = \{c\};$$

$$A_2 = \{b\}, M_2(a) = M_2(c) = \{a, c\}, M_2(b) = \{b\};$$

$$A_3 = \{c\}, M_3(b) = M_3(c) = \{b, c\}, M_3(a) = \{a\}.$$

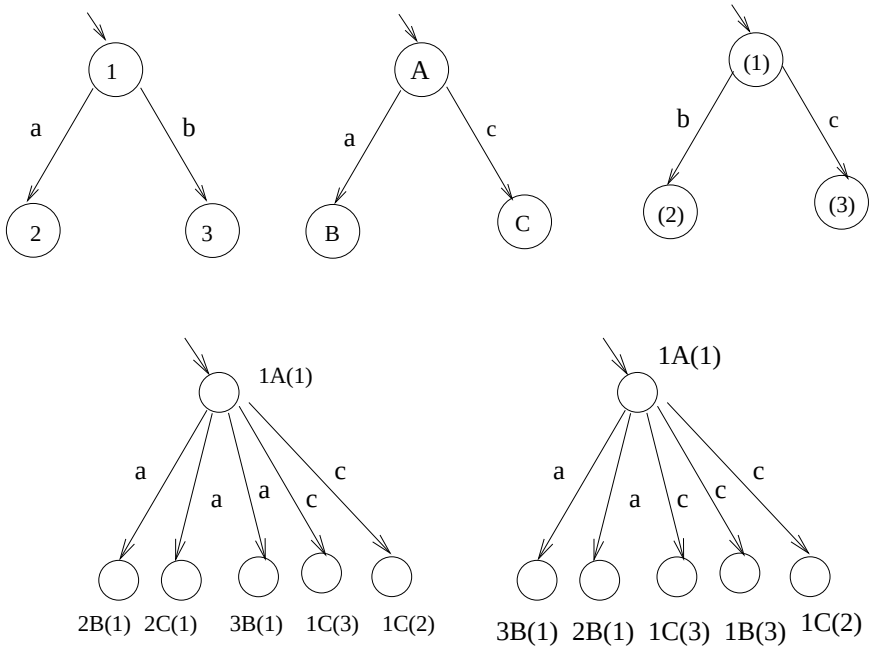

Figure 7.2 G_1 (top left), G_2 (top middle), G_3 (top right), $(G_1 {}^{M_1}_{A_1}\|^{M_2}_{A_2} G_2)$ ${}^{M_1\cap M_2}_{A_1\cup A_2}\|^{M_3}_{A_3}$ G_3 (bottom left), and $G_1 {}^{M_1}_{A_1}\|^{M_2\cap M_3}_{A_2\cup A_3}(G_2 {}^{M_2}_{A_2}\|^{M_3}_{A_3} G_3)$ (bottom right)

The state machines $(G_1 {}^{M_1}_{A_1}\|^{M_2}_{A_2} G_2)^{M_1\cap M_2}_{A_1\cup A_2}\|^{M_3}_{A_3} G_3$ and $G_1 {}^{M_1}_{A_1}\|^{M_2\cap M_3}_{A_2\cup A_3}(G_2 {}^{M_2}_{A_2}\|^{M_3}_{A_3} G_3)$ are also drawn in Figure 7.2 (details are omitted), from which the non-associativity of PSCM is clear.

As discussed below under certain assumptions, PSCM can be converted to PSC, in which case the property of associativity of PSC carries over to that of PSCM.

7.2 Augmentation for Conversion to PSC/SSC

The main feature of PSCM (when compared to PSC) is that execution of an event enabled in the composition by a system can be tracked by another system by synchronously executing an indistinguishable event. We call such synchronous execution M-synchronous executions, or

simply M-synchronizations. One purpose of augmentation is to introduce new transitions that let such M-synchronizations be computed as ordinary synchronizations, where the synchronizing transitions carry the same label. Another purpose is to also capture asynchronous executions also as ordinary synchronous executions by introducing appropriate self-loop transitions in systems that are non-participants, and for unobservable events appropriate ϵ-transitions in the systems where unobservable events are executable. It is clear that augmentation in G_j for an asynchronous execution of G_i will be a self-loop, whereas the augmentation for a transition in G_i that G_j tracks, will be along side the tracking transition, and also augmentation on ϵ-transition in G_j will be along side an unobservable event transition executable in G_j. Care must be taken so that it is always the case that an augmented transition of G_j synchronizes with an existing transition of G_i, i.e., an augmented transition of G_j should not synchronize with an augmented transition of G_i, since such a transition is not possible in the original composition.

Letting $Aug_i(x_i) \subseteq \overline{\Sigma}$ denote the set of labels in $\overline{\Sigma}$ with which state x_i in system G_i can be augmented, then so as they are not stray transitions introduced by augmentation one constraint the set of augmented events should satisfy is that they be locally enabled, i.e.,

$$\begin{aligned} Aug_i(x_i) \cap \Sigma &\subseteq \Sigma_e(x_i) \cap \Sigma = [\Sigma(x_i) \cup A_i^c] \\ &= [\Sigma(x_i) \cap A_i] \cup [\Sigma - A_i] \\ &= [\Sigma(x_i) \cap A_i] \cup [A - A_i] \cup [\Sigma - A], \end{aligned}$$

where $A := \cup_i A_i$ denotes the set of all priority events. So the above provides an upper bound for $Aug_i(x_i)$.

Consider first an event σ in the local priority set A_i. As mentioned earlier, for σ to be a candidate for augmentation at x_i (i.e., $\sigma \in Aug(x_i)$), σ must be locally enabled. Since $\sigma \in A_i$, this means σ must be locally executable (i.e., $\sigma \in \Sigma_i(x_i)$). Now if σ is locally executable, an augmentation on σ is not required if it is completely observable, i.e., if $\sigma \in \Sigma_{oi}$. Thus

$$\text{If } \sigma \in Aug_i(x_i) \cap A_i, \text{then it should hold that } \sigma \in [A_i \cap \Sigma(x_i) - \Sigma_{oi}].$$

Now if $\sigma \in A_i \cap \Sigma(x_i) - \Sigma_{oi}$ so that an augmentation on σ is performed at x_i, then for this

transition to not synchronize with another augmented transition in another system, it must be the case that there exists at least one system where no augmentation on σ is performed. This is guaranteed by having j such that $\sigma \in A_j \cap \Sigma_{oj}$. Thus

$$\text{If } \sigma \in Aug_i(x_i) \cap A_i, \text{ then it should hold that } \exists j, \sigma \in [\Sigma(x_i) - \Sigma_{oi}] \cap [A_j \cap \Sigma_{oj}]. \tag{7.1}$$

Next consider an event σ not in local priority set A_i, but in priority set for some system ($\sigma \in A - A_i$). Then it is locally enabled and so a candidate for augmentation. Also since σ is in priority set of another system G_j, it is guaranteed that σ is globally executable only if it is locally executable in G_j (i.e., $\sigma \in \Sigma(x_j)$). In order to avoid synchronization of augmented transition in one system by augmented transition in another system, it must be the case that no augmentation is performed on the same event in at least one system. This is guaranteed by further having $\sigma \in \Sigma_{oj}$. Thus

$$\text{If } \sigma \in Aug_i(x_i) \cap [A - A_i], \text{ then it should hold that } \exists j, \sigma \in [A_j \cap \Sigma_{oj}]. \tag{7.2}$$

Finally consider an event σ in priority set of none of the systems ($\sigma \in \Sigma - A$). Then σ is locally enabled in every system, and so a candidate for augmentation in each system. However, it is possible that σ is not locally executable in any one of them, i.e., it is possible that $\sigma \in \Sigma_e((x_1, \cdots, x_n)) - \cup_i \Sigma(x_i)$. Then augmentation on σ will allow synchronization of transition that are augmented transitions in all systems. This suggests that augmentation cannot be performed on events in $\Sigma - A$. In other words,

$$\sigma \in Aug(x_i) \cap [\Sigma - A] \text{ should be impossible.} \tag{7.3}$$

Combining 7.1, 7.2, and 7.3 it can be obtain that,

$$\text{If } \sigma \in Aug_i(x_i) \cap \Sigma, \text{ then it should hold that } \exists j, \sigma \in [(A_i \cap \Sigma(x_i) - \Sigma_{oi}) \cup (A - A_i)] \cap [A_j \cap \Sigma_{oj}]. \tag{7.4}$$

In other words, in order to perform augmentation on $\sigma \in \Sigma$ at state x_i of G_i, either (i) σ is a priority event of G_i, is executable at x_i, and is not completely observable by G_i, or (ii) σ is a non-priority event of G_i but a priority event of some system. In either case there must exist a

system for which σ is a priority event and is completely observable. Allowing for the possibility of augmentation on any priority event, we make the following requirement assumption.

Assumption 1 For $i \leq n$, consider NSM G_i possessing event priority set A_i and observation mask M_i. Suppose $\forall \sigma \in A = \cup_i A_i$, $\exists j$ such that $\sigma \in A_j \cap \Sigma_{oj}$.

Under Assumption 1, all events in $A = \cup_i A_i$ are candidates for augmentation. Moreover the label ϵ can be used for augmentation whenever an unobservable event is locally defined.

Algorithm 6 For $i \leq n$, consider NSM G_i possessing event priority set A_i and observation mask M_i. The following algorithm computes augmented G_i, $G_i^{Aug_i} := (X_i, \Sigma, \alpha_i^{Aug}, X_{0i})$.

1. For each state x_i of G_i,

$$Aug_i(x_i) := \begin{cases} [A_i \cap \Sigma(x_i) - \Sigma_{oi}] \cup [A - A_i] \cup \{\epsilon\} & \text{if } \Sigma(x_i) \cap A \cap M_i(\epsilon) \neq \emptyset \\ [A_i \cap \Sigma(x_i) - \Sigma_{oi}] \cup [A - A_i] & \text{otherwise} \end{cases}$$

2. For $\sigma \in Aug_i(x_i) - M_i(\epsilon)$, if $\alpha_i(x_i, M_i(\sigma)) \neq \emptyset$, add transitions on σ from x_i to all states in the set $\alpha_i(x_i, M_i(\sigma))$, otherwise add self-loop on σ at x_i;

3. For $\sigma \in Aug_i(x_i) \cap M_i(\epsilon)$, add self-loop on σ at x_i. Further if $\alpha_i(x_i, \sigma) \neq \emptyset$, add transitions on ϵ from x_i to all states in this set.

In other words, for augmentation on observable σ, we add σ-transitions along side all existing transitions on σ-indistinguishable events, and if none such transitions exist, we simply add a self-loop on σ. On the other hand (when σ is unobservable), we augment by adding a self-loop on σ, together with ϵ-transitions along side any existing σ-transitions.

The following example illustrates Algorithm 6.

Example 24 Consider G_1 and G_2 shown in Figure 7.3, with

$$A_1 = \{d, e\}, M_1(a) = M_1(b) = \{a, b\}, M_1(c) = \{c\}, M_1(d) = \{d\}, M_1(e) = \{e\};$$
$$A_2 = \{a, c\}, M_2(a) = \{a\}, M_2(c) = \{c\}, M_2(b) = M_2(d) = M_2(e) = \{\epsilon, b, d, e\}.$$

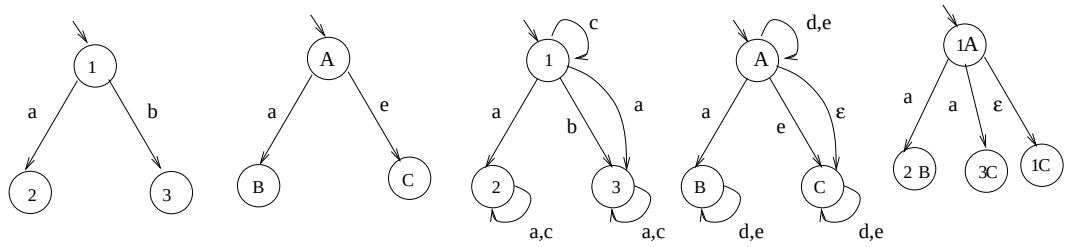

Figure 7.3 G_1 (first), G_2 (second), $G_1^{Aug_1}$ (third), $G_2^{Aug_2}$ (fourth), and $G_1 {}^{M_1}_{A_1}\|^{M_2}_{A_2} G_2$ (fifth)

Then $A = A_1 \cup A_2 = \{a, c, d, e\}$, $\Sigma_{o1} = \{c, d, e\}$ and $\Sigma_{o2} = \{a, c\}$. For state 1 in G_1, since

$$\Sigma(1) \cap A \cap M_1(\epsilon) = \{a, b\} \cap \{a, c, d, e\} \cap \emptyset = \emptyset,$$

$$\begin{aligned} Aug_1(1) &= [A_1 \cap \Sigma(1) - \Sigma_{o1}] \cup [A - A_1] \\ &= [\{d, e\} \cap \{a, b\} - \{c, d, e\}] \cup [\{a, c, d, e\} - \{d, e\}] \\ &= \{a, c\}. \end{aligned}$$

For $a \in Aug_1(1)$, since $a \notin M_1(\epsilon)$ and $\alpha_1(1, M_1(a)) = \alpha_1(1, \{a, b\}) = \{2, 3\}$, by step 2 of Algorithm 6, transition $(1, a, 3)$ is added in G_1 for augmentation. For $c \in Aug_1(1)$, since $c \notin M_1(\epsilon)$ and $\alpha_1(1, M_1(c)) = \emptyset$, by step 2 of Algorithm 6, transition $(1, c, 1)$ is added in G_1 for augmentation. Similarly, one can compute $Aug_1(2) = \{a, c\} = Aug_1(3)$. The state machine $G_1^{Aug_1}$ is drawn in Figure 7.3.

For state A in G_2, since

$$\Sigma(A) \cap A \cap M_2(\epsilon) = \{a, e\} \cap \{a, c, d, e\} \cap \{\epsilon, b, d, e\} = \{e\} \neq \emptyset,$$

$$\begin{aligned} Aug_2(A) &= [A_2 \cap \Sigma(A) - \Sigma_{o2}] \cup [A - A_2] \cup \{\epsilon\} \\ &= [\{a, c\} \cap \{a, e\} - \{a, c\}] \cup [\{a, c, d, e\} - \{a, c\}] \cup \{\epsilon\} \\ &= \{d, e, \epsilon\}. \end{aligned}$$

For $d \in Aug_2(A)$, since $d \in M_2(\epsilon)$ and $\alpha_2(A, d) = \emptyset$, by step 3 of Algorithm 6, transition (A, d, A) is added in G_2 for augmentation. For $e \in Aug_2(A)$, since $e \in M_2(\epsilon)$ and

$\alpha_2(A, e) = \{C\}$, by step 3 of Algorithm 6, transitions (A, e, A) and (A, ϵ, C) are added in G_2 for augmentation. Similarly, one can compute $Aug_2(B) = \{d, e\} = Aug_1(C)$. The state machine $G_2^{Aug_2}$ is drawn in Figure 7.3. Finally, the state machine $G_1 {}^{M_1}_{A_1}\|^{M_2}_{A_2} G_2$ is also drawn in Figure 7.3.

Since we do not augment with events in $[\Sigma - A]$, through the augmentation we are able to simulate only those asynchronous or M-synchronous executions as synchronous executions that occur on events outside $[\Sigma - A]$, i.e., on events in A. So after the augmentation, the priority sets of both the systems can only be enlarged to the set A, where all events will occur synchronously. This is summarized in the following theorem.

Theorem 18 Under the Assumption 1, $G_1 {}^{M_1}_{A_1}\|^{M_2}_{A_2} G_2 = G_1^{Aug_1} {}^{M_1}_{A}\|^{M_2}_{A} G_2^{Aug_2}$.

Proof: Since the state sets, the event sets, and the initial states of the two NSM's are all identical, we only need to show that they also have the identical set of transitions. For notational convenience, the set of events defined (resp., enabled) at a state x_i of $G_i^{Aug_i}$ is denoted by $\Sigma_{Aug}(x_i)$ (resp., $\Sigma_{Aug,e}(x_i)$). Since the priority set of $G_i^{Aug_i}$ is A, it follows that $\Sigma_{Aug,e}(x_i) = \Sigma_{Aug}(x_i) \cup A^c \cup \{\epsilon\}$.

By Algorithm 6, events defined at x_i of $G_i^{Aug_i}$ are given by,

$$
\begin{aligned}
& \Sigma_{Aug}(x_i) \\
= \;& \Sigma(x_i) \cup Aug_i(x_i) \\
= \;& \begin{cases} \Sigma(x_i) \cup [A_i \cap \Sigma(x_i) - \Sigma_{oi}] \cup [A - A_i] \cup \{\epsilon\} & \text{if } \Sigma(x_i) \cap A \cap M_i(\epsilon) \neq \emptyset \\ \Sigma(x_i) \cup [A_i \cap \Sigma(x_i) - \Sigma_{oi}] \cup [A - A_i] & \text{otherwise} \end{cases} \\
\overset{(a)}{=} \;& \begin{cases} \Sigma(x_i) \cup [A - A_i] \cup \{\epsilon\} & \text{if } \Sigma(x_i) \cap A \cap M_i(\epsilon) \neq \emptyset \\ \Sigma(x_i) \cup [A - A_i] & \text{otherwise} \end{cases}
\end{aligned}
$$

Equality (a) holds since $A_i \cap \Sigma(x_i) \subseteq \Sigma(x_i)$, which implies $A_i \cup \Sigma(x_i) - \Sigma_{oi} \subseteq \Sigma(x_i)$. So, we have

$$
\begin{aligned}
& \Sigma_{Aug,e}((x_1, x_2)) \\
= \;& \Sigma_{Aug,e}(x_1) \cap \Sigma_{Aug,e}(x_2)
\end{aligned}
$$

$$
\begin{aligned}
&= ([\Sigma_{Aug}(x_1) \cup A^c] \cup \{\epsilon\}) \cap ([\Sigma_{Aug}(x_2) \cup A^c] \cup \{\epsilon\}) \\
&= ([\Sigma(x_1) \cup [A - A_1] \cup A^c] \cap [\Sigma(x_2) \cup [A - A_2] \cup A^c]) \cup \{\epsilon\} \\
&\overset{(b)}{=} ([\Sigma(x_1) \cup A_1^c] \cap [\Sigma(x_2) \cup A_2^c]) \cup \{\epsilon\} \\
&= \Sigma_e((x_1, x_2))
\end{aligned}
$$

Equality (b) holds since

$$[A - A_i] \cup A^c = [A - A_i] \cup [\Sigma - A] = \Sigma - A_i = A_i^c.$$

It follows that the set of enabled events at (x_1, x_2) in $G_1 {}^{M_1}_{A_1}\|^{M_2}_{A_2} G_2$ is the same as the set of enabled events at (x_1, x_2) in $G_1^{Aug_1} {}^{M_1}_{A}\|^{M_2}_{A} G_2^{Aug_2}$. Since G_i is a subautomaton of $G_i^{Aug_i}$, it follows that each transition of the left NSM is also a transition of the right NSM.

Next we prove the converse that each transition of the right NSM is also a transition of the left NSM. Since we do not augment an event outside A, it holds that a transition of the right NSM on an event outside A is also a transition of the left NSM. Consider next a transition on an event $\sigma \in A$. Then without loss of generality, by Assumption 1, we can assume $\sigma \in A_2 \cap \Sigma_{o2}$. Then it suffices to prove that a transition on an event $\sigma \in A_2 \cap \Sigma_{o2}$ of the right NSM is also a transition of the left NSM.

By Definition 17, for a transition on σ to occur in the composed system, σ must be enabled by both systems, i.e., $\sigma \in \Sigma_{Aug,e}((x_1, x_2))$. Then

$$
\begin{aligned}
& \sigma \in [A_2 \cap \Sigma_{o2}] \cap \Sigma_{Aug,e}((x_1, x_2)) \\
\Rightarrow\ & \sigma \in [A_2 \cap \Sigma_{o2}] \cap [([\Sigma(x_1) \cup A_1^c] \cap [\Sigma(x_2) \cup A_2^c]) \cup \{\epsilon\}] \\
\Rightarrow\ & \sigma \in [\Sigma(x_1) \cup A_1^c] \cap [\Sigma(x_2) \cap A_2 \cap \Sigma_{o2}].
\end{aligned}
$$

Since $\sigma \in \Sigma(x_2)$, exists a transition (x_2, σ, x_2') in G_2. Also since

$$
\begin{aligned}
\sigma &\in \Sigma(x_1) \cup A_1^c \\
&= [\Sigma(x_1) \cap \Sigma_{o1}] \cup [\Sigma(x_1) - \Sigma_{o1}] \cup A_1^c \\
&= [\Sigma(x_1) \cap \Sigma_{o1}] \cup [([\Sigma(x_1) - \Sigma_{o1}] \cup A_1^c) \cap M_1(\epsilon)] \cup [([\Sigma(x_1) - \Sigma_{o1}] \cup A_1^c) - M_1(\epsilon)],
\end{aligned}
$$

the following cases exist.

1. $\sigma \in \Sigma(x_1) \cap \Sigma_{o1}$.

   In this case, since σ is observable and already defined at x_1, there is no augmentation on σ at x_1 in G_1. Since there is no augmentation on σ in G_2 at any of its states, including x_2, there is no newly introduced transition on σ in the right NSM.

2. $\sigma \in ([\Sigma(x_1) - \Sigma_{o1}] \cup A_1^c) - M_1(\epsilon)$.

   (a) If $\alpha_1(x_1, M_1(\sigma)) \neq \emptyset$, then by step 2 of Algorithm 6, for $x_1' \in \alpha_1(x_1, \sigma_1)$ with $\sigma_1 \in M(\sigma)$,

   $$\text{if transition } (x_1, \sigma_1, x_1') \text{ defined in } G_1,$$
   $$\text{then transitions } (x_1, \sigma_1, x_1'), (x_1, \sigma, x_1') \text{ defined in } G_1^{Aug_1}.$$

   Then exists synchronous transition $((x_1, x_2), \sigma, (x_1', x_2'))$ in $G_1^{Aug_1}{}_A^{M_1} \|_A^{M_2} G_2^{Aug_2}$. It also holds that by clause 1 of Definition 17, the transition $((x_1, x_2), \sigma, (x_1', x_2'))$ is in ${G_1}_{A_1}^{M_1} \|_{A_2}^{M_2} G_2$.

   (b) If $\alpha_1(x_1, M_1(\sigma)) = \emptyset$, then by step 2 of Algorithm 6,

   $$\text{transition } (x_1, \sigma, x_1) \text{ is defined in } G_1^{Aug_1}.$$

   Then exists synchronous transition $((x_1, x_2), \sigma, (x_1, x_2'))$ in $G_1^{Aug_1}{}_A^{M_1} \|_A^{M_2} G_2^{Aug_2}$. It also holds that this transition is in ${G_1}_{A_1}^{M_1} \|_{A_2}^{M_2} G_2$ by clause 4 of Definition 17.

3. $\sigma \in ([\Sigma(x_1) - \Sigma_{o1}] \cup A_1^c) \cap M_1(\epsilon)$.

   (a) If $\sigma \in \Sigma(x_1)$, then $\sigma \in \Sigma(x_1) \cap A \cap M_1(\epsilon)$, and by step 1 of Algorithm 6, $\epsilon \in Aug_1(x_1)$. By step 3 of Algorithm 6,

   $$\text{if transition } (x_1, \sigma, x_1') \text{ defined in } G_1,$$
   $$\text{then transitions } (x_1, \sigma, x_1'), (x_1, \epsilon, x_1'), (x_1, \sigma, x_1) \text{ defined in } G_1^{Aug_1}.$$

   Then exist synchronous transitions $((x_1, x_2), \sigma, (x_1, x_2'))$ and $((x_1, x_2), \sigma, (x_1', x_2'))$, and asynchronous transitions $((x_1, x_2'), \epsilon, (x_1', x_2'))$ and $((x_1, x_2), \epsilon, (x_1', x_2))$ in $G_1^{Aug_1}{}_A^{M_1} \|_A^{M_2} G_2^{Aug_2}$. It also holds that by clause 4 of Definition 17, the transition $((x_1, x_2), \sigma, (x_1, x_2'))$ is in ${G_1}_{A_1}^{M_1} \|_{A_2}^{M_2} G_2$, whereas by clause 2 of Definition 17, the transition $((x_1, x_2), \sigma, (x_1', x_2'))$

is in $G_1 {}^{M_1}_{A_1}\|^{M_2}_{A_2} G_2$, and finally, by clause 5 of Definition 17, transitions $((x_1, x_2'), \epsilon, (x_1', x_2'))$ and $((x_1, x_2), \epsilon, (x_1', x_2))$ are in $G_1 {}^{M_1}_{A_1}\|^{M_2}_{A_2} G_2$.

(b) If $\sigma \notin \Sigma(x_1)$, then by step 3 of Algorithm 6,

$$(x_1, \sigma, x_1) \text{ is defined in } G_1^{Aug_1}.$$

Then exists synchronous transition $((x_1, x_2), \sigma, (x_1, x_2'))$ in $G_1^{Aug_1} {}^{M_1}_{A}\|^{M_2}_{A} G_2^{Aug_2}$. It also holds that this transition is in $G_1 {}^{M_1}_{A_1}\|^{M_2}_{A_2} G_2$ by clause 4 of Definition 17.

This completes the proof. ■

In the special case, when the event priority sets exhaust the entire event set, i.e., when $A = \Sigma$, all M-synchronous executions can be simulated as ordinary synchronous executions. Then no M-synchronizations are needed and so all masks can be treated as identity mask. We state this formally below.

Assumption 2 For $i \leq n$, consider NSM G_i possessing event priority set A_i such that $A = \cup_i A_i = \Sigma$.

Theorem 19 Under the Assumptions 1 and 2,

$$G_1 {}^{M_1}_{A_1}\|^{M_2}_{A_2} G_2 = G_1^{Aug_1} {}^{M_1}_{\Sigma}\|^{M_2}_{\Sigma} G_2^{Aug_2} = G_1^{Aug_1} {}^{Id}_{\Sigma}\|^{Id}_{\Sigma} G_2^{Aug_2}.$$

Proof: By Theorem 18, we have $G_1 {}^{M_1}_{A_1}\|^{M_2}_{A_2} G_2 = G_1^{Aug_1} {}^{M_1}_{\Sigma}\|^{M_2}_{\Sigma} G_2^{Aug_2}$. Thus, it suffices to prove that

$$G_1^{Aug_1} {}^{M_1}_{\Sigma}\|^{M_2}_{\Sigma} G_2^{Aug_2} = G_1^{Aug_1} {}^{Id}_{\Sigma}\|^{Id}_{\Sigma} G_2^{Aug_2}. \tag{7.5}$$

Since the state sets, the event sets, and the initial states of the two NSM's are all identical, we only need to show that they also have the identical set of transitions. Note that since the mask functions of the right NSM of (7.5) are identity masks, no M-synchronizations are allowed in the right NSM of (7.5). Consequently, the right NSM of (7.5) is a subautomaton of the left NSM of (7.5). Hence, each transition of the right NSM of (7.5) is also a transition of the left NSM of (7.5). It remains to prove the converse that each transition of the left NSM

of (7.5) is also a transition of the right NSM of (7.5). The analysis carried out in proof of Theorem 18 is also valid here since Theorem 18 applies under one less assumption.

We consider transitions on an event $\sigma \in A$ in left NSM. Then as in proof of Theorem 18, we can assume $\sigma \in A_2 \cap \Sigma_{o2}$. Then as in the proof of Theorem 18, we again have the cases 1, 2(a), 2(b), 3(a), and 3(b). There is no new transition introduced in case 1. In case 2(a), each M-synchronization by a transition on σ_1 in $G_1^{Aug_1}{}_{\Sigma}^{M_1}\|_{\Sigma}^{M_2}G_2^{Aug_2}$ is duplicated by an ordinary synchronization on σ in $G_1^{Aug_1}{}_{\Sigma}^{M_1}\|_{\Sigma}^{M_2}G_2^{Aug_2}$. In cases 2(b), 3(a), and 3(b), each asynchronous execution on σ in $G_1^{Aug_1}{}_{\Sigma}^{M_1}\|_{\Sigma}^{M_2}G_2^{Aug_2}$ is duplicated by an ordinary synchronous execution on σ in $G_1^{Aug_1}{}_{\Sigma}^{M_1}\|_{\Sigma}^{M_2}G_2^{Aug_2}$, whereas each asynchronous execution on ϵ in $G_1^{Aug_1}{}_{\Sigma}^{M_1}\|_{\Sigma}^{M_2}G_2^{Aug_2}$ is duplicated by an appropriate ϵ-transition in $G_1^{Aug_1}{}_{\Sigma}^{M_1}\|_{\Sigma}^{M_2}G_2^{Aug_2}$. Thus, by replacing non-identity masks M_1 and M_2 by the identity mask, only certain duplicate transitions on $\sigma \in A$ are avoided, but no σ-transition of the left NSM is removed in the right NSM.

Finally, due to Assumption 2, $A = \Sigma$, and so the above statements are true of all transitions on all events $\sigma \in \Sigma$. This completes the proof. ■

We showed that under Assumptions 1 and 2, PSCM of two systems having non-identity masks can be computed first by augmenting each of the systems, then computing PSCM of the augmented systems possessing identity masks. Further, by Remark 20, when two systems interact through identity masks, their PSCM reduces to simply their PSC. Also note that when the event priority set of each system is the entire event set Σ, PSC reduces to SSC. This leads to the following corollary.

Corollary 4 Under the Assumptions 1 and 2,

$$G_1{}_{A_1}^{M_1}\|_{A_2}^{M_2}G_2 = G_1^{Aug_1}{}_{\Sigma}^{M_1}\|_{\Sigma}^{M_2}G_2^{Aug_2} = G_1^{Aug_1}{}_{\Sigma}^{Id}\|_{\Sigma}^{Id}G_2^{Aug_2} = G_1^{Aug_1}{}_{\Sigma}\|_{\Sigma}G_2^{Aug_2} = G_1^{Aug_1}\|G_2^{Aug_2}.$$

The result in the above corollary can be read as follows. Under Assumptions 1 and 2,

$$PSCM((G_1, A_1, M_1), (G_2, A_2, M_2)) = PSC((G_1^{Aug_1}, A), (G_2^{Aug_2}, A)) = SSC(G_1^{Aug_1}, G_2^{Aug_2}).$$

Note that Assumptions 1 and 2 automatically hold when one of systems can observe every event completely (has identity mask) and has priority over every event (priority set = Σ).

This yields the following corollary, which is useful in supervisory control setting, as a plant can "observe" every event and has priority over every event.

Corollary 5 $G_1{}^{Id}_{\Sigma}\|^{M}_{A_2}G_2 = G_1^{Aug_1}{}_{\Sigma}\|_{\Sigma}G_2^{Aug_2} = G_{1\Sigma}\|_{\Sigma}G_2^{Aug_2} = G_1\|G_2^{Aug_2}$.

Proof: Since priority set of G_1 is Σ, and its observation mask is Id, it follows that $A_1 = \Sigma$ and $\Sigma_{o1} = \Sigma$. Thus $A_1 \cap \Sigma_{o1} = \Sigma$, which implies that $\sigma \in A_1 \cap \Sigma_{o1}$ for any $\sigma \in \Sigma$. Thus, Assumption 1 holds. Also since $A_1 \cup A_2 = \Sigma \cup A_2 = \Sigma$, Assumption 2 also holds. Then the first equality in assertion of the corollary follows from Corollary 4.

To show the second equality of the assertion, it suffices to show that $G_1^{Aug_1} = G_1$. Since each event is observable in G_1, $M_1(\epsilon) = \emptyset$. So for each state x_1 in G_1, $\Sigma(x_1) \cap M_1(\epsilon) = \emptyset$. Thus

$$\begin{aligned} Aug_1(x_1) &= [A - A_1] \cup [A_1 \cap \Sigma(x_1) - \Sigma_{o1}] \\ &= [\Sigma - \Sigma] \cup [\Sigma \cap \Sigma(x_1) - \Sigma] \\ &= \emptyset. \end{aligned}$$

Since $Aug_1(x_1) = \emptyset$ for any state x_1 in G_1, no augmentation takes place in G_1, i.e., $G_1^{Aug_1} = G_1$. Finally the last equality in the assertion of the corollary follows from the equivalence of PSC and SSC when each priority set is the entire event set Σ. ■

Note that no assumptions are needed in Corollary 5.

7.3 Control for Language Equivalence via PSCM

In this section, we extend the theory of supervisory control under partial observation to the present setting where control is exercised by means of interaction via PSCM (under the restriction that the event priority set of the supervisor is a subset of the event priority set of the plant, i.e., there are no "driven" events). The plant is modeled by a NSM $G = (X, \Sigma, \alpha, X_0)$ having event priority set Σ and identity observation mask. The supervisor is modeled as another NSM $S = (Y, \Sigma, \beta, Y_0)$ having event priority set $\Sigma_c = \Sigma - \Sigma_u$, the set of controllable events, and an observation mask M. We let Σ_o denote the set of completely observable events of a supervisor. The control specification is generated by a state machine $R = (Q, \Sigma, \delta, Q_0)$.

The control task is to design a supervisor S such that the behavior of the controlled plant equals the specification language, i.e., $L(G_{\Sigma}^{Id} \| _{\Sigma_c}^{M} S) = L(R)$. We show that both the existence and synthesis of a supervisor can be determined polynomially.

In Kumar et al. (2005a), SSC based control under partial observation was studied, allowing the supervisor to be nondeterministic. Due to the use of SSC, supervisor was required to be (Σ_u, M)-compatible. It was shown that a necessary and sufficient condition for the existence of a (Σ_u, M)-compatible supervisor such that the behavior of the controlled system equals the specification language is *achievability*.

Definition 18 Kumar et al. (2005a)

1. $K \subseteq \Sigma^*$ is said to be Σ_u-*controllable* with respect to Σ^* if $\forall s \in pr(K)$ and $\forall a \in \Sigma_u$, $sa \in pr(K)$.

2. $K \subseteq \Sigma^*$ is said to be M-*recognizable* with respect to Σ^* if $\forall s, t \in \Sigma^*$ and $\forall a \in \Sigma$ with $M(a) = M(\epsilon)$, $sat \in pr(K) \Rightarrow sa^*t \subseteq pr(K)$.

3. $K \subseteq \Sigma^*$ is said to be (Σ_u, M)-*achievable* with respect to Σ^* ((Σ^*, Σ_u, M)-achievable for short) if K is Σ_u-controllable and M-recognizable with respect to Σ^*, and $\forall s, t \in \Sigma^*$, $\forall a \in \overline{\Sigma}$, $b \in \Sigma_u$ with $M(a) = M(b)$, $sat \in pr(K) \Rightarrow \{sbt\} \subseteq pr(K)$.

4. $K \subseteq L(G)$ is said to be $(L(G), \Sigma_u, M)$-*achievable* if exists (Σ^*, Σ_u, M)-achievable K' such that $pr(K) = pr(K') \cap L(G)$.

It was shown in Kumar et al. (2005a) that $(L(G), \Sigma_u, M)$-achievability of prefix-closed language is preserved under intersection and so the infimal $(L(G), \Sigma_u, M)$-achievable superlanguage of a language $K \subseteq L(G)$, denoted $inf\overline{PA}_{L(G)}(K)$, exists. Further it holds that,

$$inf\overline{PA}_{L(G)}(K) = inf\overline{PA}_{\Sigma^*}(K) \cap L(G),$$

and so $inf\overline{PA}_{(L(G)}(K)$ can be computed by first computing $inf\overline{PA}_{\Sigma^*}(K)$. The following algorithm was given in (Kumar et al., 2005a, Algorithm 1) to compute a generator $\hat{R}$ of the language $inf\overline{PA}_{\Sigma^*}(K)$.

Algorithm 7 Kumar et al. (2005a) Let $R = (Q, \Sigma, \delta, Q_0)$ be a state machine with $L(R) = pr(K)$.

A1 For each transition (q, b, q_1) with either $b \in M(\epsilon)$ or $\exists (q, b', q_2)$ such that $M(b) = M(b') \neq M(\epsilon)$, replace (q, b, q_1) by a pair of transitions (q, ϵ, q_1') and (q_1', b, q_1), where q_1' is a newly added state.

A2 For every transition (q, b, q') with $b \in M(\epsilon)$, add transitions (q, b, q) and (q, ϵ, q').

A3 For every state q and every event $b \in \Sigma_u \cap M(\epsilon)$, if b is not defined at q then let $\delta(q, b) = \delta(q, \epsilon)$.

A4 For every state q, every event $b \in \Sigma_u - M(\epsilon)$, and every transition (q, a, q') with $M(a) = M(b)$, add a transition (q, b, q').

A5 (i) For every state q and every event $b \in \Sigma_u - M(\epsilon)$, if no such an event a with $M(a) = M(b)$ is defied at q then add a transition $(q, b, dump)$, where $dump$ is an added state, (ii) $\forall \sigma \in \Sigma_u$, $\delta(dump, \sigma) = \{dump\}$.

Let the state machine constructed by Algorithm 7 be denoted $\hat{R}$. Then from (Kumar et al., 2005a, Theorem 6), $L(\hat{R}) = inf\overline{PA}_{\Sigma^*}(L(R))$.

Next we aim to derive a necessary condition for the existence of a supervisor such that $L(G_{\Sigma}^{Id} \|_{\Sigma_c}^{M} S) = L(R)$. Since G possesses priority set Σ and identity mask, from Corollary 5, $G_{\Sigma}^{Id} \|_{\Sigma_c}^{M} S = G_{\Sigma} \|_{\Sigma} S^{Aug}$. Let the augmented supervisor be given by, $S^{Aug} = (Y, \Sigma, \beta^{Aug}, Y_0)$. We first observe that $L(S^{Aug})$ is (Σ^*, Σ_u, M)-achievable and the PSCM-based controlled behavior $L(G_{\Sigma}^{Id} \|_{\Sigma_c}^{M} S)$ is $(L(G), \Sigma_u, M)$-achievable.

Lemma 17 Consider plant G with priority set Σ and identity observation mask, a supervisor S with priority set Σ_c and observation mask M. Then

1. $L(S^{Aug})$ is (Σ^*, Σ_u, M)-achievable.
2. $L(G_{\Sigma}^{Id} \|_{\Sigma_c}^{M} S)$ is $(L(G), \Sigma_u, M)$-achievable.

Proof:

1. For $y \in Y$, let $Aug(y)$ be the set of labels used for augmenting transitions in S to obtain S^{Aug}. Then from Algorithm 6,

$$\forall y \in Y, \quad Aug(y) := \begin{cases} [\Sigma_c \cap \Sigma(y) - \Sigma_o] \cup \Sigma_u \cup \{\epsilon\} & \text{if } \Sigma(y) \cap M(\epsilon) \neq \emptyset \\ [\Sigma_c \cap \Sigma(y) - \Sigma_o] \cup \Sigma_u & \text{otherwise} \end{cases} \tag{7.6}$$

Note that the set of events defined at state y of S^{Aug} is given by, $\Sigma(y) \cup Aug(y)$.

We first show that $L(S^{Aug})$ is Σ_u-controllable with respect to Σ^*. Pick $s \in L(S^{Aug})$, $a \in \Sigma_u$. Let $y \in Y$ be a state reached by s in S^{Aug}. Then since $\Sigma_u \subseteq Aug(y)$, it follows that $sa \in L(S^{Aug})$.

Next we show that $L(S^{Aug})$ is M-recognizable with respect to Σ^*. Pick $s, t \in \Sigma^*$, $a \in M(\epsilon)$ such that $sat \in L(S^{Aug})$. Let y_s, y_{sa} be states such that y_s is reached by executing s at Y_0 in S^{Aug}, y_{sa} is reached by executing a at y_s in S^{Aug}, and t is defined at y_{sa}. Then $a \in \Sigma(y_s) \cup Aug(y_s)$. We claim that $a \in Aug(y_s)$ even when $a \in \Sigma(y_s)$. Clearly this is the case when $a \in \Sigma_u$ since $\Sigma_u \subseteq Aug(y_s)$. On the other hand, if $a \in \Sigma_c$, then it also holds that $a \in \Sigma_c - \Sigma_o$ ($a \in M(\epsilon)$ implies $a \notin \Sigma_o$). This together with $a \in \Sigma(y_s)$ implies that $a \in \Sigma(y_s) \cap \Sigma_c - \Sigma_0 \subseteq Aug(y_s)$. Since $a \in Aug(y_s) \cap M(\epsilon)$, from step 3 of Algorithm 6, transitions (y_s, a, y_s) and (y_s, ϵ, y_{sa}) are added in S for augmentation. It follows that $sa^*t \subseteq L(S^{Aug})$.

Finally, we show that $L(S^{Aug})$ is (Σ_u, M)-achievable with respect to Σ^*. Pick $s, t \in \Sigma^*$, $a \in \overline{\Sigma}$, $b \in \Sigma_u$ with $M(a) = M(b)$ such that $sat \in L(S^{Aug})$. Then as before, $a \in Aug(y_s)$. Since $b \in \Sigma_u$, and $\Sigma_u \subseteq Aug(y_s)$, $b \in Aug(y_s)$. If $b \in M(a) \neq M(\epsilon)$, then from step 2 of Algorithm 6 a b-transition is added along side each a-transition in S^{Aug}. On the other hand if $b \in M(a) = M(\epsilon)$, then from step 3 of Algorithm 6, a self-loop transition (y_s, b, y_s) is added in S and also an ϵ-transition is added along side each a-transition. In either case, it follows that that $sbt \in L(S^{Aug})$. Thus, $L(S^{Aug})$ is (Σ^*, Σ_u, M)-achievable, as desired. This completes the proof of the first part of Lemma 17.

2. Now we prove the second part of Lemma 17. By Corollary 5,

$$L(G_{\Sigma}^{Id} \|_{\Sigma_c}^{M} S) = L(G \| S^{Aug}) = L(G) \cap L(S^{Aug}).$$

By Lemma 17, $L(S^{Aug})$ is (Σ^*, Σ_u, M)-achievable. So from last part of Definition 18, $L(G) \cap L(S^{Aug}) = L(G_{\Sigma}^{Id} \|_{\Sigma_c}^{M} S)$ is $(L(G), \Sigma_u, M)$-achievable. ■

From the second part of Lemma 17, if a specification language equals the language of a PSCM controlled plant, then it must be $(L(G), \Sigma_u, M)$-achievable. Thus, $(L(G), \Sigma_u, M)$-achievability serves as a necessary condition for the existence of a supervisor. Next we aim to show the converse that if a specification language is $(L(G), \Sigma_u, M)$-achievable, then it is enforcible by a PSCM-based control.

We propose the following algorithm for constructing a supervisor starting from a specification state machine.

Algorithm 8 Consider a specification state machine $R = (Q, \Sigma, \delta, Q_0)$.

1. For every transition (q, b, q_1) with either $M(b) = M(\epsilon)$ or $\exists (q, b', q_2)$ such that $M(b) = M(b') \neq M(\epsilon)$, replace (q, b, q_1) by a pair of transitions (q, ϵ, q_1') and (q_1', b, q_1), where q_1' is a newly added state. Denote the resulting state machine as $R' = (Q', \Sigma, \delta', Q_0)$.

2. For every state q' of R', every event $b \in \Sigma_u - M(\Sigma(q')) - M(\epsilon)$, add a transition $(q', b, dump)$, where $dump$ is an added state. Denote the resulting state machine as $\widetilde{R} := (\widetilde{Q}, \Sigma, \widetilde{\delta}, Q_0)(Q' \cup \{dump\}, \Sigma, \widetilde{\delta}, Q_0)$.

The following example illustrates Algorithm 8.

Example 25 Consider R shown in Figure 7.4, with

$$\Sigma = \{a, b, c, d\}, \Sigma_u = \{b, c, d\}, M(a) = M(b) = \{a, b\} \neq M(\epsilon), M(c) = \{c, \epsilon\}, M(d) = \{d\}.$$

Since at state 1, $M(a) = M(b)$, by step 1 of Algorithm 8, replace transition $(1, a, 2)$ (resp. $(1, b, 3)$) by transitions $(1, \epsilon, 2')$ and $(2', a, 2)$ (resp. $(1, \epsilon, 3')$ and $(3', b, 3)$). Next since

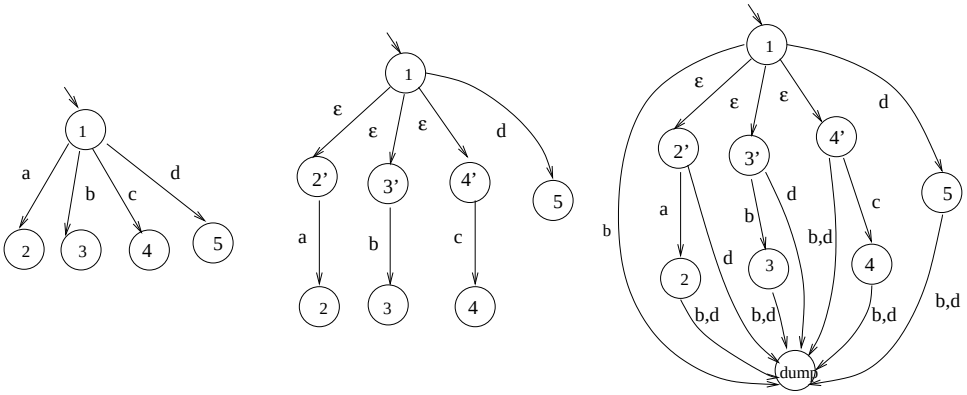

Figure 7.4 R (left), R' (middle), and $\widetilde{R}$ (right)

$M(c) = M(\epsilon)$, by step 1 of Algorithm 8, replace $(1, c, 4)$ by $(1, \epsilon, 4')$ and $(4', c, 4)$. The resulting state machine R' is shown in Figure 7.4.

At state 1, since $\Sigma_u - M(\Sigma(1)) - M(\epsilon) = \{b\}$, by step 2 of Algorithm 8, transition $(1, b, dump)$ is added in R', where $dump$ is an newly added state. At state $2'$, $\Sigma_u - M(\Sigma(2')) - M(\epsilon) = \{d\}$, so transition $(2', d, dump)$ is added in R' to obtain $\widetilde{R}$. Similarly, one can compute the set of transitions on $\Sigma_u - M(\Sigma(\cdot)) - M(\epsilon)$ for other states. The details are omitted here. The resulting state machine $\widetilde{R}$ is shown in Figure 7.4.

We claim that when $L(R)$ is $(L(G), \Sigma_u, M)$-achievable, $\widetilde{R}$ constructed in Algorithm 8 can be used a supervisor, i.e., $L(G_{\Sigma}^{Id} \| _{\Sigma_c}^{M} \widetilde{R}) = L(R)$. Note that $L(G_{\Sigma}^{Id} \| _{\Sigma_c}^{M} \widetilde{R}) = L(G) \cap L(\widetilde{R}^{Aug})$. In the following we first prove that $L(\widetilde{R}^{Aug})$ equals $inf\overline{PA}_{\Sigma^*}(L(R))$. The proof is based on showing the equivalence of the languages $L(\widetilde{R}^{Aug})$ and $L(\hat{R})$, where the construction of $\hat{R}$ is stated above in Algorithm 7.

Lemma 18 Given deterministic R, the set of controllable events Σ_c and a mask M, $L(\widetilde{R}^{Aug}) = L(\hat{R})$, where $\widetilde{R}$ is computed by Algorithm 8, $\widetilde{R}^{Aug}$ is computed by Algorithm 6, and $\hat{R}$ is computed by Algorithm 7.

Proof: In order to facilitate the proof, the computation of $\widetilde{R}^{Aug}$ is shown below by combining Algorithms 6 and 8.

B1 For every transition (q, b, q_1) with either $M(b) = M(\epsilon)$ or $\exists(q, b', q_2)$ such that $M(b) =$

$M(b') \neq M(\epsilon)$, replace (q, b, q_1) by a pair of transitions (q, ϵ, q_1') and (q_1', b, q_1), where q_1' is a newly added state.

B2 For every state q, every event $b \in \Sigma_u - M(\Sigma(q)) - M(\epsilon)$, add a transition $(q, b, dump)$, where *dump* is an added state. The result of these two steps is the state machine, $\widetilde{R} := (\widetilde{Q}, \Sigma, \widetilde{\delta}, Q_0)$.

B3 For $\widetilde{q} \in \widetilde{Q}$ and $\sigma \in Aug(\widetilde{q}) - M(\epsilon)$, (i) if $\widetilde{\delta}(\widetilde{q}, M(\sigma)) \neq \emptyset$, add transitions on σ from $\widetilde{q}$ to all states in the set $\widetilde{\delta}(\widetilde{q}, M(\sigma))$, (ii) otherwise add a self-loop on σ at $\widetilde{q}$;

B4 For $\widetilde{q} \in \widetilde{Q}$ and $\sigma \in Aug(\widetilde{q}) \cap M(\epsilon)$, (i) add self-loop on σ at $\widetilde{q}$. (ii) Further if $\widetilde{\delta}(\widetilde{q}, \sigma) \neq \emptyset$, add transitions on ϵ from $\widetilde{q}$ to all states in the set $\widetilde{\delta}(\widetilde{q}, \sigma)$.

Next we compare the construction of $\widetilde{R}^{Aug}$ and that of $\hat{R}$. Since steps A1 and B1 introduce certain new states in R the same way, whereas the steps A5 and B2 both introduce the "dump" state, the state spaces of $\widetilde{R}^{Aug}$ and $\hat{R}$ are the same. So we next compare the transitions in the two state machines.

- Steps A1 and B1 introduce the transitions in the same way.
- Step A2 introduces self-loop transition on an unobservable event σ at a state q where σ is defined, and also ϵ-transitions along side all σ-transitions at q. Since $\sigma \in \Sigma(q) - \Sigma_o$, and

$$\begin{aligned} Aug(q) &= [\Sigma(q) \cap \Sigma_c - \Sigma_o] \cup \Sigma_u \cup \{\epsilon\} \\ &\supseteq [\Sigma(q) \cap \Sigma_c - \Sigma_o] \cup [\Sigma(q) \cap \Sigma_u - \Sigma_o] \\ &= [\Sigma(q) - \Sigma_o], \end{aligned}$$

  it follows that $\sigma \in Aug(q)$. Also, $\sigma \in M(\epsilon)$ and so the same set of transitions as introduced by step A2 are introduced by steps B4(i) and B4(ii).

- At a state q, step A3 introduces transitions on an uncontrollable and unobservable event along side each ϵ-transition at q (including a self-loop at q). Only a self-loop transition on such an event is added at q by step B4(i). (Recall that $\Sigma_u \subseteq Aug(\cdot)$.) As we will see this

is the only difference between $\hat{R}$ and $\widetilde{R}^{Aug}$: For $b \in \Sigma_u \cap M(\epsilon)$ if (q, ϵ, q') is a transition in $\widetilde{R}$, then step A3 adds transitions (q, b, q) and (q, b, q'), whereas only the transition (q, b, q) is added by step B4(i). This however keeps the languages $L(\hat{R})$ and $L(\widetilde{R}^{Aug})$ to be the same since the extra transition (q, b, q') can be simulated as the self-loop transition (q, b, q) followed by the ϵ-transition (q, ϵ, q') without affecting the trace generated.

- At a state q, step A4 introduces transitions on an event $\sigma \in \Sigma_u - M(\epsilon) \cap M(\Sigma(q))$ along side all transitions in $M(\sigma) \cap \Sigma(q)$. The same set of transitions are introduced on the event $\sigma \in \Sigma_u - M(\epsilon) \cap M(\Sigma(q)) \subseteq Aug(q) - M(\epsilon)$ by step B3(i).

- Steps A5(i) and B2 introduce the uncontrollable transitions, that are not unobservable but for which no indistinguishable events are locally defined, in the same way.

- Step A5(ii) introduces self-loops on uncontrollable events at the dump state. Since in $\widetilde{R}$, no transitions are defined at the dump state, $Aug(dump) = \Sigma_u$. At the dump state, self-loops on events in $Aug(dump) - M(\epsilon) = \Sigma_u - M(\epsilon)$ are introduced by step B3(ii), whereas self-loops on events in $Aug(dump) \cap M(\epsilon) = \Sigma_u \cap M(\epsilon)$ are introduced by step B4(i).

It follows that except for one case, the transitions in $\hat{R}$ are the same as those in $\widetilde{R}^{Aug}$, and even in this exceptional case, the difference is such that the language is preserved, proving the assertion that $L(\hat{R}) = L(\overline{R}^{Aug})$. ■

The following corollary immediately follows.

Corollary 6 Given deterministic R, the set of controllable events Σ_c and a mask M, if $L(R) \subseteq L(G)$ is $(L(G), \Sigma_u, M)$-achievable, then $L(G_{\Sigma}^{Id} \| _{\Sigma_c}^{M} \widetilde{R}) = L(R)$, where $\widetilde{R}$ is computed by Algorithm 8.

Proof: From Corollary 5, $L(G_{\Sigma}^{Id} \| _{\Sigma_c}^{M} \widetilde{R}) = L(G) \cap L(\widetilde{R}^{Aug})$. By Lemma 18, $L(\widetilde{R}^{Aug}) = L(\hat{R})$, where $L(\hat{R}) = inf\overline{PA}_{\Sigma^*}(L(R))$ (Kumar et al., 2005a, Theorem 6). It follows that

$$\begin{aligned} inf\overline{PA}_{L(G)}(L(R)) &= L(G) \cap inf\overline{PA}_{\Sigma^*}(L(R)) \\ &= L(G) \cap L(\widetilde{R}^{Aug}) \end{aligned}$$

$$= \quad L(G_{\Sigma}^{Id} \| _{\Sigma_c}^{M} \widetilde{R})$$

Since $L(R)$ is given to be $(L(G), \Sigma_u, M)$-achievable, it holds that $L(R) = inf\overline{PA}_{L(G)}(L(R))$, and so the result follows. ■

By Lemma 17 and Corollary 6, we obtain the following necessary and sufficient condition for the existence of a PSCM-based supervisor enforcing a given specification.

Theorem 20 Given G, R, the set of controllable events Σ_c and a mask M, there exists a supervisor S such that $L(G_{\Sigma}^{Id} \| _{\Sigma_c}^{M} S) = L(R)$ if and only if $L(R) \subseteq L(G)$ and $L(R)$ is $(L(G), \Sigma_u, M)$-achievable.

Proof: (*Only If*) From Lemma 17, $L(G_{\Sigma}^{Id} \| _{\Sigma_c}^{M} S)$ is $(L(G), \Sigma_u, M)$-achievable, and since $L(R) = L(G_{\Sigma}^{Id} \| _{\Sigma_c}^{M} S)$, $L(R)$ is also $(L(G), \Sigma_u, M)$-achievable. Further from Corollary 5, $L(G_{\Sigma}^{Id} \| _{\Sigma_c}^{M} S) = L(G \| S^{Aug}) = L(G) \cap L(S^{Aug})$, it follows that $L(R) = L(G_{\Sigma}^{Id} \| _{\Sigma_c}^{M} S) \subseteq L(G)$.

(*If*) Since $L(R) \subseteq L(G)$ is $(L(G), \Sigma_u, M)$-achievable, using Algorithm 8 we construct NSM $\widetilde{R}$. Then from Corollary 6, $L(G_{\Sigma}^{Id} \| _{\Sigma_c}^{M} \widetilde{R}) = L(R)$, i.e., $\widetilde{R}$ is the desired supervisor. This completes the proof. ■

Remark 22 It follows from Theorem 20 that the existing tests for achievability, which is of complexity $O(|G||R|^2)$ Kumar et al. (2005a), can be applied to verify the existence of a supervisor. Further if the existence condition is satisfied, a supervisor can be obtained by applying Algorithm 8 to a generator R of the specification language, implying the synthesis of a supervisor is of complexity $O(|R|)$. Note that in general the state machine obtained by Algorithm 8 is not (Σ_u, M)-compatible. Thus, by using the control mechanism of PSCM, the requirement of (Σ_u, M)-compatibility of a supervisor has been removed, as desired.

7.4 An Illustrative Example

We illustrate Theorem 20 through a manufacturing example, taken from Kumar et al. (2005a). The manufacturing system consists of one robot, two workstations and two storage-stations. The robot moves among the workstations and storage-stations on a guide rail. Initially, the robot departs from workstation 1 (event a). Then it picks up a part either from

storage-station 1 (event b_1) or storage-station 2 (event b_2), and delivers the part to workstation 2 for processing (event c). After the processing, robot returns the part to a storage-station. After returning parts, robot goes back to workstation 1 and can repeat the whole process. A state machine model G of the system is drawn in Figure 7.5.

Not returning the part to its original storage-station is not desirable. The specification R, also shown in Figure 7.5, gives the desired behavior. According to the specification, after processing, the robot returns the part to its original storage-station. The rest of the behavior is the same as the one feasible in the system.

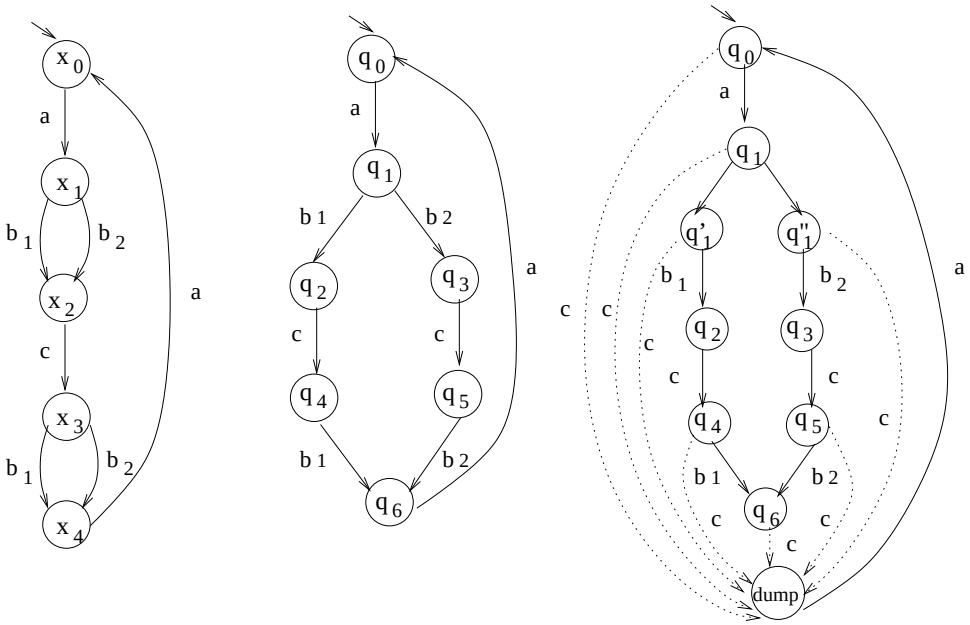

Figure 7.5 G (left), R (middle), and $\widetilde{R}$ (right)

We require that the part must be delivered to workstation 2 for processing, i.e., the event c is uncontrollable. Also, only events a and c are completely observable. Events b_1 and b_2 are observationally indistinguishable. Thus, we have $\Sigma = \{a, b_1, b_2, c\}$, $\Sigma_c = \{a, b_1, b_2\}$, and the observation mask M is given by, $M(a) = \{a\}$, $M(b_1) = M(b_2) = \{b_1, b_2\}$ and $M(c) = \{c\}$. It can be verified that $L(R)$ is not observable, i.e., a deterministic supervisor does not exist. However, $L(R)$ is $(L(G), \Sigma_u, M)$-achievable and so from Theorem 20, a PSCM-based supervisor does exist. (Thanks to PSCM-based control formalism developed here that allows supervisor

to be nondeterministic.)

We use Algorithm 8 to construct $\widetilde{R}$ that acts as a supervisor.

At state q_1 of R, $M(b_1) = M(b_2)$, by step 1 of Algorithm 8, we replace transition (q_1, b_1, q_2) (resp., (q_1, b_2, q_3)) by a pair of transitions (q_1, ϵ, q_1') and (q_1', b_1, q_2) (resp., (q_1, ϵ, q_2'') and (q_2'', b_2, q_3)). Next by step 2 of Algorithm 8, since $\Sigma_u - M(\Sigma(q)) - M(\epsilon) = \{c\}$ for state $q \in \{q_0, q_1, q_1', q_1'', q_4, q_5, q_6\}$, we add transitions on c from states $q_0, q_1, q_1', q_1'', q_4, q_5$, and q_6 to the newly added "*dump*" state.

We obtain the state machine $\widetilde{R}$ drawn in Figure 7.5. One can verify that $L(G_{\Sigma}^{Id} \|_{\Sigma_c}^{M} \widetilde{R}) = L(R)$, i.e., $\widetilde{R}$ serves as a desired supervisor. It should be noted that $\widetilde{R}$ is not (Σ_u, M)-compatible. As an example, the uncontrollable event c is undefined at the *dump* state.

7.5 Control for Bisimulation Equivalence via PSCM

In this section, we study the PSCM-based control to ensure bisimilarity of the controlled plant and the given specification.

We first show the following result for a (Σ_u, M)-compatible S and its augmentation.

Lemma 19 Given plant G and the set of controllable events Σ_c and a mask M, if S is (Σ_u, M)-compatible, then $S = S^{Aug}$.

Proof: If $\sigma \in Aug(y) - M(\epsilon)$, where $Aug(y)$ is computed by Equation 7.6, then (i) $\sigma \in \Sigma_u$, (ii) $\sigma \in \Sigma_c \cap \Sigma(y)$, or (iii) $\sigma = \epsilon$.

For case (i), since S is Σ_u-compatible, σ is defined at state y. Thus, no transition is added in the augmentation.

For case (ii) or (iii), since S is M-compatible, $\beta(y, \sigma) = \beta(y, M(\sigma) \cap \Sigma(y))$. Thus, no transition is added in the augmentation. So we conclude $S = S^{Aug}$. ∎

From Lemma 17 and Corollary 5, if we have a (Σ_u, M)-compatible supervisor such that $G \| S \simeq R$, then $G_{\Sigma}^{Id} \|_{\Sigma_c}^{M} S = G || S^{Aug} = G \| S \simeq R$. I.e., if there exists a SSC-based bisimilarity enforcing supervisor, then there exists a PSCM-based bisimilarity enforcing supervisor. This result is shown in the following corollary, its proof is omitted.

Corollary 7 Given plant G and the set of controllable events Σ_c and a mask M, if exists a (Σ_u, M)-compatible supervisor S such that $G \| S \simeq R$, then exists a supervisor S' such that $G_{\Sigma}^{Id} \|_{\Sigma_c}^{M} S' \simeq R$.

However, the reverse of Corollary 7 may not hold since S^{Aug} may not be (Σ_u, M)-compatible. For this consider G and S shown in Figure 7.6, where $\Sigma_c = \{a_2\}$ and $M(a_1) = M(a_2) = \{\epsilon, a_1, a_2\}$. S^{Aug} is also shown in Figure 7.6, which is not (Σ_u, M)-compatible.

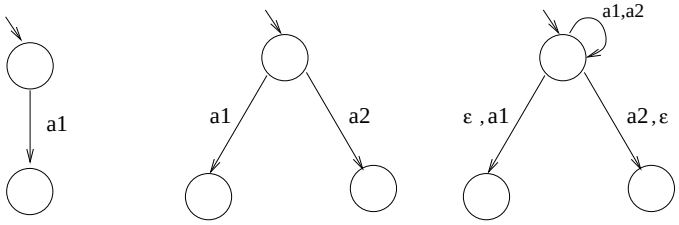

Figure 7.6 G (left), S (middle) and S^{Aug} (right)

Next we define a new augmentation from which the augmented S is (Σ_u, M)-compatible.

Algorithm 9 For $i \leq n$, consider NSM G_i possessing event priority set A_i and observation mask M_i. The following algorithm computes augmented G_i, $G_i^{\overline{Aug}_i} := (X_i, \Sigma, \alpha_i^{\overline{Aug}}, X_{0i})$.

1. Same as step 1 of Algorithm 6.
2. Same as step 2 of Algorithm 6.
3. For $\sigma \in Aug_i(x_i) \cap M_i(\epsilon)$, if $\alpha_i(x_i, M_i(\sigma)) \neq \emptyset$, add transitions on σ from x_i to all states in this set, and add self-loop on σ at x_i.

Lemma 20 Given plant G with the set of controllable events Σ_c and a mask M, and supervisor S,

1. $S^{\overline{Aug}}$ is (Σ_u, M)-compatible, and
2. $S^{\overline{Aug}} \simeq S^{Aug}$,

where $S^{\overline{Aug}}$ and S^{Aug} are computed from Algorithm 9 and 6, respectively.

Proof:

1. Since $\Sigma_u \subseteq Aug(y)$, so $S^{\overline{Aug}}$ is Σ_u-compatible. Further by step 2 and 3, σ-transition is added along every indistinguishable transition at a state. This implies that $S^{\overline{Aug}}$ is M-compatible. Thus, $S^{\overline{Aug}}$ is (Σ_u, M)-compatible.

2. Note that the state spaces of $S^{\overline{Aug}}$ and S^{Aug} are same. And also they are same if there is at most one unobservable event defined at each state of S. I.e., in this case, there exists a bisimulation relation $\Phi := \{(y, y) \mid y \in S\}$ such that $S^{\overline{Aug}} \simeq_\Phi S^{Aug}$.

   Suppose $\sigma_i \in \Sigma(y) \cap M(\epsilon)$ and $y_i \in \beta(y, \sigma_i)$ for $i = 1, 2$. Then by step 3 of Algorithm 9 and 6, the only difference of $S^{\overline{Aug}}$ and S^{Aug} is that transition σ_j is added along transition σ_i in $S^{\overline{Aug}}$ but not in S^{Aug} for $i, j = 1, 2$ and $i \neq j$. However, note that σ_i is defined as a self-loop in both machines, so $y_j \in \beta^{\overline{Aug}}(y, \sigma_i)$ and is reachable by σ_i^* in $S^{\overline{Aug}}$, and $y_j \in \beta^{Aug}(y, \sigma_i\epsilon)$ and is reachable by σ_i^* in S^{Aug}. Thus, the bisimulation relation Φ is still valid even when there are more than one unobservable events defined at each state of S. This proves the lemma. ■

From Lemma 20 and Corollary 5 , if we have a supervisor S such that $G_\Sigma^{Id} \|_{\Sigma_c}^M S \simeq R$, then there exists a (Σ_u, M)-compatible supervisor $S^{\overline{Aug}}$ such that $G \| S^{\overline{Aug}} \simeq G \| S^{Aug} = G_\Sigma^{Id} \|_{\Sigma_c}^M S \simeq R$, I.e., if there exists a PSCM-based bisimilarity enforcing supervisor, then there exists a SSC-based bisimilarity enforcing supervisor. This result is shown in the following corollary, its proof is omitted.

Corollary 8 Given plant G and the set of controllable events Σ_c and a mask M, if exists a supervisor S such that $G_\Sigma^{Id} \|_{\Sigma_c}^M S \simeq R$, then exists a (Σ_u, M)-compatible supervisor S' such that $G \| S' \simeq R$.

From Corollary 7 and 8, the following theorem is obvious, and its proof is omitted.

Theorem 21 Given plant G and the set of controllable events Σ_c and a mask M, exists a supervisor S such that $G_\Sigma^{Id} \|_{\Sigma_c}^M S \simeq R$ if and only if exists a (Σ_u, M)-compatible supervisor S' such that $G \| S' \simeq R$.

Remark 23 From Theorem 4, the SSC-based bisimilarity enforcing supervisor exists in the state space $2^{X\times Q}$. Thus, by Theorem 21, the PSCM-based bisimilarity enforcing supervisor also exists in the state space $2^{X\times Q}$.

7.6 Conclusion

In this chapter we introduced the notion of prioritized synchronous composition under mask (PSCM) to model interaction/control of discrete event systems under partial observation. This extends the formalism of prioritized synchronous composition (PSC) which assumes identity observation masks. We established a link between PSCM and PSC (and thereby SSC) under certain constraints on the priority sets and the mask functions: PSCM of two systems can be alternatively obtained by first augmenting individual systems, and next computing the PSC of the augmented systems. For this to work, the priority sets of the two systems must exhaust the entire event set, and each event must be observable to a system having priority over the event.

We showed that when PSCM is adopted as a mechanism of control, not only the control & observation-compatibility requirements are removed of a supervisor, the existence condition is given by achievability that is weaker than controllability and observability combined. (The weaker condition is required since the supervisor is allowed to be nondeterministic.) This suggests that the notion of PSCM, introduced in the paper is an appropriate generalization of PSC to account for partial observation. We also showed that both the existence verification and synthesis of a PSCM-based supervisor is polynomially solvable. The existence is linear in the size of the plant and quadratic in the size of the specification, whereas the synthesis is linear in the size of the specification. For PSCM-based bisimilarity control, we showed that the small model theorem remains valid.

CHAPTER 8. CONCLUSION AND FUTURE WORK

8.1 Summary

In this dissertation, we studied supervisory control of discrete event systems for enforcing bisimulation equivalence and simulation equivalence specifications. The bisimulation equivalence allows the specification of the full set of branching constraints (such as nonblocking), while the simulation equivalence can express specifications in the the universal fragment of branching-time logics (such as properties of all sequencing/branching behaviors of a system). We also proposed a formalism of prioritized synchronous composition under mask (PSCM) for modeling interaction/control of partially observed discrete event systems.

The main contributions of the dissertation are summarized as follows.

1. We studied a more general bisimulation equivalence control problem, namely, in which both the plant as well as the specification are nondeterministic. No prior work addresses this problem in this generality – they impose determinism either on the plant or on the specification. Our main result is a small model theorem showing that a supervisor for enforcing bisimulation equivalence between the specification and the controlled system exists if and only if it exists over a certain finite state space, namely the power set of Cartesian product of the plant and the specification state spaces.

2. We showed that the simulation equivalence represents a nice compromise between the complexity of control vs. its fineness. While bisimilarity enforcing control is the finest, the best known complexity for such a control is doubly exponential in the sizes of the plant and the specification. On the other hand, while a language equivalence control is polynomially solvable, it is the coarsest. A simulation equivalence specification is finer

than a language equivalence specification, yet it remains polynomially solvable.

3. When the system model is deterministic, we introduced a notion of state-controllability-bisimilar (resp., state-achievability-bisimilar) as part of the necessary and sufficient condition for the existence of a supervisor under complete (resp., partial) observation. We showed that under complete observation, the complexity of both verifying existence and synthesis of a bisimilarity enforcing supervisor is linear in size of plant and specification. Under partial observation, the existence of a bisimilarity enforcing supervisor can be determined polynomially in both plant and specification states, and the upper bound for the synthesis we provided is exponential.

4. We introduced PSCM to model interaction/control of partially observed discrete event systems. We showed that when PSCM is adopted as a mechanism of interaction, not only the control & observation-compatibility requirements are removed of a supervisor, the existence condition for a supervisor such that the controlled system is language equivalent to a specification is given by *achievability* that is weaker than controllability and observability combined. (The weaker condition is required since we allow supervisors to be nondeterministic.) This suggests that the notion of PSCM introduced in this work is an appropriate generalization of PSC to account for partial observation. We studied control of (nondeterministic) DESs for achieving bisimulation equivalence specifications using PSCM as a control mechanism, and established an equivalence between a PSCM-based bisimilarity enforcing controller and a SSC-based bisimilarity enforcing controller by showing that if one of them exists and the other one also exists.

8.2 Directions for Further Research

This thesis has laid a foundation for control of discrete event systems for achieving bisimulation and simulation equivalence, and opens up several avenues for future work in this area.

1. Partial specification: In many applications, the system events can be partitioned into the external and the internal events. The specification is only concerned with the behavior

projected onto the external events. Future work can study the problem of control for achieving bisimulation equivalence to a partial-specification.

2. DESs modeled as Petri-Nets: Petri-nets provides a graphical and compact representation of concurrent nondeterministic systems. It is important to study the problem of control of DESs modeled as Petri-Nets for bisimulation or simulation equivalence.

3. Timed DESs: In some application, not only the order but also the time of the occurrence of events is important. Future work can study the problem of control of DESs modeled as timed automata or timed Petri-Nets for achieving bisimulation or simulation equivalence.

BIBLIOGRAPHY

Antoniotti, M. (1995). *Synthesis and Verification of Discrete Controllers for Robotics and Manufacturing Devices with Temporal Logic and Control-D Systems*. PhD dissertation, Department of Computer Science, New York University, New York, NY.

Arnold, A., Vincent, A., and Walukiewicz, I. (2003a). Games for synthesis of controllers with partial observation. *Theoretical Computer Science*, pages 7–34.

Arnold, A., Vincent, A., and Walukiewicz, I. (2003b). Games for synthesis of controllers with partial observation. *Theoretical Computer Science*, 303:7–34.

Baeten, J. C. M., Bergstra, J. A., and Klop, J. W. (1987). Ready-trace semantics for concrete process algebra with the priority operator. *The Computer Journal*, 30(6):498–506.

Balemi, S. (1994). Input/output discrete event processes and communication delays. *Discrete Event Dynamical Systems: Theory and Applications*, 4(1):41–85.

Barrett, G. and Lafortune, S. (1998). Bisimulation, the supervisory control problem and strong model matching for finite state machines. *Journal of Discrete Event Systems: Theory and Applications*, 8:377–429.

Benedetto, M. D. D., Sangiovanni-Vincentelli, A. L., and Villa, T. (2001). Model matching for finite state machines. *IEEE Transactions on Automatic Control*, 46:1726–1743.

Bensalem, S., Bouajjani, A., Loiseaux, C., and Sifakis, J. (1992). Property preserving simulations. In *CAV*, pages 260–273.

Brave, Y. and Heymann, M. (1993). Control of discrete event systems modeled as hierarchical state machine. *IEEE Transactions on Automatic Control*, 38(12):1803–1819.

Cassandras, C. G. and Lafortune, S. (1995). *Introduction to Discrete Event Systems*. Kluwer Academic Publishers, Norwell, MA.

Chen, Y.-L., Lafortune, S., and Lin, F. (2000). Design of nonblocking modular supervisors using event priority functions. *IEEE Transactions on Automatic Control*, 45(3).

Cieslak, R., Desclaux, C., Fawaz, A., and Varaiya, P. (1988). Supervisory control of discrete event processes with partial observation. *IEEE Transactions on Automatic Control*, 33(3):249–260.

Fabian, M. (1995). *On Object Oriented Nondeterministic Supervisory Control*. PhD dissertation, Control Engineering Laboratory, Chalmers University, Goteberg.

Fabian, M. and Kumar, R. (2000). Mutually nonblocking supervisory control of discrete event systems. *Automatica*, 36:1863–1869.

Hennessey, M. and Milner, R. (1985). Algebraic laws for nondeterminism and concurrency. *Journal of ACM*, 32:137–161.

Henzinger, M. R., Henzinger, T. A., and Kopke, P. W. (1995). Computing simulations on finite and infinite graphs. In *36th Annual Symposium on Foundations of Computer Science (FOCS), IEEE Computer Society Press*, pages 453–462.

Henzinger, T. A., Kupferman, O., and Majumdar, R. (2003). On the universal and existential fragments of the mu-calculus. In *Proceedings of the Ninth International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS), Lecture Notes in Computer Science*, 2619, pages 49–64. Springer-Verlag.

Heymann, M. (1990). Concurrency and discrete event control. *IEEE Control Systems Magazine*, 10(4):103–112.

Heymann, M. and Lin, F. (1998). Discrete-event control of nondeterministic systems. *IEEE Transactions on Automatic Control*, 43(1):3–17.

Heymann, M. and Meyer, G. (1991). Algebra of discrete event processes. Technical Report NASA 102848, NASA Ames Research Center, Moffett Field, CA.

Hoare, C. A. R. (1985). *Communicating Sequential Processes*. Prentice Hall, Inc., Englewood Cliffs, NJ.

Hopcroft, J. E. and Ullman, J. D. (1979). *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, Reading, MA.

Inan, K. (1994). Nondeterministic supervision under partial observations. In Cohen, G. and Quadrat, J.-P., editors, *Lecture Notes in Control and Information Sciences 199*, pages 39–48. Springer-Verlag, New York.

Jiang, S. and Kumar, R. (2000). Decentralized control of discrete event systems with specializations to local control and concurrent systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 30(5):653–660.

Jiang, S. and Kumar, R. (2001). Supervisory control of discrete event systems with CTL* temporal logic specification. In *2001 IEEE Conference on Decision and Control*, pages 4122–4127, FL.

Jiang, S. and Kumar, R. (2002). Supervisory control of nondeterministic discrete event systems with driven events via masked prioritized synchronization. *IEEE Transactions on Automatic Control*, 47(9):1438–1449.

Jiang, S. and Kumar, R. (2006). Supervisory control of discrete event systems with CTL* temporal logic specification. *SIAM Journal on Control and Optimization*, 44(6):2079–2103.

Khatri, S., Narayan, A., Krishnan, S., McMillan, K., Brayton, R., and Sangiovanni-Vincentelli, A. (1996). Engineering change in a non-deterministic FSM setting. In *dac*, pages 451–456.

Komenda, J. (2002a). Coalgebra and supervisory control of discrete-event systems with partial observations. In *International Symposium (MTNS 2002)*, pages 12–16, Notre Dame.

Komenda, J. (2002b). Computation of supremal sublanguages of supervisory control using coalgebra. In *Workshop on Discrete-Event Systems (WODES'02)*, pages 26–33, Zaragoza.

Kozak, P. and Wonham, W. M. (1995). Fully decentralized solutions of supervisory control problems. *IEEE Transactions on Automatic Control*, 40(12):2094–2097.

Kumar, R. and Garg, V. K. (1995). *Modeling and Control of Logical Discrete Event Systems*. Kluwer Academic Publishers, Boston, MA.

Kumar, R., Garg, V. K., and Marcus, S. I. (1991). On controllability and normality of discrete event dynamical systems. *Systems and Control Letters*, 17(3):157–168.

Kumar, R. and Heymann, M. (2000a). Masked prioritized synchronization for interaction and control of discrete event systems. *IEEE Transactions on Automatic Control*, 45(11):1970–1982.

Kumar, R. and Heymann, M. (2000b). Masked prioritized synchronization for interaction and control of discrete event systems. *IEEE Transactions on Automatic Control*, 45(11):1970–1982.

Kumar, R., Jiang, S., Zhou, C., and Qiu, W. (2005a). Control using nondeterministic supervisors for partially observed discrete event systems. In *Proceedings of 2004 American Control Conference*, pages 4472–4476, Boston, MA.

Kumar, R., Jiang, S., Zhou, C., and Qiu, W. (2005b). Polynomial synthesis of supervisor for partially observed discrete-event systems by allowing nondeterminism in control. *IEEE Transactions on Automatic Control*, 50(4):463–475.

Kumar, R. and Shayman, M. A. (1996a). Nonblocking supervisory control of nondeterministic systems via prioritized synchronization. *IEEE Transactions on Automatic Control*, 41(8):1160–1175.

Kumar, R. and Shayman, M. A. (1996b). Supervisory control of real-time systems using prioritized synchronization. In Alur, R., Henzinger, T., and Sontag, E., editors, *Hybrid Systems III*, volume 1066 of *Lecture Notes in Computer Science*. Springer-Verlag, New York.

Kumar, R. and Shayman, M. A. (1997). Centralized and decentralized supervisory control of nondeterministic systems under partial observation. *SIAM Journal of Control and Optimization*, 35(2):363–383.

Kumar, R., Takai, S., Fabian, M., and Ushio, T. (2005c). Maximally permissive mutually & globally nonblocking supervision with application to switching control. *Automatica*, 41(8):1299–1312.

Kupferman, O., Madhusudan, P., Thiagarajan, P. S., and Vardi, M. Y. (2000). Open systems and reactive environments: control and synthesis. In *Proceedings of 11th Conference on Concurrency Theory, (Lecture Notes in Computer Science)*, volume 1877, pages 92–107. Springer-Verlag.

Kupferman, O. and Vardi, M. Y. (1999). Robust satisfaction. In *Proceedings of 10th Conference on Concurrency Theory, (Lecture Notes in Computer Science)*, volume 1664, pages 382–398. Springer-Verlag.

Kupferman, O., Vardi, M. Y., and Wolper, P. (2001). Module checking. *Information and Computation*, 164:322–344.

Madhusudan, P. and Thiagarajan, P. (2002). Branching time controllers for discrete event systems. *Theoretical Computer Science*, 274:117–149.

Maidi, M. (2000). The common fragment of CTL and LTL. In *Proceedings of Foundations of Computer Science*, pages 643–652.

Marchand, H. and Pinchinat, S. (2000). Supervisory control problem using symbolic bisimulation techniques. In *2000 American Control Conference*, pages 4067–4071, Chicago, Illinois, USA.

Milner, R. (1980). *A Calculus of Communicating Systems*. Springer Verlag.

Milner, R. (1989). *Communication and Concurrency*. Prentice Hall, New York.

Overkamp, A. (1994). Supervisory control for nondeterministic systems. In Cohen, G. and Quadrat, J.-P., editors, *Lecture Notes in Control and Information Sciences 199*, pages 59–65. Springer-Verlag, New York.

Phillips, I. (1987). Refusal testing. *Theoretical Computer Science*, 50:241–284.

Prosser, J., Kam, M., and Kwatny, H. (1998). Supervisor synthesis for partially-observed discrete event systems. *IEEE Transactions on Automatic Control*, 43(11).

Qin, H. and Lewis, P. (1990). Factorization of finite state machines under observational equivalence. In *Lecture Notes In Computer Science*, volume 458. Springer-Verlag.

Ramadge, P. J. and Wonham, W. M. (1987). Supervisory control of a class of discrete event processes. *SIAM Journal of Control and Optimization*, 25(1):206–230.

Ramadge, P. J. and Wonham, W. M. (1989). The control of discrete event systems. *Proceedings of IEEE: Special Issue on Discrete Event Systems*, 77:81–98.

Riedweg, S. and Pinchinat, S. (2003). Quantified mu-calculs for control synthesis. In *Mathematical Foundations of Computer Science*, Bratislava, Slovak Republic.

Rohloff, K. and Lafortune, S. (2002). On the computational complexity of verification of modular discrete event systems. In *Proceedings of 2002 IEEE Conference on Decision and Control*, Las Vegas, NV.

Rutten, J. (1995). A calculus of transition systems (towards universal coalgebra). In Ponse, A., de Rijke, M., and Venema, Y., editors, *Modal Logic and Process Algebra - A Bisimulation Perspective*, volume 53, pages 231–256, Stanford, California. CSLI Lecture Notes, CSLI Publications.

Rutten, J. (2000). Coalgebra, concurrency, and control. In *5th Workshop on Discrete Event Systems*, pages 31–38.

Shayman, M. A. and Kumar, R. (1995). Supervisory control of nondeterministic systems with driven events via prioritized synchronization and trajectory models. *SIAM Journal of Control and Optimization*, 33(2):469–497.

Shayman, M. A. and Kumar, R. (1999). Process objects/masked composition: An object oriented approach for modeling and control of discrete event systems. *IEEE Transactions on Automatic Control*, 44(10):1864–1869.

Tabuada, P. (2004). Open maps, alternating simulations and control synthesis. In *International Conference on Concurrency Theory*, pages 466–480.

Tripakis, S. (2001). Undecidable problems of decentralized control and observation. In *Proceedings of 2001 IEEE Conference on Decision and Control*, pages 4104–4109, Orlando, FL.

Tsitsiklis, J. N. (1989). On the control of discrete event dynamical systems. *Mathematics of Control Signals and Systems*, 2(2):95–107.

Wong, K. C. and Wonham, W. M. (1996). Hierarchical control of discrete event systems. *Discrete Event Dynamical Systems: Theory and Applications*, 6:241–273.

Yevtushenko, N., Villa, T., Brayton, R., Petrenko, A., and Sangiovanni-Vincentelli, A. (2001). Solution of parallel logic equations for logic synthesis. In *ICCAD*, pages 103–110.

Yoo, T. and Lafortune, S. (2001). On the computational complexity of some problems arising in partially-observed discrete event systems. In *Proceedings of 2001 American Control Conference*, Arlington, VA.

Zhong, H. and Wonham, W. M. (1990). On the consistency of hierarchical supervision in discrete-event systems. *IEEE Transactions on Automatic Control*, 35(10):1125–1134.

Zhou, C., Kumar, R., and Jiang, S. (2004). Control of nondeterministic discrete event systems for bisimulation equivalence. In *Proceedings of 2004 American Control Conference*, pages 4488–4492, Boston, MA.