Property Enforcement for Partially-Observed Discrete-Event Systems

by

Xiang Yin

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Electrical Engineering: Systems)
in The University of Michigan
2017

Xiang Yin

xiangyin@umich.edu

ORCID iD: 0000-0003-1944-1570

To my parents and my grandparents.

ACKNOWLEDGEMENTS

First of all, I would like to express my sincere gratitude to my advisor, Professor Stéphane Lafortune, for his encouragement, guidance and for giving me so much freedom to explore anything I am interested in. I would also like to thank Professor Necmiye Ozay, Professor Demosthenis Teneketzis and Professor Dawn Tilbury for accepting to sit on my dissertation committee.

I would like to thank my colleges Dr. Eric Dallal, Dr. Yi-Chin Wu, Yiding Ji, Romulo Meira Goes, Dr. Lilian Carvalho, Dr. Richard Hill, Dr. Eunsuk Kang and Dr. Christoforos Keroglou for their discussions on Discrete-Event Systems and other research topics. I would also like to thank Maxwell Morrison and Siyuan Sheng for helping me to implement some algorithms in the dissertation.

I thank my friends Zhaojian Li and Cao Gao for their supports and encouragements. I also would like to thank Dinghao Hu, Chen Zhang, Zhengye Bian and Xiaowei Wu for their remote but daily companies through our small group the Chinese Academy of "Kobe" Sciences.

I would like to acknowledge the financial support from NSF grant CCF-1138860, NSF grant CNS-1446298, NSF grant CNS-1421122 and the Rackham Predoctoral Fellowship Award from the Rackham Graduate School at the University of Michigan.

Finally, I sincerely thank my parents, Yi Zhang and Guangli Yin, and my grandparents, Shuying Chen and Zhengguo Zhang, for their constant support and love. I would not have been the person I am today without their support. I owe my fiancee, Dr. Tingting Bai, too much in past five years. I would like to express my deepest gratitude to Tingting for her love, support and sacrifice. I love you.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

CPS	Cyber-Physical Systems
DES	Discrete Event Systems
IS-based	information-state-based
AES	All Enforcement Structure
MPIEP	Maximally Permissive IS-Based Property Enforcement Problem
BTS	Bipartite Transition System
NB-AES	Non-blocking All Enforcement Structure
SCC	Strongly Connected Component
NB-MPIEP	Non-blocking Maximally Permissive IS-Based Property Enforcement Problem
EBTS	Extended Bipartite Transition System
UBTS	Unfolded Bipartite Transition System
CELC	Critical Elementary Livelock Cycle
ICS	Inter-Connected System
MPRCP	Maximally-Permissive Range Control Problem
CSR	Control Simulation Relation
BDO	Bipartite Dynamic Observer
MPO	Most Permissive Observer

ABSTRACT

Property Enforcement for Partially-Observed Discrete-Event Systems

by

Xiang Yin

Chair: Stéphane Lafortune

Engineering systems that involve physical elements, such as automobiles, aircraft, or electric power pants, that are controlled by a computational infrastructure that consists of several computers that communicate through a communication network, are called Cyber-Physical Systems. Ever-increasing demands for safety, security, performance, and certification of these critical systems put stringent constraints on their design and necessitate the use of formal model-based approaches to synthesize provably-correct feedback controllers. This dissertation aims to tackle these challenges by developing a novel methodology for synthesis of control and sensing strategies for Discrete Event Systems (DES), an important class of cyber-physical systems. First, we develop a uniform approach for synthesizing property enforcing supervisors for a wide class of properties called information-state-based (IS-based) properties. We then consider the enforcement of non-blockingness in addition to IS-based properties. We develop a finite structure called the All Enforcement Structure (AES) that embeds all valid supervisors. Furthermore, we propose novel and general approaches to solve the sensor activation problem for partially-observed DES. We extend our results for the sensor activation problem from the centralized case to the decentralized case.

The methodology in the dissertation has the following novel features: (i) it explicitly considers and handles imperfect state information, due to sensor noise, and limited controllability, due to unexpected environmental disturbances; (ii) it is a uniform information-state-based approach that can be applied to a variety of user-specified requirements; (iii) it is a formal model-based approach, which results in provably correct solutions; and (iv) the methodology and associated theoretical foundations developed are generic and applicable to many types of networked cyber-physical systems with safety-critical requirements, in particular networked systems such as aircraft electric power systems and intelligent transportation systems.

CHAPTER I

Introduction

Engineering systems that involve physical elements, such as automobiles, aircraft, or electric power pants, that are controlled by a computational infrastructure that consists of several computers that communicate through a communication network, are called Cyber-Physical Systems (CPS). In the study of CPS, verification and synthesis are two important research issues. In large complex automated systems, we are first interested in verifying whether or not the given system satisfies a certain property of interest. When the answer is negative, we wish to synthesize some strategy that provably enforces the property by a certain enforcement mechanism. Ever-increasing demands for safety, security, performance, and certification of these critical systems put stringent constraints on their design and necessitate the use of formal model-based approaches to synthesize provably-correct feedback controllers. Safety-criticality of these systems and often times the need for certification make it essential to employ methodologies that lead to provably-correct solutions. Fast and principled ways of synthesizing controllers with correctness guarantees for these systems will directly benefit the industries developing such systems as one can significantly reduce the system integration, verification and validation cycle (and therefore time-to-market).

Unfortunately, today's design tools to handle these requirements are inadequate. Ad hoc approaches are currently employed by software teams to implement case-by-case solutions, which do not have safety guarantees. These solutions consist more or less of large lists of "if-then-else" rules whose overall outcome as a system is in general impossible to decipher. Such ad hoc solutions are especially error-prone due to the fact that it is not possible to exhaustively enumerate all possible combinations of variables for all different accident scenarios.

This dissertation aims to tackle these design challenges by developing a novel methodology for synthesis of provably correct control and sensing strategies for Discrete Event Systems (DES), an important class of cyber-physical systems. DES models are widely used in the study of complex automated systems where the behavior is inherently event-driven, as well as in the study of discrete abstractions of continuous, hybrid, and/or cyber-physical systems. In particular, abstraction techniques that lift system dynamics from the underlying continuous-state and continuous-time domain to the domain of a discrete-state and event-driven (labeled) transition system have recently proven to be highly effective in solving control problems where the requirements take the form of a set of safety and liveness properties, expressed either in some kind of temporal logic or as regular language constraints over the higher-level event set. We therefore can capture the high level design requirements of the cyber-physical system at the level of the abstracted discrete transition system, as a set of logical constraints imposed on the discrete transition system.

In the context of DES, many properties have been studied. In the standard supervisory control problem [64], the properties under consideration are safety and non-blockingness: safety requires that the system should only execute legal behaviors (modeled in terms of a regular language); non-blockingness requires that the system should always be able to eventually achieve one in a set of desired behaviors. Diagnosability is related to the problem of fault diagnosis and isolation in automated systems and it requires that any type of fault event be diagnosed unambiguously within a bounded delay [77, 133]. In [10, 36, 70, 71, 73, 74, 107, 119, 124], a confidentiality property for partially-observed DES called opacity is studied. Opacity captures the plausible deniability of the system's "secret" in the presence of an outside observer that is potentially malicious. Anonymity is a type of opacity that is of interest in the study of privacy. Several other properties have been considered in the DES literature to capture different requirements on the behavior of the system; among them we mention predictability [29, 47, 126], detectability [88, 89], and attractability [9, 60, 78]. In the computer science literature on verification and reactive synthesis, linear temporal logic or branching time logic are also used to describe desired properties of systems; see, e.g., [21].

In many applications, the system of interest is partially observed due to the limited sensing capabilities. In this dissertation, we are concerned with the problem of enforcing a certain property on its set of behaviors for a partially-observed DES. Specifically, we investigate two different enforcement mechanisms: *supervisory control* and *dynamic sensor activation*. If the property is related to the *actual behavior* of the system, one of the most commonly-used enforcement mechanisms is to restrict the system behavior by supervisory control. The theory of supervisory control for DES was initiated by Ramadge and Wonham [64]; for this reason, it is often referred to as the Ramadge-Wonham Framework. In this framework, a supervisor disables/enables events of the system dynamically based its observations in order to restrict the system such that the closed-loop behavior satisfies some given property. In some situations, restricting the system's behavior via control may be infeasible. However, if the property is specifically related to the *observed behavior* of the system, i.e., to the strings of events output by the system, then an alternative approach is to change this output information by activating/deactivating sensors; this is the sensor activation problem. The principal objective of dissertation is the development of novel approaches for solving the control problem and the sensor activation problem for the purpose of property enforcement.

1.1 Literature Review

1.1.1 Property Enforcement via Supervisory Control

In the standard supervisory control problem, the properties to be enforced are *safety* and *non-blockingness*. This problem was solved in [64] in the case of full observation (e.g., no unobservable events). In the partial observation setting, different solutions methodologies have been proposed; see, e.g., the following original references and books [12, 20, 35, 42, 52, 83, 106]. In particular, in [20, 52], the necessary and sufficient conditions for exactly achieving a specification language were given. These are the well-known controllability, observability, and $\mathcal{L}_m(G)$-closure conditions. When the given specification language cannot be exactly achieved, one is interested in synthesizing solutions that are not only safe and non-blocking, but also *maximally permissive* in the sense that there does not exist another solution that is strictly larger and is still safe and non-blocking; in other words, such solutions are *locally maximal*. Since observability may not be preserved under union, no supremal solution exists in general, unless additional assumptions are made.

Many approaches have been considered in the literature for synthesizing safe and non-blocking supervisors for partially observed DES; see, e.g., [1,8,11,18,19,34,43,93]. One approach is to find the supremal controllable *normal* and closed sub-language, as initially defined in [20, 52]; see also, e.g., [8, 18, 41] for computational algorithms. However, since normality is stronger than observability, such a solution may be too restrictive, even empty in some cases. In [11, 93], solutions that are provably larger than the supremal controllable normal sub-language are provided. In [93], the authors identified a class of observable sub-languages that is invariant under the specifically defined "strict subautomaton union" operation. In [11], the authors identified a new language property, called relative observability, that is stronger than observability, weaker than normality, and preserved under the standard union of languages. The authors also provided an algorithm to compute the supremal controllable and relative observable sub-language. The solutions obtained by the techniques of [93] and [11] are incomparable and neither of them is maximal in general. Moreover, both techniques may return empty solutions even when non-empty solutions exist. The decidability of the problem of synthesizing a non-empty solution, i.e., a solution that is both safe and non-blocking, was established in [34]. If the decidability condition holds, in [132], the authors provided an algorithm that always returns a non-empty solution; however, the solution obtained is not maximal in general.

On-line and off-line approaches have been developed to compute maximal controllable and observable solutions when the non-blockingness requirement is relaxed, i.e., when the specification is given by a prefix-closed language; see, e.g., [4, 27, 31]. However, these approaches cannot be applied to the case where the specification is described by a non-prefix-closed language, since the resulting solutions may be blocking. In [44, 68], the computation of the infimal prefix-closed controllable and observable super-language of a given lower bound specification language was provided. Besides these, some other approaches have also been considered in the literature. In [34, 43], the use of nondeterministic supervisors was advocated. The problem of supervisor (or controller) synthesis under partial observation has also been investigated in other frameworks; see, e.g., [1, 17, 48, 61, 96]. To the best of our knowledge, the synthesis of non-blocking and safe deterministic supervisors that are maximally permissive for partially observed DES has remained an open problem. Also, how to synthesize a *maximal* safe supervisor that contains a given lower bound behavior is open.

Besides the standard supervisory control problem under partial observation, many different approaches have also been proposed to synthesize supervisors for different properties under partial observation. In [76], an integrated approach to control and diagnosis was studied. Specifically, the authors presented an approach for designing a maximally permissive supervisor that enforces diagnosability. This problem is also referred to as the *active diagnosis* problem. Several approaches have also been proposed in the literature for enforcing *opacity* of a given system that is not opaque at the outset; see, e.g., [2, 5, 23, 26, 72, 92]. In this context, the control problem is to synthesize a partial-observation supervisor that prevents behaviors that reveal the secret from occurring in the controlled system. In other words, the objective for the opacity-enforcing supervisor is to hide the system's secret in the presence of the external intruder. In [87], the author studied the problem of synthesizing a supervisor that enforces *detectability*. The enforcement of *attractability* was studied in [9, 60] for the fully-observed case and more recently in [78] under the partial observation assumption. The controller synthesis problem for *non-interference* was studied in [6].

While there is a wide literature on the enforcement of properties of DES using supervisory control, several open problems remain. First, except for the standard supervisor control problem under partial observation, all other works assume that $\Sigma_c \subseteq \Sigma_o$, where Σ_c and Σ_o are the sets of events that can be controlled and observed by the supervisor, respectively. In other words, the solutions to these property enforcement problems are only available under the assumption that all controllable events are observable. Second, all of the existing literature deals with different property-enforcing problems *separately*, i.e., each enforcement technique developed is only applicable to a specific property. Moreover, the enforcement of some properties, such as anonymity, has not yet been addressed in the literature.

1.1.2 Property Enforcement via Sensor Activation

The problem of sensor optimization in DES was initially studied in [25,30,37,129]; the goal in these works was to find an optimal set of observable events that is fixed for the entire execution of the system and enforces a given DES-theoretic property. This problem is referred to as optimal sensor selection for *static observations*. In the context of *dynamic observations*, where sensors can be turned on/off dynamically, the corresponding problem of optimal sensor activation has also received a lot of attention in the literature; see, e.g., [14–16, 22, 79, 80, 85, 86, 97, 101, 103, 104] for a sample of this work and the recent survey paper [82] for an extensive bibliography. In [16, 97], the problem of dynamic sensor activation for the purpose of fault diagnosis was studied; the optimal synthesis problems considered therein were solved according to numerical cost criteria. In [101, 104], both centralized and decentralized sensor activation problems for the purposes of control and diagnosis, respectively, were s-tudied. The features of these works are: (i) the properties of interest to be enforced are (co)observability or (co)diagnosability; (ii) the optimality criterion is logical; and (iii) the solutions are only sub-optimal in the sense that by enlarging the solution space (by refining the state space of the system), better solutions could be obtained in principle. In [103] and [85], online approaches were proposed for two different properties, observability and detectability, respectively. The complexity of synthesizing a minimal sensor activation policy for diagnosability was studied in [14]. In [16], a structure called the *Most Permissive Observer* (MPO) was proposed for solving the problem of dynamic sensor activation for the purpose of fault diagnosis. The MPO is a finite structure that embeds all valid sensor activation policies, i.e., all policies that enforce the property of K-diagnosability. Therefore, the MPO can serve as a basis for finding one optimal solution with respect to some cost criterion. This approach was extended to timed systems in [13] and to the problem of opacity in [15]. Recently, an information-state-based characterization of the MPO structure was proposed in [22]; this work shows that the size complexity of the MPO could be reduced, as compared with the original MPO from [16], by appropriately defining the notion of information state in the context of the enforcement of K-diagnosability.

In many large scale systems, the information structure is decentralized due to the distributed nature of the sensors. In the decentralized diagnosis problem considered in [24], the system is monitored by a set of local agents that work as a team in order to diagnose every occurrence of fault events. In [101], the problem of dynamic sensor activation for decentralized diagnosis is studied. Specifically, a "window-based partition" approach is proposed in order to obtain a solution. However, a drawback of this approach is that the solution obtained is only optimal w.r.t. a finite (restricted) solution space and may not be language-based optimal in general. In other words, by enlarging the solution space by refining the state space of the system model, better solutions could be obtained in principle. In [104], the problem of dynamic sensor activation for decentralized control is also studied, where the solution obtained is again optimal w.r.t. a finite solution space. To the best of our knowledge, the problems of *language-based* sensor optimization for *decentralized* diagnosis and decentralized control have remained open problems, as is mentioned in the recent survey [82].

1.2 Organization and Main Contributions

The main contributions and the organization of this dissertation are summarized as follows.

Chapter II: A Uniform Approach for Property Enforcement via Supervisory Control ([109, 114, 121])

In this chapter, we propose a *uniform approach* that is applicable to the enforcement, by supervisory control, of a large class of properties that can be expressed in terms of suitably-defined *information states*. We refer to such properties as *information-state-based* (or IS-based) properties. Roughly speaking, an IS-based property is a property that only depends on the current local information of the system, as available to the supervisor, and does not explicitly depend on the future behavior of the system. Specifically, our approach is based on the construction of a finite information structure called the All Enforcement Structure (AES). By construction, the AES embeds in its structure *all* property-enforcing supervisors. Therefore, it can serve as the basis for solving the synthesis problem.

Chapter III: Synthesis of Non-Blocking Supervisors for IS-Based Properties ([110, 120])

In this chapter, we tackle the supervisor synthesis problem for non-blockingness *in addition to* the enforcement of an IS-based property. We define another finite bipartite transition system that we call the Non-blocking All Enforcement Structure (NB-AES). We then provide a synthesis algorithm, based on the NB-AES, that constructs a non-blocking and maximally permissive supervisor that enforces an IS-based property, if one exists. This is the first algorithm with such properties and it answers a long standing open problem that was unsolved for more than 25 years.

Chapter IV: The Range Control Problem ([117, 118, 125])

In this chapter, we study a generalized supervisor synthesis problem called the *Maximally Permissive Range Control Problem.* In this problem, we not only want to find a locally maximal supervisor, but we also require that the synthesized maximal supervisor contain a given behavior. We only restrict our attention to the safety requirement and not do consider the issue of blockingness. More specifically, we consider two specification languages: the safety specification language K, which is also referred to as the upper bound language, and a prefix-closed lower bound language $R \subseteq K$, which models the required behavior that the closed-loop system must achieve. To solve the range control problem, we present a new synthesis algorithm based on the two notions of AES and Control Simulation Relation (CSR). Although we only consider prefix-closed languages, i.e., nonblockingness is not considered, to the best of our knowledge, the maximally-permissive range control problem we solve herein was an open problem even in this case.

Chapter V: A Uniform Approach for Centralized Sensor Activation ([112, 116])

In this chapter, we consider the problem of dynamic sensor activation in centralized and partially-observed DES. The objective in this problem is to synthesize a sensor activation policy that dynamically turns sensors on/off online in order to achieve a given objective, e.g., to control the system or to diagnose faults. This problem is important since in many applications turning more sensors on implies that more energy or bandwidth is consumed. Therefore, it is of interest to synthesize a sensor activation policy that is optimal with respect to some criterion, subject to the constraints of the problem. To solve this problem, we define a generalized version of the Most Permissive Observer (MPO). This generalized MPO embeds all valid solutions to the enforcement of an IS-based property in its finite structure. Based on the MPO, we present an algorithm for the synthesis of optimal sensor activation policies under a logical performance objective.

Chapter VI: Sensor Activation in Decentralized Decision-Making ([113, 123])

In this chapter, we investigate the problem of decentralized decision-making in DES that operate under dynamic observations. In this context, the system is monitored by a set of agents that act as a team to make a global decision. The sensors of each agent can be turned on/off online dynamically according to a sensor activation policy. We define a general decentralized decision-making problem called the decentralized state disambiguation problem, which covers the decentralized control problem, the decentralized fault diagnosis problem, and the decentralized fault prognosis problem. The goal is to find a *language-based* minimal sensor activation policy for each agent such that the agents can always make a correct global decision as a team. A novel approach to solve this problem is proposed. We adopt a *person-by-person* approach to decompose this decentralized minimization problem into two centralized constrained minimization problems. Each centralized constrained minimization problem is then reduced to the centralized sensor activation problem that we solve in Chapter V.

We prove that the solution obtained by our procedure is minimal w.r.t. the system language, in contrast to the works in the literature where minimality was with respect to a finite solution space.

Chapter VII: Conclusion and Future Work

We conclude the dissertation and discuss several potential future directions.

CHAPTER II

A Uniform Approach for Property Enforcement via Supervisory Control

2.1 Introduction

In this chapter, we propose a *uniform approach* that is applicable to the enforcement, by supervisory control, of a large class of properties that can be expressed in terms of suitably-defined *information states*. We refer to such properties as information-state-based (IS-based) properties. Roughly speaking, an IS-based property is a property that only depends on the current local information of the system, as available to the supervisor, and does not explicitly depend on the future behavior of the system. The approach that we develop to tackle this problem is significantly different from the previous approaches in the literature, which are also concerned with property enforcement by supervisory control. Specifically, our approach is based on the construction of a finite information structure called the *All Enforcement Structure* and abbreviated as AES. The AES is a game structure between the supervisor and the "environment" (aka system). By construction, the AES embeds in its structure *all* property-enforcing supervisors. Therefore, it can serve as the basis for solving the synthesis problem.

The single *uniform* solution methodology is applicable not only to safety and opacity, but to any property that can be expressed as an IS-based property. This includes, but is not restricted to, safety, diagnosability, opacity, detectability, anonymity and attractability. There are properties that cannot be formulated as IS-based properties; the prime example is non-blockingness, which will be addressed in the next chapter. In Table 2.1, we compare our proposed approach with previous work. To the best of our knowledge, the problem of synthesizing a maximally permissive supervisor that enforces anonymity has not yet been considered in the literature; this property can be enforced by our general methodology. Moreover, we relax the assumption made by previous works that all controllable events should be observable. We show that, in this more general setting, uniqueness of a maximally permissive solution is lost. Hence, our focus is on the synthesis of solutions that are provably (locally) maximally permissive.

The rest of this chapter is organized as follows. In Section 2.2, we present the model of the system to be analyzed. In Section 2.3, we formulate the information-state-based property enforcement problem that we solve in this chapter. In Section 2.4, we define a class of bipartite transition systems that is used for solving the property enforcement problem. In Section 2.5, we define the structure called AES, the key notion for the approach investigated in this chapter. We then present a general-purpose synthesis algorithm that returns a maximally-permissive partial-observation supervisor based on the AES in Section 2.6. In Section 2.7, we show how the proposed uniform approach can be applied to enforce different specific properties. Finally, we conclude this chapter in Section 2.8.

2.2 Preliminary

In this section, we review some basic concepts and notations that will be used in this chapter. Let Σ be a finite set of events and denote by Σ^* the set of all finite strings over Σ, including the empty string ϵ. A language $L \subseteq \Sigma^*$ is a subset of Σ^*.

Table 2.1: Comparison between the proposed uniform approach and previous approaches

Property	Safety	Opacity	Diagnosability	Detectability	Anonymity	Attractability
Previous Works	[4, 18, 20, 52]	[2, 5, 26, 72]	[76]	[87]	None	[78]
Previous Assumptions	None	$\Sigma_a \subseteq \Sigma_o$, $\Sigma_c \subseteq \Sigma_o$[1]	$\Sigma_c \subseteq \Sigma_o$	$\Sigma_c \subseteq \Sigma_o$	N/A	$\Sigma_c \subseteq \Sigma_o$
Assumptions in this chapter	None	$\Sigma_a = \Sigma_o$	None	None	$\Sigma_a = \Sigma_o$	None

[1] Σ_c, Σ_o and Σ_a are, respectively, the set of events that can be controlled by the supervisor, the set of events that can be observed by the supervisor, and the set of events that can be observed by the external observer.

The prefix closure of language L is the set $\overline{L} = \{t \in \Sigma^* : \exists u \in \Sigma^* \text{ s.t. } tu \in L\}$. We say that a language is *prefix-closed* if $L = \overline{L}$. Given language L and string $s \in L$, we denote the active (event) set at s in L by $\Delta_L(s) = \{\sigma \in \Sigma : s\sigma \in L\}$ and use $L/s = \{t \in \Sigma^* : st \in L\}$ to denote the set of continuations of s in L. For any string $s \in \Sigma^*$, we denote by $|s|$ the length of s with $|\epsilon| = 0$. For any $\sigma \in \Sigma, s \in \Sigma^*$, we use $\sigma \in s$ to denote that σ occurs at least once in s.

The DES of interest is modeled as a deterministic finite-state automaton

$$\mathbf{G} = (X, \Sigma, \delta, x_0, X_m), \tag{2.1}$$

where X is the finite set of states, Σ is the finite set of events, $\delta : X \times \Sigma \to X$ is the partial transition function, where $\delta(x, \sigma) = y$ means that there is a transition labeled by event σ from state x to state y, $x_0 \in X$ is the initial state and X_m is the set of marked states. The transition function δ is extended to $X \times \Sigma^*$ recursively by: $\delta(x, s\sigma) = \delta(\delta(x, s), \sigma)$, where $x \in X$, $s \in \Sigma^*$ and $\sigma \in \Sigma$. For brevity, we also write $\delta(x_0, s)$ as $\delta(s)$. The language generated by $\mathbf{G}$ is described by $\mathcal{L}(\mathbf{G}) = \{s \in \Sigma^* : \delta(x_0, s)!\}$, where ! means "is defined"; the marked language is $\mathcal{L}_m(\mathbf{G}) = \{s \in \Sigma^* : \delta(x_0, s) \in X_m\}$.

Given two automata $\mathbf{A} = (X_A, \Sigma, \delta_A, x_{A,0})$ and $\mathbf{B} = (X_B, \Sigma, \delta_B, x_{B,0})$, we say that $\mathbf{A}$ is a sub-automaton of $\mathbf{B}$, denoted by $\mathbf{A} \sqsubseteq \mathbf{B}$, if $\delta_A(x_{A,0}, s) = \delta_B(x_{B,0}, s)$ for all $s \in \mathcal{L}(\mathbf{A})$. We say that $\mathbf{A}$ is a *strict* sub-automaton [1] of $\mathbf{B}$, denoted by $\mathbf{A} \sqsubset \mathbf{B}$, if (i) $\mathbf{A} \sqsubseteq \mathbf{B}$; and (ii) $\forall x, y \in X_A, \forall s \in \Sigma^* : \delta_B(x, s) = y \Rightarrow \delta_A(x, s) = y$. Note that, for any two automata $\mathbf{A}$ and $\mathbf{B}$ such that $\mathcal{L}(\mathbf{A}) \subseteq \mathcal{L}(\mathbf{B})$, we can always refine the state spaces of $\mathbf{A}$ and $\mathbf{B}$ and obtain two new automata $\mathbf{A}'$ and $\mathbf{B}'$ such that $\mathcal{L}(\mathbf{A}') = \mathcal{L}(\mathbf{A})$, $\mathcal{L}(\mathbf{B}') = \mathcal{L}(\mathbf{B})$ and $\mathbf{A}' \sqsubset \mathbf{B}'$.

In the framework of supervisory control [64], the system $\mathbf{G}$ is controlled by a

[1]The definition of strict sub-automaton used in this dissertation is slightly stronger than the definition used in [18].

supervisor that dynamically enables/disables events of the system such that some specification is provably achieved. The event set Σ is partitioned into two disjoint subsets: $\Sigma = \Sigma_c \dot{\cup} \Sigma_{uc}$, where Σ_c is the set of controllable events and Σ_{uc} is the set of uncontrollable events. We say that a control decision $\gamma \in 2^{\Sigma}$ is admissible if $\Sigma_{uc} \subseteq \gamma$, namely, uncontrollable events can never be disabled. We define $\Gamma = \{\gamma \in 2^{\Sigma} : \Sigma_{uc} \subseteq \gamma\}$ as the set of admissible control decisions. When the system is partially observed [20, 52], Σ is also partitioned into two disjoint sets: $\Sigma = \Sigma_o \dot{\cup} \Sigma_{uo}$, where Σ_o is the set of observable events and Σ_{uo} is the set of unobservable events. The natural projection $P : \Sigma^* \to \Sigma_o^*$ is defined by

$$P(\epsilon) = \epsilon \text{ and } P(s\sigma) = \begin{cases} P(s)\sigma & \text{if } \sigma \in \Sigma_o \\ P(s) & \text{if } \sigma \in \Sigma_{uo} \end{cases} \tag{2.2}$$

P is extended to $P : 2^{\Sigma^*} \to 2^{\Sigma_o^*}$ by $P(L) = \{t \in \Sigma_o^* : \exists s \in L \text{ s.t. } P(s) = t\}$, where $L \subseteq \Sigma^*$. The inverse projection $P^{-1} : \Sigma_o^* \to 2^{\Sigma^*}$ is defined by $P^{-1}(t) := \{s \in \Sigma^* : P(s) = t\}$; P^{-1} is also extended to $P^{-1} : 2^{\Sigma_o^*} \to 2^{\Sigma^*}$ by $P^{-1}(s) = \{s \in \Sigma^* : \exists t \in L \text{ s.t. } P(s) = t\}$, where $L \subseteq \Sigma_o^*$. Since a supervisor can only make decisions based on its observations, a partial-observation supervisor is a function $S : P(\mathcal{L}(\mathbf{G})) \to \Gamma$. We use the notation $S/\mathbf{G}$ to represent the controlled system and the language generated by $S/\mathbf{G}$, denoted by $\mathcal{L}(S/\mathbf{G})$, is defined recursively as follows:

i) $\epsilon \in \mathcal{L}(S/\mathbf{G})$; and

ii) $[s \in \mathcal{L}(S/\mathbf{G}) \wedge s\sigma \in \mathcal{L}(G) \wedge \sigma \in S(s)] \Leftrightarrow [s\sigma \in \mathcal{L}(S/\mathbf{G})]$.

We also define $\mathcal{L}_m(S/\mathbf{G}) = \mathcal{L}(S/\mathbf{G}) \cap \mathcal{L}_m(\mathbf{G})$.

Given a prefix-closed language $\overline{K} = K$, we say that K is *controllable* (w.r.t. $\mathcal{L}(\mathbf{G})$ and Σ_c) if $(\forall s \in K, \sigma \in \Sigma_{uc})(s\sigma \in \mathcal{L}(\mathbf{G}) \Rightarrow s\sigma \in K)$; we say that K is *observable* (w.r.t. $\mathcal{L}(\mathbf{G})$, Σ_c and Σ_o) if $(\forall s, s' \in K, \sigma \in \Sigma_c)(P(s) = P(s') \wedge s\sigma \in K \wedge s'\sigma \in \mathcal{L}(\mathbf{G}) \Rightarrow s'\sigma \in K)$; we say that K is *normal* (w.r.t. $\mathcal{L}(\mathbf{G})$ and Σ_o) if $\overline{K} = P^{-1}[P(\overline{K})] \cap \mathcal{L}(\mathbf{G})$. It is well known that there exists a supervisor S such that $\mathcal{L}(S/\mathbf{G}) = K$ if and only if K is controllable and observable [20,52]. In general, observability is not preserved under union, unless additional assumptions are made. For instance, it was shown in [54] that if $\Sigma_c \subseteq \Sigma_o$, then controllability and observability together imply normality, which is preserved under union. However, this assumption is not required here.

We define several operators that will be used later. The set of *all possible states* in $\mathbf{G}$ reachable from the initial state x_0 via some string in sublanguage $L \subseteq \mathcal{L}(\mathbf{G})$ with the same projection as $s \in L$, is given by

$$R_{\mathbf{G}}(s, L) := \{x \in X : \exists t \in L \text{ s.t. } P(t) = P(s) \wedge x = \delta(x_0, t)\} \tag{2.3}$$

The *unobservable reach* of the subset of states $S \subseteq X$ under the subset of events $\gamma \subseteq \Sigma$ is given by

$$\mathrm{UR}_\gamma(S) := \{x \in X : \exists u \in S, \exists \sigma \in (\Sigma_{uo} \cap \gamma)^* \text{ s.t. } x = \delta(u, \sigma)\}. \tag{2.4}$$

The *observable reach* of the subset of states $S \subseteq X$ under observable event $\sigma \in \Sigma_o$ is given by

$$\mathrm{Next}_\sigma(S) := \{x \in X : \exists u \in S \text{ s.t. } x = \delta(u, \sigma)\}. \tag{2.5}$$

2.3 Problem Formulation

In this section, we define the class of information-state-based properties and formulate the *Property Enforcement Problem* that we solve in this chapter.

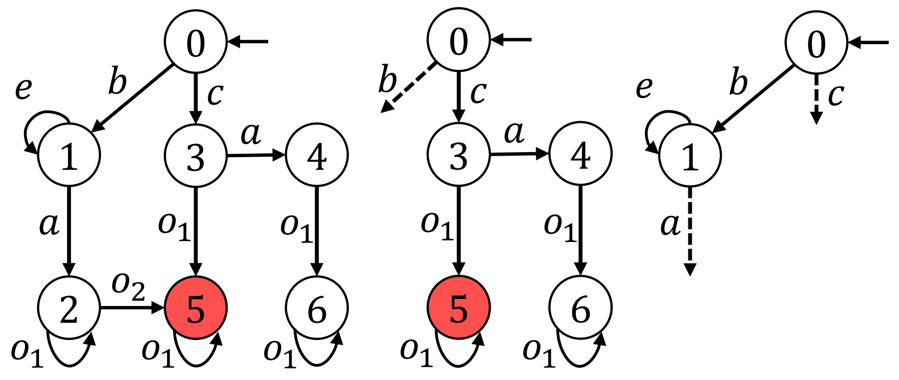
In many applications, we are interested in enforcing some properties on the system behavior via supervisory control. In general, a property on a language (or a language-based property) is a function $\varphi : 2^{\Sigma^*} \to \{0, 1\}$, where for any language L, $\varphi(L) = 1$ means that L satisfies property φ. We write that $L \models \varphi$ if φ is a language-based property and $\varphi(L) = 1$. For example, safety is a typical property of interest. Let $K \in 2^{\Sigma^*}$ be a specification language. Then the safety property $\varphi : 2^{\Sigma^*} \to \{0, 1\}$ can be defined by: for any $L \in 2^{\Sigma^*}$, $\varphi(L) = 1 \Leftrightarrow L \subseteq K$. To enforce a given property on the system, we need to synthesize a supervisor that restricts the system behavior to a sublanguage that satisfies the property; moreover, it is desired that this sublanguage be as large as possible w.r.t. set inclusion. In other words, the supervisor should only disable an event if it is necessary to do so. By considering a general property instead of only safety, the standard supervisory control problem under partial observation [20,52] is generalized to the *Maximally Permissive Property Enforcement Problem* defined as follows.

Problem 1. *(Maximally Permissive Property Enforcement Problem). Given system $\boldsymbol{G}$ and language-based property $\varphi : 2^{\Sigma^*} \to \{0, 1\}$, synthesize a partial observation supervisor $S : P(\mathcal{L}(\boldsymbol{G})) \to \Gamma$, such that*

1. *$\mathcal{L}(S/\boldsymbol{G}) \models \varphi$;*
2. *For any S' satisfying 1) and 2), we have that $\mathcal{L}(S/\boldsymbol{G}) \not\subset \mathcal{L}(S'/\boldsymbol{G})$.*

In the formulation of the above problem, the property of interest is defined over languages; hence, it may not be possible to bound a priori the memory needed for its verification. To simplify our problem, hereafter, we will investigate a particular class of properties called *Information-State-based* (IS-based) properties. Since we are dealing with partially observed systems, we define the notion of an *information state* as a subset $IS \subseteq X$ of states of $\mathbf{G}$ and denote by $I = 2^X$ the set of all information states. IS-based properties are defined as follows.

Definition 2.3.1. *(IS-Based Property). Given an automaton $\boldsymbol{G}$, an IS-based property φ w.r.t. $\boldsymbol{G}$ is a function $\varphi : I \to \{0, 1\}$, where $\forall i \in 2^X$, $\varphi(i) = 1$ means that i satisfies this property. We say that sublanguage $L \subseteq \mathcal{L}(\boldsymbol{G})$ satisfies φ w.r.t. $\boldsymbol{G}$, which is denoted by $L \models_G \varphi$, if $\forall s \in L : \varphi(R_G(s, L)) = 1$.*



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

Figure 2.1: For $\mathbf{G}$: $\Sigma_c = \{a, b, c\}, \Sigma_o = \{o_1, o_2\}$, and $X_S = \{5\}$

We will show later in Section 2.7 that by some proper state space refinements, many important properties in the DES literature, e.g., safety, diagnosability, opacity, detectability and attractability, can be formulated as IS-based properties.

Example 2.3.1. *Let us consider the system* $\boldsymbol{G}$ *in Figure 2.1(a), where the set of observable events is* $\Sigma_o = \{o_1, o_2\}$. *Consider the subset of states* $X_S = \{5\}$. *We define the IS-based property* $\varphi : I \rightarrow \{0, 1\}$ *by*

$$\varphi(i) = 0 \Leftrightarrow i \subseteq X_S \tag{2.6}$$

$\forall i \in I$. *We will show later in Section 2.7 that the IS-based property defined above essentially captures a security property called* current-state opacity. *One may interpret* X_S *as the set of secret states that the system wants to hide from a potentially malicious external observer, referred to as the intruder. We say that property* φ *holds if the intruder can never determine unambiguously that the secret has occurred based on its observation capabilities. If the intruder's observable set is* $\Sigma_o = \{o_1, o_2\}$, *then the system language* $\mathcal{L}(\boldsymbol{G})$ *does not satisfy* φ, *since* $R_G(bao_2, \mathcal{L}(\boldsymbol{G})) = \{5\} \subseteq X_S$, *i.e., upon the occurrence of string* bao_2, *the secret state* 5 *will be revealed to the intruder.*

Similarly to the property enforcement problem, we formulate the Maximally Permissive IS-Based Property Enforcement Problem (MPIEP) as follows.

Problem 2. *(Maximally Permissive IS-Based Property Enforcement Problem). Given system $\boldsymbol{G}$ and IS-based property $\varphi : 2^X \to \{0,1\}$ w.r.t. $\boldsymbol{G}$, synthesize a partial observation supervisor $S : P(\mathcal{L}(\boldsymbol{G})) \to \Gamma$, such that*

1. $\mathcal{L}(S/\boldsymbol{G}) \models_{\boldsymbol{G}} \varphi$;

2. *For any S' satisfying 1) and 2), we have that $\mathcal{L}(S/\boldsymbol{G}) \not\subset \mathcal{L}(S'/\boldsymbol{G})$.*

We will show in Section 2.6 that under the assumption that $\Sigma_c \subseteq \Sigma_o$, there always exists a unique *supremal* solution to MPIEP. However, this is not true in general, as illustrated in the following example.

Example 2.3.2. *Consider again the system $\boldsymbol{G}$ in Figure 2.1(a). Let the set of controllable events be $\Sigma_c = \{a, b, c\}$, which is incomparable with Σ_o. To enforce property φ defined in Example 2.3.1, we need to find a controllable, observable, and live sublanguage of $\mathcal{L}(\boldsymbol{G})$ that satisfies φ. It is easy to verify that solutions $\mathcal{L}(\boldsymbol{G}_1)$ and $\mathcal{L}(\boldsymbol{G}_2)$, shown in Figure 2.1(b) and Figure 2.1(c), respectively, are two maximal controllable and observable solutions satisfying φ. However, the union of these two solutions is not a valid solution, since the system needs to enable event a at state 1 and to disable event a at state 3; but states 1 and 3 are indistinguishable in $\mathcal{L}(\boldsymbol{G}_1) \cup \mathcal{L}(\boldsymbol{G}_2)$. This violates the property of observability.*

2.4 Bipartite Transition System

In this section, we define the general notion of Bipartite Transition System (BTS).

Definition 2.4.1. *A bipartite transition system T w.r.t. $\boldsymbol{G}$ is a 7-tuple*

$$T = (Q_Y^T, Q_Z^T, h_{YZ}^T, h_{ZY}^T, \Sigma, \Gamma, y_0) \tag{2.7}$$

where

- $Q_Y^T \subseteq I$ *is the set of* Y*-states;*
- $Q_Z^T \subseteq I \times \Gamma$ *is the set of* Z*-states and* $I(z)$ *and* $\Gamma(z)$ *denote, respectively, the information state and the control decision components of a* Z*-state* z*, so that* $z = (I(z), \Gamma(z))$*;*
- $h_{YZ}^T : Q_Y^T \times \Gamma \rightarrow Q_Z^T$ *is the partial transition function from* Y*-states to* Z*-states, which satisfies the following constraint: for any* $y \in Q_Y^T, z \in Q_Z^T$ *and* $\gamma \in \Gamma$*, we have*
$$h_{YZ}^T(y, \gamma) = z \Rightarrow [I(z) = UR_\gamma(y)] \wedge [\Gamma(z) = \gamma] \tag{2.8}$$
- $h_{ZY}^T : Q_Z^T \times \Sigma \rightarrow Q_Y^T$ *is the partial transition function from* Z*-states to* Y*-states, which satisfies the following constraint: for any* $y \in Q_Y^T, z \in Q_Z^T$ *and* $\sigma \in \Sigma$*, we have*
$$h_{ZY}^T(z, \sigma) = y \Leftrightarrow [\sigma \in \Gamma(z) \cap \Sigma_o] \wedge [y = Next_\sigma(I(z))] \tag{2.9}$$
- Σ *is the set of events of* $\boldsymbol{G}$*;*
- Γ *is the set of admissible control decisions of* $\boldsymbol{G}$*;*
- $y_0 \in Q_Y^T$ *is the initial* Y*-state where* $y_0 = \{x_0\}$*.*

The purpose of defining the notion of BTS is to describe, in a general manner that will be specialized later, the "game" between the "supervisor/controller" and the "system/environment" (**G**). To capture this game, we need a bipartite structure, with two types of nodes (states). A Y-state is an information state, from which the supervisor issues control decisions. A Z-state is an information state augmented with control decisions, from which the system "selects" observable events to occur within the set of enabled events. A transition from a Z-state to a Y-state represents the observable reach, i.e, y in the above definition is the set of states reachable from some state of the information state component of the preceding Z-state through the single observed event just executed by $\mathbf{G}$. A transition from a Y-state to a Z-state represents the unobservable reach and "remembers" the set of enabled events from the Y-state that leads to it. This means that $I(z)$ is the set of states reachable from some state in the preceding Y-state through some enabled unobservable event string, and that $\Gamma(z)$ is the control decision made in the preceding Y-state. Since the supervisor cannot choose which event will occur once it has made a control decision, all enabled and feasible observable events should be defined at a Z-state; this is why we put "$\Leftrightarrow$" in Equation (2.9). Finally, we say that a Z-state z is *terminal* if $(\forall x \in I(z))(\forall \sigma \in \Sigma_o \in \Gamma(z))[\delta(x,\sigma)\neg!]$.

Example 2.4.1. *Consider again the system* $\boldsymbol{G}$ *in Figure 2.1(a). As an example of a BTS, the reader is referred directly to Figure 2.2(b), which is a particular type of BTS that we will discuss later in this chapter. From the initial* Y*-state* $y_0 = \{0\}$*, by making control decision* $\gamma = \{a, c, o_1, o_2\}$ *(the uncontrollable events* o_1 *and* o_2 *are omitted in the figure), we will reach* Z*-state* $z = h^T_{YZ}(y_0, \gamma) = (\{0,3,4\}, \{a, c, o_1, o_2\})$*. From* z*, only one observable event,* o_1*, can happen, and it leads to the next* Y*-state* $y_1 = h^T_{ZY}(z, o_1) = \{5, 6\}$*. This BTS includes another control decision at* $y_0 = \{0\}$*,* $\gamma = \{b, o_1, o_2\}$*, from which no observation will occur. Finally, at* Y*-state* $\{5, 6\}$*, this BTS includes a single control decision, where only the uncontrollable events are included.*

Given two BTSs T_1 and T_2, we say that T_1 is a *subsystem* of T_2, denoted by $T_1 \sqsubseteq T_2$, if $Q^{T_1}_Y \subseteq Q^{T_2}_Y, Q^{T_1}_Z \subseteq Q^{T_2}_Z$, and for any $y \in Q^{T_1}_Y, z \in Q^{T_1}_Z, \gamma \in \Gamma$, and $\sigma \in \Sigma$, we have that

1) $h^{T_1}_{YZ}(y,\gamma) = z \Rightarrow h^{T_2}_{YZ}(y,\gamma) = z$; and

2) $h^{T_1}_{ZY}(z,\sigma) = y \Rightarrow h^{T_2}_{ZY}(z,\sigma) = y$.

For example, we see that the BTS in Figure 2.2(b) is a subsystem of the BTS in Figure 2.2(a). Also, the union of T_1 and T_2 is defined as a BTS $T_1 \cup T_2$ such that:

$Q_Y^{T_1 \cup T_2} = Q_Y^{T_1} \cup Q_Y^{T_2}, Q_Z^{T_1 \cup T_2} = Q_Z^{T_1} \cup Q_Z^{T_2}$; and for any $y \in Q_Y^{T_1 \cup T_2}, z \in Q_Z^{T_1 \cup T_2}, \gamma \in \Gamma$ and $\sigma \in \Sigma$, we have that

1) $h_{YZ}^{T_1 \cup T_2}(y, \gamma) = z \Leftrightarrow \exists i \in \{1, 2\} : h_{YZ}^{T_i}(y, \gamma) = z$; and

2) $h_{ZY}^{T_1 \cup T_2}(z, \sigma) = y \Leftrightarrow \exists i \in \{1, 2\} : h_{ZY}^{T_i}(z, \sigma) = y$.

In general, the control decision defined at a Y-state may not be unique. Therefore, given a BTS T, we define $C_T(y) := \{\gamma \in \Gamma : h_{YZ}^T(y, \gamma)!\}$ to be the set of control decisions defined at $y \in Q_Y^T$. For simplicity, we also write $y \xrightarrow{\gamma}_T z$ if $z = h_{YZ}^T(y, \gamma)$ and $z \xrightarrow{\sigma}_T y$ if $z = h_{ZY}^T(z, \sigma)$. Note that, for two BTSs T_1 and T_2, we have that $h_{YZ}^{T_1}(y, \gamma) = h_{YZ}^{T_2}(y, \gamma)$ whenever they are defined. Therefore, we will drop the superscript in $h_{YZ}^T(y, \gamma)$ and write it as $h_{YZ}(y, \gamma)$ and $y \xrightarrow{\gamma} z$ if it is defined for some T; the same holds for h_{ZY} and $z \xrightarrow{\sigma} y$. We call $\gamma_0\sigma_1\gamma_1\sigma_2 \ldots \sigma_n\gamma_n$, where $\gamma_i \in \Gamma, \sigma_i \in \Sigma_o$, a *run*. A run also induces a *sequence*

$$y_0 \xrightarrow{\gamma_0} z_0 \xrightarrow{\sigma_1} y_1 \xrightarrow{\gamma_1} \ldots \xrightarrow{\gamma_{n-1}} z_{n-1} \xrightarrow{\sigma_n} y_n \xrightarrow{\gamma_n} z_n$$

We say that a run is generated by T if its induced sequence is defined in T.

We say that a BTS T is

- *complete*, if $\forall y \in Q_Y^T : C_T(y) \neq \emptyset$; and
- *deterministic*, if $\forall y \in Q_Y^T : |C_T(y)| = 1$.

If T is deterministic, then we also use notation $c_T(y)$ to denote the unique control decision defined at $y \in Q_Y^T$, i.e., $C_T(y) = \{c_T(y)\}$. The completeness condition says that for any Y-state, we need to be able to pick at least one control decision; the determinism condition says that such a control decision is unique at each Y-state.

Let $S : P(\mathcal{L}(\mathbf{G})) \to \Gamma$ be a partial observation supervisor. It works as follows. Initially, it makes control decision $S(\epsilon)$. Then new control decision $S(\sigma)$ is made upon the occurrence of (enabled) observable event σ, and so forth. Let $s = \sigma_1 \dots \sigma_n \in P(\mathcal{L}(S/\mathbf{G}))$ be an observed string. Then the execution of s induces a well-defined sequence

$$y_0 \xrightarrow{S(\epsilon)} z_0 \xrightarrow{\sigma_1} y_1 \xrightarrow{S(\sigma_1)} \dots \xrightarrow{\sigma_n} y_n \xrightarrow{S(\sigma_1 \dots \sigma_n)} z_n$$

We denote by $IS_S^Y(s)$ and $IS_S^Z(s)$, the last Y-state and Z-state in $y_0 z_0 y_1 z_2 \dots z_{n-1} y_n z_n$, respectively, i.e., $IS_S^Y(s) = y_n$ and $IS_S^Z(s) = z_n$. That is, $IS_S^Y(s)$ and $IS_S^Z(s)$ are the Y-state and the Z-state that result from the occurrence of string s under supervisor S, respectively.

The next result states that given a supervisor S and a string $s \in \mathcal{L}(S/\mathbf{G})$, the Z-state defined above is, in fact, equivalent to the set of all possible states the system can be in after observing $P(s)$.

Lemma 2.4.1. *Given a system $\boldsymbol{G}$ and a supervisor S, for any string $s \in \mathcal{L}(S/\boldsymbol{G})$, we have*

$$I(IS_S^Z(P(s))) = R_{\boldsymbol{G}}(s, \mathcal{L}(S/\boldsymbol{G})). \tag{2.10}$$

Proof. We prove by induction on the length of $P(s)$. For any string s, let $|P(s)| = n$. Let s_k denote the string that consists of the first k events in $P(s)$ for $k = 0, \dots, n$ and e_k denote the $(k+1)^{th}$ event in $P(s)$ for $k = 0, \dots, n-1$, so that $s_0 = \epsilon$, $s_1 = e_0$, etc.. . Define y_0 as usual. For $k = 0, \dots, n$, let $z_k = h_{YZ}(y_k, S(s_k))$, and for $k = 0, \dots, n-1$, define $y_{k+1} = h_{ZY}(z_k, e_k)$. By definition, we know that

$$R_{\mathbf{G}}(s, \mathcal{L}(S/\mathbf{G})) = \{v \in X : \exists s' \in \mathcal{L}(S/\mathbf{G}) \text{ s.t. } P(s) = P(s') \wedge v = \delta(x_0, s')\} \tag{2.11}$$

Therefore, the inductive hypothesis is that:

$$I(z_k) = \{v \in X : \exists s'_k \in \mathcal{L}(S/\mathbf{G}) \text{ s.t. } P(s'_k) = s_k \wedge v = \delta(x_0, s'_k)\} \tag{2.12}$$

Induction Basis: $s_0 = \epsilon$

$$\begin{aligned} I(z_0) = \mathrm{UR}_{S(\epsilon)}(y_0) &= \{v \in X : \exists t \in (S(\epsilon) \cap \Sigma_{uo})^* \text{ s.t. } v = \delta(x_0, t)\} \\ &= \{v \in X : \exists t \in \mathcal{L}(S/\mathbf{G}) \text{ s.t. } P(t) = \epsilon \wedge v = \delta(x_0, t)\} \end{aligned}$$

Induction Step: Assume that the induction hypothesis is true at k. Then

$$\begin{aligned} y_{k+1} &= h_{ZY}(z_k, e_k) \\ &= \{v \in X : \exists u \in I(z_k) \text{ s.t. } v = \delta(u, e_k)\} \\ &= \{v \in X : \exists s'_k \in \mathcal{L}(S/\mathbf{G}) \text{ s.t.} P(s'_k) = s_k \wedge v = \delta(x_0, s'_k e_k)\} \end{aligned} \tag{2.13}$$

and

$$\begin{aligned} I(z_{k+1}) &= \mathrm{UR}_{S(s_k e_k)}(y_{k+1}) \\ &= \{v \in X : \exists u \in y_{k+1}, \exists t \in (S(s_k e_k) \cap \Sigma_{uo})^* \text{ s.t. } v = \delta(u, t)\} \\ &= \{v \in X : \exists s'_{k+1} \in \mathcal{L}(S/\mathbf{G}) \text{ s.t. } P(s'_{k+1}) = s_k e_k \wedge v = \delta(x_0, s'_{k+1})\} \end{aligned}$$

This completes the proof. □

By the above Lemma and the definition of $\mathcal{L}(S/\mathbf{G})$, we can also express $IS_S^Y(s)$ and the information state component of $IS_S^Z(s)$ as follows

$$\begin{aligned} IS_S^Y(s) &= \{x \in X : \exists t \in (\Sigma^* \Sigma_o \cup \{\epsilon\}) \cap \mathcal{L}(S/\mathbf{G}) \text{ s.t. } P(t) = s \wedge \delta(x_0, t) = x\} \\ I(IS_S^Z(s)) &= \{x \in X : \exists t \in \mathcal{L}(S/\mathbf{G}) \text{ s.t. } P(t) = s \wedge \delta(x_0, t) = x\} \end{aligned}$$

With the above notions, we can "decode" supervisors from a BTS as explained in the following definition.

Definition 2.4.2. *A supervisor S is said to be included in a complete BTS T if*

$$(\forall s \in P(\boldsymbol{\mathcal{L}}(S/\boldsymbol{G})))[S(s) \in C_T(IS_S^Y(s))]. \tag{2.14}$$

We denote by $\mathbb{S}(T)$ the set of all supervisors included in T.

Example 2.4.2. *The BTS shown in Figure 2.2(b) is a complete BTS. By picking control decision $\{b, o_1, o_2\}$ (shown as $\{b\}$ in the figure) at the initial Y-state $\{0\}$, no future observable behavior can occur, since the only enabled and feasible event e forms an unobservable self-loop at state 1. This leads to a BTS-included supervisor S defined by $S(\epsilon) = \{b, o_1, o_2\}$.*

If a BTS T is deterministic, then the supervisor included in T is unique, since the control decision at each Y-state is unique. In this case, we denote by S_T the unique supervisor included in T, i.e., $\mathbb{S}(T) = \{S_T\}$. Essentially, T is a *realization* of supervisor S_T. However, not all supervisors can be realized by a BTS, since a supervisor may make different control decisions at different visits to the same Y-state. We say that a supervisor S is *information-state-based* (IS-based) if

$$\forall s, t \in P(\mathcal{L}(S/\mathbf{G})) : IS_S^Y(s) = IS_S^Y(t) \Rightarrow S(s) = S(t).$$

Then, a supervisor can be realized by a BTS iff it is IS-based.

2.5 A Uniform Approach for Enforcing Properties

In this section, we define the All Enforcement Structure, a specific type of BTS that embeds all supervisors that enforce a given IS-based property in its transition structure. We then discuss its properties and its construction.

2.5.1 All Enforcement Structure for a Given Property

In Lemma 2.4.1, we have shown that given a supervisor S, for any string $s \in P(\mathcal{L}(\mathbf{G}))$, the Z-state $IS_S^Z(s)$ reached is the set of all possible states the system could be in after s. As a consequence, if we construct a BTS that is "as large as possible" and in which all reachable Z-states satisfy the IS-based property, then the resulting structure should contain all valid property-enforcing supervisors. This leads to the definition of the All Enforcement Structure for a given property.

Definition 2.5.1. *(All Enforcement Structure). Given a system* $\boldsymbol{G}$ *and an IS-based property* $\varphi : I \to \{0, 1\}$ *w.r.t.* $\boldsymbol{G}$*, the All Enforcement Structure (AES) for property* φ*, denoted by*

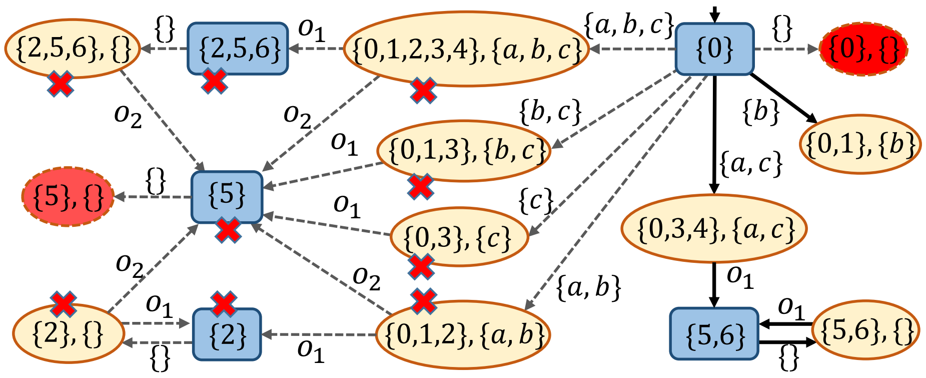
$$\mathcal{AES}_\varphi(\boldsymbol{G}) = (Q_Y^{AES}, Q_Z^{AES}, h_{YZ}^{AES}, h_{ZY}^{AES}, \Sigma, \Gamma, y_0), \tag{2.15}$$

is defined as the largest complete BTS w.r.t. $\boldsymbol{G}$ *such that* $\forall z \in Q_Z^{AES} : \varphi(I(z)) = 1$*. By "largest" subsystem, we mean that for any* T *satisfying the above conditions, we have that* $T \sqsubseteq \mathcal{AES}_\varphi(\boldsymbol{G})$*.*

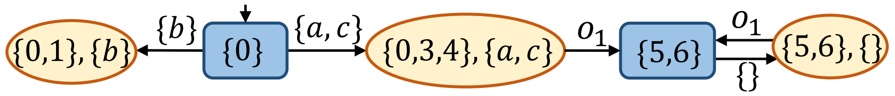
Note that if T_1 and T_2 are two complete BTSs in which all Z-states satisfy φ, then it is easy to see that the union of them is still a BTS in which all Z-states satisfy φ. Therefore, the notion of "largest BTS" in the definition is well defined. This will also be seen when we present the algorithm for the construction of the AES later.

Example 2.5.1. *We return to system* $\boldsymbol{G}$ *in Figure 2.1(a) with the IS-based property in Equation (2.6). The BTS shown in Figure 2.2(b) is, in fact, its AES. For example, at initial* Y*-state* $\{0\}$*, we cannot make control decision* $\{a, b, c\}$*, which would lead us to* Z*-state* $(\{0, 1, 2, 3, 4\}, \{a, b, c\})$*. This is because upon the occurrence of event* o_2*,* Y*-state* $\{5\}$ *would be reached, from which no matter what control decision we take, the secret will be revealed. We will discuss later that how to construct the AES.*

Remark 2.5.1. In Figure 2.2(a), we can also take control decision $\{a\}$ at the initial Y-state $y_0 = \{0\}$. However, this control decision is equivalent to decision $\{\}$, since



(a) The resulting structure after procedure DoDSF



(b) The constructed AES

Figure 2.2: Example of the construction of the AES. In the diagrams, rectangular (blue) states correspond to Y-states and oval (yellow) states correspond to Z-states. For simplicity, in the diagrams, we omit all uncontrollable events in the control decisions, e.g., decision $\{\}$ represents $\{o_1, o_2\}$, and so forth.

event a will never be executed within the unobservable reach. Formally, we say that a control decision $\gamma \in \Gamma$ is *irredundant* at information state $i \in I$ if, for any $\sigma \in \gamma$, there exists $x \in UR_\gamma(i)$ such that $\delta(x, \sigma)$ is defined. From now on, we only consider irredundant control decisions in the AES, which will clearly not affect its properties. Similarly, we say that a supervisor S is irredundant if for any $s \in P(\mathcal{L}(S/\mathbf{G}))$, control decision $S(s)$ is irredundant at information state $IS_S^Y(s)$. Hereafter, we also assume without loss of generality that any S is irredundant.

The following theorem shows that the AES (only) contains valid solutions to the property enforcement problem.

Theorem II.1. *Suppose that φ is an IS-based property w.r.t. $\boldsymbol{G}$. A supervisor enforces property φ if and only if it is an AES-included supervisor. Mathematically,*

$$\mathcal{L}(S/\boldsymbol{G}) \models_{\boldsymbol{G}} \varphi \Leftrightarrow S \in \mathbb{S}(\mathcal{AES}_\varphi(\boldsymbol{G})) \tag{2.16}$$

Proof. By Lemmas 2.4.1, we know that the LHS of Equation (2.16) holds if and only if $\forall s \in P(\mathcal{L}(S/\mathbf{G})) : \varphi(I(IS_S^Z(s)))=1$, Therefore, the "if" part follows immediately from Definitions 2.4.2 and 2.5.1.

Next, we prove the "only if" part by contradiction. Suppose that $\mathcal{L}(S/\mathbf{G}) \models_{\mathbf{G}} \varphi$, i.e., $\forall s \in P(\mathcal{L}(S/\mathbf{G})) : \varphi(I(IS_S^Z(s))) = 1$. We assume that $S \notin \mathbb{S}(\mathcal{AES}_\varphi(\mathbf{G}))$. First, we know that there exists a complete BTS T such that $S \in \mathbb{S}(T)$. Specifically, the complete BTS T can be constructed as follows: $Q_Y^T := \{y \in I : \exists t \in P(\mathcal{L}(S/\mathbf{G})) \text{ s.t. } y = IS_S^Y(t)\}$, $Q_Z^T := \{z \in I \times \Gamma : \exists t \in P(\mathcal{L}(S/\mathbf{G})) \text{ s.t. } z = IS_S^Z(t)\}$ and for any $y \in Q_Y^T$, $C_T(y) := \{\gamma \in \Gamma : \exists t \in P(\mathcal{L}(S/\mathbf{G})) \text{ s.t. } y = IS_S^Y(t) \wedge \gamma = S(t)\}$. In other words, any Y or Z-state in T are a Y or Z-state reached by supervisor S under some string $t \in P(\mathcal{L}(S/\mathbf{G}))$, respectively. Clearly, we know that $\forall z \in Q_Z^T : \varphi(I(z)) = 1$. Since $S \notin \mathbb{S}(\mathcal{AES}_\varphi(\mathbf{G}))$, we know that there exists a string $s \in P(\mathcal{L}(S/\mathbf{G}))$, such that $S(s) \notin C_{\mathcal{AES}_\varphi(\mathbf{G})}(IS_S^Y(s))$. In this case, the union of T and $\mathcal{AES}_\varphi(\mathbf{G})$ is strictly larger than $\mathcal{AES}_\varphi(\mathbf{G})$, since control decision $S(s)$ is defined at Y-state $IS_S^Y(s)$ in $T \cup \mathcal{AES}_\varphi(\mathbf{G})$ but not in $\mathcal{AES}_\varphi(\mathbf{G})$. This is a contradiction since by definition, $\mathcal{AES}_\varphi(\mathbf{G})$ is the largest BTS satisfying the condition in Definition 2.5.1. □

2.5.2 Construction of the AES

The construction algorithm for the AES follows directly from its definition and proceeds in two steps. First, we construct the BTS that enumerates all possible behaviors for each state by a depth-first search and remove all Z-states that violate the IS-based property. Second, we prune states that violate the completeness from the remaining part of the BTS, until convergence is achieved. In practice, in the depth-first search part, we do not need to search the whole state space and we can stop the search of a branch once a Z-state that violates the IS-based property is encountered.

The above procedure is formally described in Algorithm FIND-AES whose pa-

Algorithm 1: FIND-AES

```
input  : $\mathbf{G}$ and $\varphi$
output: $AES$
1  $AES.Y \leftarrow \{y_0\}, AES.Z \leftarrow \emptyset$ and $AES.h \leftarrow \emptyset$;
2  DoDFS($\mathbf{G}, y_0, AES$);
3  Prune($AES$);
4  $AES \leftarrow$ Accessible($AES$);
procedure DoDFS($\mathbf{G}, y, AES, \varphi$);
5  for $\gamma \in \Gamma$ do
6      $z \leftarrow h_{YZ}(y, \gamma)$;
7      if $\varphi(I(z)) = 1$ then
8          $AES.h \leftarrow AES.h \cup \{(y, \gamma, z)\}$;
9          if $z \notin AES.Z$ then
10             $AES.Z \leftarrow AES.Z \cup \{z\}$;
11             for $\sigma \in \gamma \cap \Sigma_o$ do
12                 $y' \leftarrow h_{ZY}(z, \sigma)$;
13                 $AES.h \leftarrow AES.h \cup \{(z, \sigma, y')\}$;
14                 if $y' \notin AES.Y$ then
15                     $AES.Y \leftarrow AES.Y \cup \{y'\}$;
16                     DoDFS($\mathbf{G}, y', AES, \varphi$);
procedure Prune($AES$);
17 while exists $Y$-state in AES that has no successor do
18     Delete all such $Y$-states in AES and delete all their predecessor $Z$-states;
```

rameters are as follows: (i) AES represents the AES that we want to construct; (ii) AES.Y and AES.Z are its sets of Y- and Z-states, respectively; and (iii) AES.h is its transition function. Initially, AES.Y is set to be $y_0 = \{x_0\}$. The depth-first search is then started; it is implemented by the procedure DoDFS. Line 7 is used to determine whether the Z-state encountered satisfies property φ. If not, we terminate the search of this branch. Otherwise, we compute all possible Y-state successors and make a recursive call. This recursive procedure allows us to traverse the whole reachable space of Y- and Z-states. The above procedure may result in Y-states that have no successors. Therefore, we need to iteratively prune: (i) all Y-states that have no successor states; and (ii) all Z-states for which at least one observation is not defined. This step is captured by procedure Prune. Finally, states that are no longer accessible from the initial state of the AES need to be removed before the algorithm returns. Algorithm FIND-AES will terminate in finite steps, since the number of Y- and Z-states is finite.

Example 2.5.2. *Consider our running example. We apply Algorithm FIND-AES to construct the corresponding AES. The resulting BTS after running the procedure DoDSF is shown in Figure 2.2(a). At the initial Y-state, we cannot take control decision $\{\}$ since this will lead to a deadlock Z-state $(\{0\}, \{\})$. The depth-first search DoDSF terminates at Y-state $\{5\}$, since no matter what control decision we take from $\{5\}$, a Z-state (marked in red in Figure 2.2(a)) that violates the IS-based property (i.e., that reveals the secret) will be encountered. After procedure DoDSF is done, we need to run procedure Prune. This starts by removing Y-state $\{5\}$, since no successor state is defined from it. Since Y-state $\{5\}$ has been removed, all its predecessor Z-states, i.e., $(\{0,3\}, \{c\}), (\{0,1,2\}, \{a,b\})$ and so forth, must also be removed. Finally, we remove inaccessible states $\{2,5,6\}$ and $\{2\}$ and obtain the AES shown in Figure 2.2(b).*

Theorem II.2. *Algorithm FIND-AES correctly constructs the AES.*

Proof. Let T be the BTS returned by Algorithm FIND-AES. T is a complete BTS by procedure Prune. Since procedure DoDSF only traverses the state space where all Z-states satisfy φ, we know that $\forall z \in Q_Z^T : \varphi(I(z)) = 1$. Therefore, it remains to show that T is the largest BTS with the desired properties; for that proof, we proceed by contradiction.

Assume that T' is another complete BTS such that $\forall z \in Q_Z^{T'} : \varphi(I(z)) = 1$ and it is strictly larger than T, i.e., $T \sqsubseteq T'$ and $Q_Y^T \cup Q_Z^T \subset Q_Y^{T'} \cup Q_Z^{T'}$. Therefore, in Algorithm FIND-AES, T is obtained by pruning states from some BTS T'' (which may not satisfy the desired properties) such that $T' \sqsubseteq T''$; e.g., T'' can be the resulting BTS after procedure DoDFS. Then, any Y- or Z-states in T' will not be removed in T'' by procedure Prune. Therefore, Algorithm FIND-AES will converge to a BTS that is strictly lager than T (at least as large as T'). This contradicts the fact that Algorithm FIND-AES converges to T. $\square$

2.6 Synthesis of Maximally Permissive Supervisors

In this section, we present a synthesis algorithm that returns a solution to MPIEP. We first discuss the general case, where Σ_c and Σ_o need not be comparable. Then we show that, under the assumption that $\Sigma_c \subseteq \Sigma_o$, there always exists a unique supremal solution to MPIEP.

2.6.1 General Case

Given an IS-based property, Theorem II.1 provides us with a straightforward procedure for synthesizing a property-enforcing supervisor. We can simply start from the initial Y-state and pick *one* control decision defined in the AES; then we pick *all* possible observations for the successor Z-state, and so forth, until reaching a Z-state that has no successor state. However, this procedure may result in a solution with infinite domain, since we may select different control decisions upon different visits to the same information state. Therefore, we will restrict our attention to an *information-state-based* (IS-based) solution. We will show later that, in fact, such a restriction is without loss of generality.

We present a synthesis algorithm, called Algorithm MAX-SYNT, for constructing an IS-based supervisor S^* that solves MPIEP. This algorithm starts from y_0. For each reachable Y-state y, it picks *one* control that is *locally maximal* and for each reachable Z-state, it picks *all* possible observations, until: (i) a terminal Z-state is reached; or (ii) a Y-state that has already been visited is reached. This is implemented by procedure Expand in Algorithm MAX-SYNT, which is simply a depth-first search. In other words, we pick a locally maximal control decision and fix it for each Y-state. This will result in a BTS T that includes a *unique* supervisor S_T, which is our solution.

The follow theorem establishes the correctness of Algorithm MAX-SYNT.

```
Algorithm 2: MAX-SYNT
  input  : $\mathcal{AES}_\varphi(\mathbf{G})$
  output: $S^*$
1     $T.Y \leftarrow \{y_0\}, T.Z \leftarrow \emptyset$ and $T.h \leftarrow \emptyset$;
2     Expand($T, \mathcal{AES}_\varphi(\mathbf{G}), y_0$);
3     $S^* \leftarrow S_T$;
  procedure Expand($T, \mathcal{AES}_\varphi(\mathbf{G}), y$);
4     Find a locally maximal control decision
       $\gamma \in C_{\mathcal{AES}_\varphi(\mathbf{G})}(y)$ s.t. $\forall \gamma' \in C_{\mathcal{AES}_\varphi(\mathbf{G})}(y) : \gamma \not\subset \gamma'$;
5     $z \leftarrow h_{YZ}(y, \gamma)$;
6     $T.h \leftarrow T.h \cup \{(y, \gamma, z)\}$;
7     if $z \notin T.Z$ then
8         $T.Z \leftarrow T.Z \cup \{z\}$;
9         for $\sigma \in \gamma \cap \Sigma_o$ do
10            $y' \leftarrow h_{ZY}(z, \sigma)$;
11            $T.h \leftarrow T.h \cup \{(z, \sigma, y')\}$;
12            if $y' \notin T.Y$ then
13                $T.Y \leftarrow T.Y \cup \{y'\}$;
14                Expand($T, \mathcal{AES}_\varphi(\mathbf{G}), y'$);
```

Theorem II.1. *Let S^* be a solution returned by Algorithm MAX-SYNT. Then S^**

solves MPIEP.

Proof. First, we note that $\mathcal{L}(S/\mathbf{G})$]satisfies φ; this follows from Theorem II.1 and the fact that, by construction, S^* is an AES-included supervisor. Therefore, it remains to show that S^* is maximal; for that proof, we proceed by contradiction.

Assume that S^* is not maximal, which means that there exists another AES-included supervisor $S' \in \mathbb{S}(\mathcal{AES}_\varphi(\mathbf{G}))$ such that $\mathcal{L}(S^*/\mathbf{G}) \subset \mathcal{L}(S'/\mathbf{G})$. Therefore, there exists a string $w \in P(\mathcal{L}(S^*/\mathbf{G})) \subseteq P(\mathcal{L}(S'/\mathbf{G}))$ such that $S^*(w) \subset S'(w)$ and $S^*(w') = S'(w'), \forall w' \in \overline{\{w\}} \setminus \{w\}$. We know that $IS^Y_{S^*}(w) = IS^Y_{S'}(w)$; let us call this Y-state y. But this means that the control decision $S^*(w)$ at y violates the locally maximal construction rule, i.e., there should not exist a control decision $\gamma \in C_{\mathcal{AES}_\varphi(\mathbf{G})}(y) : S^*(w) \subset \gamma$. This is a contradiction. Hence no such S' exists. □

By Theorem II.1, we know that the AES is non-empty if MPIEP has a solution. Moreover, when the AES is non-empty, Algorithm MAX-SYNT always returns a solution to MPIEP. Therefore, we have the following result.

Corollary 2.6.1. *MPIEP is solvable if and only if the AES is non-empty.*

Since supervisor S^* is IS-based by construction, we also have the following result.

Corollary 2.6.2. *For any IS-based property φ, there exists an IS-based supervisor that solves MPIEP iff $\mathcal{AES}_\varphi(\boldsymbol{G})$ is non-empty.*

Example 2.6.1. *We return to our running example. If we pick locally maximal control decision $\{a, c\}$ at the initial Y-state $\{y_0\}$ and pick the unique control decision $\emptyset$ at the reachable Y-state, which means that all controllable events are disabled, then we will obtain the maximal solution that was shown earlier in Figure 2.1(b). On the other hand, if we pick control decision $\{b\}$ at $\{0\}$, which is also locally maximal, then no observable behavior can occur thereafter; this corresponds to the maximal solution shown in Figure 2.1(c).*

Remark 2.6.1. The running time of the entire synthesis procedure is $O(2^{2|X|+2|\Sigma_c|})$. First, we need to construct the AES by Algorithm FIND-AES, which consists of two procedures, DoDSF and Prune. The procedure DoDSF may result in a BTS that, in the worst case, has $2^{|X|+|\Sigma_c|} + 2^{|X|}$ states. The complexity of procedure Prune is quadratic in the size of the above BTS. The complexity of Algorithm MAX-SYTN is linear in the size of the AES that, in the worst case, also has $2^{|X|+|\Sigma_c|} + 2^{|X|}$ states. Therefore, our synthesis procedure is exponential in the size of **G**. However, it was shown in [99] that synthesizing a partial observation safe supervisor, which is a special case of our problem, is NP-hard. Therefore, this exponential complexity seems to be unavoidable and it is due to the partial observation feature of our problem.

2.6.2 Case of $\Sigma_c \subseteq \Sigma_o$

It was shown in [54] that, under the assumption that $\Sigma_c \subseteq \Sigma_o$, observability and controllability together imply normality. Therefore, there exists a supremal controllable and observable sublanguage when $\Sigma_c \subseteq \Sigma_o$. It was also reported in [76] (respectively, [26] and [78]) that, under the assumption that $\Sigma_c \subseteq \Sigma_o$, there exists a supremal controlable, observable and diagnosable (respectively, opaque and attractable) sublanguage. In fact, we can prove the corresponding general result for *any IS-based property* in our framework.

The following lemma reveals that, under the assumption that $\Sigma_c \subseteq \Sigma_o$, the information state encountered does not depend on the control policy we take.

Lemma 2.6.1. *Let S^1 and S^2 be two supervisors. Under the assumption that $\Sigma_c \subseteq \Sigma_o$, we have that*

$$(\forall s \in \mathcal{L}(S^1/\boldsymbol{G}) \cap \mathcal{L}(S^2/\boldsymbol{G}))[I(IS^Z_{S^1}(P(s))) = I(IS^Z_{S^2}(P(s)))] \tag{2.17}$$

Proof. We prove this lemma by induction on the length of $P(s)$. For any string s, let

$|P(s)| = n$. Let s_k and e_k be the same notations defined in the proof of Lemma 2.4.1. For any $i = 1, 2$, define y_0^i as usual and for $k = 0, \ldots, n$, let $z_k^i = h_{YZ}(y_k^i, S^i(s_k))$, and for $k = 0, \ldots, n-1$, define $y_{k+1}^i = h_{ZY}(z_k^i, e_k)$. Therefore, the inductive hypothesis is that:

$$I(z_k^1) = I(z_k^2) \tag{2.18}$$

Induction Basis ($s_0 = \epsilon$):

$$\begin{aligned} I(z_0^2) = \mathrm{UR}_{S^2(\epsilon)}(y_0) &= \{v \in X : \exists t \in (S^2(\epsilon) \cap \Sigma_{uo})^* \text{ s.t. } v{=}\delta(x_0, t)\} \\ &= \{v \in X : \exists t \in (S^1(\epsilon) \cap \Sigma_{uo})^* \text{ s.t. } v{=}\delta(x_0, t)\} \\ &= \mathrm{UR}_{S^1(\epsilon)}(y_0) = I(z_0^1) \end{aligned}$$

where $S^2(\epsilon) \cap \Sigma_{uo} = S^1(\epsilon) \cap \Sigma_{uo}$ holds because $\Sigma_c \cap \Sigma_{uo} {=} \emptyset$.

Induction Step:

Assume that the induction hypothesis is true at k. Then

$$\begin{aligned} y_{k+1}^2 = h_{ZY}(z_k^2, e_k) &= \{v \in X : \exists u \in I(z_k^2) \text{ s.t. } v = \delta(u, e_k)\} \\ &= \{v \in X : \exists u \in I(z_k^1) \text{ s.t. } v = \delta(u, e_k)\} \\ &= y_{k+1}^1 \end{aligned}$$

$$I(z_{k+1}^2) = \mathrm{UR}_{S^2(s_k e_k)}(y_{k+1}^2) = \mathrm{UR}_{S^2(s_k e_k)}(y_{k+1}^1) = \mathrm{UR}_{S^1(s_k e_k)}(y_{k+1}^1) = I(z_{k+1}^1)$$

where $\mathrm{UR}_{S^2(s_k' e_k)}(y_{k+1}^1) = \mathrm{UR}_{S^1(s_k' e_k)}(y_{k+1}^1)$ follows from the same argument as in the induction basis. This completes the proof by induction. $\square$

Consider two different supervisors; under the assumption that $\Sigma_c \subseteq \Sigma_o$, the information state components of the Z-states encountered upon the occurrence of the same string are identical. Therefore, in this scenario, state estimation (from observed events) does not depend on the control policy we take. In [3], the authors show that for centralized partial observation control problems, a given Z-state (termed as maximal information set in [3]) is independent from the control policy the supervisor will take in the future. (This is not true in general in decentralized control; see again [3].) In essence, Lemma 2.6.1 extends this result and says that, under the assumption that $\Sigma_c \subseteq \Sigma_o$, control and state estimation are one-way "fully separated", i.e., in addition to the non-dependency of state estimation on the future control actions, the Z-state even does not depend on the past control actions. This separability also leads to the follows theorem, which says that for any IS-based property, under the assumption that $\Sigma_c \subseteq \Sigma_o$, there exists a unique maximal permissive supervisor that enforces the property.

Theorem II.2. *Assume that* $\Sigma_c \subseteq \Sigma_o$*. Then there exists a unique (supremal) solution to MPISEP.*

Proof. By contradiction. Suppose that $\varphi : I \to \{0, 1\}$ is the IS-based property that we want to enforce. We assume that S^1 and S^2 are two different solutions to MPISEP, i.e., $\mathcal{L}(S^1/\mathbf{G})$ and $\mathcal{L}(S^2/\mathbf{G})$ are two incomparable maximal controllable and observable sublanguages satisfying φ. Under the assumption that $\Sigma_c \subseteq \Sigma_o$ and that the two given languages are controllable and observable, we know that their union will also be controllable and observable. Hence, there exists a partial observation supervisor S^* such that $\mathcal{L}(S^*/\mathbf{G}) = \mathcal{L}(S^1/\mathbf{G}) \cup \mathcal{L}(S^2/\mathbf{G})$. Specifically, for any $s \in P(\mathcal{L}(S^*/\mathbf{G}))$ we have $S^*(s) = S^1(s) \cup S^2(s)$.

Next, we show that S^* also enforces φ. Let us assume that S^* does not enforce property φ, i.e., $\exists s \in \mathcal{L}(S^*/\mathbf{G})$ s.t. $\varphi(R_{\mathbf{G}}(s, \mathcal{L}(S^*/\mathbf{G}))) = 0$. Since $\mathcal{L}(S^*/\mathbf{G}) = \mathcal{L}(S^1/\mathbf{G}) \cup \mathcal{L}(S^2/\mathbf{G})$, we know that $\exists i \in \{1, 2\}$ s.t. $s \in \mathcal{L}(S^i/\mathbf{G})$. By Lemma 2.6.1, we know that $I(IS^Z_{S^*}(P(s))) = I(IS^Z_{S^i}(P(s)))$. Moreover, by Lemma 2.4.1, we know that $R_{\mathbf{G}}(s, \mathcal{L}(S^*/\mathbf{G})) = R_{\mathbf{G}}(s, \mathcal{L}(S^i/\mathbf{G}))$. However, $\varphi(R_{\mathbf{G}}(s, \mathcal{L}(S^i/\mathbf{G}))) = 1$, since S^i enforces property φ. This implies that $\varphi(R_{\mathbf{G}}(s, \mathcal{L}(S^*/\mathbf{G}))) = 1$, which is a contradiction. Therefore, S^* also enforces property φ.

The above result contradicts the fact that $\mathcal{L}(S^1/\mathbf{G})$ and $\mathcal{L}(S^2/\mathbf{G})$ are maximal. Therefore, there only exists a unique solution to MPISEP. □

We have shown that Algorithm MAX-SYNT always returns a maximal solution; moreover, under the assumption that $\Sigma_c \subseteq \Sigma_o$, this maximal solution is unique. Therefore, in this scenario, Algorithm MAX-SYNT returns the unique supremal solution.

Corollary 2.6.3. *Let S^* be the solution returned by Algorithm MAX-SYNT. When $\Sigma_c \subseteq \Sigma_o$, S^* is the unique supremal solution to MPIEP.*

Remark 2.6.2. In the standard supervisory control problem, the supremal controllable and observable sublanguage can be obtained under the assumption that $\Sigma_c \subseteq \Sigma_o$ by computing the supremal controllable and normal sublanguage [18]. Since both the supremal normal approach and Algorithm MAX-SYNT take exponential complexity in the size of the system, our approach does not improve upon the complexity of the previous result under this restrictive assumption. Instead, Algorithm MAX-SYNT provides an alternative approach for the computation of supermal controllable and normal sublanguage for this special case.

2.6.3 The Issue of Liveness

In additional to IS-based property, in many applications, we also need to consider the (weaker) *liveness* property. Liveness is an important property in many cyber and cyber-physical systems, e.g., software systems [50] and flexible manufacturing systems [49]. Formally, we say that a language L is live if for any $s \in L$, we have $\Delta_L(s) \neq \emptyset$. We say that system $\mathbf{G}$ is live if its generated language $\mathcal{L}(\mathbf{G})$ is live. In fact, the definitions of many properties, e.g., diagnosability and detectability, are based on the assumption that the system under consideration is live. Therefore, we need to assume that $\mathbf{G}$ is live and we must ensure that the controlled system is also live. The liveness assumption on $\mathbf{G}$ is without essential loss of generality, since it can be relaxed by adding observable self-loops at terminal states, as is done in [77]. (Essentially, this means that system deadlock is observable.) In order to enforce liveness, we can added the following requirement to Definition 2.5.1: for any Z-state $z \in Q_Z^{AES}$, we have

$$\forall x \in I(z), \exists \sigma \in \Gamma(z) : \delta(x, \sigma)! \tag{2.19}$$

It is straightforward to show that in the resulting modified AES, instead of the result in Theorem II.1, we have instead that

$$[S/\mathbf{G} \text{ is live}] \wedge [\mathcal{L}(S/\mathbf{G}) \models_{\mathbf{G}} \varphi] \Leftrightarrow S \in \mathbb{S}(\mathcal{AES}_{\varphi}(\mathbf{G})) \tag{2.20}$$

In other words, the modified AES will contain all property-enforcing supervisors, resulting in live behavior. The modified AES can be constructed in the same manner as the construction the AES. Specifically, in line 7 of FIND-AES, in addition to check if $\varphi(I(z)) = 1$, we also need to check if Equation (2.19) holds. Then we can apply Algorithm MAX-SYNT based the modified AES, which will return a live property-enforcing supervisor; the correctness of this approach is proved in [121].

2.7 Applications of the Uniform Approach

In this section, we show that how to apply the uniform approach described in this chapter to the enforcement of several specific properties commonly encountered in the study of DES. Our uniform approach comprises three steps:

1. Formulate the property to be enforced as an IS-based property;
2. Construct the AES using Algorithm FIND-AES;
3. Find a maximal solution based on the AES using Algorithm MAX-SYNT.

In Sections 2.5 and 2.6, we have discussed Steps 2 and 3, respectively; it remains to discuss how to formulate a given property as an IS-based property, whenever feasible. As was mentioned earlier, there are properties that cannot be formulated as IS-based properties; one such example is non-blockingness [120]. However, as we will see in this section, many important properties in the DES literature, including but not restricted to safety, opacity, diagnosability, detectbility, anonymity and attractability, can be formulated as IS-based properties. Therefore, all of them can be enforced by using the above three-step methodology.

2.7.1 Enforcement of Safety

Given a prefix-closed specification language K, we say that language $L \subseteq \mathcal{L}(\mathbf{G})$ is safe if $L \subseteq K$. When the uncontrolled system is not safe, the standard supervisory control and observation problem [20,52] asks to synthesize a least restrictive supervisor such that the controlled system is safe. We show that this can be solved by our uniform approach.

Let $K = \mathcal{L}(\mathbf{K})$, for some automaton $\mathbf{K}$. In [18], the authors provide an algorithm to construct refined automata $\mathbf{K}_S = (X_{K_S}, \Sigma, \delta_{K_S}, x_{0,K_S})$ and $\mathbf{G}_S = (X_{G_S}, \Sigma, \delta_{G_S}, x_{0,G_S})$ such that the following holds: 1) $\mathcal{L}(\mathbf{G}_S) = \mathcal{L}(\mathbf{G})$ and $\mathcal{L}(\mathbf{K}_S) = \mathcal{L}(\mathbf{K})$; 2) $\mathbf{K}_S$ is a sub-automaton of $\mathbf{G}_S$; 3) $\mathbf{K}_S \sqsubset \mathbf{G}_S$. For the construction of $\mathbf{G}_S$ and $\mathbf{K}_S$, the reader is referred to [18]. The above conditions imply that X_{K_S} captures the legal behaviors, i.e., any string in $\mathcal{L}(\mathbf{G}_S)$ that leads to a state in X_{K_S} is safe and any string in $\mathcal{L}(\mathbf{G}_S)$ that leads to a state in $X_{G_S} \setminus X_{K_S}$ is unsafe.

For the refined system model $\mathbf{G}_S$, we define the IS-based property $\varphi_{safe} : 2^{X_{G_S}} \to \{0, 1\}$ w.r.t. $\mathbf{G}_S$ as follows. For any information state $i \in 2^{X_{G_S}}$, $\varphi_{safe}(i) = 1 \Leftrightarrow i \subseteq X_{K_S}$. Then we have the following result.

Proposition 2.7.1. *Let $\boldsymbol{K}$ be the specification automaton and $\boldsymbol{G}_S$ be the refined system automaton defined above, then language $L \subseteq \mathcal{L}(\boldsymbol{G})$ is safe if and only if*

$L \models_{\boldsymbol{G}_S} \varphi_{safe}$.

Proof. The proof follows directly from Definition 2.3.1 and Lemma 2.4.1, since $L \not\models_{\mathbf{G}_s} \varphi_{safe}$ if and only if $\exists s \in L : \delta_{G_S}(x_{0,G_S}, s) \notin X_{K_S}$, which is equivalent to $L \not\subseteq \mathcal{L}(\mathbf{K}_S) = \mathcal{L}(\mathbf{K})$. □

Hence, to solve the safety control problem, it suffices to synthesize a supervisor that enforces the IS-based property φ_{safe} w.r.t. the refined state space of $\mathbf{G}_S$. Therefore, the maximally permissive safety control problem can be solved by our uniform approach.

2.7.2 Enforcement of Current-State Opacity

Opacity is a confidentiality property for partially-observed systems. It captures the plausible deniability of the system's "secret" in the presence of an outside observer that is potentially malicious. First, we recall the definition of *current-state opacity*, as it is presented in [51, 107].

Definition 2.7.1. *Let $\boldsymbol{G} = (X, \Sigma, \delta, x_0)$ be the system automaton. Language $L \subseteq \mathcal{L}(\boldsymbol{G})$ is said to be* current-state opaque *w.r.t. $X_S \subseteq X$, $\boldsymbol{G}$ and P if*

$$(\forall s \in L : \delta(x_0, s) \in X_S)(\exists t \in L)\,[P(s) = P(t) \wedge \delta(x_0, t) \notin X_S] \tag{2.21}$$

Note that we assume in this section that the external observer and the supervisor have the same observation set, Σ_o.

To formulate the current-state opacity enforcement problem in our framework, we define the IS-based current-state opacity property $\varphi_{opa} : 2^X \to \{0, 1\}$ as follows. For any information state $i \in 2^X$, we have

$$\varphi_{opa}(i) = 0 \Leftrightarrow i \subseteq X_S \tag{2.22}$$

The following result says that the IS-based property φ_{opa} correctly captures the opacity property.

Proposition 2.7.2. *Let $\boldsymbol{G}$ be the system automaton, $X_S \subseteq X$ be the subset of secret states, and φ_{opa} be the IS-based property defined above. Language $L \subseteq \mathcal{L}(\boldsymbol{G})$ is current-state opaque if and only if $L \models_{\boldsymbol{G}} \varphi_{opa}$.*

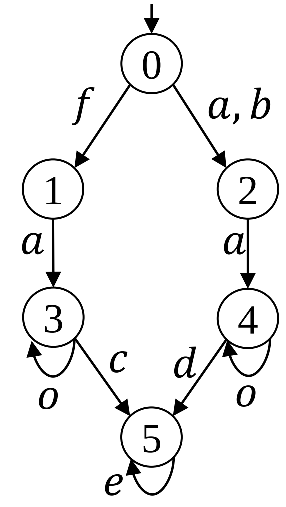
Proof. We proceed by contrapositive. By definition, $\mathbf{G}$ is not current-state opaque if and only if $(\exists s \in L)(\forall t \in L)[P(s) = P(t) \Rightarrow \delta(x_0, t) \in X_S]$, which is equivalent to $(\exists s \in L)(\forall x \in R_{\mathbf{G}}(s, L))[x \in X_S]$. This is equivalent to $\exists s \in L : \varphi_{opa}(R_{\mathbf{G}}(s, L)) = 0$, i.e., $L \not\models_{\mathbf{G}} \varphi_{opa}$. $\square$

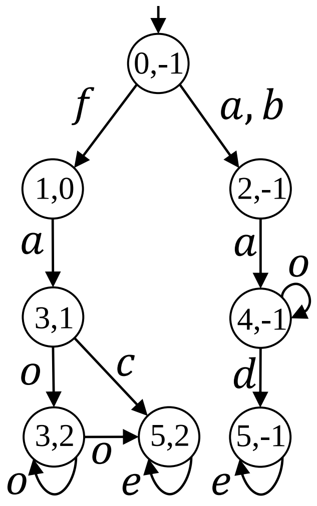
Consequently, the opacity enforcement problem can be solved by using φ_{opa} in our uniform approach. Our running example has already shown how to synthesize a maximally permissive supervisor enforcing opacity.

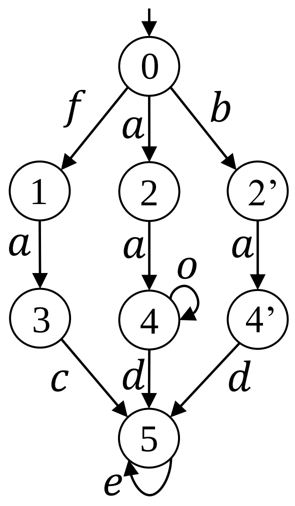
Remark 2.7.1. It was shown in [107] that several other notions of opacity, e.g., language-based opacity, initial-state opacity, and initial-and-final-state opacity, can be transformed to current-state opacity in polynomial time. Therefore, the enforcement of these notions of opacity can be done by first transforming them to current-state opacity and then enforcing current-state opacity as discussed above.

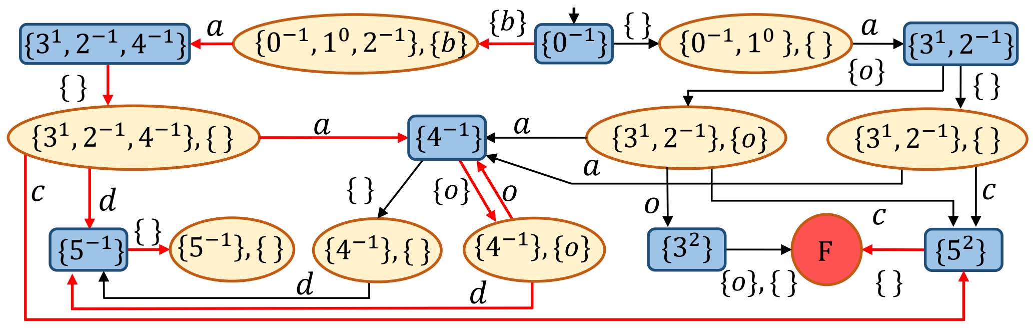
2.7.3 Enforcement of K-Diagnosability

In fault diagnosis problems, $e_d \in \Sigma_{uo}$ is a fault event whose occurrences must be diagnosed by the diagnoser within a finite number of steps. Suppose L is the language to be diagnosed. We define $\Psi(e_d, L) = \{se_d \in L : s \in \Sigma^*\}$ to be the set of strings that end with the fault event. We say that a language is K-diagnosable if this diagnosis delay is uniformly bounded by a given number K. The formal definition of K-diagnosability is recalled from [16, 22, 77].


(a) $\mathbf{G}$


(b) $\mathbf{G}_D$


(c) $\mathcal{L}(S^*/\mathbf{G})$



(d) $\mathcal{AES}_{\varphi_{diag}}(\mathbf{G}_D)$

Figure 2.3: For $\mathbf{G}$: $\Sigma_c = \{b, o\}, \Sigma_o = \{a, c, d, o\}$, and f is the fault event. For the sake of brevity, in the diagram of the AES, we write state (x, n) in the form of x^n and all uncontrollable events in the control decisions are omitted. We also represent all Z-states z such that $\forall x^n \in I(z) : n \geq 0$ as a single state F, since we can diagnose the failure unambiguously at such states.

Definition 2.7.2. *(K-Diagnosability). A live language L is said to be K-diagnosable w.r.t. P and $e_d \in \Sigma_{uo}$ if*

$$(\forall s \in \Psi(e_d, L))(\forall t \in L/s)[|t| \geq K \Rightarrow (\forall w \in P^{-1}(P(st) \cap L) : e_d \in w)] \tag{2.23}$$

To formulate K-diagnosability as an IS-based property, we need to refine the state space of the original system $\mathbf{G}$, which is similar to the refinement procedure in [22]. Given $\mathbf{G} = (X, \Sigma, \delta, x_0)$ and non-negative integer K, we define the new automaton $\mathbf{G}_D = (X_D, \Sigma, \delta_D, x_{D,0})$, where

- $X_D \subseteq X \times \{-1, 0, 1, \ldots, K\}$ is the set of states;
- Σ is the set of events (same as defined in $\mathbf{G}$);
- $\delta_D : X_D \times \Sigma \to X_D$ is the partial transition function that is built from δ in $\mathbf{G}$ as follows: for any $u = (x, n) \in X_D, \sigma \in \Sigma$,

$$\delta_D(u, \sigma) = \begin{cases} (\delta(x,\sigma), -1), & \text{if } n = -1 \text{ and } \sigma \in \Sigma \setminus \{e_d\} \\ (\delta(x,\sigma), n+1), & \text{if } 0 \leq n < K \text{ or } n = -1 \wedge \sigma = e_d \\ (\delta(x,\sigma), K), & \text{if } n = K \end{cases} \tag{2.24}$$

- $x_{D,0} = (x_0, -1) \in X_D$ is the initial state.

By construction, we have that $\mathcal{L}(\mathbf{G}) = \mathcal{L}(\mathbf{G}_D)$, i.e., $\mathbf{G}_D$ is language-equivalent to $\mathbf{G}$ but refines its state space. Therefore, we can analyze the (language-based) property of diagnosability based on the refined system $\mathbf{G}_D$. To this end, we define the IS-based property termed K-diagnosability.

Definition 2.7.3. *The property of IS-based K-diagnosability $\varphi_{diag} : 2^{X_D} \to \{0, 1\}$ w.r.t. $\mathbf{G}_D$ is defined by: for any $i \in 2^{X_D}$,*

$$\varphi_{diag}(i) = 0 \Leftrightarrow (\exists u, v \in i)[[u]_n = -1 \wedge [v]_n = K] \tag{2.25}$$

where $[u]_n$ denotes the integer component of state u.

The following result establishes that to enforce K-diagnosablility, it suffices to enforce the property of IS-based K-diagnosability defined above.

Proposition 2.7.3. *A live language $L \subseteq \mathcal{L}(\boldsymbol{G}) = \mathcal{L}(\boldsymbol{G}_D)$ is K-diagnosable w.r.t. P and e_d if and only if $L \models_{\boldsymbol{G}_D} \varphi_{diag}$.*

Proof. We proceed by contrapositive.

$$\begin{aligned}
&L \text{ is not } K\text{-diagnosable} \\
\Leftrightarrow &\exists t_v = t_{v,1}t_{v,2}, t_u \in L \text{ s.t. } t_{v,1} \in \Psi(e_d, L) \text{ and} \\
&t_{v,2} \geq K \text{ and } \Sigma_F \notin t_u \text{ and } P(t_u) = P(t_v) && \text{Def. 2.7.2} \\
\Leftrightarrow &\exists t_v, t_u \in L \text{ s.t. } [\delta_K(x_{D,0}, t_v)]_n = K \text{ and } [\delta_D(x_{D,0}, t_u)]_n = -1 \text{ and } P(t_u) = P(t_v) \\
\Leftrightarrow &\exists t_v \in L \text{ s.t. } \varphi_{diag}(R_{\mathbf{G}_D}(t_v, L)) = 0 && \text{Def. 2.7.3} \\
\Leftrightarrow &L \not\models_{\mathbf{G}_D} \varphi_{diag} && \text{Def. 2.3.1}
\end{aligned}$$

The second equivalence is from the definition of $\mathbf{G}_D$. □

Example 2.7.1. *Let us consider the system $\boldsymbol{G}$ in Figure 2.3(a), where the set of controllable events is $\Sigma_c = \{b, o\}$ and the set of observable events is $\Sigma_o = \{a, c, d, o\}$; these two sets are incomparable. Event f is the unique fault event. Consider a desired diagnosis delay of $K = 2$. The corresponding unfolded system $\boldsymbol{G}_D$ is shown in Figure 2.3(b). The corresponding AES $\mathcal{AES}_{\varphi_{diag}}(\boldsymbol{G}_D)$ for $\boldsymbol{G}_D$ w.r.t. φ_{diag} is given in Figure 2.3(d). Note that, to construct the AES, we need to consider the issue of liveness discussed in Section 2.6.3. For the sake of brevity, we write state (x, n) in the form of x^n. For example, at Y-state $\{3^1, 2^{-1}, 4^{-1}\}$, we cannot enable event o, since no matter what control decision we take after the occurrence of o, a Z-state that contains both states 3^2 and 4^{-1} will be encountered, i.e., the IS-based property φ_{diag} will be violated.*

By applying Algorithm MAX-SYNT to $\mathcal{AES}_{\varphi_{diag}}(\boldsymbol{G}_D)$, a maximally permissive supervisor S^* is obtained; we highlight the chosen locally maximal control decision at each reachable Y-state (which in this example is unique) and all feasible observable events at each reachable Z-state in the diagram. The corresponding controlled behavior is given in Figrue 2.3(c). By Theorem II.1, $\mathcal{L}(S^*/\boldsymbol{G})$ is a maximal live, controllable, observable and 2*-diagnosable sublanguage of $\mathcal{L}(\boldsymbol{G})$.*

2.7.4 Enforcement of Strong Detectability

Detectability is a property arising in state estimation of DES. In [87], the enforcement of strongly detectability is studied under the assumption that $\Sigma_c \subseteq \Sigma_o$. Here, we show that strongly detectability with a pre-specified detection delay K, or *strongly K-detectability*, can be enforced without such an assumption by using the uniform approach. First, we recall the formal definition of strongly K-detectability from [87, 89].

Definition 2.7.4. *(Strongly K-Detectability). A live language $L \subseteq \mathcal{L}(\boldsymbol{G})$ is said to be strongly K-detectable w.r.t. P and $\boldsymbol{G}$ if*

$$(\forall s \in L)[|P(s)| \geq K \Rightarrow |R_{\boldsymbol{G}}(s, L)| = 1] \tag{2.26}$$

Analogous to the enforcement of diagnosability, given an automaton $\mathbf{G} = (X, \Sigma, \delta, x_0)$, we can build a new automaton $\mathbf{G}_T = (X_T, \Sigma, \delta_T, x_{T,0})$, where

- $X_T \subseteq X \times \{0, 1, \ldots, K\}$ is the set of states;
- Σ is the set of events;
- $\delta_T : X_T \times \Sigma \to X_T$ is the partial transition function and for any $u = (x, n) \in$

$X_T, \sigma \in \Sigma$, δ_T is defined by

$$\delta_T(u,\sigma) = \begin{cases} (\delta(x,\sigma), n), & \text{if } \sigma \in \Sigma_{uo} \wedge n < K \\ (\delta(x,\sigma), n+1), & \text{if } \sigma \in \Sigma_o \wedge n < K \\ (\delta(x,\sigma), K), & \text{if } n = K \end{cases} \tag{2.27}$$

- $x_{T,0} = (x_0, 0)$ is the initial state.

With the refined system $\mathbf{G}_T$, we define IS-based K-detectability as follows.

Definition 2.7.5. *(IS-Based Strongly K-Detectability). The property of IS-based K-detectability $\varphi_{det} : 2^{X_T} \to \{0,1\}$ w.r.t. $\boldsymbol{G}_T$ is defined by: for any $i \in 2^{X_T}$,*

$$\varphi_{det}(i) = 0 \Leftrightarrow (\exists u \in i : [u]_n = K) \wedge |i| > 1, \tag{2.28}$$

where $[u]_n$ denotes the integer component of u.

The following result says that to enforce strongly K-detectability it suffices to enforce the IS-based property φ_{det} defined above.

Proposition 2.7.4. *A live language L is strongly K-detectable w.r.t. P and $\boldsymbol{G}$ if and only if $L \models_{\boldsymbol{G}_T} \varphi_{det}$.*

Proof. We proceed by contrapositive.

$$\begin{aligned}
& L \text{ is not strongly } K\text{-detectable w.r.t. } P \text{ and } \mathbf{G} \\
\Leftrightarrow & \exists s \in L \text{ s.t. } |P(s)| \geq K \text{ and } |R_{\mathbf{G}}(s,L)| > 1 && \text{Def. 2.7.4} \\
\Leftrightarrow & \exists s \in L \text{ s.t. } |P(s)| \geq K \text{ and } |R_{\mathbf{G}_T}(s,L)| > 1 \\
\Leftrightarrow & \exists s \in L \text{ s.t. } [\delta_T(x_{T,0}, s)]_n = K \text{ and } |R_{\mathbf{G}_T}(s,L)| > 1 \\
\Leftrightarrow & \exists s \in L \text{ s.t. } \varphi_{det}(R_{\mathbf{G}_T}(s,L)) = 0 \\
\Leftrightarrow & L \not\models_{\mathbf{G}_T} \varphi_{det} && \text{Def. 2.3.1}
\end{aligned}$$

The third and fourth equivalences follow from the construction of $\mathbf{G}_T$ and the definition of φ_{det}, respectively. For the second equivalence, first we have the following observations:

(1) $|R_{\mathbf{G}}(s, L)| > 1$ if and only if

$$\exists t \in L : P(s) = P(t) \wedge \delta(x_0, t) \neq \delta(x_0, s) \tag{2.29}$$

(2) $|R_{\mathbf{G}_T}(s, L)| > 1$ if and only if

$$\exists t \in L : [P(s) = P(t)] \wedge [\delta(x_0, t) \neq \delta(x_0, s) \text{ or } \max\{|P(t)|, K\} \neq \max\{|P(s)|, K\}] \tag{2.30}$$

However, is always true that $\max\{|P(t)|, K\} = \max\{|P(s)|, K\}$ if $P(s) = P(t)$. Therefore, Equation (2.30) is equivalent to Equation (2.29), which implies that $|R_{\mathbf{G}}(s, L)| > 1 \Leftrightarrow |R_{\mathbf{G}_T}(s, L)| > 1$. $\square$

2.7.5 Enforcement of Anonymity

Strong detectability requires that the supervisor eventually be able to determine the exact system state. In security and privacy applications, when the system is monitored by a potentially malicious observer, we may want to enforce the exact opposite, i.e., the exact system state should never be revealed. This is related to the notion of opacity discussed earlier and it is termed *anonymity* [90], which is defined as follows.

Definition 2.7.6. *(Anonymity). Language $L \subseteq \mathcal{L}(\boldsymbol{G})$ is said to be anonymous w.r.t. P and $\boldsymbol{G}$ if*

$$(\forall s \in L)(\exists t \in L)[P(s) = P(t) \wedge \delta(x_0, s) \neq \delta(x_0, t)] \tag{2.31}$$

Anonymity is different from either detectability or opacity. However, anonymity can be easily formulated as an IS-based property, which means that it can enforced by using the uniform approach. To this end, we define the IS-based property $\varphi_{ano} : 2^X \to \{0,1\}$ w.r.t. $\mathbf{G}$ by: for any $i \in 2^X$, $\varphi_{ano}(i) = 0 \Leftrightarrow |i| = 1$. Then we have the following result, which says that enforcing anonymity is equivalent to enforcing IS-based property φ_{ano}.

Proposition 2.7.5. *Language L is anonymous w.r.t. P and $\boldsymbol{G}$ if and only if $L \models_{\boldsymbol{G}} \varphi_{ano}$.*

Proof. L is not anonymous if and only if $(\exists s \in L)(\forall t \in L)[P(s) = P(t) \Rightarrow \delta(x_0, s) = \delta(x_0, t)]$, This is equivalent to $(\exists s \in L)[|R_{\mathbf{G}}(s, L)| = 1]$, i.e., $L \not\models_{\mathbf{G}} \varphi_{ano}$. □

2.7.6 Enforcement of Attractability

The last property enforcement problem we study in this section is the state attraction problem. In this problem, the goal is to design a supervisor such that the controlled system will converge to a desired *attractor* in a bounded number of event occurrences. In [78], the state attraction problem under partial observation is studied under the assumption that $\Sigma_c \subseteq \Sigma_o$. We show that this assumption can be relaxed by taking the uniform approach developed in this chapter. Hereafter, instead of allowing arbitrary bounded convergence delay, we require that the system converge to the attractor in a pre-specified number of steps, leading to the notion of K-attractability.

Definition 2.7.7. *(K-Attractability). Let $\boldsymbol{G} = (X, \Sigma, \delta, x_0)$ be the system automaton. Language $L \subseteq \mathcal{L}(\boldsymbol{G})$ is said to be K-attractable w.r.t. $\boldsymbol{G}$ and $A \subseteq X$ if for any $s \in L$, we have*

1. *$|s| \geq K \Rightarrow \delta(x_0, s) \in A$;*
2. *$\delta(x_0, s) \in A \Rightarrow \forall st \in L : \delta(x_0, st) \in A$.*

Remark 2.7.2. In [78], the authors assume that $A \subseteq X$ is an invariant set, i.e., $(\forall x \in A)(\forall s \in \Sigma^*)[\delta(x, s) \in A]$. In this case, the second requirement in Definition 2.7.7

will be satisfied trivially. Therefore, the definition of attractability we consider here is more general.

Given an automaton $\mathbf{G} = (X, \Sigma, \delta, x_0)$, to formulate K-attractability as an IS-based property, we first construct the new automaton $\mathbf{G}_A = (X_A, \Sigma, \delta_A, x_{A,0})$, where

- $X_A \subseteq X \times \{0, 1, \ldots, K\}$ is the set of states;
- Σ is the set of events;
- $\delta_A : X_A \times \Sigma \to X_A$ is the partial transition function and for any $u = (x, n) \in X_A, \sigma \in \Sigma$, δ_T is defined by

$$\delta_A(u,\sigma) = \begin{cases} (\delta(x,\sigma), n+1), & \text{if } n < K \wedge x \notin A \\ (\delta(x,\sigma), K), & \text{if } n < K \wedge x \in A \text{ or } n = K \end{cases} \tag{2.32}$$

- $x_{A,0} = (x_0, 0)$ is the initial state.

We define IS-based K-attractability as follows.

Definition 2.7.8. *The property of IS-based K-attractability $\varphi_{att} : 2^{X_A} \to \{0,1\}$ w.r.t. $\boldsymbol{G}_A$ is defined by: for any $i \in 2^{X_A}$,*

$$\varphi_{att}(i) = 0 \Leftrightarrow \exists u \in i : [u]_x \notin A \wedge [u]_n = K \tag{2.33}$$

where $[u]_x$ and $[u]_n$ are the state component and the integer component of u, respectively.

The following result says that to enforce K-attractability w.r.t. $\mathbf{G}$, it suffices to enforce the IS-based property φ_{att} w.r.t. $\mathbf{G}_A$ defined above.

Proposition 2.7.6. *A live language L is K-attractable w.r.t. $\boldsymbol{G}$ if and only if $L \models_{\boldsymbol{G}_A} \varphi_{att}$.*

Proof. We proceed by contrapositive.

$$\begin{aligned}
& L \text{ is not } K\text{-attractable w.r.t. } \mathbf{G} \\
\Leftrightarrow & \exists s \in L \text{ s.t. } [|s| \geq K \wedge \delta(x_0, s) \notin A] \text{ or } [\delta(x_0, s) \in A \wedge \exists t \in L/s : \delta(x_0, st) \notin A] \\
\Leftrightarrow & \exists w \in L \text{ s.t. } [\delta_A(x_{A,0}, w)]_x \notin A \text{ and } [\delta_A(x_{A,0}, w)]_n = K \\
\Leftrightarrow & \exists w \in L \text{ s.t. } \varphi_{att}(R_{\mathbf{G}_A}(w, L)) = 0 && \text{Def. 2.7.8} \\
\Leftrightarrow & L \not\models_{\mathbf{G}_A} \varphi_{att} && \text{Def. 2.3.1}
\end{aligned}$$

For the second equivalence, the proof of the "$\Rightarrow$" direction can be done by taking $w = s$ if the first case holds and $w = st$ if the second case holds. For the "$\Leftarrow$" direction, since $[\delta_A(x_{A,0}, w)]_n = K$, by the construction of $\mathbf{G}_A$, we know that (i) $|w| \geq K$ or (ii) $\exists w_1 w_2 \in \overline{\{w\}}$ s.t. $\delta(x_0, w_1) \in A$ and $\delta(x_0, w_1 w_2) \notin A$. These two cases correspond to the two cases after the first equivalence, respectively. □

Remark 2.7.3. So far, we have discussed the enforcement of K-diagnosability, K-detectability and K-attractability. Since K-diagnosability (respectively, K-detectability and K-attractability) is stronger than diagnosability (respectively, detectability and attractability), enforcing the former one implies that the latter one is also enforced. Moreover, the uniform approach guarantees the diagnosis (respectively, detection and attraction) delay of the controlled system, which cannot be guaranteed by the previous approaches. In this sense, enforcing these properties with desired delay K is a new feature of the uniform solution rather than a restrictive assumption. However, if one does not care about the diagnosis (respectively, detection and attraction) delay and just wants to enforce diagnosability (respectively, detectability and attractability), then the maximally permissive solution obtained for K-diagnosability (respectively, K-detectability and K-attractability) may not be the maximally permissive solution for diagnosability (respectively, detectability and attractability). Moreover, as K increases, the permissiveness of the solution increases, but the complexity of the synthesis algorithm also increases, since we need to “unfold” the system for more steps. In other words, there is a tradeoff between the permissiveness of the solution and the complexity of synthesis algorithm when there is no delay K required a priori. In this case, one may proceed as follows. First, one may start with a solution by choosing a relatively small K. If the permissiveness of this solution satisfies the design requirement, then stop. Otherwise, choose a larger K and repeat the above procedure until a desirable solution is found.

2.8 Conclusion

In this chapter, we presented a uniform approach to the problem of synthesizing a maximally permissive supervisor that enforces a certain property for a partially-observed discrete-event system that does not originally satisfy the property. To this end, we defined a class of properties called Information-State-Based properties and a novel information structure called the All Enforcement Structure that embeds all valid supervisors enforcing any IS-based property. Based on the AES, a synthesis algorithm was provided to synthesize a locally maximal solution to this problem, without making any assumptions about the observability properties of the controllable events. In this regard, our approach relaxes the assumption that all controllable events are observable in the existing works on property enforcement by supervisory control. We showed that many important properties in the DES literature can be enforced by the uniform approach described in this chapter. Moreover, this approach can be applied to enforce other properties, such as anonymity, for which no synthesis methodologies exist in the current literature. In addition, the AES can be used for solving quantitative optimal property enforcement control problems when a cost structure is imposed on this problem. Since the AES embeds all valid property-enforcing supervisors, it provides a suitable solution space over which to solve such optimal control problems.

CHAPTER III

Synthesis of Non-blocking Supervisors for IS-Based Properties

3.1 Introduction

In Chapter II, we have proposed a uniform approach that is applicable to the enforcement of a large class of properties called the IS-based properties. Unfortunately, non-blockingness, one of the most important property in the supervisory control theory cannot be formulated as an IS-based property. This is because that non-blockingness requires that for any string in the closed-loop language, there *exists* a continuation of the string leading to a marking state; this information depends on the future behaviors of the system, which cannot be simply evaluated based on the current information-state.

In this chapter, we tackle the supervisor synthesis problem for non-blockingness *in addition to* IS-based property. We define another finite bipartite transition system that we call the "Non-Blocking All Enforcement Structure" (or NB-AES hereafter). The NB-AES contains in its transition structure all supervisors that are *deadlock-free*. We obtain the necessary and sufficient conditions for the solvability of the maximally permissive control problem. We then provide a synthesis algorithm, based on the NB-AES, that constructs a non-blocking and maximally permissive supervisor that enforces an IS-based property, if one exists. This is the first algorithm with such properties. Unlike the case of IS-based property, for which an IS-based supervisor is always sufficient when a solution exists, we show that 2^X may not be sufficient to represent a non-blocking supervisor; additional memory is required in general. However, we show that a finite memory is also sufficient to represent a non-blocking supervisor.

This Chapter is organized as follows. In Section 3.2, we revisit some basic terminologies and formulate the problem we want to solve. In Section 3.3, we define a new BTS called the NB-AES. In Section 3.4, we present an algorithm based on the NB-AES that returns a solution to the non-blocking synthesis problem (if one exists) and the correctness proof of the proposed algorithm. An illustrative example of our synthesis algorithm, for which previous approaches return empty solutions is provided in Section 3.5. Finally, we conclude this chapter in Section 3.6. In addition, Appendix 3.7.1 discusses in more detail implementation issues that arise in the synthesis algorithm of Section 3.5. The computational complexity of the synthesis algorithm of Section 3.5 is analyzed in Appendix 3.7.2.

3.2 Problem Formulation

Let $L \subseteq \mathcal{L}(\mathbf{G})$ be a prefix-closed language. We say that L is *non-blocking* (w.r.t. $\mathbf{G}$) if $\overline{L \cap \mathcal{L}_m(\mathbf{G})} = L$. Given an automaton $\mathbf{G}$, an *execution* is a sequence $\langle x_1, \sigma_1, \ldots, \sigma_{k-1}, x_k \rangle$, where $x_i \in X, \sigma_i \in \Sigma$ and $x_{i+1} = \delta(x_i, \sigma_i), \forall i \in \{1, 2, \ldots, k-1\}$. We say that an execution forms a *cycle* if $x_1 = x_k$; we say that a cycle is an *elementary cycle* if $\forall i, j \in \{1, 2, \ldots, k-1\} : i \neq j \Rightarrow x_i \neq x_j$. A Strongly Connected Component (SCC) in $\mathbf{G}$ is a maximal set of states $C \subseteq X$ such that $\forall x, y \in C, \exists s \in \Sigma^* : \delta(x, s) = y$; a SCC C is said to be non-trivial if $\forall x, y \in C, \exists s \in \Sigma^* \setminus \{\epsilon\} : \delta(x, s) = y$. A *livelock* in $\mathbf{G}$ is a non-trivial SCC C such that: (i) $C \cap X_m = \emptyset$, i.e., there is no marked state in it; and (ii) $\forall x \in C, \forall \sigma \in \Sigma : \delta(x, \sigma) \in C$, i.e., there is no transition defined out of it. We say that $\langle x_1, \sigma_1, \ldots, \sigma_{k-1}, x_k \rangle$ is an *elementary livelock cycle* if: (i) it is an elementary cycle; and (ii) there exists a livelock C, such that $\{x_1, x_2, \ldots, x_{k-1}\} \subseteq C$. We say that $L \subseteq \mathcal{L}(\mathbf{G})$ is a *livelock language* if any automaton generating L contains a livelock; otherwise, we say that L is *livelock-free.* Also, we say that $L \subseteq \mathcal{L}(\mathbf{G})$ is a *deadlock language* if $\exists s \in L : \Delta_L(s) = \emptyset \wedge s \notin \mathcal{L}_m(G)$; otherwise, we say that L is *deadlock-free.* Clearly, L is non-blocking if and only if it is both deadlock-free and livelock-free. We can also extend these concepts to a supervisor by evaluating its generated langauge. Specifically, let $S : P(\mathcal{L}(\mathbf{G})) \to \Gamma$ be a supervisor for $\mathbf{G}$. We say that S is

- *non-blocking* (w.r.t. $\mathbf{G}$) if $\mathcal{L}(S/\mathbf{G})$ is non-blocking w.r.t. $\mathbf{G}$
- *deadlock-free* (w.r.t. $\mathbf{G}$) if $\mathcal{L}(S/\mathbf{G})$ is deadlock-free w.r.t. $\mathbf{G}$.

Similarly to the Maximally Permissive IS-Based Property Enforcement Problem (MPIEP) we formulate the Non-blocking Maximally Permissive IS-Based Property Enforcement Problem (NB-MPIEP) as follows.

Problem 3. *(Non-blocking Maximally Permissive IS-Based Property Enforcement Problem). Given system $\boldsymbol{G}$ and IS-based property $\varphi : 2^X \to \{0, 1\}$ w.r.t. $\boldsymbol{G}$, synthesize a partial observation supervisor $S : P(\mathcal{L}(\boldsymbol{G})) \to \Gamma$, such that*

1. *$\mathcal{L}(S/\boldsymbol{G})$ is non-blocking w.r.t. $\boldsymbol{G}$;*
2. *$\mathcal{L}(S/\boldsymbol{G}) \models_{\boldsymbol{G}} \varphi$;*
3. *For any S' satisfying 1)-2), we have that $\mathcal{L}(S/\boldsymbol{G}) \not\subset \mathcal{L}(S'/\boldsymbol{G})$.*

3.3 Non-blocking All Enforcement Structure

In this section, we tackle the non-blockingness requirement. We first define the Non-Blocking AES (NB-AES), a bipartite transition system obtained from the AES that contains all *non-blocking* control policies satisfying the IS-based property; then we investigate its construction and properties.

3.3.1 Definition of the NB-AES

Definition 3.3.1. *(Live decision string). Given a BTS T, for any Y-state $y \in Q_Y^T$ and state $x \in y$ in it, we say that a decision string $\gamma_1\gamma_2\ldots\gamma_n$, where $\gamma_i \in \Gamma$ for $i = 1,\ldots n$, is* live *for (y,x) in T if there exists a string $s = \xi_1\sigma_1\xi_2\ldots\sigma_{n-1}\xi_n$, where $\xi_i \in (\Sigma_{uo} \cap \gamma_i)^*, \sigma_i \in \Sigma_o \cap \gamma_i$, such that $\delta(x,s) \in X_m$ and $\forall i < n : \gamma_{i+1} \in C_T(y_i)$, where $y \xrightarrow{\gamma_1\sigma_1\ldots\gamma_i\sigma_i}_T y_i$. We say that y is live in T if for any $x \in y$, (y,x) is live in T.*

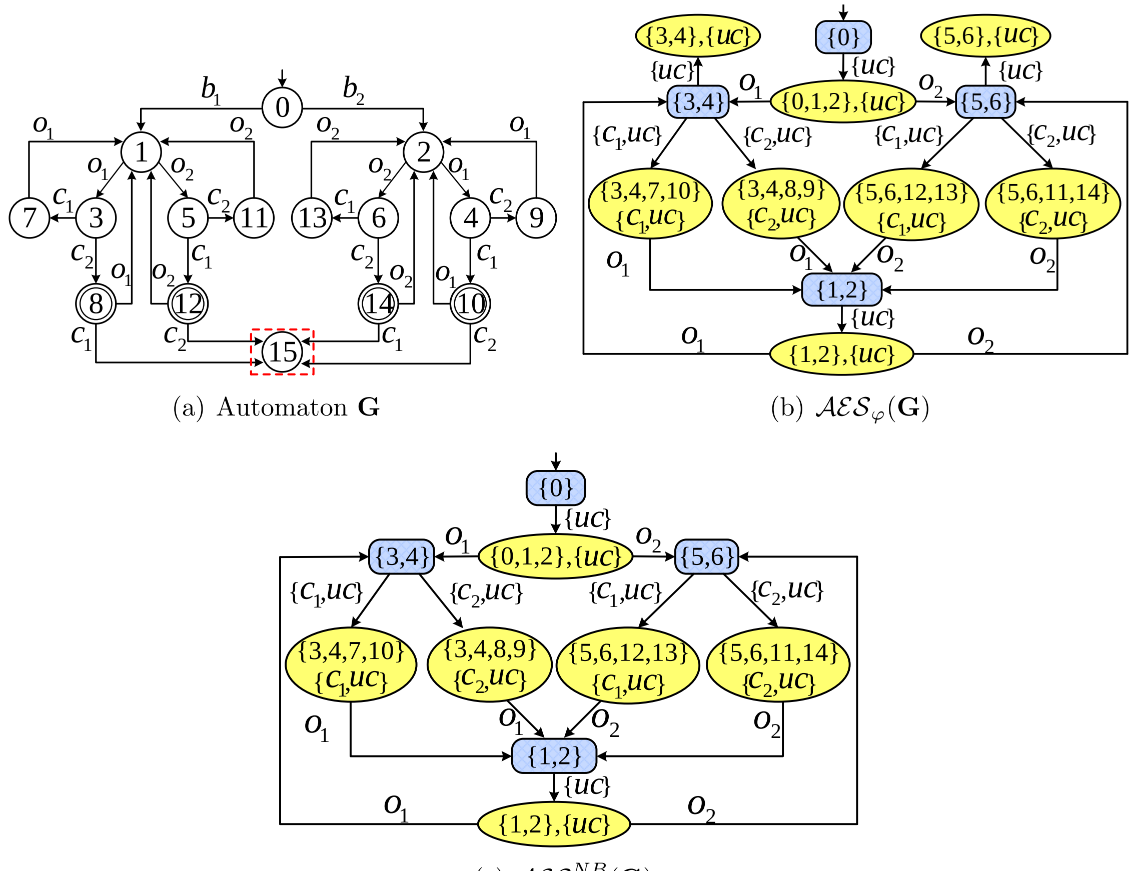

(a) Automaton **G**
(b) $\mathcal{AES}_\varphi(\mathbf{G})$
(c) $\mathcal{AES}_\varphi^{NB}(\mathbf{G})$

Figure 3.1: An example of (NB-)AES. For **G**: $\Sigma_c = \{c_1,c_2\}$, $\Sigma_o = \{o_1,o_2\}$ where state 15 is illegal. *uc* denotes all uncontrollable events.

Example 3.3.1. *Let $\boldsymbol{G}$ be the automaton shown in Figure 3.1(a). Let us consider safety specification φ defined by $\varphi(i) = 0 \Leftrightarrow 15 \in i$, i.e., state 15 is the unique illegal state. The resulting AES w.r.t. $\boldsymbol{G}$ for φ is shown in Figure 3.1(b). Then $\{uc\}\{c_2, uc\}$ is a live decision string for state $1 \in \{1, 2\}$, since string o_1c_2, which leads state 1 to marked state 8, exists under this decision string.*

Intuitively, the liveness property of a Y-state simply says that given a current information state, for each state in it, we can always find a sequence of control decisions under which this state will be able to reach some marked state through some string. The verification of the liveness property of a Y-state is a reachability problem in an automaton that is built from the original BTS by explicitly adding transitions to capture reachability within states in Z-states. Details can be found in the appendix.

The purpose of the above notion of liveness of information states is to eliminate one source of blocking: clearly, if a Y-state is not live, then no matter what control decision we take at that Y-state, we will always be blocked by some state in it.

In the case of Z-states, we introduce a notion of deadlock-freeness to complement the notion of liveness of Y-states. Specifically, for a Z-state z, we require that any state $x \in I(z)$ should either have an unobservable path to a marked state or a path that goes outside of the Z-state; otherwise, it will also be a source of blocking. This leads to the following definition, which depends on Z-state z and on $\mathbf{G}$, but not on the BTS that z is part of.

Definition 3.3.2. *(Deadlock-free Z-state). A Z-state z is said to be deadlock-free if for all $x \in I(z)$ we have*

$$(\exists s \in (\Gamma(z) \cap \Sigma_{uo})^*)[\delta(x, s) \in X_m] \vee (\exists s \in (\Gamma(z) \cap \Sigma_{uo})^*(\Gamma(z) \cap \Sigma_o))[\delta(x, s)!] \quad (3.1)$$

Otherwise, z is said to be a deadlock Z-state.

We are now ready to define the NB-AES structure, which contains all safe and non-blocking solutions.

Definition 3.3.3. *(Non-blocking All Enforcement Structure). Given a system $\boldsymbol{G}$ and an IS-based property $\varphi : I \to \{0,1\}$ w.r.t. $\boldsymbol{G}$, the Non-blocking All Enforcement Structure (NB-AES) for property φ, denoted by*

$$\mathcal{AES}_{\varphi}^{NB}(\boldsymbol{G}) = (Q_Y^{NB}, Q_Z^{NB}, h_{YZ}^{NB}, h_{ZY}^{NB}, \Sigma, \Gamma, y_0), \tag{3.2}$$

is defined as the largest complete *BTS w.r.t. $\boldsymbol{G}$ such that*

1 $\forall y \in Q_Y^{NB}$: y is live in $\mathcal{AES}_{\varphi}^{NB}(\boldsymbol{G})$; and

2 $\forall z \in Q_Z^{NB} : \varphi(I(z)) = 1$ and z is deadlock-free.

In the above definition, the largest non-blocking subsystem of the AES is uniquely defined, since the union of any subsystems satisfying the above properties still satisfies these properties. Similar to the case of the AES, we also only consider the reachable part of the NB-AES hereafter.

Example 3.3.2. *Going back to Figure 3.1, the NB-AES w.r.t. $\boldsymbol{G}$ for φ is shown in Figure 3.1(c). Comparing with its AES, since all Y-states in it are live, the deadlock Z-states that are removed are $(\{3,4\},\{uc\})$ and $(\{5,6\},\{uc\})$.*

3.3.2 Properties and Construction Algorithm

By definition, the NB-AES is also a complete BTS. Thus, we can talk about the properties of its included supervisors, which are given in the following theorem.

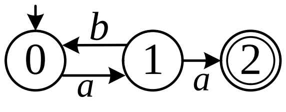
Theorem III.1. *For the set of NB-AES included supervisors, the following two properties are satisfied:*

1. *If $S \in \mathbb{S}(\mathcal{AES}_{\varphi}^{NB}(\boldsymbol{G}))$, then S is a* deadlock-free *supervisor satisfying φ;*

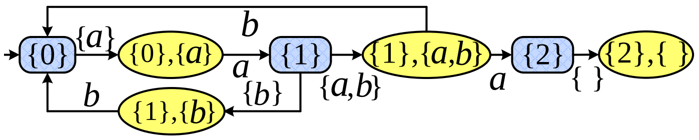
2. *If S is a* non-blocking *supervisor satisfying φ, then $S \in \mathbb{S}(\mathcal{AES}_{\varphi}^{NB}(\boldsymbol{G}))$.*

Proof. 1) Since the NB-AES is a subsystem of the AES, we know that $\mathcal{L}(S/\mathbf{G}) \models_{\mathbf{G}} \varphi$. Now, let us assume that L has a deadlock, which implies that there exists $s \in L$ such that $\delta(x_0, s) \notin X_m$ and $\delta_L(s) = \emptyset$. In terms of information state evolution, we know that $\delta(x_0, s) \in IS_S^Z(P(s))$. By Definition 3.3.2, this implies that the Z state $IS_S^Z(P(s))$ is a deadlock state, which contradicts the definition of the NB-AES. Thus, S is deadlock-free.

2) We prove by contrapositive, i.e., we show that if $S \notin \mathbb{S}(\mathcal{AES}_\varphi^{NB}(\mathbf{G}))$ then S cannot cannot simultaneously be non-blocking and φ-enforcing. Since the NB-AES is a subsystem of the AES, there are two cases for $S \notin \mathbb{S}(\mathcal{AES}_\varphi^{NB}(\mathbf{G}))$:
Case 1: $S \notin \mathbb{S}(\mathcal{AES}_\varphi(\mathbf{G}))$. By Theorem II.1, S is not φ-enforcing.
Case 2: $S \in \mathbb{S}(\mathcal{AES}_\varphi(\mathbf{G}))$ but $S \notin \mathbb{S}(\mathcal{AES}_\varphi^{NB}(\mathbf{G}))$. We now show that in this case S is blocking. By Definition 3.3.3, it can be shown by contradiction that there exists $s \in \mathcal{L}(S/\mathbf{G})$ such that one of the two following cases holds: (i) $IS_S^Z(P(s))$ is a deadlock Z-state. By Definition 3.3.2, L is blocking; (ii) $IS_S^Y(P(s))$ is not live. If $y = IS_S^Y(P(s))$ is not live, then by Definition 3.3.1, there exists at least one state in y where no control decision can be made to lead it to a marked state. Specifically, $(\exists t \in \mathcal{L}(S/\mathbf{G}) : P(t) = P(s))(\forall v \in \Sigma^* : tv \in \mathcal{L}(S/\mathbf{G}))[\delta(x_0, tv) \notin X_m]$. Thus, S is blocking. □



(a) Automaton **G**



(b) The corresponding NB-AES

Figure 3.2: For **G**: $\Sigma_{uo} = \emptyset$ and $\Sigma_{uc} = \{b\}$.

Note that for $S \in \mathbb{S}(\mathcal{AES}_\varphi^{NB}(\mathbf{G}))$, S need not be livelock-free in general. Let us consider the automaton **G** in Figure 3.2(a) and its corresponding NB-AES shown in Figure 3.2(b). Clearly, supervisor S such that $\mathcal{L}(S/\mathbf{G}) = (ab)^*$ is included in the NB-AES, but it is a livelock supervisor. However, the above statement is true when **G** is acyclic, i.e., there is no cycle in $\mathbf{G}$, since in this case, the deadlock-freeness condition and the non-blockingness condition are equivalent. Therefore, we have the following result.

Corollary 3.3.1. *If $\boldsymbol{G}$ is acyclic, then S is non-blocking and φ-enforcing* iff $S \in \mathbb{S}(\mathcal{AES}_{\varphi}^{NB}(\boldsymbol{G}))$.

Algorithm 3: FIND-NB-AES

```
input  : $\mathcal{AES}_{\varphi}(\mathbf{G})$
output: $\mathcal{AES}_{\varphi}^{NB}(\mathbf{G})$
1    $A \leftarrow$FIND-AES($\mathbf{G}$);
2    Delete all $Z$-states in $A$ that are deadlock states;
3    while exists $Y$-state in $A$ that is not live do
4        Delete all $Y$-states in $A$ that are not live;
5        while exists $Y$-state in $A$ that has no successor do
6            Delete all such $Y$-states in $A$ and delete all their predecessor
               $Z$-states;
7    if the initial $Y$-state has been removed then
8        return the NB-AES does not exist;
     else
9        $\mathcal{AES}_{\varphi}^{NB}(\mathbf{G}) \leftarrow$ Accessible($A$);
```

The construction procedure for the NB-AES is given by Algorithm FIND-NB-AES. The basic idea of the construction algorithm follows directly from the definition. We need to keep pruning states from the AES structure until convergence. Specifically, there are three kinds of states that we need to prune:

(i) All Z-state that are deadlock states;

(ii) All Y-states that are not live; and

(iii) All Y or Z-states that violate the definition of completeness.

In the algorithm, the elimination of (i), (ii) and (iii) are implemented in line-2, line-4 and line-6, respectively. Note that for (ii) and (iii), iteration steps are required, since pruning states may change the liveness or the completeness of the transition system. However, (i) just needs to be executed once, since the deadlock property does not depend on T.

Proposition 3.3.1. *The running time of FIND-NB-AES is in* $O(|X||\Sigma|2^{2|X|+|\Sigma_c|-1})$.

Proof. The proof is given in the appendix. □

3.4 Synthesis of Non-blocking Supervisors

3.4.1 Synthesis Algorithm

We now tackle the synthesis problem for non-prefix-closed specification languages, i.e., non-blockingness must be ensured in addition to an IS-based property. Formally, we show how to synthesize a maximal non-blocking supervisor from the NB-AES.

In the prefix-closed case, once the AES is built, we can randomly pick *one* control decision and fix it at *each* reachable information state and this will give us a (IS-based) supervisor for IS-based property φ. However, this strategy may not work in the non-prefix-closed case, since the NB-AES only guarantees that there *exists* a good decision, but arbitrarily choosing one control decision may return a livelock solution. This phenomenon was already pointed out by the example in Figure 3.2. Moreover, if we go back to the example in Figure 3.2, we find that we cannot remove any (Y or Z) state from the NB-AES, otherwise, some nonblocking solutions will be excluded. This means that the NB-AES is already the most "compact" structure that contains all non-blocking solutions, even if it contains some livelock solutions. One conjecture is that we can search through the space of IS-based supervisors, which is finite, for the desired maximal solution. Unfortunately, an IS-based solution does not exist in general; an example where this occurs is presented in Section 3.5.

The non-existence, in general, of an IS-based supervisor that is both φ-enforcing and non-blocking implies immediately that state space refinement is required if we want to synthesize a solution from the NB-AES. Our synthesis algorithm, which is described formally below, is based on the idea of suitably "unfolding" the NB-AES. To begin with, we need to build an IS-based supervisor (**Step 1**) and then determine whether or not there exists a livelock in it (**Step 2**). If not, then we are done and return the solution. If yes, then we need to break the livelock at some point and resolve it by unfolding the NB-AES at that point such that a live decision string can be added at the livelock point (**Steps 3 and 4**). This will give us a new (non-IS-based) supervisor. Finally, we need to go back to **Step 2** and test again until the iteration converges (**Step 5**). However, two questions arise: (i) *Where* should we break a livelock? and (ii) *How* can we unfold the NB-AES? In order to answer these two questions, we first define the concept of "extended BTS" and then we use this notion to define "unfolded" BTS.

Let $\mathbb{Z}$ be the set of integers and $\mathbb{N}$ be the set of non-negative integers. E is called an Extended Bipartite Transition System (EBTS) of T if it is a partial unfolding of a BTS T resulting in sets $Q_Y^E = Q_Y^T \times \mathbb{Z}$ and $Q_Z^E = Q_Z^T \times \mathbb{Z}$ with corresponding transition functions $h_{YZ}^E : Q_Y^E \times \Gamma \to Q_Z^E$ and $h_{ZY}^E : Q_Z^E \times \Sigma \to Q_Y^E$ over the extended state space, such that the restrictions of h_{YZ}^E and h_{ZY}^E to domains Q_Y^T and Q_Z^T, respectively, are consistent with h_{YZ}^T and h_{ZY}^T whenever h_{YZ}^E and h_{ZY}^E are defined. Specifically, $h_{YZ}^E((y,n),\gamma)$ (respectively, $h_{ZY}^E((z,n),\sigma)$) is of the form $(h_{YZ}^T(y,\gamma), \delta(y,n,\gamma))$ (respectively, $(h_{ZY}^T(z,\sigma), \delta(z,n,\sigma))$), where $\delta : (Q_Y^T \cup Q_Z^T) \times \mathbb{Z} \times (\Gamma \cup E) \to \mathbb{Z}$ is some updating function for the integer component of the state. (The exact form of δ is left unspecified for the purpose of this general definition.) Given an EBTS E, its included supervisors is defined analogously as before for a BTS in Definition 2.4.2 we will still use the notations $\mathbb{S}(E)$ to represent the supervisors included in E. Clearly, if E is a complete EBTS of a complete BTS T, then $\mathbb{S}(E) \subseteq \mathbb{S}(T)$.

The definition of an EBTS only requires that the restriction of the transition function to domains Q_Y^T and Q_Z^T be consistent with the BTS. However, we also want that the restriction of the transition function to domain $\mathbb{Z}$ satisfy certain rules (namely, it should "remember" the number of times the current state has been visited). This leads to the notion of an Unfolded Bipartite Transition System (UBTS), which is a particular type of EBTS defined as follows. For simplicity, we will write state (y, n) as y^n. Given an extended state $x^n \in Q_Y^E \cup Q_Z^E$, $Pre_Y^E(x^n)$ and $Pre_Z^E(x^n)$ denote, respectively, the set of Y-states and the set of Z-states that can reach this state through some runs in E, excluding itself; also, we call x^n a *control state* if $n \in \mathbb{N}$ and a *transient state* if $n \in \mathbb{Z} \setminus \mathbb{N}$.

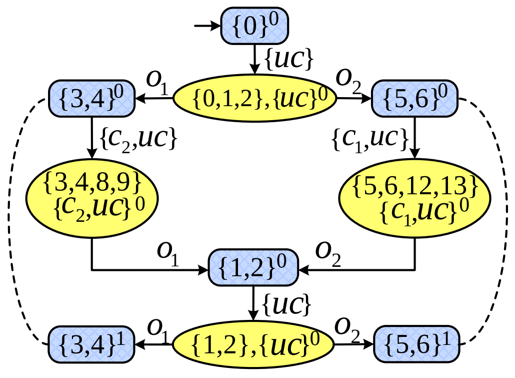
Definition 3.4.1. *We say that U is an unfolded BTS of a complete BTS T if it is an EBTS of T, such that:*

1. $(\forall y^n \in Q_Y^U)[|C_U(y^n)| \leq 1]$;
2. $(\forall z^n \in Q_Z^U)(\forall \sigma \in \Sigma)[h_{ZY}(z, \sigma)! \Rightarrow h_{ZY}^U(z^n, \sigma)!]$;
3. *There are no cycles in U;*
4. *For any $y^n \in Q_Y^U$, if $n \in \mathbb{N}$, then $n = |\{y^{\tilde{n}} \in Pre_Y^U(y^n) : \tilde{n} \in \mathbb{N}\}|$. Similarly, for any $z^n \in Q_Z^U$, if $n \in \mathbb{N}$, then $n = |\{z^{\tilde{n}} \in Pre_Z^U(z^n) : \tilde{n} \in \mathbb{N}\}|$.*
5. *The terminal states of U are either (i) terminal Z-states or (ii) Y-states of the form y^n with $n \geq 1$.*

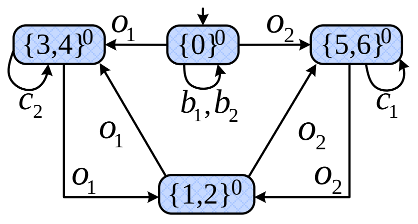
For brevity, hereafter, we also write $y^n \xrightarrow{c}_U z^{n'}$ for $h_{YZ}^U(y^n, c) = z^{n'}$ and $z^n \xrightarrow{\sigma}_U y^{n'}$ for $h_{ZY}^U(z^n, \sigma) = y^{n'}$.

Conditions 1) and 2) together imply that except for Y-states with no defined control decision, a UBTS will be complete. Condition 4) says that the integer component of any control state in U is n if there are n control states in its predecessors that have the same Y- or Z-state component. By condition 5), any branch of the UBTS ends up with a repeated control Y-state or a terminal Z-state. Thus, given a UBTS U, we can merge each terminal Y-state $y^n, n \geq 1$ with its predecessor state y^0 and denote the resulting new EBTS by $\tilde{U}$. Specifically, $\tilde{U}$ is obtained by removing states $R := \{y^n \in Q_Y^U : |C_U(y^n)| = 0\}$ from U and for any $y^n \in R$, any transition that originally goes to state y^n in U will go to the corresponding state y^0 in $\tilde{U}$. By definition of a UBTS, $\tilde{U}$ is a complete EBTS. Moreover, we note that the set of supervisors $\mathbb{S}(\tilde{U})$ included in $\tilde{U}$ is a singleton, since there is only one control decision at each Y-state in $\tilde{U}$. Thus, we call the unique supervisor included in $\tilde{U}$ the *supervisor induced by UBTS U* and denote it by S_U. Similarly, for any Y-state $y \in Q_Y^{\tilde{U}}$, we denote by $c_y^{\tilde{U}}$ the unique control decision defined at y, i.e., $C_{\tilde{U}}(y) = \{c_y^{\tilde{U}}\}$. The supervisor S_U can be *realized* by an automaton $A_U = (Q_Y^{\tilde{U}}, \Sigma, \xi, q_0, Q_Y^{\tilde{U}})$, where q_0 is the initial Y-state of $\tilde{U}$ and $\xi : Q_Y^{\tilde{U}} \times \Sigma \to Q_Y^{\tilde{U}}$ is a partial function defined by: for any $q \in Q_Y^{\tilde{U}}, \sigma \in \Sigma$, we have (i) $\xi(q, \sigma) = q$ if $\sigma \in c_y^{\tilde{U}} \cap \Sigma_{uo}$; (ii) $\xi(q,\sigma) = h_{ZY}^{\tilde{U}}(h_{YZ}^{\tilde{U}}(y, c_y^{\tilde{U}}), \sigma)$ if $\sigma \in c_y^{\tilde{U}} \cap \Sigma_o$; and (iii) $\xi(q, \sigma)$ is undefined if $\sigma \notin c_y^{\tilde{U}}$. Then we can compute the controlled behavior by $\mathcal{L}(S_U/\mathbf{G}) = \mathcal{L}(A_U \times \mathbf{G})$, where "$\times$" denotes the usual product composition operation of automata; see, e.g., [12] (p. 78).

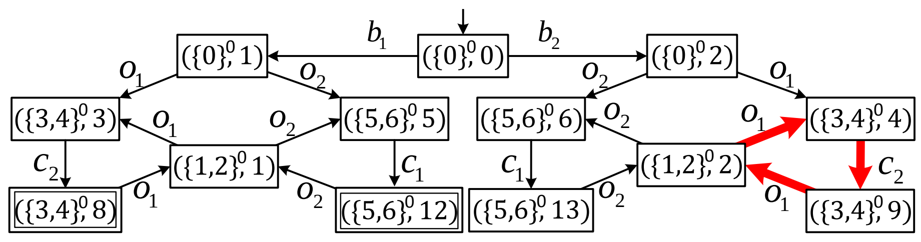
If $\mathcal{L}(S_U/\mathbf{G})$ is a livelock language, then there exists an elementary livelock cycle $\langle q_1, \sigma_1, \ldots, \sigma_{k-1}, q_k \rangle$ in $A_U \times \mathbf{G}$ such that $\exists i \in \{1, \ldots, k-1\} : \sigma_i \in \Sigma_o$, since U only contains deadlock-free Z-states. We call such a cycle a Critical Elementary Livelock Cycle (CELC). In our problem, any CELC in a livelock of $A_U \times \mathbf{G}$ corresponds to the presence of some elementary cycle in $\tilde{U}$. Moreover, since a cycle in $\tilde{U}$ is obtained by merging some terminal Y-state y^m and its corresponding y^0 in U, then for a CELC, there exists some terminal Y-state in U that leads to it. We call such a terminal Y-state an *entrance Y-state* of the CELC. More specifically, let $\langle q_1, \sigma_1, \ldots, \sigma_{k-1}, q_k \rangle$ be a CELC. Note that q_i is in the form of $(y_i^{n_i}, x_i)$. Then, there exists an observable event $\sigma_i, i \in \{1, \ldots, k-1\}$ such that $q_{i+1} = (y_{i+1}^0, x_{i+1})$ but $h_{ZY}^U(h_{YZ}^U(y_i^{n_i}, c_{y_i^{n_i}}^{\tilde{U}}), \sigma_i) = y_{i+1}^m, m \neq 0$, where $c_{y_i^{n_i}}^{\tilde{U}}$ is the unique control decision defined at $y_i^{n_i}$ in $\tilde{U}$. In other words, y_{i+1}^m is a terminal Y-state of U, which is not in $\tilde{U}$. Then y_{i+1}^m is an entrance



(a) UBTS U_0 (without the dashed lines)



(b) A_{U_0}



(c) $\mathcal{L}(S_{U_0}/\mathbf{G}) = \mathcal{L}(A_{U_0} \times \mathbf{G})$

Figure 3.3: Example of Steps 1 and 2.

Y-state of the CELC and we call $x_{i+1} \in y_{i+1}$ a corresponding state in the entrance Y-state. In Definition 3.3.1, we introduced the notion of live decision string for a state pair $(y, x), y \in Q_Y^T, x \in y$ in a BTS T. We say that a live decision string $\gamma_1\gamma_2\ldots\gamma_n$ is *locally maximal* for (y, x) if there does not exist another live decision string $\gamma_1'\gamma_2'\ldots\gamma_n'$ for (y, x) in T such that $\forall i \in \{1, 2, \ldots, n\} : \gamma_i \subseteq \gamma_i'$ and $\exists j \in \{1, 2, \ldots, n\} : \gamma_j \subset \gamma_j'$.

Example 3.4.1. *Consider the automaton $\boldsymbol{G}$ shown in Figure 3.1. An example of UBTS is given in Figure 3.3(a); it is an unfolding of $\mathcal{AES}_\varphi^{NB}(\boldsymbol{G})$. By merging state pairs $(\{3,4\}^0, \{3,4\}^1)$ and $(\{5,6\}^0, \{5,6\}^1)$ in U_0 (connected by the dashed lines), we get the corresponding EBTS $\tilde{U}_0$. The induced supervisor S_{U_0} is realized by the automaton A_{U_0} shown in Figure 3.3(b). The language of the controlled system $\mathcal{L}(S_{U_0}/\boldsymbol{G}) = \mathcal{L}(A_{U_0} \times \boldsymbol{G})$ is given in Figure 3.3(c). By the properties of the NB-AES, we know that S_{U_0} is φ-enforcing and deadlock-free. However, we see that it is blocking. In $A_{U_0} \times \boldsymbol{G}$, we see that $\langle(\{3,4\}^0, 4), c_2, (\{3,4\}^0, 9), o_1, (\{1,2\}^0, 2), o_1, (\{3,4\}^0, 4)\rangle$ is a CELC, which is due to the presence of the cycle $\{3,4\}^0 \to \{1,2\}^0 \to \{3,4\}^0$ in $\tilde{U}_0$*

(we omit the Z-states in the cycle since they are uniquely determined). Therefore, $\{3,4\}^1$ is an entrance Y-state of this CELC and $4 \in \{3,4\}$ is a corresponding state in it.

We are now ready to state our synthesis algorithm, which is formally presented in Algorithm NB-SOLU. For the sake of readability, we decompose Algorithm NB-SOLU into five steps that are mapped to the corresponding lines in the statement of the algorithm.

Step 1: Generate an initial UBTS (lines 1-2): The goal of this step is to initially generate an IS-based supervisor via building a UBTS from the NB-AES. First, we set U_0 to be the UBTS that only contains the initial state y_0^0 of the NB-AES and call procedure EXPAND (lines 13-26). This procedure expands the initial state and constructs a UBTS by a breadth-first search in the NB-AES. First, pick a locally maximal control decision for y_0^0; then, for the Z-state encountered, find all its Y-state successors and pick one locally maximal control decision for each of them, and so forth, until: (i) a terminal Z-state is reached; or (ii) a Y-state y^n whose information state component has already been visited is reached, i.e., $n \neq 0$. Note that, all the states added by EXPAND are control states, since the integer components are always greater than or equal to zero. Since the construction procedure stops once a Y-state is repeated, the largest index for a Y-state in the UBTS at this step should be 1 and the UBTS induced supervisor is IS-based. Note that the language $\mathcal{L}(S_{U_0}/\mathbf{G})$ is a maximal language, since we take locally maximal control decisions in the construction procedure; however, it may be blocking in general.

Step 2: Detect livelock (lines 4-5): The goal of this step is to detect a livelock (if one is present) and find a state where it can be properly broken. If $\mathcal{L}(S_{U_i}/\mathbf{G})$ is livelock-free, then we stop the algorithm and return the current UBTS as the solution. If not, we need to find *one* CELC causing livelock and a corresponding entrance Y-state, as defined earlier.

Algorithm 4: NB-SOLU)

```
input  : $\mathcal{AES}_{\varphi}^{NB}(\mathbf{G})$
output: $S_{U_k}$
1  Set $i \leftarrow 0, Q_Y^{U_i} \leftarrow \{y_0^0\}, M = 0$;
2  EXPAND($U_i$);
3  $i \leftarrow i + 1, U_i \leftarrow U_{i-1}$;
   while $\mathcal{L}(S_{U_{i-1}}/\boldsymbol{G})$ is a livelock language do
4      find an entrance state $y_e^k \in Q_Y^{U_i}$ for one CELC and a corresponding
        state $x_e \in y_e$ that is also in the livelock.;
5      Find a locally maximal live decision string $\gamma_1\gamma_2 \ldots \gamma_n$ for $(y_e, x_e)$ in the
        NB-AES.;
6      From state $y_e^k$, augment $U_i$ with run $\gamma_1\sigma_1 \ldots \sigma_{n-1}\gamma_n$ and the $Y$ and
        $Z$-states reachable along its prefixes, where $\sigma_j$ is defined in Def. 3.3.1.
        Specifically, we augment $U_i$ with the following transitions:

        $$y_e^k \xrightarrow{\gamma_1}_{U_i} z_1^{k_1} \xrightarrow{\sigma_1}_{U_i} y_1^{k_2} \ldots \xrightarrow{\sigma_{n-1}}_{U_i} y_{n-1}^{k_{2n-2}} \xrightarrow{\gamma_n}_{U_i} z_n^{k_{2n-1}}$$

        where the values of $y_j$ and $z_j$ are determined by $h_{ZY}$ and $h_{YZ}$,
        respectively, by the definition of an EBTS and $k_j = M - j$, for any
        $j = 1, \ldots, 2n - 1$.;
7      $M \leftarrow M - 2n + 1$.;
8      EXPAND($U_i$);
9      $i \leftarrow i + 1, U_i \leftarrow U_{i-1}$;
10 return $S_{U_{i-1}}$;
   procedure Expand($U$);
11 while $\exists y^n \in Q_Y^U$ such that $C_U(y^n) = \emptyset \wedge n = 0$ or $\exists z^n \in Q_Z^U$ such that

        $$\exists \sigma \in \Gamma(z) \cap \Sigma_o : h_{ZY}(z, \sigma)! \wedge h_{ZY}^U(z^n, \sigma) \text{ is not defined} \qquad (3.3)$$

     do
12     for $y^n \in Q_Y^U$ such that $C_U(y^n) = \emptyset \wedge n = 0$ do
13         Find a control decision $\gamma \in C_{\mathcal{AES}_{\varphi}^{NB}(\mathbf{G})}(y)$ in $\mathcal{AES}_{\varphi}^{NB}(\mathbf{G})$ such that
            $\forall \gamma' \in C_{\mathcal{AES}_{\varphi}^{NB}(\mathbf{G})}(y) : \gamma \not\subset \gamma'$;
14         Augment $U$ with transition: $y^n \xrightarrow{\gamma}_U z^{n'}$, where $z = h_{YZ}(y, \gamma)$ and

            $$n' = |\{\tilde{z}^{\tilde{n}} \in Pre_Z^U(y^n) : \tilde{z} = z \text{ and } \tilde{n} \geq 0\}|$$

15     for $z^n \in Q_Z^U$ such that (3.3) holds do
16         for $\sigma \in \Gamma(z) \cap \Sigma_o$ satisfying (3.3) do
17             Augment $U$ with transition: $z^n \xrightarrow{\sigma}_U y^{n'}$, where $y = h_{ZY}(z, \sigma)$ and

                $$n' = |\{\tilde{y}^{\tilde{n}} \in Pre_Y^U(z^n) : \tilde{y} = y \text{ and } \tilde{n} \geq 0\}|$$
```

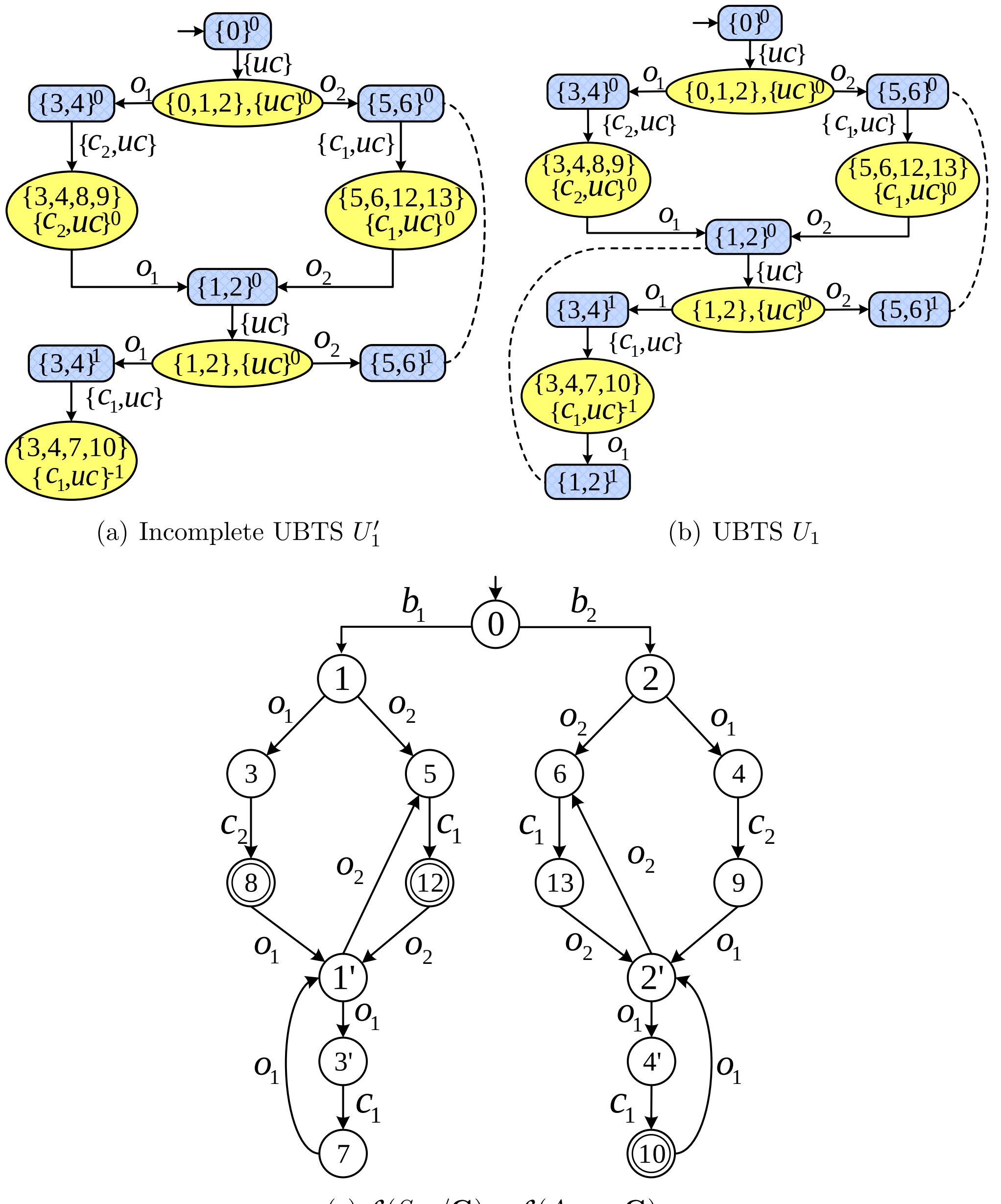
Step 3: Resolve livelock (lines 6-7): This step aims to resolve the livelock found in Step 2. Specifically, we unfold the UBTS from an entrance Y-state of the livelock by finding a live decision string in the NB-AES. The states added at this step are transient states and we use a global variable M in Algorithm NB-SOLU to remember how many transient states we have added to U. Consequently, all the transient states in U have different (negative) integer components. Also, to achieve maximality, all newly added control decisions are locally maximal.

Remark 3.4.1. To find such locally maximal live decision strings, one approach is to first find an arbitrary live string and then sequentially replace each control decision in it by a larger one, whenever feasible, from γ_1 to γ_n. A formal algorithm for this construction is given in the appendix.

Step 4: Complete the UBTS (line 8): After Step 3, the resulting transition system may no longer be a UBTS. Thus, we need to complete U_i as a UBTS such that we can again induce a supervisor from it. This step is implemented by calling again the procedure EXPAND, which finds one control decision for each Y-state that has no successors, and adds all observations for each Z-state that has some defined observations (i.e., is not terminal).

Step 5: Iteration: Finally, we need to go back to **Step 2** until the iteration stops, i.e., until all livelocks have been resolved.

Example 3.4.2. *Consider the automaton $\boldsymbol{G}$ and its NB-AES from Figure 3.1. Consider the UBTS U_0 and its induced language $\mathcal{L}(S_{U_0}/\boldsymbol{G})$ shown in Figure 3.3. We see that U_0 is a valid UBTS generated after **Step 1**, which ends up with the repeated Y-states $\{3,4\}^1$ and $\{5,6\}^1$, but it induces a livelock solution. Consider the CEL-C highlighted in Figure 3.3 as we have discussed in Example 3.4.1. In **Step 2**, we find that $y_e = \{3,4\}^1$ is an entrance Y-state of this livelock and return $(\{3,4\}^1, 4)$. For **Step 3**, one possible choice is to take control decision $\{c_1, uc\}$ at $\{3,4\}^1$, since state 4 will be able to reach marked state 10 via c_1. Therefore, a transient*



(a) Incomplete UBTS U_1'

(b) UBTS U_1

(c) $\mathcal{L}(S_{U_1}/\mathbf{G}) = \mathcal{L}(A_{U_1} \times \mathbf{G})$

Figure 3.4: Example of Steps 3, 4 and 5. Note that states in $A_{U_1} \times \mathbf{G}$ have been renamed for simplicity.

Z-state $(\{3,4,7,10\},\{c_1,uc\})^{-1}$ is added and the resulting BTS U_1' is shown in Figure 3.4(a). However, in U_1', the enabled observable event o_1 is not defined at Z-state $(\{3,4,7,10\},\{c_1,uc\})^{-1}$. Thus, ***Step 4*** *will call procedure EXPAND again to complete the UBTS by adding a new Y-state $\{1,2\}^1$ that can be reached by observing o_1 into U_1'. Since $\{1,2\}$ already exists in the UBTS, we stop the procedure EXPAND and get U_1 shown in Figure 3.4(b) and its induced language $\mathcal{L}(S_{U_1}/\boldsymbol{G})$ is shown Figure 3.4(c). Since $\mathcal{L}(S_{U_1}/\boldsymbol{G})$ is livelock-free, we stop the synthesis procedure and return it as a maximal controllable, observable, safe, and non-blocking solution.*

Remark 3.4.2. In Figure 3.3(a), we could also select control decision $\{c_2, uc\}$ at state $\{5,6\}^0$. It can be easily verified that this will induce a non-blocking and IS-based solution. Thus we can stop the synthesis at Step 2 and return this solution. However, as discussed earlier, the above situation may not always hold. This is why we chose the non-IS-based solution to illustrate all the steps of Algorithm NB-SOLU.

3.4.2 Correctness of the Synthesis Algorithm

In this section, we show that (i) the synthesis algorithm presented in the previous section converges in a finite number of iterations and (ii) the resulting solution is maximal.

In the synthesis steps of Algorithm NB-SOLU, the supervisor should not only know its current information state, but it also needs to remember the number of times the current state has been visited. However, this does not tell us how much memory we need to realize the supervisor. The following theorem reveals that the supervisor can be represented in a finite structure, i.e., the resulting language is regular.

Theorem III.1. *Algorithm NB-SOLU converges in a finite number of iterations.*

Proof. Suppose that $x \in y^n$ is detected in $A_{U_i} \times \mathbf{G}$ at **Step 2**, where x is a corresponding state of an entrance Y-state y^n, $n \geq 1$ and $i \geq 1$. This implies that there exists a CELC $\langle (y^0, x), \sigma_1, \ldots, \sigma_k, (y^0, x)\rangle$ in $A_{U_i} \times \mathbf{G}$. We define the pair being *resolved* for the CELC as the last state in the CELC before the final state (y^0, x) such that (i) its first component is y^0; and (ii) it is entered by an observable event. More specifically, we can write this CELC in the form of

$$(y^0, x) \xrightarrow{\sigma_1^1\sigma_2^1\ldots\sigma_{k_1}^1} (y^0, x^2) \xrightarrow{\sigma_1^2\sigma_2^2\ldots\sigma_{k_2}^2} (y^0, x^3) \cdots \xrightarrow{\sigma_1^j\sigma_2^j\ldots\sigma_{k_j}^j} (y^0, x^{j+1}) \xrightarrow{\sigma_1^{j+1}\sigma_2^{j+1}\ldots\sigma_{k_{j+1}}^{j+1}} \cdots$$
$$\xrightarrow{\sigma_1^{r-1}\sigma_2^{r-1}\ldots\sigma_{k_{r-1}}^{r-1}} (y^0, x^r) \xrightarrow{\sigma_1^r\sigma_2^r\ldots\sigma_{k_r}^r} (y^0, x) \tag{3.4}$$

where $\sigma_{k_j}^j \in \Sigma_o, j = 1, \ldots, r$ and (y^0, x^r) is the pair being *resolved*. Note that, inside of the CELC, cycle $\langle y^0, \ldots, y^{n-1}, y^0\rangle$ in A_{U_i} may be involved for r times. Figure 3.5 illustrates the notions of detected and resolved states in the context of the CELC.

In order to prove the theorem, it suffices to prove that any pair $(y^0, x), x \in y$ can be resolved at most once in **Step 3**. To see this, let us first suppose that $x \in y^n$ was detected in $A_{U_i} \times \mathbf{G}$ at **Step 2**, where $n \geq 1$ and $i \geq 1$. Let $(y^0, x^r), x^r \in y$ be the corresponding pair being resolved. Note that x^r and x need not necessarily be the same state. Since **Step 3** is executed after detecting $x \in y^n$, the above CELC will be broken and a path from $x \in y^n$ to a marked state is introduced. Formally, in $A_{U_{i+1}} \times \mathbf{G}$, we know that $f_{[A_{U_{i+1}} \times \mathbf{G}]}((y^0, x^r), \sigma_1^r\sigma_2^r \ldots \sigma_{k_r}^r) = (y^n, x)$ and there exists a string $t \in \Sigma^*$ such that $f_{[A_{U_{i+1}} \times \mathbf{G}]}((y^0, x^r), \sigma_1^r\sigma_2^r \ldots \sigma_{k_r}^r t) \in Q_Y^{U_{i+1}} \times X_m$, where $f_{[A_{U_{i+1}} \times \mathbf{G}]}$ denotes the transition function of $A_{U_{i+1}} \times \mathbf{G}$. This path is also illustrated in Figure 3.5. In fact, t can be the corresponding live path for the live decision string introduced at y^n as defined in Definition V.1.

Now, let us assume that after some iteration steps, $x^r \in y^0$ is resolved again for a CELC in $A_{U_{i+q}} \times \mathbf{G}, q \geq 2$. This means that (y^0, x^r) is in a livelock of $A_{U_{i+q}} \times \mathbf{G}$. Moreover, we have already shown that there exists a string $\sigma_1^r\sigma_2^r \ldots \sigma_{k_r}^r t$ from (y^0, x^r) to a marked state in $A_{U_{i+1}} \times \mathbf{G}$. Such a marked state is also reachable from (y^0, x^r) in $A_{U_{i+q}} \times \mathbf{G}$, since U_{i+q} is unfolded from U_{i+1} and any states reachable in U_{i+1} are

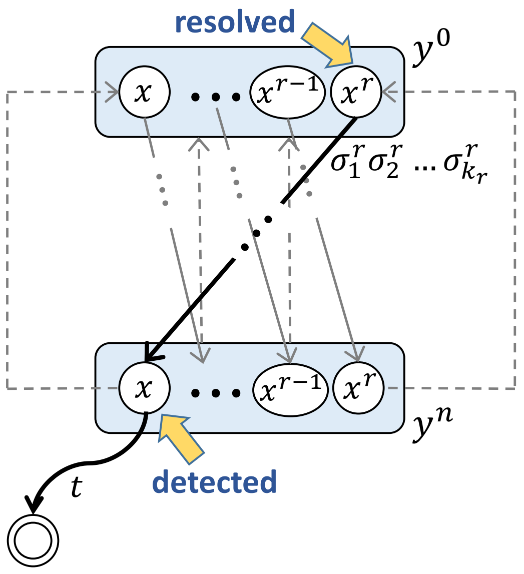


Figure 3.5: Conceptual illustration of the proof of Theorem III.1.

still reachable in U_{i+q}. More specifically, such a marked state can be reached from (y^0, x^r) via the same string $\sigma_1^r\sigma_2^r \ldots \sigma_{k_r}^r t$ as above. Therefore, (y^0, x^r) cannot be in any livelock, which gives us a contradiction. Thus we conclude that any pair $(y, x), x \in y$ can be resolved at most once in **Step 3**.

Let the set of Y-states of $\mathcal{AES}_{\varphi}^{NB}(\mathbf{G})$ be denoted by Q_Y^{NB}. Then, $\sum_{y\in Q_Y^{NB}} |y| \leq \sum_{i=1}^{|X|} i\binom{|X|}{i} = |X|2^{|X|-1}$ gives an upper bound for the number of iterations. □

Proposition 3.4.1. *If the NB-AES has been constructed, then the running time of Algorithm NB-SOLU is* $O([|X|^3 2^{|X|} + |\Sigma|]|X|2^{2|X|+|\Sigma|})$.

Proof. The proof is given in the appendix. □

Suppose that Algorithm NB-SOLU stops after n steps of iteration and returns UBTS U_n; then the induced supervisor S_{U_n} has the following properties.

Theorem III.2. *S_{U_n} is a non-blocking supervisor that enforces φ.*

Proof. Follows directly from Theorem III.1 and the livelock-free stopping condition in **Step 2**. □

Theorem III.3. *S_{U_n} is maximal, i.e., for any non-blocking S' satisfying φ, we have $\mathcal{L}(S_{U_n}/\boldsymbol{G}) \not\subset \mathcal{L}(S'/\boldsymbol{G})$.*

Proof. We prove this theorem by contradiction. Assume that $\mathcal{L}(S_{U_n}/\mathbf{G})$ is not maximal, i.e., $\exists S' \in \mathbb{S}(\mathcal{AES}_{\varphi}^{NB}(\mathbf{G}))$ such that S' is non-blocking and $\mathcal{L}(S_{U_n}/\mathbf{G}) \subset \mathcal{L}(S'/\mathbf{G})$. This implies the following two facts[1]:

1. $(\forall s \in \mathcal{L}(S_{U_n}/\mathbf{G}))[S_{U_n}(s) \subseteq S'(s)]$;
2. $(\exists t \in \mathcal{L}(S_{U_n}/\mathbf{G}))[S_{U_n}(t) \subset S'(t)]$.

Let us consider the string $t \in \mathcal{L}(S_{U_n}/\mathbf{G})$ such that $S_{U_n}(P(t)) \subset S'(P(t))$ and $S_{U_n}(t') = S'(t'), \forall t' \in \overline{\{P(t)\}} \setminus \{P(t)\}$. Then we know that $IS^Y_{S_{U_n}}(P(t)) = IS^Y_{S'}(P(t))$, and we call this Y-state y. Then, for the control decision at y in S_{U_n}, i.e., $S_{U_n}(P(t))$, one of the two following cases holds:

(i) $S_{U_n}(P(t))$ is a control decision returned by **Step 1** or **4**. By the construction rule, we know that $\forall \gamma' \in C_{\mathcal{AES}_{\varphi}^{NB}(\mathbf{G})}(y) : S_{U_n}(P(t)) \not\subset \gamma'$. Since $S' \in \mathbb{S}(\mathcal{AES}_{\varphi}^{NB}(\mathbf{G}))$, by Definition 2.4.2, we know that $S_{U_n}(P(t)) \subset S'(P(t))$ cannot happen.

(ii) $S_{U_n}(P(t))$ is a control decision returned by **Step 3**. Suppose that $S_{U_n}(P(t))$ is in a live control decision string $\gamma_1\gamma_2\ldots\gamma_n$ and let $w := \xi_1\sigma_1\xi_2\ldots\xi_{i-1}\sigma_{i-1}\xi_n$ be the corresponding live path as defined in Definition 3.3.1. We assume, without loss of generality, that $S_{U_n}(P(t)) = \gamma_1, S_{U_n}(P(t)\sigma_1) = \gamma_2, \ldots, S_{U_n}(P(t)\sigma_1\ldots\sigma_{n-1}) = \gamma_n$. Consider another live control decision string $\gamma'_1\gamma'_2\ldots\gamma'_n$, where $\gamma'_i := S'(P(t)\sigma_1\ldots\sigma_{i-1}), 1 \leq i \leq n$. Such a live control decision string is well defined since $\mathcal{L}(S_{U_n}/\mathbf{G}) \subset \mathcal{L}(S'/\mathbf{G})$ and tw is also in $\mathcal{L}(S'/\mathbf{G})$. By fact 2) above we know that $\gamma_i \subseteq \gamma'_i, i \geq 2$. Moreover, we know that $\gamma_1 \subset \gamma'_1$. Thus, $\gamma'_1\gamma'_2\ldots\gamma'_n$ is strictly lager than $\gamma_1\gamma_2\ldots\gamma_n$, which contradicts the fact that $\gamma_1\gamma_2\ldots\gamma_n$ is locally maximal.

For each case, we obtain a contradiction. Thus, no more permissive supervisor exists. □

[1]Without loss of generality, we assume that the supervisors are irredundant.

Remark 3.4.3. The intuition behind the above proof is that it is impossible to construct a supervisor that generates a language strictly larger than the one obtained by the proposed algorithm, since we have taken either locally maximal control decisions (case (i)) or locally maximal control decision strings (case (ii)). For the first case, it is easy to see that the control decision $S_{U_n}(P(t))$ is locally maximal. For the second case, it does not mean that we cannot find a *single* control decision γ_1' such that $\gamma_1 \subset \gamma_1'$. However, if we do so, then $\gamma_1'\gamma_2 \ldots \gamma_n$ will not be a live decision string. The intuition behind this phenomenon is that, in partially-observed DES, enabling more events at the current state may result in more conservative decisions in the future. In other words, the control decision string $\gamma_1\gamma_2 \ldots \gamma_n$ is locally maximal *as a whole.*

Recall that the NB-AES exists if there exists a non-empty solution to the problem under consideration and Algorithm NB-SOLU always returns a maximal solution in a finite number of iterations if the NB-AES exists. Consequently, we have the following theorem.

Theorem III.4. *NB-MPIEP is solvable if and only if* $\mathcal{AES}_{\varphi}^{NB}(\boldsymbol{G})$ *exists.*

Hence, the existence of the NB-AES provides the solvability condition for NB-MPIEP.

3.4.3 Discussion

We have solved the maximally permissive supervisor synthesis problem for IS-based property with prefix-closed and non-prefix-closed specification languages. It was shown in [64] that when the plant can be fully observed, under the assumption that $\mathbf{H} \sqsubseteq \mathbf{G}$, the maximal permissive supervisor (for safety and non-blockingness) can be repressed in the form of $S : X \rightarrow \Gamma$. For the partially-observed prefix-closed specification case, since the information state we defined captures all the information we need to solve the problem, the supervisor for any IS-based property we synthesized is in the form of $S : I \rightarrow \Gamma$. For the non-prefix-closed specification case, we have shown that the information state is not sufficient anymore to carry all the information we need for synthesis purposes; in this case, the "real" information state is the information state originally defined augmented with an integer that represents the number of times the current state has been visited. Thus, the maximally permissive supervisor is in fact in the form of $S : I \times \mathbb{Z} \to \Gamma$.

3.5 Illustrative Example

We illustrate the synthesis algorithm of Section 3.4, Algorithm NB-SOLU, by an illustrative example. In particular, this example shows that: (i) IS-based non-blocking supervisors may not always exist in general; and (ii) a maximal solution can still be obtained by using Algorithm NB-SOLU, even when other algorithm return empty solutions.

System Model: Consider the following guideway problem: A town is divided into two zones, zone 1 and zone 2, with single-way streets as shown in Figure 3.6. At the top of the zones, there is a recycling station. Everyday, *only one* zone will send a robot (r_1 or r_2) to clean up the streets. The robot sent by zone 1 can only move counter-clockwise, i.e., move forward or turn left; the robot sent by zone 2 can only move clockwise.

Control: There are two traffic lights, L_1 and L_2, close to the bottom intersection as shown in the figure. The lights control the robots as follows: When L_1 is red, if robot r_1 is at point a, then it must wait until the light turns green; if robot r_2 is at point c, then it can choose to wait there or turn right. The effect of L_2 is analogous.

Sensing: There is a radar around the traffic lights that detects whether there is a robot in region D, which is in front of each light, every time unit. However, the radar cannot distinguish which zone the detected robot belongs to.

Specification: Since all streets are one-way streets, with legal directions shown in Figure 3.6, we do not want movement in the reverse direction to happen. Without

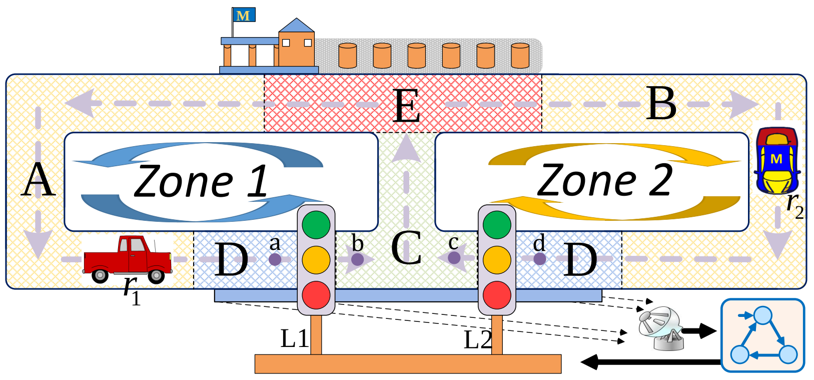
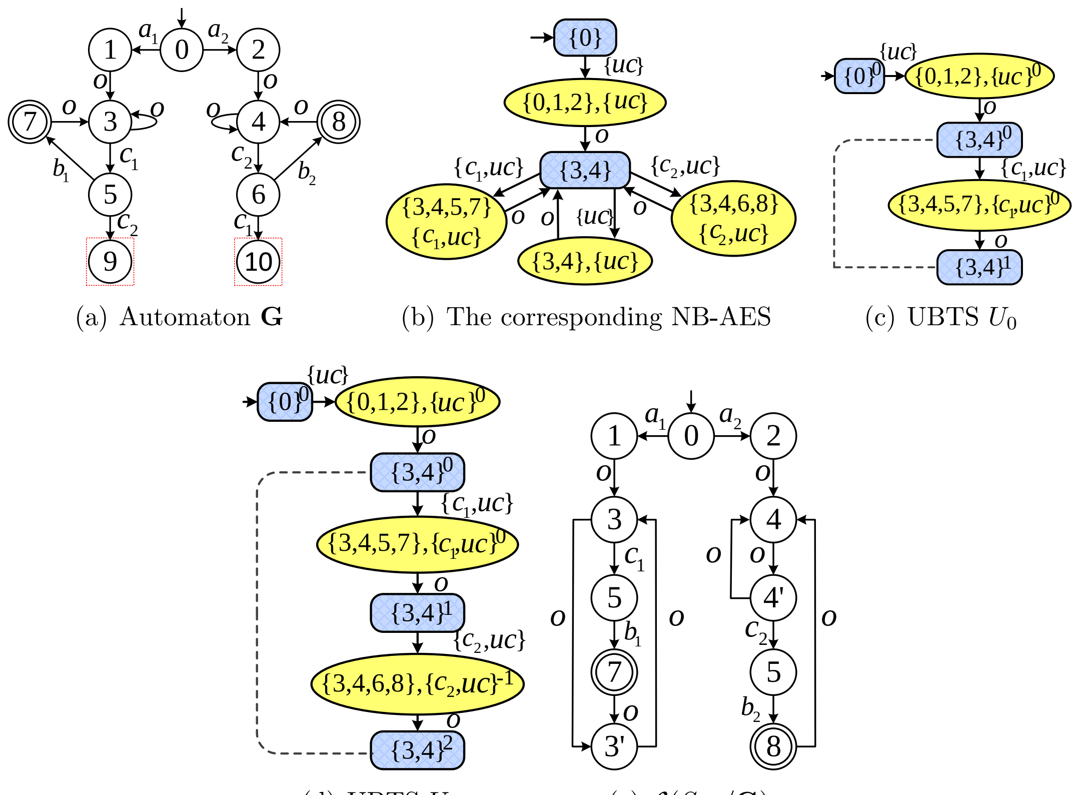


Figure 3.6: Example discussed in Section 3.5.

any traffic light, the robot from zone 1 could possibly violate this specification by entering zone 2 through the points a, b, c and d. Clearly, if both L_1 and L_2 are kept red, then the above specification can be satisfied trivially. However, in order for the robot to be able to unload the trash it collected along the streets, we require that the robot should always be able to enter region E. In summary, the goal for us is to design a control policy for the traffic lights for one day's operation based on the radar information and such that the above requirements are satisfied.

The above problem can be modeled as a supervisory control problem under partial observation. First, we use unobservable and uncontrollable events a_1 and a_2 to represent the nondeterministic initial setting, since we do not know where the robot starts from. Event o is used to model the event that the radar detects a robot in region D, which is observable but not controllable. We use event c_1 to represent that there is a robot that crosses L_1 (from the RHS to the LHS or from the LHS to the RHS); this event is controllable but not observable. We define c_2 analogously for the control effect of L_2. Events b_1 and b_2 represent that robots r_1 and r_2 unload their trash, respectively; these events are unobservable and uncontrollable. The automaton model $\mathbf{G}$ of this system is shown in Figure 3.7(a), in which states 9 and 10 are illegal states.

The corresponding NB-AES the safety specification w.r.t. $\mathbf{G}$ is shown in Fig-



(a) Automaton **G**

(b) The corresponding NB-AES

(c) UBTS U_0

(d) UBTS U_1

(e) $\mathcal{L}(S_{U_1}/\mathbf{G})$

Figure 3.7: For **G**: $\Sigma_c = \{c_1, c_2\}, \Sigma_o = \{o\}$, and states 9 and 10 are illegal.

ure 3.7(b). By applying Algorithm NB-SOLU, we first obtain the initial UBTS U_0 shown in Figure 3.7(c), which induces a livelock solution. Thus, we need to unfold from the entrance Y-state $\{3, 4\}$, which results in the UBTS U_1 shown in Figure 3.7(d). UBTS U_1 induces the controllable, observable, safe, and non-blocking sublanguage $\mathcal{L}(\tilde{U}_1/\mathbf{G})$ shown in Figure 3.7(e). Moreover, this language is maximal.

This example, while simple, has important implications. First, note that the solution obtained by Algorithm NB-SOLU is a non-IS-based solution, since it enables c_1 when state $\{3, 4\}$ is visited for $2k + 1$ times and it enables c_2 when state $\{3, 4\}$ is visited for $2k$ times, $k \in \mathbb{N}$. Moreover, we see that any fixed control decision at Y-state $\{3, 4\}$ will result in a livelock solution. This verifies our earlier assertion in Section 3.4 that IS-based solutions may not exist in general and that the unfolding steps of Algorithm NB-SOLU are indeed needed. Second, for this problem, the supremal controllable normal solution and the solutions obtained by using the methods in [11, 93] are all empty, even though a solution exists.

3.6 Conclusion

In this chapter, we solves the problem of synthesizing a *non-blocking* supervisor for an IS-based property. This extends the results in Chapter II from the prefix-closed case to the non-prefix-closed case. This results in a maximally permissive non-blocking supervisor enforcing an IS-based property for a partially observed DES. For this purpose, defined the Non-Blocking All enforcement Structure, another new bipartite transition system obtained from the AES that takes non-blockingness into account in addition to IS-based property. We provided a synthesis algorithm that uses the NB-AES to synthesize the desired maximal, controllable, and observable sublanguage. Finally, the convergence and maximality of this algorithm were proved. This also solve the previously open problem of synthesizing a safe and non-blocking supervisor, which is a special case of NB-MPIEP.

3.7 Appendix

3.7.1 Implementation of the Algorithms

In this section, we discuss implementation issues related to the construction and synthesis algorithms in the chapter. Specifically, we answer the following two questions.

1). How to verify the liveness property defined in Definition 3.3.1?

2). How to find a local maximal live decision string $\gamma_1\gamma_2\dots\gamma_n$ for any state pair $(y,x), y \in Q_Y^T, x \in y$ in a BTS T?

The key to these two problems is to build an automaton that contains all state connections *inside* of each Y-or Z-state in the BTS. We call such an automaton the Inter-Connected System (ICS).

Definition 3.7.1. *(Inter-Connected System). Given a bipartite transition system T (w.r.t. $\boldsymbol{G}$), its corresponding* Inter-Connected System *is defined as the automaton $ICS^T = (Q^{ICS^T}, \Sigma^{ICS^T}, \delta^{ICS^T}, q_0^{ICS^T}, Q_m^{ICS^T})$, where*

- $Q^{ICS^T} \subseteq (Q_Y^T \times X) \cup (Q_Z^T \times X)$ *is the set of states defined by*
  - $(y,x) \in Q^{ICS^T}$ *if* $y \in Q_T^Y$ *and* $x \in y$,
  - $(z,x) \in Q^{ICS^T}$ *if* $z \in Q_T^Z$ *and* $x \in I(z)$;
- $\Sigma^{ICS^T} = \Sigma \cup \Gamma$ *is the set of events;*
- $\delta^{ICS^T} : Q^{ICS^T} \times \Sigma \to Q^{ICS^T}$ *is the partial transition function defined by: for any* $\gamma \in \Gamma, \sigma \in \Sigma$
  - $\delta^{ICS^T}((y,x_1),\gamma) = (z,x_2)$ *if* $x_1 = x_2$ *and* $h_{YZ}^T(y,\gamma) = z$
  - $\delta^{ICS^T}((z,x_1),\sigma) = (z,x_2)$ *if* $\delta(x_1,\sigma) = x_2$ *and* $\sigma \in \Gamma(z) \cap \Sigma_{uo}$
  - $\delta^{ICS^T}((z,x_1),\sigma) = (y,x_2)$ *if* $\delta(x_1,\sigma) = x_2$, $\sigma \in \Gamma(z) \cap \Sigma_o$ *and* $h_{ZY}^T(z,\sigma) = y$

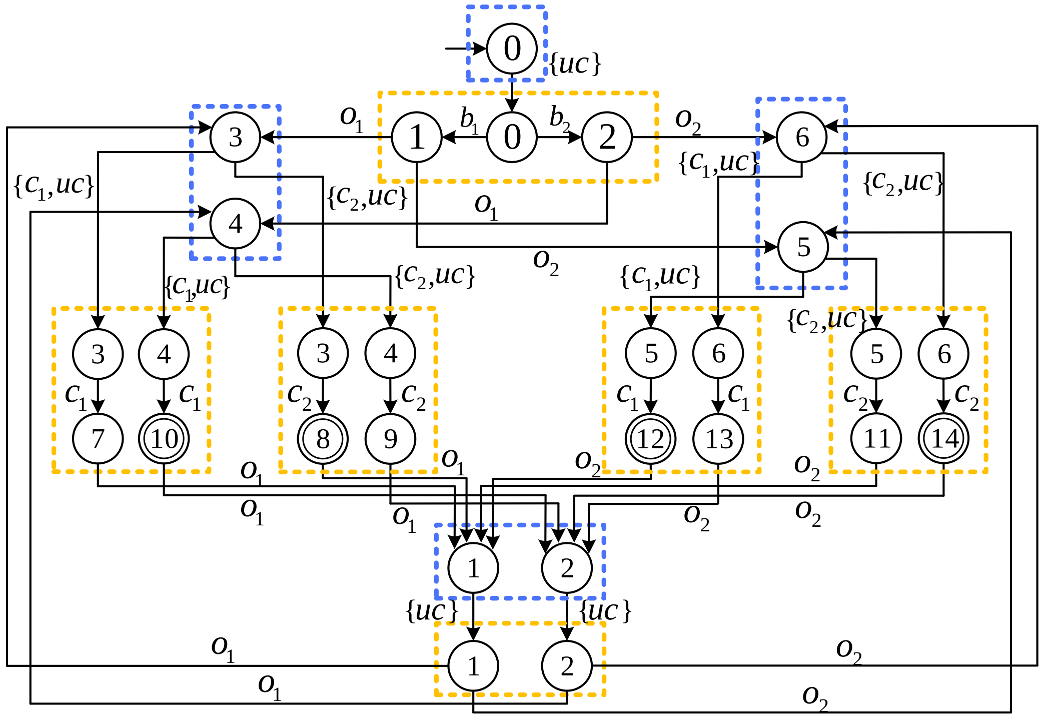


Figure 3.8: Example of Inter-Connected System: The figure shows the corresponding ICS for the automaton and its NB-AES shown in Figure 3.1. The blue dashed rectangles and yellow dashed rectangles correspond to the Y-states and the Z-states in the BTS, respectively.

- $q_0^{ICS^T} = (\{x_0\}, x_0)$ *is the initial state;*
- $Q_m^{ICS^T} = \{(z,x) \in Q^{ICS^T} : x \in X_m\}$ *is the set of marked states.*

The ICS for an EBTS U is defined analogously.

Given an automaton, we say a state is *co-accessible* if there is a string from this state to a marked state and we say an automaton is *co-accessible* if all states in it are co-accessible; see, e.g., [12]. The following result says that to verify the liveness a Y-state in T it suffices to verify the co-accessibility of its corresponding states in ICS^T.

Proposition 3.7.1. *Given a BTS T and its Inter-Connected System ICS^T, a Y-state y in T is live if and only if for any state $x \in y$ in it, $(y,x) \in Q^{ICS^T}$ is co-accessible in ICS^T.*

Proof. ($\Rightarrow$) Since the Y-state y is live in T, then Definition 3.3.1 implies that for any state $x \in y$ in it, there exists a decision string $\gamma_1\gamma_2 \ldots \gamma_n$ such that under this decision string there exists a string $s = \xi_1\sigma_1\xi_2 \ldots \sigma_{n-1}\xi_n, \xi_i \in (\Sigma_{uo} \cap \gamma_i)^*, \sigma_i \in \Sigma_o \cap \gamma_i$, such that $\delta(x, s) \in X_m$. By the definition of the ICS, such a string $w = \gamma_i\xi_1\sigma_1\gamma_2 \ldots \sigma_{n-1}\gamma_n\xi_n$ also exists in ICS^T and $\delta^{ICS^T}((y, x), w) \in Q_m^{ICS^T}$. Thus, (y, x) is co-accessible.

($\Leftarrow$) By construction. Recall that $\Sigma^{ICS^T} = \Sigma_o \cup \Sigma_{uo} \cup \Gamma$. Then we first define two natural projections $P_C : (\Sigma_o \cup \Sigma_{uo} \cup \Gamma)^* \to \Gamma^*$ and $P_{CO} : (\Sigma_o \cup \Sigma_{uo} \cup \Gamma)^* \to (\Sigma_o \cup \Gamma)^*$, i.e., for any $s \in (\Sigma^{ICS^T})^*$, $P_C(s)$ is of the form $\gamma_1\gamma_2\gamma_3 \ldots, \gamma_i \in \Gamma$ and $P_{CO}(s)$ is of the form $\gamma_1\sigma_1\gamma_2\sigma_2 \ldots, \gamma_i \in \Gamma, \sigma_i \in \Sigma_o$.

Since for any $x \in y$, (y, x) is co-accessible in ICS^T, we can find a string $t = e_1e_2 \ldots e_m \in (\Sigma^{ICS^T})^*$ such that $\delta^{ICS^T}((y, x), t) \in Q_m^{ICS^T}$. By Definition 3.3.1, it is clear that $P_C(t)$ is a live decision string for (y, x). Consequently, y is live in T. $\square$

Corollary 3.7.1. *Given a BTS T, all Y-states in T are live if and only ICS^T is co-accessible.*

In the construction algorithm of the NB-AES, we need to check whether or not there exists a Y-state in a BTS that is not live. By Corollary 3.7.1, this suffices to check the co-accessibility of the ICS^T. Specifically, we need to first build $ICS^{\mathcal{AES}_\varphi(\mathbf{G})}$, the ICS of the AES; then, for each iteration step, we check whether or not $ICS^{\mathcal{AES}_\varphi(\mathbf{G})}$ is co-accessible. If a state $(y, x) \in Q^{ICS^{\mathcal{AES}_\varphi(\mathbf{G})}}$ is not co-accessible, then (i) the Y-state y in $\mathcal{AES}_\varphi(\mathbf{G})$ should be removed and; (ii) the set of states $\{(y', x') \in Q^{ICS^{\mathcal{AES}_\varphi(\mathbf{G})}} : y' = y\}$ should also be removed from the ICS; we then repeat until the ICS is accessible.

Now we are ready to show how to find a locally maximal live decision string $\tilde{\gamma}_1\tilde{\gamma}_2 \ldots \tilde{\gamma}_n$ for (y, x), as needed in Algorithm NB-SOLU. In the proof of Proposition 3.7.1, we have already shown how to find a live decision string for a given (y, x). For computation simplicity, we can find a shortest live path s in ICS^T such that $\delta^{ICS^T}((y, x), s) \in Q_m^{ICS^T}$ and get the shortest live decision string $\gamma_1\gamma_2 \ldots \gamma_n = P_C(s)$

and its corresponding run $\gamma_1\sigma_1\gamma_2\sigma_2\ldots\sigma_{n-1}\gamma_n = P_{CO}(s)$. To find a locally maximal decision string, our approach is simply to sequentially replace each single control decision in $P_C(s)$ by one that is as large as possible. Specifically, we start from γ_1 and see whether or not we can pick a control decision γ_1' in $C_{\mathcal{AES}_\varphi^{NB}(\mathbf{G})}(y)$ such that: (i) $\gamma_1 \subset \gamma_1'$ and (ii) $\gamma_1'\gamma_2\ldots\gamma_n$ is also a live decision string. If γ_1' satisfies these two conditions, then we replace γ_1 by γ_1', and try to grow γ_1', and so forth, until we cannot find a larger one. Then we proceed to analyze $\gamma_2, \gamma_3, \ldots$ by the same manner. The only difference is that when we try to replace γ_i by γ_i', we just need to consider the existence of the decision string $\gamma_i'\gamma_{i+1}\ldots\gamma_n$ and do not need to consider those that have already been grown to be maximal (namely, γ_1 to γ_{i-1}). This procedure is formally described by Algorithm L-MAX.

Algorithm 5: L-MAX

```
input  : $y$ and $\gamma_1\sigma_1\gamma_2\sigma_2\ldots\sigma_{n-1}\gamma_n$
output: $\tilde{\gamma}_1\tilde{\gamma}_2\ldots\tilde{\gamma}_n$
1  $i \leftarrow 1, y_1 \leftarrow y$;
2  while $i \leq n$ do
3      for $\gamma' \in C_{\mathcal{AES}_\varphi^{NB}(\boldsymbol{G})}(y_i)$ do
4          if $\gamma_i \subset \gamma'$ and the run $\gamma'\sigma_i\gamma_{i+1}\sigma_{i+1}\ldots\sigma_{n-1}\gamma_n$ is defined at $y_i$ in the NB-AES then
5              $\tilde{\gamma}_i \leftarrow \gamma'$;
           else
6              $\tilde{\gamma}_i \leftarrow \gamma_i$;
7      $y_{i+1} \leftarrow h_{ZY}(h_{YZ}(y_i, \tilde{\gamma}_i), \sigma_i)$;
8      $i \leftarrow i + 1$;
```

3.7.2 Complexity Analysis

Proof of Proposition 3.3.1.

Proof. First, we need to build $\mathcal{AES}_\varphi(\mathbf{G})$, which can be done in $O(|X||\Sigma|2^{|X|+|\Sigma_c|})$ as discussed earlier. For each Z-state in $\mathcal{AES}_\varphi(\mathbf{G})$, checking whether it is deadlock-free can be done in $O(|X||\Sigma|)$. Thus, line 2 in the algorithm can be done in $O(|X||\Sigma|2^{|X|+|\Sigma_c|})$.

As discussed in Appendix 3.7.1, before starting the iteration, we need to build $ICS^{\mathcal{AES}_\varphi(\mathbf{G})}$, which has $\sum_{y\in Q_Y^{AES}}|y| + \sum_{z\in Q_z^{AES}}|I(z)| \leq (1+2^{|\Sigma_c|})\sum_{i=1}^{|X|} i\binom{|X|}{i} = (1+2^{|\Sigma_c|})|X|2^{|X|-1}$ number of states and $2^{|\Sigma_c|}\sum_{y\in Q_Y^{AES}}|y| + |\Sigma|\sum_{z\in Q_z^{AES}}|I(z)| \leq (2^{|\Sigma_c|}+2^{|\Sigma_c|}|\Sigma|)|X|2^{|X|-1}$ number of transitions; we denote these upper bounds by n_1 and n_2, respectively. Thus, the construction of $ICS^{\mathcal{AES}_\varphi(\mathbf{G})}$ can be done in $O(|X||\Sigma|2^{|X|+|\Sigma_c|-1})$.

Since we need to remove at least one Y-state for each iteration step, the whole iteration procedure will execute at most $|Q_Y^{AES}|$ number of times, which is bounded by $2^{|X|}$. For each iteration step, by Corollary 3.7.1, we need to verify the co-accessability of $ICS^{\mathcal{AES}_\varphi(\mathbf{G})}$, which can be done in $O(n_1+n_2)$; then we search through the state space of Q_Y^{AES} and remove the Y-states that have no successors and the corresponding states in the ICS, which is still bounded by $O(n_1+n_2)$. Thus, the total complexity for the construction of the NB-AES is $O(|X||\Sigma|2^{2|X|+|\Sigma_c|-1})$. □

Proof of Proposition 3.4.1

Proof. First, in Algorithm NB-SOLU, the EBTS $\tilde{U}_i$ contains at most $|X|^2 2^{2|X|}$ Y-states and the same number of Z-states. Therefore, in the ICS $ICS^{\tilde{U}_i}$, there are at most $n_1' := |X|^3 2^{2|X|+1}$ states and $n_2' := |X|^3 2^{2|X|+|\Sigma|+1}$ transitions. The above n_1' and n_2' are estimated based on the fact that the largest superscript of any control Y-state y is $|y|$ and for each iteration we introduce at most $|X|2^{|X|}$ transient Y-states. Now we are ready to analyze the complexity of the synthesis algorithm.

First, let us consider the complexity of each single iteration step (**Step 2-4**):

- **Step 2** is a livelock detection problem in the ICS of $\tilde{U}_i$, which can be done in $O(n_1'+n_2')$.

- **Step 3** involves two problems:

  1) a shortest path search problem in the ICS of $\tilde{U}_i$, which requires $O(n_1'+n_2')$ and

2) the problem of growing this path to be maximal. For this problem, since such path has a length $N = |X|2^{|X|+1}$, in the worst case, then it requires a complexity of $O(N2^{|\Sigma_c|} + (N-2)2^{|\Sigma_c|} + \cdots + 2^{|\Sigma_c|}) = O(\frac{(N+1)^2}{4}2^{|\Sigma_c|})$.

- **Step 4** calls the procedure EXPAND, which can be done in $O(|\Sigma_o|2^{|X|+|\Sigma_c|} + |\Sigma_o||X|2^{|X|})$.

Thus, the complexity of a single iteration step is of $O([|X|^3 2^{|X|} + |\Sigma|]2^{|X|+|\Sigma|})$.

In the convergence proof of Algorithm NB-SOLU, we have already shown that $|X|2^{|X|-1}$ provides an upper bound for the number of iterations. Combining this with the above results, we get that the total complexity of Algorithm NB-SOLU is $O([|X|^3 2^{|X|} + |\Sigma|]|X|2^{2|X|+|\Sigma|})$. □

CHAPTER IV

The Range Control Problem

4.1 Introduction

So far, we have solved the problem of synthesizing a non-blocking supervisor that enforces an IS-based property. As we have shown in Chapters II and III, the supervisor synthesis problem may not have a unique supremal solution in general. Instead, there may be several incomparable *locally maximal* solutions. Our approach "randomly" selects one solution among these locally maximal solution. In particular, the maximal solution obtained is a particular type of maximal solutions, namely, *greedy maximal* solutions. In a greedy maximal solution, the supervisor tries to enable *as many events as possible* at each control decision instant. However, no consideration is given to including some minimum *required* behavior in these solutions, a meaningful criterion when choosing among locally maximal solutions. This phenomenon is illustrated in Figure 4.1.

In order to resolve the above issue, we consider in this chapter a generalized supervisor synthesis problem called the *Maximally-Permissive Range Control Problem.* In this problem, we not only want to find a locally maximal supervisor, but we also require that the synthesized maximal supervisor contain a given behavior. Namely, we want to find a "meaningful" maximal solution. However, instead of investigating the general IS-based property with non-blockingness requirement as we did in Chap-

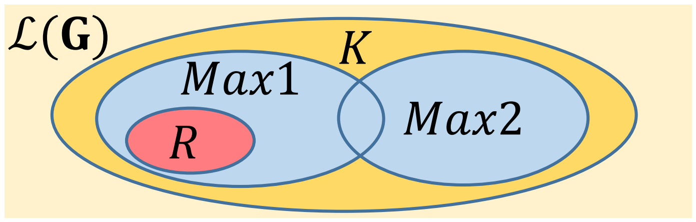


Figure 4.1: Let $\mathbf{G}$ be the system, K be the legal behavior and R be the required behavior. $Max1$ and $Max2$ are two incomparable maximal solutions in K, i.e., $Max1 \not\subseteq Max2$ and $Max2 \not\subseteq Max1$. However, $Max1$ contains the required behavior R, while $Max2$ does not contain any string in R.

ters II and III, in this chapter, we only restrict our attention to safety requirement and not do consider the issue of blockingness. More specifically, we consider two specification languages: the safety specification language K, which is also referred to as the upper bound language, and a prefix-closed lower bound language $R \subseteq K$, which models the required behavior that the closed-loop system must achieve. To solve the range control problem, we present a new synthesis algorithm based on the two notions of AES and Control Simulation Relation (CSR). Although we only consider prefix-closed languages, i.e., nonblockingness is not considered, to the best of our knowledge, the maximally-permissive range control problem we solve herein was an open problem even in this case.

This chapter is organized as follows. In Section 4.2, we formulate the maximally-permissive range control problem that we solve in the chapter. In Section 4.3, we first reveal that the notion of strict sub-automaton plays an important role in the range control problem. Then we provide a new constructive approach for computing the infimal supervisor. In Section 4.4, we define the notion of Control Simulation Relation (CSR). The CSR is used to resolve the future dependency issue, which is the main difficulty in handling maximal permissiveness with the lower bound constraint. In Section 4.5, we first provide an algorithm to synthesize a maximally-permissive supervisor that contains the required behavior. Then we prove the correctness of the proposed algorithm. We also discuss how to verify whether a given supervisor is maximal or not. Finally, we conclude the chapter in Section 4.6.

4.2 Problem Formulation

In this chapter, we consider a generalized supervisory control synthesis problem, called the *range control problem*, where we have two prefix-closed specification languages:

- the upper bound language $K=\overline{K}\subseteq\mathcal{L}(\mathbf{G})$; and
- the lower bound language $R=\overline{R}\subseteq K$.

The upper bound K describes the *legal* behavior of the system and we say that a supervisor S is *safe* if $\mathcal{L}(S/\mathbf{G}) \subseteq K$. Recall that the maximal safe supervisor may not be unique and there may be two incomparable maximal supervisors S_1 and S_2 such that $\mathcal{L}(S_1/\mathbf{G}) \not\subseteq \mathcal{L}(S_2/\mathbf{G})$ and $\mathcal{L}(S_2/\mathbf{G}) \not\subseteq \mathcal{L}(S_1/\mathbf{G})$. In order to synthesize a "meaningful" maximal solution, we introduce a lower bound language R describing the *required* behavior that the closed-loop system must achieve. Examples of using the range requirement to impose design constraints can be found in [38, 52, 53, 58]. We now formulate the Maximally-Permissive Range Control Problem (MPRCP):

Problem 4. *(Maximally-Permissive Range Control Problem). Given system $\boldsymbol{G}$, lower bound language R and upper bound language K, synthesize a maximally-permissive supervisor $S^* : P(\mathcal{L}(\boldsymbol{G})) \to \Gamma$ such that $R \subseteq \mathcal{L}(S^*/\boldsymbol{G}) \subseteq K$.*

Remark 4.2.1. We make several comments on MPRCP.

1. First, under the assumption that $\Sigma_c \subseteq \Sigma_o$, MPRCP has a unique solution, if one exists. Specifically, it suffices to compute the *supremal controllable and normal sub-language* of K, denoted by $K^{\uparrow CN}$, and test whether or not $R \subseteq K^{\uparrow CN}$. If so, then $K^{\uparrow CN}$ is the unique supremal solution; otherwise, there does not exist a solution to MPRCP.

2. Second, when the lower bound requirement is relaxed, i.e., $R = \{\epsilon\}$, MPRCP is solved by the results in Chapter II, since safety is an IS-based property and it suffices to synthesize an *arbitrary* maximal supervisor without taking the lower bound into consideration.

3. Finally, if the maximal permissiveness requirement is relaxed, then we just need to compute infimal controllable and observable language of R, denoted by $R^{\downarrow CO}$, and test whether or not $R^{\downarrow CO} \subseteq K$. If so, then $R^{\downarrow CO}$ is the *most conservative* solution; otherwise, MPRCP does not have a solution.

Hence, many existing problems solved in the literature are special cases of MPRCP. However, to the best of our knowledge, MPRCP is still open for the general case, which is clearly more difficult than the above special cases.

Throughout this chapter, we use $\mathbf{K} = (X_K, \Sigma, \delta_K, x_{0,K})$ to denote the automaton generating K, and use $\mathbf{R} = (X_R, \Sigma, \delta_R, x_{0,R})$ to denote the automaton generating R. For the sake of simplicity and without loss of generality, as we have discussed in Section 2.7.1. we assume that $\mathbf{K} \sqsubset \mathbf{G}$. Therefore, in the chapter, we consider IS-based property φ_{safe}, which is essentially the upper bound (safety) requirement, defined by:

$$\forall i \in 2^X : \varphi_{safe}(i) = 1 \Leftrightarrow i \subseteq X_K \tag{4.1}$$

Since φ_{safe} is uniquely defined based on $\mathbf{K}$, hereafter, we also write the AES $\mathcal{AES}_{\varphi_{safe}}(\mathbf{G})$ as $\mathcal{AES}(\mathbf{G}, \mathbf{K})$.

4.3 Synthesis of the Infimal Supervisor

Although there does not exist a unique maximal safe supervisor in general, there does exist a unique infimal supervisor that contains the lower bound. Here we explain the intuition of the existence of the infimal supervisor; formal proof can be found in [68]. Let S_1 and S_2 be two safe supervisors. Then we can always construct a new supervisor S_{12} defined by $\forall s \in \Sigma_o^* : S_{12}(s) = S_1(s) \cap S_2(s)$. It is easy to show that $\mathcal{L}(S_{12}/\mathbf{G}) = \mathcal{L}(S_1/\mathbf{G}) \cap \mathcal{L}(S_2/\mathbf{G})$. Therefore, if both S_1 and S_2 achieves R, then S_{12} also achieves R. Hence, there always exists a unique infimal supervisor that achieves R. We denote by $R^{\downarrow CO}$ the closed-loop behavior of the unique infimal supervisor; this language is also referred to as the infimal prefix-closed controllable and observable super-language of R [68].

The goal of this section is to synthesize a BTS T_R that realizes the infimal supervisor achieving the lower bound, i.e., $\mathcal{L}(S_{T_R}/\mathbf{G}) = R^{\downarrow CO}$. This infimal supervisor will be further used as a basis to solve MPRCP. Before we start this section, we introduce the following monotonicity properties of the AES defined for the safety specification. Let $y \in Q_Y^{AES}$ be a Y-state in $\mathcal{AES}(\mathbf{G}, \mathbf{K})$. We say that a control decision $\gamma \in \Gamma$ is *safe* at y if $\gamma \in C_{\mathcal{AES}(\mathbf{G},\mathbf{K})}(y)$. Then we have the following monotonicity properties.

Proposition 4.3.1. *(Monotonicity Properties [109]).*

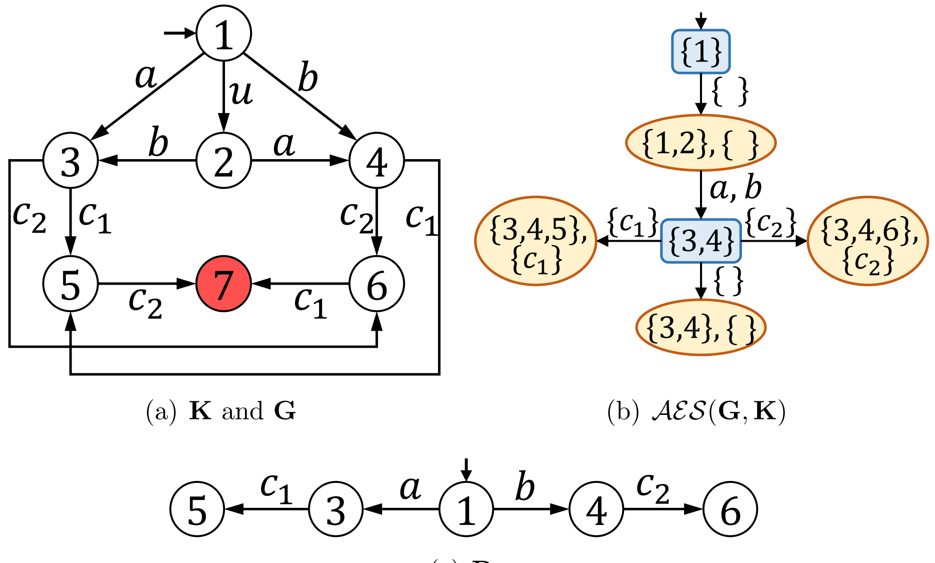
1. *Any control decision that is safe at Y-state y_1 is also safe at Y-state $y_2 \subseteq y_1$.*

2. *If control decision γ_1 is safe at Y-state y, then so is any control decision $\gamma_2 \subseteq \gamma_1$.*

Note that the monotonicity properties only hold for safety specification in the prefix-closed case. In general, they do not hold for other IS-based property in the non-prefix-closed case.

4.3.1 The Role of Strict Sub-automaton

Recall that the goal of this section is to construct a BTS T_R such that $\mathcal{L}(S_{T_R}/\mathbf{G}) = R^{\downarrow CO}$. Although we know that $R^{\downarrow CO}$ is shown to be a regular language, this fact in itself is not sufficient for the purpose of synthesis. Specifically, we are interested in whether or not $R^{\downarrow CO}$ can be achieved by an IS-based supervisor which can be realized by a BTS. This question is very important, since it essentially asks *what is*



(a) $\mathbf{K}$ and $\mathbf{G}$ (b) $\mathcal{AES}(\mathbf{G},\mathbf{K})$ (c) $\mathbf{R}$

Figure 4.2: For $\mathbf{G}$, we have $\Sigma_o = \{a, b\}$ and $\Sigma_c = \{c_1, c_2\}$.

the right state space in order to realize the infimal supervisor that contains R. One may conjecture that there always exists a BTS T_R such that $\mathcal{L}(S_{T_R}/\mathbf{G}) = R^{\downarrow CO}$ when $R^{\downarrow CO} \subseteq K$. However, this is not true in general as illustrated by the following example.

Example 4.3.1. *Let us consider the system $\boldsymbol{G}$ shown in Fig. 4.2(a). The upper bound automaton $\boldsymbol{K}$ is obtained by removing the single illegal state 7 from $\boldsymbol{G}$. Then the AES for safety specification $\mathcal{AES}(\boldsymbol{G},\boldsymbol{K})$ is shown in Fig. 4.2(b). We consider a lower bound language R which is generated by automaton $\boldsymbol{R}$ shown in Fig. 4.2(c). One can easily check that any IS-based supervisor S does not contain R. This is because events a and b lead to the same Y-state $\{3,4\}$ in any BTS and control decision $S(a)$ and $S(b)$ should always be the same in any IS-based supervisor S. However, we can find a non-IS-based supervisor S', which enables c_1 after observing a and enables c_2 after observing b, such that $R^{\downarrow CO} = \mathcal{L}(S'/\boldsymbol{G})$.*

The reason why there may not exist an IS-based supervisor that achieves $R^{\downarrow CO}$ is explained as follows. Suppose that $\mathbf{R}$ is a sub-automaton of $\mathbf{K}$ such that we can match the state space of $\mathbf{R}$ with the state space of $\mathbf{K}$. Let $s \in P(R)$ be an observable string in $P(s)$ and define

$$y_R(s) = \{x \in X_R : \exists t \in R \cap (\Sigma^* \Sigma_o \cup \{\epsilon\}) \text{ s.t. } \delta_R(x_0, t) = x \wedge P(t) = s\}$$

as the "information state" of $\mathbf{R}$ reached upon observing s, which is analogous to a Y-state in a BTS. Then it is possible that two different "information states" under the original control strategy can be merged as a single information state under the new (more permissive) control strategy. As a consequence, information is lost by using the newly reached information state. We call this phenomenon *information merge*. For example, for the lower bound automaton $\mathbf{R}$ shown in Fig. 4.2(c), we have that $y_R(a) = \{3\}$ and $y_R(b) = \{4\}$. In order to achieve R, in addition to enabling events a and b initially, we also need to enable event u, since it is uncontrollable. Then the two different "information states" $\{3\}$ and $\{4\}$ in $\mathbf{R}$, which are reached by observing a and b, respectively, will be merged as a single state $\{3, 4\}$. However, simply knowing state $\{3, 4\}$ is not sufficient for making control decisions in order to contain the lower bound behavior. To find an IS-based solution, state $\{3, 4\}$ has to be split into two states: one is reached by observing a and the other is reached by observing b.

Let $y \in 2^X$ be an information state and suppose that $X_R \subseteq X$. We denote by $y|_{\mathbf{R}}$ the restriction of y to the state space of $\mathbf{R}$, i.e., $y|_{\mathbf{R}} = \{x \in X_R : x \in y\}$. The following result says that the state merging phenomenon described above will not occur when $\mathbf{R}$ is a *strict* sub-automaton of $\mathbf{K}$.

Proposition 4.3.2. *Assume that $\boldsymbol{R} \sqsubset \boldsymbol{K} \sqsubset \boldsymbol{G}$. Then for any supervisor S such that $R \subseteq \mathcal{L}(S/\boldsymbol{G})$, we have that*

1. $\forall s \in P(R) : IS_S^Y(s)|_{\boldsymbol{R}} = y_R(s)$;

2. $\forall s, t \in P(R) : y_R(s) \neq y_R(t) \Rightarrow IS_S^Y(s) \neq IS_S^Y(t)$.

Proof. First, we show the first statement. By the definition of IS_S^Y, we have that

$$\begin{aligned} IS_S^Y(s) =& \{x \in X : \exists w \in \{\epsilon\} \cup (\mathcal{L}(S/\mathbf{G}) \cap \Sigma^*\Sigma_o) \text{ s.t. } \delta(x_0, w) = x \wedge P(w) = s\} \\ =& \{x \in X_R : \exists w \in \{\epsilon\} \cup (R \cap \Sigma^*\Sigma_o) \text{ s.t. } \delta_R(x_0, w) = x \wedge P(w) = s\} \cup A_{G\setminus R} \\ =& y_R(s) \cup A_{G\setminus R} \end{aligned}$$

where $A_{G\setminus R} = \{x \in X : \exists w \in \{\epsilon\} \cup ((\mathcal{L}(S/\mathbf{G})\setminus R) \cap \Sigma^*\Sigma_o) \text{ s.t. } \delta(x_0, w) = x \wedge P(w) = s\} \subseteq X \setminus X_R$. The reason why we know that $A_{G\setminus R}$ does not contain a state in $\mathbf{R}$ is that we have already assumed that $\mathbf{R}$ is a strict sub-automaton of both $\mathbf{K}$ and $\mathbf{G}$. Hence, any string that goes outside of R will not go back to the state space of $\mathbf{R}$. Therefore, we have that

$$IS_S^Y(s)|_{\mathbf{R}} = y_R(s)|_{\mathbf{R}} \cup A_{G\setminus R}|_{\mathbf{R}} = y_R(s) \tag{4.2}$$

Next, we show the second statement. Let us consider two arbitrary strings $s, t \in P(\mathcal{L}(\mathbf{R}))$ such that $y_R(s) \neq y_R(t)$. By the first statement, s we can write $IS_S^Y(s)$ in the form of $IS_S^Y(s) = y_R(s) \cup A_{G\setminus R}$, where $A_{G\setminus R} \subseteq X \setminus X_R$. Similarly, we can write $IS_S^Y(t)$ in the form of $IS_S^Y(t) = y_R(t) \cup B_{G\setminus R}$, where $B_{G\setminus R} \subseteq X \setminus X_R$. Note that $y_R(s), y_R(t) \subseteq X_R$. Therefore, since $y_R(s) \neq y_R(t)$, we have $IS_S^Y(s) \neq IS_S^Y(t)$. □

Remark 4.3.1. The intuition of the above result is explained as follows. Since $\mathbf{R} \sqsubset \mathbf{K}$, any newly introduced string, namely a string in $\mathcal{L}(\mathbf{K}) \setminus \mathcal{L}(\mathbf{R})$, must lead to a state in $X_K \setminus X_R$. Therefore, if strings s and t lead to two distinct "information states" $y_1 = y_R(s)$ and $y_2 = y_R(t)$ under the original supervisor, respectively, then the newly reached Y-states y_1' and y_2' under a supervisor whose closed-loop language contains R must be in the form of $y_1' = y_1 \cup \hat{y}_1$ and $y_2' = y_2 \cup \hat{y}_2$, respectively, where $\hat{y}_1, \hat{y}_2 \subseteq X_K \setminus X_R$. Since $y_1 \neq y_2$, we know that $y_1' \neq y_2'$.

Based on Lemma 4.3.2, we make the following assumption hereafter.

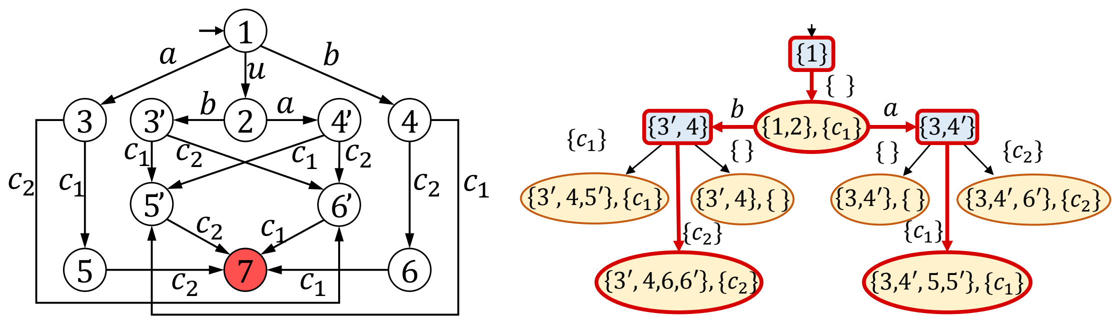
Assumption 1: $\mathbf{R} \sqsubset \mathbf{K} \sqsubset \mathbf{G}$.

Remark 4.3.2. Note that the above assumption is without loss of generality: if $\mathbf{R}$, $\mathbf{K}$ and $\mathbf{G}$ do not satisfy this assumption, then we can always refine the state spaces of $\mathbf{R}$, $\mathbf{K}$ and $\mathbf{G}$ by constructing new automata $\mathbf{R}'$, $\mathbf{K}'$ and $\mathbf{G}'$ such that 1) $\mathbf{R}' \sqsubset \mathbf{K}' \sqsubset \mathbf{G}'$; and 2) $\mathcal{L}(\mathbf{R}) = \mathcal{L}(\mathbf{R}')$, $\mathcal{L}(\mathbf{K}) = \mathcal{L}(\mathbf{K}')$ and $\mathcal{L}(\mathbf{G}) = \mathcal{L}(\mathbf{G}')$. Such a pre-processing algorithm can be found in the Appendix, which generalizes the procedure in [18] from two automata to three automata. In the worst case, the refined system model $\mathbf{G}'$ contains $|X| \times (|X_K| + 1) \times (|X_R| + 1)$ states. Therefore, only polynomial-space refinement is needed to make Assumption 1 hold; this is different from the state-partition-automata-based refinement in the literature, which has an exponential complexity. This assumption and Proposition 4.3.2 play important roles in this paper; they will also be involved several times in our later development. Finally, we remark that the reason why we assume that $\mathbf{K} \sqsubset \mathbf{G}$ and the reason why we assume that $\mathbf{R} \sqsubset \mathbf{K}$ are different. We assume that $\mathbf{K} \sqsubset \mathbf{G}$ to guarantee that legality of strings is fully captured by states. We assume that $\mathbf{R} \sqsubset \mathbf{K}$ to make sure that the information merge phenomenon will not occur.

Example 4.3.2. *Let us return to Example 4.3.1. The original automata $\boldsymbol{R}$ and $\boldsymbol{K}$ in Figs. 4.2(c) and 4.2(a) do not satisfy the assumption that $\boldsymbol{R} \sqsubset \boldsymbol{K}$. Therefore, we refine the state spaces of $\boldsymbol{K}$ and $\boldsymbol{G}$ and obtain new automata $\boldsymbol{K}'$ and $\boldsymbol{G}'$ shown in Fig. 4.3(a) such that $\boldsymbol{R} \sqsubset \boldsymbol{K}' \sqsubset \boldsymbol{G}'$, $\mathcal{L}(\boldsymbol{K}) = \mathcal{L}(\boldsymbol{K}')$ and $\mathcal{L}(\boldsymbol{G}) = \mathcal{L}(\boldsymbol{G}')$. The AES $\mathcal{AES}(\boldsymbol{G}', \boldsymbol{K}')$ for the refined system is shown in Fig. 4.3(b). We see that the original state $\{3, 4\}$ in $\mathcal{AES}(\boldsymbol{G}, \boldsymbol{K})$ splits into two states $\{3', 4\}$ and $\{3, 4'\}$ in $\mathcal{AES}(\boldsymbol{G}')$.*

4.3.2 Synthesis Algorithm

We are now ready to show how to compute the supervisor that achieves $R^{\downarrow CO}$. In particular, we show that such a supervisor can be realized by a BTS.



(a) $\mathbf{K}'$ and $\mathbf{G}'$ (b) $\mathcal{AES}(\mathbf{G}',\mathbf{K}')$

Figure 4.3: In Fig. 4.3(a), $\mathbf{G}'$ is the entire automaton and $\mathbf{K}'$ is obtained by removing illegal state 7 from $\mathbf{G}'$.

Let $y \subseteq X_R$ be a set of states in $\mathbf{R}$. We first define the following set of events

$$\Gamma_R(y) := \{\sigma \in \Sigma : \exists x \in y, \exists s \in \Sigma_{uo}^* \text{ s.t. } \delta_R(x, s\sigma)!\}$$

The following result reveals that $\Gamma_R(y)$ is indeed the set of events that should be enabled at y in order to achieve R.

Proposition 4.3.3. *For any supervisor $S : P(\mathcal{L}(\boldsymbol{G})) \to \Gamma$, $R \subseteq \mathcal{L}(S/\boldsymbol{G})$, if and only if,*

$$\forall s \in P(\mathcal{L}(S/\boldsymbol{G})) : IS_S^Y(s)|_{\boldsymbol{R}} \neq \emptyset \Rightarrow \Gamma_R(IS_S^Y(s)|_{\boldsymbol{R}}) \subseteq S(s).$$

Proof. ($\Rightarrow$) By contradiction. Assume that $\exists s \in P(\mathcal{L}(S/\mathbf{G}))$ such that $IS_S^Y(s)|_{\mathbf{R}} \neq \emptyset$ and $\Gamma_R(IS_S^Y(s)|_{\mathbf{R}}) \not\subseteq S(s)$. Let σ be an event in $\Gamma_R(IS_S^Y(s)|_{\mathbf{R}}) \setminus S(s)$. By the definition of $\Gamma_R(\cdot)$, we have that $\exists x \in IS_S^Y(s)|_{\mathbf{R}}, \exists w \in \Sigma_{uo}^*$ s.t. $\delta_R(x, w\sigma)!$. Since $x \in IS_S^Y(s)|_{\mathbf{R}}$, there exists a string $t \in R$ such that $P(t) = s$ and $\delta_R(x_0, t) = x$, which implies that $tw\sigma \in R$. However, since $\sigma \notin S(s) = S(P(tw))$, we know that $tw\sigma \notin \mathcal{L}(S/\mathbf{G})$. This contradicts the fact that $R \subseteq \mathcal{L}(S/\mathbf{G})$.

($\Leftarrow$) It suffices to show that, $t \in R \Rightarrow t \in \mathcal{L}(S/\mathbf{G})$. We proceed by induction on the length of the projection of t.

Induction Basis: For string $t \in R$ such that $|P(t)| = 0$, we know that $t \in$

$(\Gamma_R(\{x_0\}) \cap \Sigma_{uo})^* \cap R$. Since $\Gamma_R(\{x_0\}) = \Gamma_R(IS_S^Y(\epsilon)) \subseteq S(\epsilon)$, we know that $t \in (S(\epsilon) \cap \Sigma_{uo})^* \cap \mathcal{L}(\mathbf{G}) \subseteq \mathcal{L}(S/\mathbf{G})$, i.e., the induction basis holds.

Induction Hypothesis: Assume that $t \in R \Rightarrow t \in \mathcal{L}(S/\mathbf{G})$ for any t such that $|P(t)| = k$.

Induction Step: To prove the induction step, we show that $v\sigma w \in R \Rightarrow v\sigma w \in \mathcal{L}(S/\mathbf{G})$, where $|P(v)| = k, \sigma \in \Sigma_o$ and $w \in \Sigma_{uo}^*$. Note that any string t such that $|P(t)| = k+1$ can be written in the above form. Let $v' \in \overline{\{v\}}$ be the longest prefix of v that ends up with an observable event and let $x = \delta(x_0, v') \in X_R$. Since $|P(v')| = |P(v)| = k$, by the induction hypothesis, we know that $v' \in \mathcal{L}(S/\mathbf{G})$. Therefore, $x = \delta(x_0, v') \in IS_S^Y(P(v))$. By the definition of $\Gamma_R(\cdot)$, all events between v' and $v\sigma$ are in $\Gamma_R(\{x\})$. Since $x \in X_R$ and $x \in IS_S^Y(P(v))$, we know that $IS_S^Y(P(v))|_{\mathbf{R}} \neq \emptyset$. Therefore, $\Gamma_R(\{x\}) \subseteq S(P(v))$, which implies that $v\sigma \in \mathcal{L}(S/\mathbf{G})$. Similarly, let $x' = \delta(x_0, v\sigma) \in X_R$. We know that $x' \in IS_S^Y(P(v)\sigma)$ and $IS_S^Y(P(v)\sigma)|_{\mathbf{R}} \neq \emptyset$. Again, since $\Gamma_R(\{x'\}) \subseteq S(P(v)\sigma)$, we have $s\sigma w \in \mathcal{L}(S/\mathbf{G})$. This completes the induction step. $\square$

Now, we are ready to present the algorithm that constructs the BTS T_R such that $\mathcal{L}(S_{T_R}/\mathbf{G}) = R^{\downarrow CO}$. Specifically, the BTS T_R is constructed by a depth-first search as follows. Initially, we start from the initial Y-state y_0. For each Y-state y encountered, if $y|_{\mathbf{R}} \neq \emptyset$, we choose $\Gamma_R(y|_{\mathbf{R}}) \cup \Sigma_{uc}$ as the unique control decision defined at y. Note that $y|_{\mathbf{R}} \neq \emptyset$ implies that y can be reached by some string in $P(R)$, i.e., the supervisor is not sure whether or not the system has already gone outside the lower bound language R. Therefore, we choose $\Gamma_R(y|_{\mathbf{R}}) \cup \Sigma_{uc}$ as the control decision since it is the smallest control decision we need in order to contain R. If $y|_{\mathbf{R}} = \emptyset$, then we know for sure that the system has already gone outside R and we just choose Σ_{uc} as the control decision, i.e., all controllable events are disabled. To summarize the above rule, in the constructed BTS T_R, we have that

$$\forall y \in Q_Y^{T_R} : c_{T_R}(y) = \begin{cases} \Gamma_R(y|_{\mathbf{R}}) \cup \Sigma_{uc} & \text{if } y|_{\mathbf{R}} \neq \emptyset \\ \Sigma_{uc} & \text{if } y|_{\mathbf{R}} = \emptyset \end{cases}$$

Based on the above discussion, Algorithm INF-SYNT is proposed to constructed T_R. Namely, for each Y-state encountered, we choose *one* control decision based on the above-discussed rules; for each Z-state encountered, we need to consider *all* observable events that are feasible. Such a depth-first search is implemented by the recursive procedure termed DoDFS. Moreover, Algorithm INF-SYNT returns "No Solution" when a Y-state y such that $\Gamma_R(y|_{\mathbf{R}}) \cup \Sigma_{uc} \notin C_{\mathcal{AES}(\mathbf{G},\mathbf{K})}(y)$ is encountered. This implies that achieving the lower bound R will violate the safety specification, either immediately or unavoidably in the future. In this case, MPRCP has no solution. Of course, $R^{\downarrow CO}$ always exists, but our focus herein is on solving MPRCP. (If the focus is solely on the computation of $R^{\downarrow CO}$, then it suffices to set $K = \mathcal{L}(\mathbf{G})$ in the above development.)

Next, we first illustrate Algorithm INF-SYNT by an example. Then we prove its correctness.

Example 4.3.3. *Let us return to the running example in this chapter. The input parameters of Algorithm INF-SYNT are $\boldsymbol{R}$ and $\mathcal{AES}(\boldsymbol{G}', \boldsymbol{K}')$ shown in Figs. 4.2(c) and 4.3(b), respectively. We start procedure DoDFS from the initial Y-state $y_0 = \{1\}$. Since $\{1\}|_{\boldsymbol{R}} \neq \emptyset$, we take control decision $\Gamma_R(\{1\}) \cup \Sigma_{uc} = \Sigma_{uc}$ (which is depicted as $\{\}$ in Fig. 4.3(b) for simplicity since all events in it are uncontrollable events), and move to the successor Z-state $(\{1, 2\}, \{\})$. Then we need to consider all possible event occurrences from this Z-state. If a occurs, then Y-state $\{3, 4'\}$ is reached. Since $\{3, 4'\}|_{\boldsymbol{R}} = \{3\} \neq \emptyset$, we need to take control decision $\Gamma_R(\{3\}) \cup \Sigma_{uc} = \{c_1\} \cup \Sigma_{uc}$. Similarly, we need to take control decision $\{c_2\} \cup \Sigma_{uc}$ if Y-state $\{3', 4\}$ is reached. The above procedure yields deterministic BTS T_R, which is the part highlighted in*

Fig. 4.3(b).

Algorithm 6: INF-SYNT

```
input  : $\mathbf{R}$ and $\mathcal{AES}(\mathbf{G},\mathbf{K})$.
output: $T_R$.
1   $Q_Y^{T_R} \leftarrow \{y_0\}, Q_Z^{T_R} \leftarrow \emptyset$;
2   DoDFS($y_0, T_R$);
3   return $T_R$;
procedure DoDFS($y, T_R$);
4   if $y|_{\mathbf{R}} \neq \emptyset$ then
5       if $\Gamma_R(y|_{\mathbf{R}}) \cup \Sigma_{uc} \in C_{\mathcal{AES}(\mathbf{G},\mathbf{K})}(y)$ then
6           $Act \leftarrow \Gamma_R(y|_{\mathbf{R}}) \cup \Sigma_{uc}$;
        else
7           return "No Solution";
    else
8       $Act \leftarrow \Sigma_{uc}$;
9   $z \leftarrow h_{YZ}(y, Act)$;
10  Add transition $y \xrightarrow{Act} z$ to $h_{YZ}^{T_R}$;
11  if $z \notin Q_Z^{T_R}$ then
12      $Q_Z^{T_R} \leftarrow Q_Z^{T_R} \cup \{z\}$;
13      for $\sigma \in \Sigma_o : h_{ZY}(z, \sigma)!$ do
14          $y' \leftarrow h_{ZY}(z, \sigma)$;
15          Add transition $z \xrightarrow{\sigma} y'$ to $h_{ZY}^{T_R}$;
16          if $y' \notin Q_Y^{T_R}$ then
17              $Q_Y^{T_R} \leftarrow Q_Y^{T_R} \cup \{y'\}$;
18              DoDFS($y', T_R$);
```

We now prove the correctness of Algorithm INF-SYNT. First, we show that, under the assumption that $\mathbf{R} \sqsubset \mathbf{K} \sqsubset \mathbf{G}$, Algorithm INF-SYNT will never return "No Solution" when a solution exists.

Theorem IV.1. *Algorithm INF-SYNT returns "No Solution" if and only if $R^{\downarrow CO} \nsubseteq K$.*

Proof. ($\Leftarrow$) By contraposition. Suppose that Algorithm INF-SYNT returns BTS T_R. Since S_{T_R} is an IS-based supervisor, we know that, for any $s \in P(\mathcal{L}(S_{T_R}/\mathbf{G}))$ such that $IS^{Y}_{S_{T_R}}(s)|_{\mathbf{R}} \neq \emptyset$, we have

$$S_{T_R}(s) = c_{T_R}(IS^{Y}_{S_{T_R}}(s)) = \Gamma_R(IS^{Y}_{S_{T_R}}(s)|_{\mathbf{R}}) \cup \Sigma_{uc}$$

Therefore, by Proposition 4.3.3, we know that $R \subseteq \mathcal{L}(S_{T_R}/\mathbf{G})$. Then, by the definition of ${}^{\downarrow CO}$, we know that $R^{\downarrow CO} \subseteq \mathcal{L}(S_{T_R}/\mathbf{G})$. Moreover, S_{T_R} is safe since it is an AES-included supervisor, i.e., $\mathcal{L}(S_{T_R}/\mathbf{G}) \subseteq K$. Overall, we know that $R^{\downarrow CO} \subseteq K$.

($\Rightarrow$) By contradiction. Assume that Algorithm INF-SYNT returns "No Solution" but $R^{\downarrow CO} \subseteq K$. Therefore, we know that there exists a supervisor S such that $R \subseteq \mathcal{L}(S/\mathbf{G}) \subseteq K$ and there exists a sequence in the form of

$$y_0 \xrightarrow{\Gamma_R(y_0|_{\mathbf{R}})\cup\Sigma_{uc}} z_1 \xrightarrow{\sigma_1} y_1 \ldots \xrightarrow{\Gamma_R(y_{n-1}|_{\mathbf{R}})\cup\Sigma_{uc}} z_n \xrightarrow{\sigma_n} y_n \tag{4.3}$$

in procedure DoDFS in Algorithm INF-SYNT such that

1. $\forall i = 0, \ldots, n : y_i|_{\mathbf{R}} \neq \emptyset$; and

2. $\forall i = 0, \ldots, n-1 : \Gamma_R(y_i|_{\mathbf{R}}) \cup \Sigma_{uc} \in C_{\mathcal{AES}(\mathbf{G},\mathbf{K})}(y_i)$; and

3. $\Gamma_R(y_n|_{\mathbf{R}}) \cup \Sigma_{uc} \notin C_{\mathcal{AES}(\mathbf{G},\mathbf{K})}(y_n)$.

Next, we show by induction that, for any $i = 0, \ldots, n$, we have that

$$y_i \subseteq IS^{Y}_{S}(\sigma_1 \ldots \sigma_i) \text{ and } y_i|_{\mathbf{R}} = IS^{Y}_{S}(\sigma_1 \ldots \sigma_i)|_{\mathbf{R}} \tag{4.4}$$

Clearly, the induction basis holds for $i = 0$, since $y_0 = IS^{Y}_{S}(\epsilon)$. Let us assume that Equation (4.4) holds for $i = k$; we need to show that Equation (4.4) holds for $i = k+1$.

By definition, we know that

$$\begin{aligned}
&y_{k+1}\\
=&\{x\in X : \exists x'\in y_k, \exists w\in((\Gamma_R(y_k|_{\mathbf{R}})\cup\Sigma_{uc})\cap\Sigma_{uo})^* \text{ s.t. } \delta(x', w\sigma_{k+1}) = x\} && (4.5)\\
=&\{x\in X_R : \exists x'\in y_k|_{\mathbf{R}}, \exists w\in((\Gamma_R(y_k|_{\mathbf{R}})\cup\Sigma_{uc})\cap\Sigma_{uo})^* \text{ s.t. } \delta_R(x', w\sigma_{k+1}) = x\}\cup A_{G\setminus R} && (4.6)
\end{aligned}$$

where $A_{G\setminus R}\subseteq X\setminus X_R$. Note that the second equality is a consequence of the assumption that $\mathbf{R}\sqsubset\mathbf{G}$, since any string that leaves the state space of $\mathbf{R}$ must lead to a state in $X\setminus X_R$. Similarly, we can write

$$\begin{aligned}
&IS_S^Y(\sigma_1\ldots\sigma_{k+1})\\
=&\{x\in X : \exists x'\in IS_S^Y(\sigma_1\ldots\sigma_k), \exists w\in(S(\sigma_1\ldots\sigma_k)\cap\Sigma_{uo})^* \text{ s.t. } \delta(x', w\sigma_{k+1})=x\} && (4.7)\\
=&\{x\in X_R : \exists x'\in IS_S^Y(\sigma_1\ldots\sigma_k)|_{\mathbf{R}}, \exists w\in(S(\sigma_1\ldots\sigma_k)\cap\Sigma_{uo})^* \text{ s.t. } \delta_R(x', w\sigma_{k+1}) = x\}\\
&\cup B_{G\setminus R} && (4.8)\\
=&\{x\in X_R : \exists x'\in IS_S^Y(\sigma_1\ldots\sigma_k)|_{\mathbf{R}}, \exists w\in((\Gamma_R(IS_S^Y(\sigma_1\ldots\sigma_k)|_{\mathbf{R}})\cup\Sigma_{uc})\cap\Sigma_{uo})^*\\
&\qquad \text{s.t. } \delta_R(x', w\sigma_{k+1}) = x\}\cup B_{G\setminus R} && (4.9)\\
=&\{x\in X_R : \exists x'\in y_k|_{\mathbf{R}}, \exists w\in((\Gamma_R(y_k|_{\mathbf{R}})\cup\Sigma_{uc})\cap\Sigma_{uo})^* \text{ s.t. } \delta_R(x', w\sigma_{k+1}) = x\}\cup B_{G\setminus R} && (4.10)
\end{aligned}$$

where $B_{G\setminus R}\subseteq X\setminus X_R$. Note that the second equality also comes from the assumption that $\mathbf{R}\sqsubset\mathbf{G}$. The third equality comes from the fact that, for any string

$$w\in((S(\sigma_1\ldots\sigma_k)\setminus\Gamma_R(IS_S^Y(\sigma_1\ldots\sigma_k)|_{\mathbf{R}}))\cap\Sigma_{uo})^*$$

$\delta_R(x', w\sigma_{k+1})$ is not defined for $x'\in X_R$. The last equality follows from the induction hypothesis that $y_k|_{\mathbf{R}} = IS_S^Y(\sigma_1\ldots\sigma_k)|_{\mathbf{R}}$.

Therefore, by Equations (4.6) and (4.10), we know that $y_{k+1}|_{\mathbf{R}} = IS_S^Y(\sigma_1\ldots\sigma_{k+1})|_{\mathbf{R}}$.

Moreover, by the induction hypothesis and Proposition 4.3.3, we know

$$\Gamma_R(y_k|_{\mathbf{R}}) = \Gamma_R(IS_S^Y(\sigma \dots \sigma_k)|_{\mathbf{R}}) \subseteq S(\sigma_1 \dots \sigma_k). \tag{4.11}$$

Since Equation (4.4) holds for $i = k$, combing Equations (4.5), (4.7) and (4.11) together, we know that $y_{k+1} \subseteq IS_S^Y(\sigma_1 \dots \sigma_k \sigma_{k+1})$, i.e., Equation (4.4) holds for $i = k+1$.

Now, let us go back to the sequence in Equation (4.3). Since $\mathcal{L}(S/\mathbf{G}) \subseteq K$, we know that

$$S(\sigma_1 \dots \sigma_n) \in C_{\mathcal{AES}(\mathbf{G},\mathbf{K})}(IS_S^Y(\sigma_1 \dots \sigma_n)) \tag{4.12}$$

We have proved that $y_n \subseteq IS_S^Y(\sigma_1 \dots \sigma_n)$. Then, by Proposition 4.3.1, we know that $S(\sigma_1 \dots \sigma_n) \in C_{\mathcal{AES}(\mathbf{G},\mathbf{K})}(y_n)$. Moreover, since we have shown that $\Gamma_R(y_n|_{\mathbf{R}}) = \Gamma_R(IS_S^Y(\sigma_1 \dots \sigma_n)|_{\mathbf{R}})$, by Proposition 4.3.3, we know that $\Gamma_R(y_n|_{\mathbf{R}}) \cup \Sigma_{uc} \subseteq S(\sigma_1 \dots \sigma_n)$. Then, by Proposition 4.3.1 again, we know that $\Gamma_R(y_n|_{\mathbf{R}}) \cup \Sigma_{uc} \in C_{\mathcal{AES}(\mathbf{G},\mathbf{K})}(y_n)$. However, this is a contradiction. $\square$

Remark 4.3.3. Note that the "only if" part of the proof of the above theorem relies on the assumption that $\mathbf{R} \sqsubset \mathbf{K}$. In fact, if $\mathbf{R} \sqsubset \mathbf{G}$ does not hold, then Algorithm INF-SYNT may return "No Solution" even when $R^{\downarrow CO} \subseteq K$. For example, let us use $\mathbf{R}$ and $\mathcal{AES}(\mathbf{G},\mathbf{K})$ shown in Figs. 4.2(c) and 4.2(b), respectively, as the input parameters of Algorithm INF-SYNT, where $\mathbf{R}$ is a sub-automaton of $\mathbf{G}$ but not a *strict* sub-automaton. Then, after taking control decision Σ_{uc} at the initial state and observing event a, we will reach Y-state $\{3,4\}$. Since $\Gamma_R(\{3,4\}) \cup \Sigma_{uc} = \{c_1, c_2\} \cup \Sigma_{uc} \notin C_{\mathcal{AES}(\mathbf{G},\mathbf{K})}(\{3,4\})$, Algorithm INF-SYNT returns "No Solution". However, a solution does exist since $R^{\downarrow CO} \subseteq K$. This highlights our earlier assertion that the strict sub-automaton condition plays an important role in the synthesis algorithm.

The next result reveals that the BTS returned by Algorithm INF-SYNT is indeed the one that realizes the infimal safe supervisor.

Theorem IV.2. *Suppose that Algorithm INF-SYNT returns BTS T_R. Then we have $\mathcal{L}(S_{T_R}/\boldsymbol{G}) = R^{\downarrow CO}$.*

Proof. We prove this by contradiction. Let us assume that $\mathcal{L}(S_{T_R}/\mathbf{G}) \neq R^{\downarrow CO}$. In the proof of Theorem IV.1, we have shown that $R \subseteq \mathcal{L}(S_{T_R}/\mathbf{G})$. Then we know that there exists a supervisor S' such that $R \subseteq \mathcal{L}(S'/\mathbf{G}) \subset \mathcal{L}(S_{T_R}/\mathbf{G})$. Therefore, we know that there exists an observable string $s \in P(\mathcal{L}(S'/\mathbf{G})) \cap P(\mathcal{L}(S_{T_R}/\mathbf{G}))$ such that $S'(s) \subset S_{T_R}(s)$. For string s, we have the following two cases.

Case 1: $IS^Y_{S_{T_R}}(s)|_{\mathbf{R}} = \emptyset$.

In this case, by Algorithm INF-SYNT, we know that $S_{T_R}(s) = c_{T_R}(IS^Y_{S_{T_R}}(s)) = \Sigma_{uc}$. However, it contradicts the fact that $S'(s) \subset S_{T_R}(s)$, since $S'(s)$ always contains Σ_{uc}.

Case 2: $IS^Y_{S_{T_R}}(s)|_{\mathbf{R}} \neq \emptyset$.

In this case, by Algorithm INF-SYNT, we know that $S_{T_R}(s) = \Gamma_R(IS^Y_{S_{T_R}}(s)|_{\mathbf{R}}) \cup \Sigma_{uc}$. Since $R \subseteq \mathcal{L}(S'/\mathbf{G})$, by Proposition 4.3.3, we know that $\Gamma_R(IS^Y_{S'}(s)|_{\mathbf{R}}) \cup \Sigma_{uc} \subseteq S'(s)$. Moreover, by Proposition 4.3.2, we know that $IS^Y_{S'}(s)|_{\mathbf{R}} = IS^Y_{S_{T_R}}(s)|_{\mathbf{R}} = y_R(s)$, since both S' and S_{T_R} contains R. This implies that $\Gamma_R(IS^Y_{S_{T_R}}(s)|_{\mathbf{R}}) \cup \Sigma_{uc} \subseteq S'(s)$. However, this also contradicts the fact that $S'(s) \subset S_{T_R}(s)$. $\square$

Remark 4.3.4. Although language-based formulas for $R^{\downarrow CO}$ were provided in [44, 68], the formula-based approach does not tell us what is the right structure to *realize* the supervisor achieving $R^{\downarrow CO}$. To the best of our knowledge, no constructive approach for $R^{\downarrow CO}$, in terms of supervisor, is provided in the literature. The results in this section not only provide a direct *constructive approach* to compute the infimal prefix-closed controllable and observable super-language, but also provides a new *structural property* about the corresponding infimal supervisor. In particular, we show that, under the assumption that $\mathbf{R} \sqsubset \mathbf{K} \sqsubset \mathbf{G}$, 2^X is sufficient enough to represent this supervisor, i.e., the infimal supervisor can be written in the form of $S_{T_R} : 2^X \rightarrow \Gamma$. Moreover, the BTS T_R that realizes the infimal supervisor will be further used as a basis to synthesize a maximal safe supervisor containing R. This will be discussed in

Section 4.5.

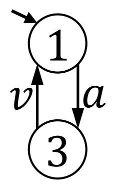
4.4 Control Simulation Relation

In this section, we first discuss the difficulty that arises in solving the range control problem. Then we define the notion of Control Simulation Relation (CSR) as the tool to overcome the difficulty.

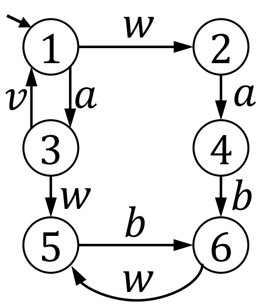
4.4.1 Difficulty in Handling the Lower Bound

In order to synthesize a maximal supervisor, the general idea is to guarantee by construction that the control decision made by the supervisor at each instant cannot be improved any further. However, this is not an easy task. Suppose that $y \in Q_Y^{AES}$ is a Y-state in the AES; we know that any control decision in $C_{\mathcal{AES}(\mathbf{G},\mathbf{K})}(y)$ is a safe control decision. Therefore, if there is no lower bound requirement and one is only interested in the safety upper bound K, then we can simply pick a "greedy maximal" decision from $C_{\mathcal{AES}(\mathbf{G},\mathbf{K})}(y)$. This is essentially the strategy we use in Chapters II and II; a similar strategy (but not based on the AES) is used in [4]. However, the following example illustrates how to choose a control decision from $C_{\mathcal{AES}(\mathbf{G},\mathbf{K})}(y)$ becomes much more complicated when the lower bound specification R has to be considered.

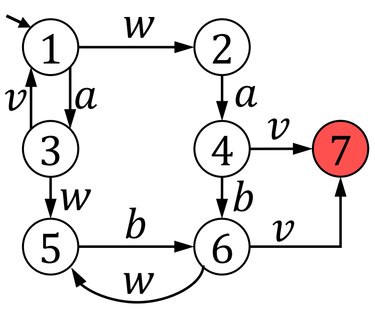
Example 4.4.1. *Let us consider automata $\boldsymbol{R}, \boldsymbol{K}$ and $\boldsymbol{G}$ shown in Figs. 4.4(a), 4.4(b) and 4.4(c), respectively, where we have $\boldsymbol{R} \sqsubset \boldsymbol{K} \sqsubset \boldsymbol{G}$. Let $\Sigma_c = \{v, w\}$ and $\Sigma_o = \{a, b, v\}$. The AES $\mathcal{AES}(\boldsymbol{G}, \boldsymbol{K})$ is shown in Fig. 4.4(d). By applying Algorithm INF-SYNT, we construct BTS T_R that realizes the infimal supervisor achieving $R^{\downarrow CO}$; T_R is shown in Fig. 4.4(e). Initially, T_R chooses to disable w, i.e., $c_{T_R}(y_0) = \{\}$, while enabling w is also a safe choice at the initial Y-state according to the AES. It seems that choosing $\{w\}$ provides more behavior than choosing $\{\}$. However, if we choose*



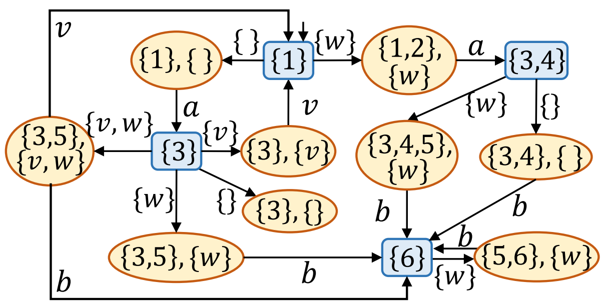
(a) **R**



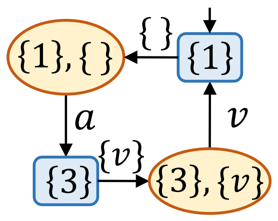
(b) **K**



(c) **G**



(d) $\mathcal{AES}(\mathbf{G}, \mathbf{K})$



(e) T_R

Figure 4.4: For **R**, **K** and **G**: $\Sigma_c = \{v, w\}$ and $\Sigma_o = \{a, b, v\}$.

$\{w\}$, then upon the occurrence of a, we can only choose to disable v, since we are not sure whether the current state is 3 or 4. This leads to failure to contain the lower bound behavior $(av)^*$, where we need to enable v after observing a. Therefore, the lower bound behavior can only be achieved by choosing $\{\}$ at the beginning rather than choosing $\{w\}$, which is greedy maximal.

The above example illustrates the following issue. In some scenario, enabling more events is not a good choice, since it may introduce more information uncertainty. Consequently, to maintain safety, the control decision may become more conservative in the future due to this information uncertainty. This may make the lower bound behavior unachievable. More problematically, we do not know whether or not enabling an event will lead to failure to contain the lower bound behavior, unless we get stuck at some instant in the future, e.g., after observing event o in the previous example. Moreover, we do not know a priori, when or whether or not this phenomenon will occur in the future. In other words, whether or not a decision defined in the AES is a "good" control decision depends on its effects in the future. This future dependency is the fundamental difficulty of the range control problem and it is in fact the essential difference between MPRCP and the standard supervisor synthesis problem without a lower bound requirement.

4.4.2 Definition of the CSR

In order to resolve the future dependency issue discussed above, we propose a simulation-like relation, called the Control Simulation Relation (CSR), to pre-process this future dependency and transform it to local information. The key idea is to compare two BTSs T_1 and T_2 and to establish a formal relationship between states in T_1 and states in T_2. The formal definition of the CSR is presented next.

Definition 4.4.1. *(Control Simulation Relation). Let T_1 and T_2 be two BTSs. A relation $\Phi = \Phi_Y \cup \Phi_Z \subseteq (Q_Y^{T_1} \times Q_Y^{T_2}) \cup (Q_Z^{T_1} \times Q_Z^{T_2})$ is said to be a* control simulation relation *from T_1 to T_2 if the following conditions hold:*

1. $(y_0, y_0) \in \Phi$;

2. For every $(y_1, y_2) \in \Phi_Y$ we have that:
$y_1 \xrightarrow{\gamma_1}_{T_1} z_1$ in T_1 implies the existence of $y_2 \xrightarrow{\gamma_2}_{T_2} z_2$ in T_2 such that $\gamma_1 \subseteq \gamma_2$ and $(z_1, z_2) \in \Phi_Z$;

3. For every $(z_1, z_2) \in \Phi_Z$ we have that:
$z_1 \xrightarrow{\sigma}_{T_1} y_1$ in T_1 implies the existence of $z_2 \xrightarrow{\sigma}_{T_2} y_2$ in T_2 such that $(y_1, y_2) \in \Phi_Y$.

We say that T_1 is control-simulated by T_2 or that T_2 control-simulates T_1, denoted by $T_1 \preceq T_2$, if there exists a control simulation relation from T_1 to T_2.

Intuitively, the control simulation relation captures whether or not T_2 is able to match an arbitrary control decision made by T_1 by either taking the same control decision or a control decision that is strictly larger than the one made by T_1 and maintain this ability for all possible future behaviors.

Given two BTSs T_1 and T_2, a relevant question is whether or not there exists a CSR from T_1 to T_2. To answer this question, we define an operator

$$F : 2^{Q_Y^{T_1} \times Q_Y^{T_2}} \cup 2^{Q_Z^{T_1} \times Q_Z^{T_2}} \to 2^{Q_Y^{T_1} \times Q_Y^{T_2}} \cup 2^{Q_Y^{T_1} \times Q_Y^{T_2}}$$

as follows. For any $\Phi = \Phi_Y \cup \Phi_Z \subseteq (Q_Y^{T_1} \times Q_Y^{T_2}) \cup (Q_Z^{T_1} \times Q_Z^{T_2})$, we have that

1. $(y_1, y_2) \in F(\Phi_Y)$ if $(y_1, y_2) \in \Phi_Y$ and for any transition $y_1 \xrightarrow{\gamma_1}_{T_1} z_1$ in T_1, there exists $y_2 \xrightarrow{\gamma_2}_{T_2} z_2$ in T_2 such that $\gamma_1 \subseteq \gamma_2$ and $(z_1, z_2) \in \Phi_Z$.

2. $(z_1, z_2) \in F(\Phi_Z)$ if $(z_1, z_2) \in \Phi_Z$ and for any transition $z_1 \xrightarrow{\sigma}_{T_1} y_1$ in T_1, there exists $z_2 \xrightarrow{\sigma}_{T_2} y_2$ in T_2 such that $(y_1, y_2) \in \Phi_Y$.

The following results reveal how operator F is related to the CSR.

Proposition 4.4.1. *The operator F has following properties:*

1. Φ is a control simulation relation from T_1 to T_2, if and only if, $\Phi \subseteq F(\Phi)$ and $(y_0, y_0) \in \Phi$;

2. $\Phi_1 \subseteq \Phi_2 \Rightarrow F(\Phi_1) \subseteq F(\Phi_2)$.

Proof. The proof is similar to the proof in [91] for the standard simulation relation. Suppose that $\Phi = \Phi_Y \cup \Phi_Z$ is a control simulation relation from T_1 to T_2. Let $(y_1, y_2) \in \Phi_Y$. Since $(\forall y_1 \xrightarrow{\gamma_1}_{T_1} z_1)(\exists y_2 \xrightarrow{\gamma_2}_{T_2} z_2)[\gamma_1 \subseteq \gamma_2 \wedge (z_1, z_2) \in \Phi_Z]$, we know that $(y_1, y_2) \in F(\Phi)$. Similarly, for any $(z_1, z_2) \in \Phi_Z$, since $(\forall z_1 \xrightarrow{\sigma}_{T_1} y_1)(\exists z_2 \xrightarrow{\sigma}_{T_2} y_2)[(y_1, y_2) \in \Phi_Y]$, we know that $(z_1, z_2) \in F(\Phi_Z)$. Therefore, we conclude that $\Phi \subseteq F(\Phi)$ and $(y_0, y_0) \in \Phi$.

Suppose that $\Phi \subseteq F(\Phi)$ and $(y_0, y_0) \in \Phi$. Clearly, the first requirement in Definition 4.4.1 is satisfied. For any $(y_1, y_2) \in \Phi_Y$, we know that the first requirement in the definition of F implies that second requirement in Definition 4.4.1. Similarly, for any $(z_1, z_2) \in \Phi_Z$, we know that the second requirement in the definition of F implies that third requirement in Definition 4.4.1. Hence, we know that Φ is a control simulation relation from T_1 to T_2.

Now we prove the second property. For any $(y_1, y_2) \in F(\Phi_1) \cap (Q_Y^{T_1} \times Q_Y^{T_2})$, we have that $(y_1, y_2) \in \Phi_1$ and

$$(\forall y_1 \xrightarrow{\gamma_1}_{T_1} z_1)(\exists y_2 \xrightarrow{\gamma_2}_{T_2} z_2)[\gamma_1 \subseteq \gamma_2 \wedge (z_1, z_2) \in \Phi_1] \tag{4.13}$$

Since $\Phi_1 \subseteq \Phi_2$, we know that $(y_1, y_2), (z_1, z_2) \in \Phi_2$. Therefore, Equation (4.13) implies that $(y_1, y_2) \in F(\Phi_2)$. Similarly, for any $(z_1, z_2) \in F(\Phi_1) \cap (Q_Z^{T_1} \times Q_Z^{T_2})$, we have that $(z_1, z_2) \in \Phi_1$ and $(\forall z_1 \xrightarrow{\sigma}_{T_1} y_1)(\exists z_2 \xrightarrow{\sigma}_{T_2} y_2)[(y_1, y_2) \in \Phi_1]$. Since $\Phi_1 \subseteq \Phi_2$, we also know that $(y_1, y_2), (z_1, z_2) \in \Phi_2$. Therefore, $(z_1, z_2) \in F(\Phi_2)$. □

The above results have the following implications. First, since $\Phi \subseteq F(\Phi)$ for any CSR Φ, we know that the maximal relation Φ is a fixed-point of operator F, i.e., $F(\Phi) = \Phi$. Note that $F(\Phi) \subseteq \Phi$ always holds. By the second property in Proposition 4.4.1, we know that F is monotone. Therefore, by Tarski's fixed-point theorem [94], we know that the supremal fixed-point of F, denoted by $\Phi^*(T_1, T_2)$, exists and it can be computed as follows

$$\Phi^*(T_1, T_2) = \lim_{k \to \infty} F^k((Q_Y^{T_1} \times Q_Y^{T_2}) \cup (Q_Z^{T_1} \times Q_Z^{T_2})) \tag{4.14}$$

In other words, $\Phi^*(T_1, T_2)$ is a maximal control simulation relation from T_1 to T_2 if $(y_0, y_0) \in \Phi^*(T_1, T_2)$. Otherwise, $T_1 \not\precsim T_2$ if $(y_0, y_0) \notin \Phi^*(T_1, T_2)$. This is similar to the standard simulation relation; see, e.g., [91]. Note that the limit in Equation (4.14) can be achieved within at most $|Q_Y^{T_1}||Q_Y^{T_2}| + |Q_Z^{T_1}||Q_Z^{T_2}|$ iterations.

Example 4.4.2. *We consider again the AES $\mathcal{AES}(\boldsymbol{G}, \boldsymbol{K})$ and BTS T_R shown in*

Figs. 4.4(e) and 4.4(d), respectively. We compute the maximal CSR between T_R and $\mathcal{AES}(\boldsymbol{G},\boldsymbol{K})$; we write $\Phi^*(T_R,\mathcal{AES}(\boldsymbol{G},\boldsymbol{K}))=\Phi^*_Y\cup\Phi^*_Z$, where $\Phi^*_Y\subseteq Q_Y^{T_R}\times Q_Y^{AES}$ and $\Phi^*_Z\subseteq Q_Z^{T_R}\times Q_Z^{AES}$. Then we have

$$\Phi^*_Y=\{(\{1\},\{1\}),(\{3\},\{3\}))\}$$
$$\Phi^*_Z=\{((\{1\},\{\}),(\{1\},\{\})),((\{3\},\{v\}),(\{3\},\{v\})),((\{3\},\{v\}),(\{3,5\},\{v,w\}))\}$$

The reason why $(\{3\},\{3,4\})\notin\Phi^*_Y$ is that $\{v\}$ is defined at $\{3\}$ in T_R but there is no decision containing $\{v\}$ defined at $\{3,4\}$ in the AES. Consequently, we know that $((\{1\},\{\}),(\{1,2\},\{w\}))\notin\Phi^*_Z$, where $(\{1\},\{\})$ and $(\{1,2\},\{w\})$ are the predecessor Z-states that enter $\{3\}$ and $\{3,5\}$ with the same event a, respectively.

4.4.3 Properties of the CSR

Hereafter, we present properties of the CSR that will be used later.

The first result reveals that the CSR indeed captures whether or not any possible behavior from a state in a BTS can be matched by another BTS from some different state.

Proposition 4.4.2. *Let T_1 and T_2 be two complete BTSs and $z_1\in Q_Z^{T_1}$ and $z_1'\in Q_Z^{T_2}$ be two Z-states in T_1 and T_2, respectively. Then $(z_1,z_1')\in\Phi^*(T_1,T_2)$, if and only if, for any sequence*

$$z_1\xrightarrow{\sigma_1}y_1\xrightarrow{\gamma_1}\dots\xrightarrow{\gamma_{n-1}}z_{n-1}\xrightarrow{\sigma_n}y_n\xrightarrow{\gamma_n}z_n \tag{4.15}$$

in T_1, there exists a sequence

$$z_1'\xrightarrow{\sigma_1}y_1'\xrightarrow{\gamma_1'}\dots\xrightarrow{\gamma_{n-1}'}z_{n-1}'\xrightarrow{\sigma_n}y_n'\xrightarrow{\gamma_n'}z_n' \tag{4.16}$$

in T_2, such that $\gamma_i\subseteq\gamma_i',\forall i\geq 0$.

Proof. The "only if" part is straightforward. For a sequence in Equation (4.15), we can always construct a sequence in Equation (4.16) by choosing γ_i' for each $i \geq 0$ such that $\gamma_i \subseteq \gamma_i'$ and $(h_{YZ}(y_i, \gamma_i), h_{YZ}(y_i', \gamma_i')) \in \Phi^*(T_1, T_2)$. The definition of the CSR guarantees the existence of such γ_i' at each y_i' encountered.

Next, we show the "if part" by contraposition. Suppose that $(z_1, z_1') \notin \Phi^*(T_1, T_2)$. Then, by Equation (4.14), either there exists an event $\sigma_1 \in \Sigma_o : h_{ZY}(z_1, \sigma_1)!$ but $\sigma_1 \in \Sigma_o : h_{ZY}(z_1, \sigma_1)\neg!$, where "$\neg!$" means "is not defined"; or there exists $\Phi_1 \supset \Phi^*(T_1, T_2)$ such that $(z_1, z_1') \in \Phi_1$ but

$$(\exists \sigma_1 \in \Sigma_o)[(y_1, y_1') \notin \Phi_1] \tag{4.17}$$

where $y_1 = h_{ZY}(z_1, \sigma_1)$ and $y_1' = h_{ZY}(z_1', \sigma_1)$.

For the first case, we know immediately that there exists a sequence $z_1 \xrightarrow{\sigma_1} y_1 \xrightarrow{\gamma_1} z_2$ in T_1, where γ_1 is an arbitrary control decision in $C_{T_1}(y_1)$, such that there does not exist a sequence $z_1' \xrightarrow{\sigma_1} y_1' \xrightarrow{\gamma_1'} z_2'$ in T_2 satisfying $\gamma_1 \subseteq \gamma_1'$. Hereafter, we consider the case where Equation (4.17) holds. Again, by Equation (4.14), $(y_1, y_1') \notin \Phi_1$ implies that either

$$(\exists \gamma_1 \in C_{T_1}(y_1))(\forall \gamma_1' \in C_{T_2}(y_1'))[\gamma_1 \nsubseteq \gamma_1'] \tag{4.18}$$

or there exists $\Phi_2 \supset \Phi_1$ such that $(y_1, y_1') \in \Phi_2$ but

$$(\exists \gamma_1 \in C_{T_1}(y_1)(\forall \gamma_1' \in C_{T_2}(y_1') : \gamma_1 \subseteq \gamma_1')[(z_2, z_2') \notin \Phi_2] \tag{4.19}$$

where $z_2 = h_{YZ}(y_1, \gamma_1)$ and $z_2' = h_{YZ}(y_1', \gamma_1')$.

Suppose that Equation (4.18) holds, then we also know immediately that there exists a sequence $z_1 \xrightarrow{\sigma_1} y_1 \xrightarrow{\gamma_1} z_2$ in T_1, such that there does not exist a sequence $z_1' \xrightarrow{\sigma_1} y_1' \xrightarrow{\gamma_1'} z_2'$ in T_2 satisfying $\gamma_1 \subseteq \gamma_1'$. Suppose that Equation (4.19) holds. Let $\gamma_1 \in C_{T_1}(y_1)$ be a control decision satisfying Equation (4.19) and let $\gamma_1' \in C_{T_2}(y_1')$ be an arbitrary control decision such that $\gamma_1 \subseteq \gamma_1'$. Note that $\gamma_1 \subseteq \gamma_1'$ implies that $\forall\sigma \in \Sigma_o : h_{ZY}^{T_1}(z_2, \sigma)! \Rightarrow h_{ZY}^{T_2}(z_2', \sigma)!$. Since $(z_2, z_2') \notin \Phi_2$, by Equation (4.14), $\exists\Phi_3 \supset \Phi_2$ such that $(z_2, z_2') \in \Phi_3$ but

$$(\exists\sigma_2 \in \Sigma_o)[(y_2, y_2') \notin \Phi_3] \tag{4.20}$$

where $y_2 = h_{ZY}(z_2, \sigma_2)$ and $y_2' = h_{ZY}(z_2', \sigma_2)$.

By iteratively applying the above arguments, suppose that, for some $m \geq 1$, we have that

$$(\exists\gamma_m \in C_{T_1}(y_m))(\forall\gamma_m' \in C_{T_2}(y_m'))[\gamma_m \not\subseteq \gamma_m'] \tag{4.21}$$

and

$$(\forall 1 \leq i \leq m)(\exists\Phi_{2i-1} \supset \Phi_{2i-2} : (z_i, z_i') \in \Phi_{2i-1})(\exists\sigma_i \in \Sigma_o :)[(y_i, y_i') \notin \Phi_{2i-1}] \tag{4.22}$$

where $y_i = h_{ZY}(z_i, \sigma_i)$, $y_i' = h_{ZY}(z_i', \sigma_i)$ and $\Phi_0 = \Phi^*(T_1, T_2)$; and

$$\begin{aligned}&(\forall 1 \leq i \leq m-1)(\exists\Phi_{2i} \supset \Phi_{2i-1} : (y_i, y_i') \in \Phi_{2i})(\exists\gamma_i \in C_{T_1}(y_i))\\&(\forall\gamma_i' \in C_{T_2}(y_i') : \gamma_i \subseteq \gamma_i')[(z_{i+1}, z_{i+1}') \notin \Phi_{2i}]\end{aligned} \tag{4.23}$$

where $z_{i+1} = h_{YZ}(y_i, \gamma_i)$ and $z_{i+1}' = h_{YZ}(y_i', \gamma_i')$. In particular, Equations (4.21), (4.22), and (4.23) are the generalizations of Equations (4.18), (4.20) and (4.19), respectively. Then we know that there exists a sequence $z_1 \xrightarrow{\sigma_1} y_1 \xrightarrow{\gamma_1} \ldots \xrightarrow{\gamma_{m-1}} z_{m-1} \xrightarrow{\sigma_m} y_m \xrightarrow{\gamma_m} z_m$ in T_1 such that, for any sequence $z_1' \xrightarrow{\sigma_1} y_1' \xrightarrow{\gamma_1'} \ldots \xrightarrow{\gamma_{m-1}'} z_{m-1}' \xrightarrow{\sigma_m} y_m' \xrightarrow{\gamma_m'} z_m'$ in T_2, if $\gamma_i \subseteq \gamma_i', i = 0, \ldots, m-1$, then there does not exist a control decision $\gamma_m' \in C_{T_2}(y_m')$ such that $\gamma_m \subseteq \gamma_m'$. This completes the contrapositive proof.

Note that, since $\Phi_1 \subset \Phi_2 \subset \cdots \subset \Phi_{2m}$ is strictly increasing, such a m always exists. To see this, let $\Phi_{2m} = (Q_Y^{T_1} \times Q_Y^{T_2}) \cup (Q_Z^{T_1} \times Q_Z^{T_2})$, which is the largest possible relation. Suppose that for any $i < m$, there exists $\Phi_{2i} \supset \Phi_{2i-1}$ such that $(y_i, y_i') \in \Phi_{2i}$ but $(\exists\gamma_i \in C_{T_1}(y_i))(\forall\gamma_i' \in C_{T_2}(y_i') : \gamma_i \subseteq \gamma_i')[(z_{i+1}, z_{i+1}') \notin \Phi_{2i}]$ Then for Φ_{2m}, it must be $(\exists \gamma_m \in C_{T_1}(y_m))(\forall \gamma'_m \in C_{T_2}(y'_m))[\gamma_m \nsubseteq \gamma'_m]$ since there does not exist a relation that is larger than Φ_{2m} anymore. $\square$

The next result reveals the relationship between the CSR and the closed-loop behavior of the system.

Proposition 4.4.3. *Let T_1 and T_2 be two deterministic BTSs. Then $\mathcal{L}(S_{T_1}/\boldsymbol{G}) \subseteq \mathcal{L}(S_{T_2}/\boldsymbol{G})$, if and only if, $T_1 \preceq T_2$.*

Proof. ($\Rightarrow$) By contraposition. Suppose that $T_1 \npreceq T_2$, which means that $(y_0, y_0) \notin \Phi^*(T_1, T_2)$. Therefore, either (i) $c_{T_1}(y_0) \nsubseteq c_{T_2}(y_0)$; or (ii) $(z_1^1, z_1^2) \notin \Phi^*(T_1, T_2)$, where $z_1^i = h_{YZ}(y_0, c_{T_i}(y_0)), i = 1, 2$. If case (i) holds, then we know immediately that $\mathcal{L}(S_{T_1}/\mathbf{G}) \nsubseteq \mathcal{L}(S_{T_2}/\mathbf{G})$, since $S_{T_1}(\epsilon) = c_{T_1}(y_0) \nsubseteq c_{T_2}(y_0) = S_{T_2}(\epsilon)$. If case (ii) holds, then by Proposition 4.4.2, there exists a string $\sigma_1 \dots \sigma_n \in P(\mathcal{L}(S_{T_1}/\mathbf{G}))$ such that $S_{T_1}(\sigma_1 \dots \sigma_i) \subseteq S_{T_2}(\sigma_1 \dots \sigma_i), \forall i = 1, \dots, n-1$ but $S_{T_1}(\sigma \dots \sigma_n) \nsubseteq S_{T_2}(\sigma \dots \sigma_n)$. Therefore, we still have that $\mathcal{L}(S_{T_1}/\mathbf{G}) \nsubseteq \mathcal{L}(S_{T_2}/\mathbf{G})$.

($\Leftarrow$) By contraposition. Suppose that $\mathcal{L}(S_{T_1}/\mathbf{G}) \nsubseteq \mathcal{L}(S_{T_2}/\mathbf{G})$. Then we know that there exists $\sigma_1 \dots \sigma_n \in P(\mathcal{L}(S_{T_1}/\mathbf{G}))$ such that $S_{T_1}(\sigma_1 \dots \sigma_{n-1}) \subseteq S_{T_2}(\sigma_1 \dots \sigma_{n-1})$ but $S_{T_1}(\sigma_1 \dots \sigma_n) \nsubseteq S_{T_2}(\sigma_1 \dots \sigma_n)$. Since T_1 is deterministic, the above string $\sigma_1 \dots \sigma_n$ uniquely determines the following sequence in T_1

$$y_0 \xrightarrow{\gamma_0^1} z_1^1 \xrightarrow{\sigma_1} y_1^1 \xrightarrow{\gamma_1^1} \dots \xrightarrow{\gamma_{n-1}^1} z_{n-1}^1 \xrightarrow{\sigma_n} y_n^1 \xrightarrow{\gamma_n^1} z_n^1 \tag{4.24}$$

where $\gamma_i^1 = S_{T_1}(\sigma_1 \dots \sigma_i)$ is the unique control decision defined at y_i^1. However, there does not exist a sequence

$$y_0 \xrightarrow{\gamma_0^2} z_1^2 \xrightarrow{\sigma_1} y_1^2 \xrightarrow{\gamma_1^2} \dots \xrightarrow{\gamma_{n-1}^2} z_{n-1}^2 \xrightarrow{\sigma_n} y_n^2 \xrightarrow{\gamma_n^2} z_n^2 \tag{4.25}$$

in T_2 such that $\gamma_i^1 \subseteq \gamma_i^2, \forall i = 1, \dots, n$, since the control decision from each y_i^2 in T_2 is uniquely defined and the only control decision defined at y_n^2, i.e., $S_{T_2}(\sigma_1 \dots \sigma_n)$, does not contain $\gamma_n^1 = S_{T_1}(\sigma_1 \ldots \sigma_n)$. □

The last result reveals that the CSR is transitive.

Proposition 4.4.4. *Let T_1, T_2 and T_3 be three BTSs such that $T_1 \preceq T_2$ and $T_2 \preceq T_3$. For $i = 1, 2, 3$, let $y_i \in Q_Y^{T_i}$ and $\gamma_i \in C_{T_i}(y_i)$ be a Y-state in T_i and a control decision defined at this state, respectively. Then*

$$[(z_1, z_2) \in \Phi^*(T_1, T_2) \wedge (z_2, z_3) \in \Phi^*(T_2, T_3)] \Rightarrow [(z_1, z_3) \in \Phi^*(T_1, T_3)]$$

where $z_i = h_{YZ}(y_i, \gamma_i), i = 1, 2, 3$.

Proof. Let $z_1 \xrightarrow{\sigma_1} y_1^1 \xrightarrow{\gamma_1^1} \ldots \xrightarrow{\gamma_{n-1}^1} z_{n-1}^1 \xrightarrow{\sigma_n} y_n^1 \xrightarrow{\gamma_n^1} z_n^1$ be an arbitrary sequence in T_1. Since $(z_1, z_2) \in \Phi^*(T_1, T_2)$, by Proposition 4.4.2, there exists a sequence $z_2 \xrightarrow{\sigma_1} y_1^2 \xrightarrow{\gamma_1^2} \ldots \xrightarrow{\gamma_{n-1}^2} z_{n-1}^2 \xrightarrow{\sigma_n} y_n^2 \xrightarrow{\gamma_n^2} z_n^2$ in T_2 such that $\gamma_i^1 \subseteq \gamma_i^2, \forall i = 1, \ldots, n$. Similarly, since $(z_2, z_3) \in \Phi^*(T_2, T_3)$, by Proposition 4.4.2, there exists a sequence $z_3 \xrightarrow{\sigma_1} y_1^3 \xrightarrow{\gamma_1^3} \ldots \xrightarrow{\gamma_{n-1}^3} z_{n-1}^3 \xrightarrow{\sigma_n} y_n^3 \xrightarrow{\gamma_n^3} z_n^3$ in T_3 such that $\gamma_i^2 \subseteq \gamma_i^3, \forall i = 1, \ldots, n$. Therefore, $(z_1, z_3) \in \Phi^*(T_1, T_3)$ by Proposition 4.4.2. □

4.5 Synthesis of a Maximally-Permissive Supervisor

In this section, we first present the main synthesis algorithm that solves MPRCP. Then we prove its correctness.

4.5.1 Synthesis Algorithm

As we discussed earlier, to synthesize a maximally permissive supervisor containing R, we need to consider some information in the future. Fortunately, such future information has been transformed to local information by the CSR. The idea of the synthesis algorithm is as follows. First, we construct BTS T_R that includes the infimal supervisor S_{T_R} achieving $R^{\downarrow CO}$. Then we compute the maximal CSR between BTS

T_R and the AES $\mathcal{AES}(\mathbf{G}, \mathbf{K})$. Next, we construct a new BTS, denoted by T^*, such that $T_R \preceq T^*$, by using a depth-first search procedure. Specifically, suppose that y is a Y-state in T^* at which we need to choose a control decision. First, this control decision should be chosen from $C_{\mathcal{AES}(\mathbf{G},\mathbf{K})}(y)$ in order to guarantee safety. In order to take care of the lower bound behavior, we need to make sure that this control decision preserves the CSR. The reason why we consider the CSR between T_R and $\mathcal{AES}(\mathbf{G}, \mathbf{K})$ is that T_R realizes the infimal supervisor containing R; namely, any BTS whose induced supervisor contains R should "simulate" the behavior of T_R.

In order to formalize the above idea, let $y \in Q_Y^{AES}$ be a Y-state in the AES and $\hat{y} \in Q_Y^{T_R}$ be a Y-state that "tracks" y in T_R such that $y|_{\mathbf{R}}, \hat{y}|_{\mathbf{R}} \neq \emptyset$. (How $\hat{y}$ "tracks" y will be clear later.) We denote by $\Phi_R^* := \Phi^*(T_R, \mathcal{AES}(\mathbf{G}, \mathbf{K}))$ the maximal CSR from T_R to $\mathcal{AES}(\mathbf{G}, \mathbf{K})$. Then we define

$$\Xi(y, \hat{y}) := \left\{ \gamma \in \Gamma : \begin{array}{c} \gamma \in C_{\mathcal{AES}(\mathbf{G},\mathbf{K})}(y) \textbf{ and } \gamma \supseteq c_{T_R}(\hat{y}) \textbf{ and} \\ (h_{YZ}(\hat{y}, c_{T_R}(\hat{y})), h_{YZ}(y, \gamma)) \in \Phi_R^* \end{array} \right\}$$

Set $\Xi(y, \hat{y})$ will be the key in the synthesis algorithm. Intuitively, $\gamma \in \Xi(y, \hat{y})$ is a control decision such that:

1. It is safe at y, i.e., $\gamma \in C_{\mathcal{AES}(\mathbf{G},\mathbf{K})}(y)$; and

2. It contains the corresponding control decision made by S_{T_R} at $\hat{y}$, i.e., $c_{T_R}(\hat{y})$; and

3. Any behavior that can occur from the corresponding Y-state $\hat{y}$ in T_R can still occur from y in the AES by taking γ.

We are now ready to present the main synthesis algorithm, which is formally presented in Algorithm MAX-RANGE. Let us explain how it works. Initially, we construct T_R and compute the maximal CSR $\Phi^*(T_R, \mathcal{AES}(\mathbf{G}, \mathbf{K}))$ from T_R to $\mathcal{AES}(\mathbf{G}, \mathbf{K})$. Then we construct a new deterministic BTS T^* by a depth-first search as follows. Initially, we start from the initial Y-state y_0. We pick *one* control decision from the AES for each Y-state encountered (how this control decision is picked will be specified soon) and pick *all* observations for each Z-state encountered. This depth-first search is implemented by recursive procedure DoDFS in Algorithm MAX-RANGE, which traverses the reachable state space of T^*. Moreover, during the construction BTS of T^*, we use T_R to track the sequence that reaches the Y- or Z-state in T^*. Specifically, whenever T^* moves from a Y-state y to a Z-state z (line 10), we need to move from Y-state $\hat{y}$ to its (unique) successor Z-state $\hat{z}$ in T_R (line 12). Similarly, whenever T^* moves from a Z-state z to a Y-state y via observable event σ (line 18), we need to move from Z-state $\hat{z}$ to a successor Y-state $\hat{y}$ in T_R through the same observable event σ (line 20). In other words, Y-state $\hat{y}$ in T_R essentially "tracks" Y-state y in the AES or in T^*, since they are always reached by sequences that have the same projected string. Note that we use T_R to track T^* only when the current Y-state y encountered in T^* satisfies $y|_{\mathbf{R}} \neq \emptyset$. Whenever $y|_{\mathbf{R}} = \emptyset$, then we just set $\hat{y} = \emptyset$ (line 21). This means that we know for sure that the string is already outside of $\mathcal{L}(\mathbf{R})$.

Now it still remains to discuss how to choose the control decision at each Y-state in T^*. To this end, we need to consider two cases for each Y-state y encountered:
1) Suppose that $y|_{\mathbf{R}} \neq \emptyset$; this means that y must be reached by a sequence $y_0 \xrightarrow{\gamma_1\sigma_1\ldots\gamma_n\sigma_n} y$ such that the projected string in this sequence is in $P(R)$, i.e., $\sigma_1 \ldots \sigma_n \in P(R)$. Then we know that there exists a sequence in T_R that "tracks" the above sequence, which means that $\hat{y} \neq \emptyset$. In this case, we choose a locally maximal decision in $\Xi(y, \hat{y})$, since we still need to be able to match any behavior in R in the future. This case is implemented by line 8 of Algorithm MAX-RANGE.
2) Suppose that $y|_{\mathbf{R}} = \emptyset$; this means that y must be reached by a sequence $y_0 \xrightarrow{\gamma_1\sigma_1\ldots\gamma_n\sigma_n} y$ such that $\sigma_1 \ldots \sigma_n \notin P(R)$. This also implies that $\hat{y} = \emptyset$. Then we simply chose a locally maximal decision in $C_{\mathcal{AES}(\mathbf{G},\mathbf{K})}(y)$, since we know for sure that the string

is already outside of $\mathcal{L}(\mathbf{R})$. This case is implemented by line 9 of Algorithm MAX-RANGE.

We illustrate Algorithm MAX-RANGE in the next example.

Example 4.5.1. *Let us return to the system we have considered in Example 4.4.1. The inputs are BTS T_R that includes the infimal supervisor and the AES $\mathcal{AES}(\boldsymbol{G}, \boldsymbol{K})$, which are shown in Figs. 4.4(e) and 4.4(d), respectively. We first start procedure DoDFS from the pair of initial Y-states, i.e., $y = \hat{y} = y_0 = \{1\}$. Since $((\{1\}, \{\}), (\{1, 2\}, \{w\})) \notin \Phi_R^*$, we know that $\Xi(\{1\}, \{1\}) = \{\emptyset\}$. Therefore, the only control decision we can choose is $\{\}$ and we have $z = \hat{z} = h_{YZ}(y, \{\}) = (\{1\}, \{\})$. Then upon observing o, we reach new Y-states $y = \hat{y} = \{3\}$. This time we have $\Xi(\{3\}, \{3\}) = \{\{w\}, \{v, w\}\}$, since $(\{3\}, \{v\})$ is related to both $(\{3\}, \{v\})$ and $(\{3, 5\}, \{v, w\})$. Therefore, we choose $\{v, w\}$ at state $\{3\}$ in T^*. Then we move to $z = (\{3, 5\}, \{v, w\})$ and $\hat{z} = (\{3\}, \{v\})$.*

Now, from Z-state $(\{3, 5\}, \{v, w\})$, if event v occurs, T^* moves to Y-state $y = \{1\}$, which has already been visited. If event b occurs, T^* moves to Y-state $y = \{6\}$. However, T_R cannot track this move since b is not defined at $\hat{z} = (\{3\}, \{v\})$ in T_R. Therefore, we set $\hat{y} = \emptyset$, which means that the string is already outside of R. Therefore, for Y-state $\{6\}$, we just choose a locally maximal control decision in $C_{\mathcal{AES}(\boldsymbol{G},\boldsymbol{K})}(\{6\})$, i.e., $\{w\}$, and move to $z = (\{5, 6\}, \{w\})$ and $\hat{z} = \emptyset$. Finally, by observing b again, T^* moves back to Y-state $\{6\}$ that has been visited. This completes the depth-first search and returns the deterministic BTS T^* shown in Fig. 4.5(a), which includes a supervisor S_{T^*} such that $R \subseteq \mathcal{L}(S_{T^*}/\boldsymbol{G}) \subseteq K$, where $\mathcal{L}(S_{T^*}/\boldsymbol{G})$ is shown in Fig. 4.5(b). (We will prove later that this supervisor is indeed maximal.)

Remark 4.5.1. One can verify that the language shown in Fig. 4.5(c) is a maximal controllable and observable sub-language of K. In fact, this solution is obtained by using the strategies proposed in [4], i.e., we pick a locally maximal decision in $C_{\mathcal{AES}(\mathbf{G}),\mathbf{K}}(y)$ for each Y-state y and disregard the lower bound requirement. However,

Algorithm 7: MAX-RANGE

```
input  : $\mathbf{R}$ and $\mathcal{AES}(\mathbf{G},\mathbf{K})$.
output: $T^*$.
1    $T_R \leftarrow$INF-SYNT$(\mathbf{R}, \mathcal{AES}(\mathbf{G},\mathbf{K}))$ ;
2    $\Phi^*_R \leftarrow \Phi^*(T_R, \mathcal{AES}(\mathbf{G},\mathbf{K}))$ ;
3    $Q_Y^{T^*} \leftarrow \{y_0\}, Q_Z^{T^*} \leftarrow \emptyset$;
4    DoDFS$(y_0, y_0, T^*)$;
5    return $T^*$;
6  procedure DoDFS$(y, \hat{y}, T^*)$;
7    if $y|_{\boldsymbol{R}} \neq \emptyset$ then
8        Find a locally maximal element $Act$ in $\Xi(y,\hat{y})$, i.e.,
          $\forall \gamma \in \Xi(y,\hat{y}) : Act \not\subset \gamma$;
     else
9        Find a locally maximal element $Act$ in $C_{\mathcal{AES}(\mathbf{G},\mathbf{K})}(y)$, i.e.,
          $\forall \gamma \in C_{\mathcal{AES}(\mathbf{G},\mathbf{K})} : Act \not\subset \gamma$;
10   $z \leftarrow h_{YZ}(y, Act)$;
11   if $\hat{y} \neq \emptyset$ then
12       $\hat{z} \leftarrow h_{YZ}(\hat{y}, c_{T_R}(\hat{y}))$;
     else
13       $\hat{z} \leftarrow \emptyset$;
14   Add transition $y \xrightarrow{Act} z$ to $h_{YZ}^{T^*}$;
15   if $z \notin Q_Z^{T^*}$ then
16       $Q_Z^{T^*} \leftarrow Q_Z^{T^*} \cup \{z\}$;
17       for $\sigma \in \Sigma_o : h_{ZY}(z,\sigma)!$ do
18           $y' \leftarrow h_{ZY}(z,\sigma)$;
19           if $\hat{z} \neq \emptyset$ and $h_{ZY}(\hat{z},\sigma)!$ then
20               $\hat{y}' \leftarrow h_{ZY}(\hat{z},\sigma)$;
             else
21               $\hat{y}' \leftarrow \emptyset$;
22           Add transition $z \xrightarrow{\sigma} y'$ to $h_{ZY}^{T^*}$;
23           if $y' \notin Q_Y^{T^*}$ then
24               $Q_Y^{T^*} \leftarrow Q_Y^{T^*} \cup \{y'\}$;
25               DoDFS$(y', \hat{y}', T^*)$;
```

this solution does not fully contain R although it is maximal.

Note that, given arbitrary Y-states y and $\hat{y}$, set $\Xi(y,\hat{y})$ may be empty. For example, in Fig. 4.4, if we take $y = \{3,4\}$ and $\hat{y} = \{3\}$, the we know that $\Xi(y,\hat{y}\} = \emptyset$, since $c_{T_R}(\hat{y}) = \{v\}$ but no control decision defined at y in the AES contains $\{v\}$. If such a scenario occurs, then Algorithm MAX-RANGE may get stuck before it correctly returns T^*. However, the following result reveals that $\Xi(y,\hat{y})$ is always non-empty for any Y-states y and $\hat{y}$ encountered in Algorithm MAX-RANGE, i.e., the control decision *Act* in line 8 of Algorithm MAX-RANGE is always well-defined.

Proposition 4.5.1. *For any Y-state y reached in procedure DoDFS, if $\hat{y} \neq \emptyset$, then $\Xi(y,\hat{y}) \neq \emptyset$. Moreover, $y|_{\boldsymbol{R}} = \hat{y}|_{\boldsymbol{R}}$.*

Proof. We prove by induction on the length of the sequence that reaches y in procedure DoDFS.

Induction Basis: The induction basis holds, since for the initial state, we have that $y_0|_{\mathbf{R}} = y_0$, i.e., $c_{T_R}(y_0|_{\mathbf{R}}) \in \Xi(y_0, y_0)$.

Induction Hypothesis: We assume that, for any Y-state reached by sequence in the form of

$$y_0 \xrightarrow{\gamma_0} z_1 \xrightarrow{\sigma_1} y_1 \dots \xrightarrow{\gamma_{n-1}} z_n \xrightarrow{\sigma_n} y_n$$

in procedure DoDFS, if $\hat{y}_n \neq \emptyset$, we have $\Xi(y_n,\hat{y}_n) \neq \emptyset$ and $y_n|_{\mathbf{R}} = \hat{y}_n|_{\mathbf{R}}$.

Induction Step: To proceed, we show that, for any Y-state reached by sequence in the form of

$$y_0 \xrightarrow{\gamma_0} z_1 \xrightarrow{\sigma_1} y_1 \dots \xrightarrow{\gamma_{n-1}} z_n \xrightarrow{\sigma_n} y_n \xrightarrow{\gamma_n} z_{n+1} \xrightarrow{\sigma_{n+1}} y_{n+1}$$

in procedure DoDFS, if $\hat{y}_{n+1} \neq \emptyset$, we have that $\Xi(y_{n+1},\hat{y}_{n+1}) \neq \emptyset$ and $y_{n+1}|_{\mathbf{R}} = \hat{y}_{n+1}|_{\mathbf{R}}$, where $\hat{y}_{n+1}$ is the state reached by the following sequence in T_R

$$y_0 \xrightarrow{c_{T_R}(y_0)} \hat{z}_1 \xrightarrow{\sigma_1} \hat{y}_1 \xrightarrow{c_{T_R}(\hat{y}_1)} \dots \xrightarrow{c_{T_R}(\hat{y}_{n-1})} \hat{z}_n \xrightarrow{\sigma_n} \hat{y}_n \xrightarrow{c_{T_R}(\hat{y}_n)} \hat{z}_{n+1} \xrightarrow{\sigma_{n+1}} \hat{y}_{n+1}$$

First, we show that $y_{n+1}|_{\mathbf{R}} = \hat{y}_{n+1}|_{\mathbf{R}}$. To see this, we write

$$\begin{aligned}
& y_{n+1}|_{\mathbf{R}} \\
=& \{x \in X : \exists x' \in y_n, \exists w\sigma_{n+1} \in \mathcal{L}(\mathbf{G}) \text{ s.t. } w \in (\gamma_n \cap \Sigma_{uo})^* \text{ and } \delta(x', w\sigma_{n+1}) = x\}|_{\mathbf{R}} \\
=& \{x \in X_R : \exists x' \in y_n|_{\mathbf{R}}, \exists w\sigma_{n+1} \in \mathcal{L}(\mathbf{R}) \text{ s.t. } w \in (\gamma_n \cap \Sigma_{uo})^* \text{ and } \delta(x', w\sigma_{n+1}) = x\}|_{\mathbf{R}} \\
=& \{x \in X_R : \exists x' \in \hat{y}_n|_{\mathbf{R}}, \exists w\sigma_{n+1} \in \mathcal{L}(\mathbf{R}) \text{ s.t. } w \in (c_{T_R}(\hat{y}_n) \cap \Sigma_{uo})^* \text{ and } \delta(x', w\sigma_{n+1}) = x\}|_{\mathbf{R}} \\
=& \{x \in X : \exists x' \in \hat{y}_n, \exists w\sigma_{n+1} \in \mathcal{L}(\mathbf{G}) \text{ s.t. } w \in (c_{T_R}(\hat{y}_n) \cap \Sigma_{uo})^* \text{ and } \delta(x', w\sigma_{n+1}) = x\}|_{\mathbf{R}} \\
=& \hat{y}_{n+1}|_{\mathbf{R}}
\end{aligned}$$

The second and the fourth equalities follow from the assumption that $\mathbf{R} \sqsubset \mathbf{G}$, since any string that leaves the state space of $\mathbf{R}$ must lead to a state in $X \setminus X_R$. The third equality follows from the induction hypothesis that $y_n|_{\mathbf{R}} = \hat{y}_n|_{\mathbf{R}}$ and the fact that $\Gamma_R(\hat{y}_n|_{\mathbf{R}}) = c_{T_R}(\hat{y}_n) \subseteq \gamma_n$.

Next, we show that $\Xi(y_{n+1}, \hat{y}_{n+1}) \neq \emptyset$. According to line 8 in Algorithm MAX-RANGE, we know that γ_n is chosen such that $\gamma_n \in \Xi(y_n, \hat{y}_n)$. Note that $\Xi(y_n, \hat{y}_n)$ is non-empty by the induction hypothesis. Therefore, we know that $c_{T_R}(\hat{y}_n) \subseteq \gamma_n$ and

$$(h_{YZ}(\hat{y}_n, c_{T_R}(\hat{y}_n)), h_{YZ}(y_n, \gamma_n)) \in \Phi_R^*$$

This implies that $(\hat{y}_{n+1}, y_{n+1}) \in \Phi_R^*$. Therefore, we know that for any sequence

$$\hat{y}_{n+1} \xrightarrow{c_{T_R}(\hat{y}_{n+1})} \hat{z}_{n+2} \xrightarrow{\sigma_{n+2}} \ldots \xrightarrow{c_{T_R}(\hat{y}_{n+k-1})} \hat{z}_{n+k}$$

in T_R, there exists a sequence

$$y_{n+1} \xrightarrow{\gamma_{n+1}} z_{n+2} \xrightarrow{\sigma_{n+2}} \ldots \xrightarrow{\gamma_{n+k-1}} z_{n+k}$$

in the AES, such that $c_{T_R}(\hat{y}_{n+i}) \subseteq \gamma_{n+i}, \forall i \geq 1$. Hence, $\gamma_{n+1} \in C_{\mathcal{AES}(\mathbf{G},\mathbf{K})}(y_{n+1})$ and

$(\hat{z}_{n+2}, z_{n+2}) \in \Phi_R^*$, i.e.,

$$(h_{YZ}(\hat{y}_{n+1}, c_{T_R}(\hat{y}_{n+1})), h_{YZ}(y_{n+1}, \gamma_{n+1})) \in \Phi_R^* \tag{4.26}$$

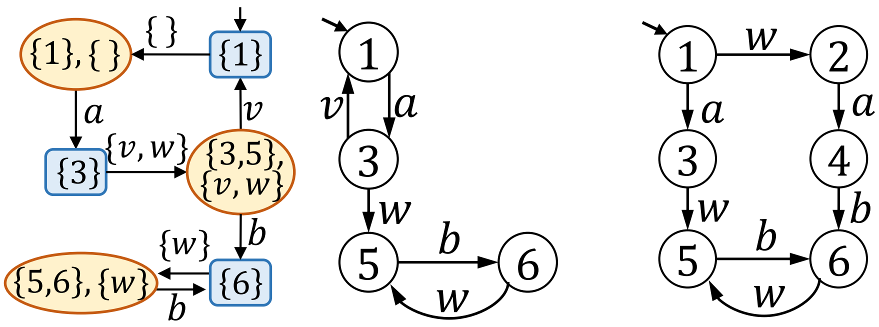
Therefore, we know that $\gamma_{n+1} \in \Xi(y_{n+1}, \hat{y}_{n+1})$, i.e., $\Xi(y_{n+1}, \hat{y}_{n+1})$ is also non-empty. This completes the induction step. □

Remark 4.5.2. Let us discuss the complexity of Algorithm MAX-RANGE. First, we need to construct the AES, which takes $O(|X||\Sigma|2^{|X|+|\Sigma|})$. Then Algorithm INF-SYNT takes $O(|X||\Sigma|2^{|X|})$ to construct T_R, since there are at most $2^{|X|}$ Y-states and the same number of Z-states in T_R; for each Y-state it takes $O(|X||\Sigma|)$ to determine its control decision and for each Z-state it takes $O(|\Sigma|)$ to consider all possible observations. Computing the maximal CSR Φ_R^* takes $O(2^{2|X|+2|\Sigma|})$. For procedure DoDFS in Algorithm MAX-RANGE, it takes $O(2^{|\Sigma|})$ to determine control decision *Act* for each Y-state and it takes $O(|\Sigma|)$ to consider all observations for each Z-state. In the worst case, there are still $2^{|X|}$ Y-states and the same number of Z-states in T^*, which implies that procedure DoDFS takes $O(2^{|X|+|\Sigma|})$ to construct T^*. Therefore, the overall complexity of Algorithm MAX-RANGE is $O(2^{2|X|+2|\Sigma|})$, which is exponential is the size of **G**. As we mentioned earlier, it is well-known that the supervisor synthesis problem under partial observation is NP-hard even without the lower bound requirement [99]. Therefore, it is highly unlikely that there exists a polynomial-time algorithm for MPRCP.

4.5.2 Correctness of the Algorithm

In this section, we establish the correctness of Algorithm MAX-RANGE, i.e., it effectively solves MPRCP.

Hereafter, we still denote by T^* the BTS returned by Algorithm MAX-RANGE and denote by S_{T^*} the supervisor induced by T^*. First, we show that S_{T^*} is s a safe



(a) T^* (b) $\mathcal{L}(S_{T^*}/\mathbf{G})$ (c) Another maximal solution

Figure 4.5: Figures in Example 4.5.1.

supervisor.

Lemma 4.5.1. *$\mathcal{L}(S_{T^*}/\boldsymbol{G}) \subseteq K$, i.e., S_{T^*} is safe.*

Proof. This follows directly from Theorem II.1. Since for each Y-state y encountered, $c_{T^*}(y)$ is chosen from $\Xi(y, \hat{y})$, which is a subset of $C_{\mathcal{AES}(\mathbf{G},\mathbf{K})}(y)$. Therefore, S_{T^*} is an AES-included supervisor, which means that it is safe. □

Next, we show that language R is contained in $\mathcal{L}(S_{T^*}/\mathbf{G})$.

Lemma 4.5.2. *$R \subseteq \mathcal{L}(S_{T^*}/\boldsymbol{G})$.*

Proof. We use Proposition 4.3.3 to show that S_{T^*} contains the lower bound R. Let us consider an arbitrary observable string $s \in P(\mathcal{L}(S_{T^*}/\mathbf{G}))$ such that $IS^Y_{S_{T^*}}(s)|_{\mathbf{R}} \neq \emptyset$. For simplicity, we denote $y = IS^Y_{S_{T^*}}(s)$. Since $y|_{\mathbf{R}} \neq \emptyset$, when y is reached for the first time in procedure DoDFS of Algorithm MAX-RANGE, i.e., when state y is added, it is reached by a sequence $y_0 \xrightarrow{\gamma_1\sigma_1\ldots\gamma_n\sigma_n} y$ in T^*, where $\sigma_1 \ldots \sigma_n \in P(R)$. Since $\sigma_1 \ldots \sigma_n \in P(R)$, we know that there exists a corresponding sequence $y_0 \xrightarrow{\gamma'_1\sigma_1\ldots\gamma'_n\sigma_n} \hat{y}$ in T_R that tracks the above sequence leading to y in T^*, i.e., $\hat{y}$ is the Y-state that tracks y in the depth-first search. Note that $\sigma_1 \ldots \sigma_n$ need not be equal to s since there may exist multiple sequences that lead to y and the depth-first search just randomly picks one of them. Therefore, $\hat{y}$ may depend on the specific implementation of the depth-first search.

By Algorithm MAX-RANGE, we know that $c_{T^*}(y)$ is chosen such that $c_{T_R}(\hat{y}) \subseteq c_{T^*}(y)$. By Algorithm INF-SYNT, we know that $c_{T_R}(\hat{y})$ is chosen such that $\Gamma_R(\hat{y}|_{\mathbf{R}}) \cup \Sigma_{uc} = c_{T_R}(\hat{y})$. By Proposition 4.5.1, we know that $y|_{\mathbf{R}} = \hat{y}|_{\mathbf{R}}$. Moreover, S_{T^*} is an IS-based supervisor, which implies that $S_{T^*}(s) = c_{T^*}(y)$. Overall, we know that

$$\Gamma_R(IS^Y_{S_{T^*}}(s)|_{\mathbf{R}}) = \Gamma_R(\hat{y}|_{\mathbf{R}}) \subseteq c_{T_R}(\hat{y}) \subseteq c_{T_R}(y) = S_{T^*}(s).$$

Recall that s is an arbitrary string in $P(\mathcal{L}(S_{T^*}/\mathbf{G}))$. Therefore, by Proposition 4.3.3, we know that $R \subseteq \mathcal{L}(S_{T^*}/\mathbf{G})$. $\square$

Finally, we show that S_{T^*} is maximal.

Lemma 4.5.3. *S_{T^*} is a maximally-permissive supervisor, i.e., for any safe supervisor S', $\mathcal{L}(S_{T^*}/\boldsymbol{G}) \not\subset \mathcal{L}(S'/\boldsymbol{G})$.*

Proof. By contradiction. Assume that S_{T^*} is not maximal. This implies that there exists another safe supervisor S' such that $\mathcal{L}(S_{T^*}/\mathbf{G}) \subset \mathcal{L}(S'/\mathbf{G})$. This implies that

1. $\forall s \in \mathcal{L}(S_{T^*}/\mathbf{G}) : S_{T^*}(P(s)) \subseteq S'(P(s))$; and
2. $\exists s \in \mathcal{L}(S_{T^*}/\mathbf{G}) : S_{T^*}(P(s)) \subset S'(P(s))$.

Let us consider an observable string $t \in P(\mathcal{L}(S_{T^*}/\mathbf{G}))$ such that $S_{T^*}(t) \subset S'(t)$ and $\forall t' \in \overline{\{t\}} \setminus \{t\} : S_{T^*}(t') = S'(t')$. Then we have that $IS^Y_{S_{T^*}}(t) = IS^Y_{S'}(t)$; we call this Y-state y.

We claim that, for the above Y-state y and control decision $S'(t)$, we have

$$(h_{YZ}(y, c_{T^*}(y), h_{YZ}(y, S'(t))) \in \Phi^*(T^*, \mathcal{AES}(\mathbf{G}, \mathbf{K})) \tag{4.27}$$

Too see this, let us consider an arbitrary sequence

$$y \xrightarrow{c_{T^*}(y)} z_1 \xrightarrow{\sigma_1} y_1 \xrightarrow{c_{T^*}(y_1)} \dots z_n \xrightarrow{\sigma_n} y_n \xrightarrow{c_{T^*}(y_n)} z_{n+1} \tag{4.28}$$

in T^*. Since, $\mathcal{L}(S_{T^*}/\mathbf{G}) \subseteq \mathcal{L}(S'/\mathbf{G})$, S' induces the following sequence

$$y \xrightarrow{S'(t)} z'_1 \xrightarrow{\sigma_1} y'_1 \xrightarrow{S'(t\sigma_1)} \dots z'_n \xrightarrow{\sigma_n} y'_n \xrightarrow{S'(t\sigma_1 \dots \sigma_n)} z'_{n+1} \tag{4.29}$$

Since S' is a safe supervisor, by Theorem II.1, we know that S' is an AES-included supervisor. This implies that the above sequence exists in the AES. Therefore, by Proposition 4.4.2, we know that Equation (4.27) holds.

Next, we consider two cases for this Y-state y to show the contradiction.

Case 1: $y|_{\mathbf{R}} = \emptyset$.

Since S' is a safe supervisor, by Theorem II.1, we know that $S'(t) \in C_{\mathcal{AES}(\mathbf{G},\mathbf{K})}(y)$. Moreover, $c_{T^*}(y)$ is chosen as a maximal element in $C_{\mathcal{AES}(\mathbf{G},\mathbf{K})}(y)$. Therefore, we obtain a contradiction immediately since $c_{T^*}(y) \subset S'(t)$ is not possible.

Case 2: $y|_{\mathbf{R}} \neq \emptyset$.

Suppose that Y-state y is reached, for the first time, by the following sequence

$$y_0 \xrightarrow{\gamma_1} z_1 \xrightarrow{\sigma_1} y_1 \xrightarrow{\gamma_2} \dots \xrightarrow{\gamma_n} z_n \xrightarrow{\sigma_n} y \tag{4.30}$$

in procedure DoDFS in Algorithm MAX-RANGE. Since $y|_{\mathbf{R}} \neq \emptyset$, we know that $\sigma_1 \dots \sigma_n \in P(R)$, i.e., $\hat{y} \neq \emptyset$. Let $\hat{y}$ be the corresponding Y-state reached by the sequence in T_R that tracks the above sequence, i.e.,

$$y_0 \xrightarrow{c_{T_R}(y_0)} \hat{z}_1 \xrightarrow{\sigma_1} \hat{y}_1 \xrightarrow{c_{T_R}(\hat{y}_1)} \dots \xrightarrow{c_{T_R}(\hat{y}_n)} \hat{z}_n \xrightarrow{\sigma_n} \hat{y} \tag{4.31}$$

Since $\mathcal{L}(S_{T_R}/\mathbf{G}) \subseteq \mathcal{L}(S_{T^*}/\mathbf{G})$, by Proposition 4.4.3, we know that $T_R \preceq T^*$. Therefore, by the definition of the CSR, Equations (4.30) and (4.31) imply that $(y, \hat{y}) \in \Phi^*(T_R, T^*)$. This further implies that

$$(h_{YZ}(\hat{y}, c_{T_R}(\hat{y})), h_{YZ}(y, c_{T^*}(y))) \in \Phi^*(T_R, T^*) \tag{4.32}$$

Overall, by Eqs. (4.27) and (4.32) and by Proposition 4.4.4, we get

$$(h_{YZ}(\hat{y}, c_{T_R}(\hat{y})), h_{YZ}(y, S'(t))) \in \Phi^*(T_R, \mathcal{AES}(\mathbf{G}, \mathbf{K}))$$

Note that we also have that $c_{T_R}(\hat{y}) \subseteq c_{T^*}(y) \subset S'(t)$ and $S'(t) \in C_{\mathcal{AES}(\mathbf{G},\mathbf{K})}(y)$. Therefore, we know that $S'(t) \in \Xi(y, \hat{y})$. However, $c_{T^*}(y) \subset S'(t)$ is not possible, since $c_{T^*}(y)$ is chosen as a maximal control decision in $\Xi(y, \hat{y})$. This is a contradiction. $\square$

Finally, combining Lemmas 4.5.1, 4.5.2 and 4.5.3 together, we have the following theorem.

Theorem IV.1. *S_{T^*} is a maximally-permissive supervisor such that $R \subseteq \mathcal{L}(S_{T^*}/\boldsymbol{G}) \subseteq K$, i.e., Algorithm MAX-RANGE effectively solves MPRCP.*

Since the resulting supervisor S_{T^*} is realized by BTS T^*, we also have the following corollary.

Corollary 4.5.1. *S_{T^*} is an IS-based solution, which implies that the closed-loop language $\mathcal{L}(S_{T^*}/\boldsymbol{G})$ is regular.*

Remark 4.5.3. We have shown that Algorithm MAX-RANGE solves MPRCP. In fact, it also solves the *maximal-permissiveness verification problem*. Specifically, suppose that there exists a given supervisor $S : P(\mathcal{L}(\mathbf{G})) \to \Gamma$ and we want to *verify* whether it is maximal or not. In this case, we can just set $R = \mathcal{L}(S/\mathbf{G})$ as the lower bound requirement and apply Algorithm MAX-RANGE to find a maximal safe supervisor S^* that contains R. If $\mathcal{L}(S/\mathbf{G}) = \mathcal{L}(S^*/\mathbf{G})$, then we know that the given supervisor S is already maximally permissive, since we cannot improve it any further. Otherwise, if $\mathcal{L}(S/\mathbf{G}) \subset \mathcal{L}(S^*/\mathbf{G})$, then we know that S is not maximal. To the best of our knowledge, the maximality verification problem was open in the literature; it is now solved as a special case of the synthesis problem.

4.6 Conclusion

In this chapter, we have solved a generalized supervisor synthesis problem, called the range control problem, for partially-observed DES. We considered both a standard upper bound specification that describes the legal behavior and a lower bound specification that describes the desired behavior. We provided new information-state-based constructive approaches for computing both infimal and maximal supervisors satisfying these requirements. The proposed approach combines the three notions of AES, strict sub-automaton, and CSR, in a novel manner; each of them plays a different role in the synthesis problem. This results in a "meaningful" maximally-permissive safe supervisor that contains a given behavior. An interesting future direction is to extend the results in this chapter to the non-prefix-closed case.

Finally, all results in Chapters II to IV have been implemented in software DPO-SYNT [122]; it can be downloaded from `https://github.com/xiang-yin/DPO-SYNT`.

4.7 Appendix

This appendix provides a state space refinement algorithm, which generalizes the procedure in [18] from two automata to three automata.

Let $\mathbf{R} = (X_R, \Sigma, \delta_R, x_{0,R})$, $\mathbf{K} = (X_K, \Sigma, \delta_K, x_{0,K})$ and $\mathbf{G} = (X_G, \Sigma, \delta_G, x_{0,G})$ be three automata such that $\mathcal{L}(\mathbf{R}) \subseteq \mathcal{L}(\mathbf{K}) \subseteq \mathcal{L}(\mathbf{G})$. We construct three new automata $\mathbf{R}', \mathbf{K}'$ and $\mathbf{G}'$ by the following algorithm.

Algorithm PRE-PROCESS

Input: $\mathbf{R}, \mathbf{K}$ and $\mathbf{G}$ such that $\mathcal{L}(\mathbf{R}) \subseteq \mathcal{L}(\mathbf{K}) \subseteq \mathcal{L}(\mathbf{G})$.

Output: $\mathbf{R}', \mathbf{K}'$ and $\mathbf{G}'$.

Step 1

1-1 Obtain $\mathbf{A} = (X_A, \Sigma, \delta_A, x_{0,A})$ from $\mathbf{R}$ by adding a new state called $Dead_A$ to

X_R and completing the transition function δ_A by: for any $x \in X_A$, $\sigma \in \Sigma$, we have $\delta_R(x, \sigma)\neg! \Rightarrow \delta_A(x, \sigma) = Dead_A$. In particular, we have $X_A = X_R \cup \{Dead_A\}, x_{0,A} = x_{0,R}$ and $\mathcal{L}(\mathbf{A}) = \Sigma^*$.

1-2 Obtain $\mathbf{B} = (X_B, \Sigma, \delta_B, x_{0,B})$ from $\mathbf{K}$ by adding a new state called $Dead_B$ to X_K and completing the transition function δ_B by: for any $x \in X_B$, $\sigma \in \Sigma$, we have $\delta_K(x, \sigma)\neg! \Rightarrow \delta_B(x, \sigma) = Dead_B$. In particular, we have $X_B = X_K \cup \{Dead_B\}, x_{0,B} = x_{0,K}$ and $\mathcal{L}(\mathbf{B}) = \Sigma^*$.

Step 2

2-1 Form the product automaton $\mathbf{ABG} := \mathbf{A} \times \mathbf{B} \times \mathbf{G} = (X_{ABG}, \Sigma, \delta_{ABG}, x_{0,ABG})$, where "$\times$" denotes the usual product composition operation of automata; see, e.g., [12] (p. 78).

Step 3

3-1 Obtain $\mathbf{G}'$ by taking $\mathbf{ABG}$.

3-2 Obtain $\mathbf{K}'$ by taking the largest sub-automaton of $\mathbf{ABG}$ where the second component is not equal to $Dead_B$; that is, delete all state of $\mathbf{ABG}$ where the second state component is $Dead_B$.

3-3 Obtain $\mathbf{R}'$ by taking the largest sub-automaton of $\mathbf{ABG}$ where the first component is not equal to $Dead_A$; that is, delete all state of $\mathbf{ABG}$ where the first state component is $Dead_A$.

After Step 2, we have that $\mathcal{L}(\mathbf{ABG}) = \mathcal{L}(\mathbf{G}') = \mathcal{L}(\mathbf{G})$ since $\mathcal{L}(\mathbf{A}) = \mathcal{L}(\mathbf{B}) = \Sigma^*$. By construction, $\mathcal{L}(\mathbf{K}') = \mathcal{L}(\mathbf{K})$ since we only delete states that have $Dead_B$ as their second component. Similarly, we know that $\mathcal{L}(\mathbf{R}') = \mathcal{L}(\mathbf{R})$. Moreover, it is clear from Step 3 that $\mathbf{K}' \sqsubset \mathbf{G}'$. since for any string $s \in \mathcal{L}(\mathbf{G}) \setminus \mathcal{L}(\mathbf{G})$, it leads to a state whose second component is $Dead_B$, but $\mathbf{K}'$ does not contain such a state. Similarly, we know that $\mathbf{R}' \sqsubset \mathbf{K}'$. Overall, have

1. $\mathcal{L}(\mathbf{G}') = \mathcal{L}(\mathbf{G})$, $\mathcal{L}(\mathbf{K}') = \mathcal{L}(\mathbf{K})$ and $\mathcal{L}(\mathbf{R}') = \mathcal{L}(\mathbf{R})$;

2. $\mathbf{R}' \sqsubset \mathbf{K}' \sqsubset \mathbf{G}'$.

In the worst case, $\mathbf{G}'$ contains $|X| \times (|X_K| + 1) \times (|X_R| + 1)$ states. Therefore, only polynomial state space refinement is needed to fulfill the assumption that $\mathbf{R} \sqsubset \mathbf{K} \sqsubset \mathbf{G}$.

CHAPTER V

A Uniform Approach for Centralized Sensor Activation

5.1 Introduction

In this chapter, we consider the problem of dynamic sensor activation in centralized and partially-observed DES. The objective in this problem is to synthesize a sensor activation policy that dynamically turns sensors on/off online in order to achieve a given objective, e.g., to control the system or to diagnose faults. This problem is important since in many applications turning more sensors on implies that more energy or bandwidth is consumed. Therefore, it is of interest to synthesize a sensor activation policy that is optimal with respect to some criterion, subject to the constraints of the problem. For instance, in control problems these constraints involve the property of observability of DES, while in diagnosis problems they involve the property of diagnosability of DES.

We use the MPO approach [16,22] to investigate the sensor activation problem for centralized partially-observed DES. However, instead of investigating the enforcement of a particular property, e.g., observability, diagnosability, or opacity, as was done in previous works, we study a general class of properties called Information-State-based (IS-based) properties, that captures all properties previously considered, and more.

This is similar to the supervisory control problem studied in Chapters II and III. However, the IS-based property is defined in a different manner due to the difference between the supervisory control problem and the sensor activation problem. We first formulate the problem of dynamic sensor activation for any property that can be expressed as an IS-based property. We show that this problem formulation is more general than both the state disambiguation problem and the opacity problem that have been studied previously in the literature. To solve this problem, we define a generalized version of the most permissive observer. This generalized MPO embeds all valid solutions to the enforcement of an IS-based property in its finite structure. Based on the MPO, we present an algorithm for the synthesis of optimal sensor activation policies under a logical performance objective.

Compared with prior works where the MPO was employed [15, 16, 22], our contributions are twofold. First, we define the MPO directly from the new notion of bipartite dynamic observer without using the recursive definition used in [22]. Second, the MPO defined in this chapter is more general since we consider a general class of properties and we show that the most permissive observer for K-diagnosability studied by [16, 22] and the most permissive dynamic mask for opacity studied by [15] are essentially special cases of the generalized MPO. Moreover, most permissive observers for observability, detectability, and predictability, which have not been studied so far in the literature, can all be defined as special cases of the generalized MPO. Therefore, the dynamic sensor activation problems for these properties can all be solved by our approach. Moreover, the problem of optimal sensor activation for predictability, which to the best of our knowledge has not been considered in the literature, can also be solved by our approach. Similarly, our approach can be employed to solve sensor activation problems for the enforcement of a wide class of *user-defined* properties that can be expressed as IS-based properties. Compared with other solution approaches for dynamic sensor activation problems, our methodology has the following features.

First, the optimal solution that we obtain is language-based. Recall that the solutions obtained by [101, 104] are optimal only w.r.t. finite (restricted) solution spaces, based on the state space of the system model. Moreover, the generalized MPO that we define embeds *all* solutions in its *single* finite structure. Therefore, it can serve as a basis for optimization w.r.t. a numerical cost criterion, which cannot be done by the online approaches described in [85, 103].

The remainder of this chapter is organized as follows. In Section 5.2, we introduce some basic terminologies. In Section 5.3, we formulate the optimal sensor activation problem for IS-based properties that we solve in this chapter. In Section 5.4, the generalized MPO is defined. A synthesis algorithm for solving the problem formulated in Section 5.2 based on the MPO is provided in Section 5.5. In Section 5.6, we show that: (i) the MPO defined in Section 5.3 generalizes the previous versions of this notion in the literature; and (ii) new problems, e.g., dynamic sensor activation for the purpose of fault prediction, can be solved by our approach. Finally, we conclude the chapter in Section 5.7.

5.2 Preliminary

In this section, we introduce some basic terminologies and notations in the dynamic sensor activation problem.

5.2.1 Information Mapping

The DES of interest is still modeled as a deterministic finite-state automaton $\mathbf{G} = (X, \Sigma, \delta, x_0)$. The set of marked states is omitted, since we are only interested in the generated language in the sensor activation problem. In dynamic sensor activation problems, the sensors are turned on/off dynamically based on the observation history. When the sensor corresponding to an event $\sigma \in \Sigma$ is turned "on", we say that the event is being *monitored*. While an event is monitored, any occurrence of it will be *observed* by the supervisor, diagnoser, predictor, or external observer, according to the problem under consideration (e.g., control, diagnosis, prediction, or opacification). At any point in the execution of the system, the set of events $\theta \in 2^{\Sigma}$ that we decide to monitor (by turning their sensors on), is called a *sensing decision*.

In the setting of dynamic observation, we assume that Σ is partitioned into three disjoint sets, $\Sigma = \Sigma_o \dot{\cup} \Sigma_s \dot{\cup} \Sigma_{uo}$, where:

1. Σ_o is the set of events whose occurrences are always observed, i.e., their sensors are always turned on and they are continuously monitored;

2. Σ_s is the set of events that we can choose to monitor or not (by turning their sensors on/off);

3. Σ_{uo} is the set of events that are always unobservable (i.e., there are no sensors for them).

We say that a sensing decision $\theta \in 2^{\Sigma}$ is *admissible* if $\Sigma_o \subseteq \theta \subseteq \Sigma_o \cup \Sigma_s$ and we let Θ denote the set of all admissible sensing decisions.

We consider a general dynamic observations setting, where the observability properties of events can be controlled by a *sensor activation policy* during the evolution of the system. A sensor activation policy is defined as a deterministic labeled automaton $\Omega = (R, L)$, where

$$R = (X_R, \Sigma, \delta_R, x_{0,R}) \tag{5.1}$$

is a deterministic (finite state or infinite state) automaton and $L : X_R \rightarrow \Theta$ is a labeling function that specifies the current set of "observable" events within $\Sigma_o \cup \Sigma_s$. Specifically, for any $s \in (\Sigma_o \cup \Sigma_s)^*$, $\Sigma_o \subseteq L(\delta_R(s)) \subseteq \Sigma_o \cup \Sigma_s$ denotes the set of events that are *monitored* after observing s. While an event is monitored, any occurrence of it will be observed by the observer. In other words, after string s, events not in $L(\delta_R(s))$ are currently "unobervable" (i.e., their sensors are turned off). To make Ω

implementable, the pair (R, L) needs to satisfy the constraint that

$$(\forall x, x' \in X_R)(\forall \sigma \in \Sigma : \delta_R(x, \sigma) = x')[x \neq x' \Rightarrow \sigma \in L(x)] \tag{5.2}$$

This condition says that the sensing decision can be updated (by updating the state of R) only when a monitored event occurs. In general, X_R could be an infinite set. However, we will show later that the optimal sensor activation policies of interest in this chapter can always be constructed with finite state spaces.

We say that the observations are *static* if the set of observable events is fixed a priori. We denote by Ω_{Σ_o} the corresponding sensor activation policy for the static observation with the set of observable events Σ_o. Specifically, $\Omega_{\Sigma_o} = (R, L)$ is given by: 1) $X_R = \{x_{0,R}\}$; 2) $\forall \sigma \in \Sigma_o : \delta_A(x_{0,R}, \sigma) = x_{0,R}$; and 3) $L(x_{0,R}) = \Sigma_o$.

Given a sensor activation policy $\Omega = (R, L)$, we define the corresponding information mapping $P_\Omega : \mathcal{L}(\mathbf{G}) \rightarrow (\Sigma_o \cup \Sigma_s)^*$ recursively as follows:

$$P_\Omega(\epsilon) = \epsilon, \quad P_\Omega(s\sigma) = \begin{cases} P_\Omega(s)\sigma & \text{if } \sigma \in L(\delta_R(s)) \\ P_\Omega(s) & \text{if } \sigma \notin L(\delta_R(s)) \end{cases}$$

That is, $P_\Omega(s)$ is the observation of string s under Ω. For any language $L \subseteq \Sigma^*$, we define $P_\Omega(L) = \{t \in \Sigma_o^* : \exists s \in L \text{ s.t. } P_\Omega(s) = t\}$.

Let $s \in \mathcal{L}(\mathbf{G})$. For the sake of simplicity, hereafter, we also denote by $\Omega(s)$ [1] the sensing decision after observing $P_\Omega(s)$, i.e., $\Omega(s) = L(\delta_R(s))$.

For any two sensor activation policies $\Omega = (R, L)$ and $\Omega' = (R', L')$, we write that $\Omega' \leq \Omega$ if

$$\forall s \in \mathcal{L}(\mathbf{G}) : \Omega'(s) \subseteq \Omega'(s) \tag{5.3}$$

[1]In fact, a sensor activation policy can also be represent by a mapping $\Omega : \mathcal{L}(\mathbf{G}) \rightarrow 2^{\Sigma_o \cup \Sigma_s}$ such that $\forall s, t \in \mathcal{L}(\mathbf{G}) : P_\Omega(s) = P_\Omega(t) \Rightarrow \Omega(s) = \Omega(t)$; see, e.g., [101, 104].

and write that $\Omega' < \Omega$ if

$$[\Omega' \leq \Omega] \wedge [\exists s \in \mathcal{L}(\mathbf{G}) : \Omega'(s) \subset \Omega'(s)] \tag{5.4}$$

5.2.2 The Observer

For any $i \in 2^X, \sigma \in \Sigma_o \cup \Sigma_s$ and $\theta \in 2^{\Sigma_o \cup \Sigma_s}$. Recall that

$$Next_\sigma(i) = \{x_1 \in X : \exists x_2 \in i \text{ s.t. } \delta(x_2, \sigma) = x_1\} \tag{5.5}$$

We also define

$$UOR_\theta(i) = \{x_1 \in Q : \exists x_2 \in i, \exists s \in (\Sigma \setminus \theta)^* \text{ s.t. } \delta(x_2, s) = x_1\} \tag{5.6}$$

That is, $Next_\sigma(i)$ is the set of states that can be reached from some state in i immediately after observing σ and $UOR_\theta(i)$ is the set of states that can be reached unobservably from some state in i under the set of monitored events θ. Note that operator UOR is slightly different from operator UR defined in Equation (2.4)

Let $\mathbf{G} = (X, \Sigma, \delta, x_0)$ be the system automaton and $\Omega = (R, L), R = (X_R, \Sigma, \delta_R, x_{0,R})$ be a sensor activation policy. The observer for $\mathbf{G}$ under Ω is

$$Obs_\Omega(\mathbf{G}) = (Q, \Sigma_o \cup \Sigma_s, f, q_0), \tag{5.7}$$

where $Q \subseteq 2^X \times X_R$ is the state space and for any state $q \in Q$, we write $q = (I(q), R(q))$ where $I(q) \in 2^X$ and $R(q) \in X_R$. The partial transition function of the observer is denoted by $f : Q \times (\Sigma_o \cup \Sigma_s) \to Q$ and is defined as follows. For any

$q = (i, x), q' = (i', x') \in Q$ and $\sigma \in \Sigma_o \cup \Sigma_s$, $f(q, \sigma) = q'$ iff

$$\begin{cases} x' = \delta_R(x, \sigma) \\ i' = UR_{L(x')}(Next_\sigma(i)) \end{cases} \tag{5.8}$$

Finally, the initial state of $Obs_\Omega(\mathbf{G})$ is $q_0 = (UOR_{L(x_{0,R})}(\{x_0\}), x_{0,R})$. For simplicity, we only consider the reachable part of $Obs_\Omega(\mathbf{G})$. By the above definition, we have that $\mathcal{L}(Obs_\Omega(\mathbf{G})) = P_\Omega(\mathcal{L}(\mathbf{G}))$.

We define the *state estimator function* (or simply "state estimator") under Ω, $\mathcal{E}_\Omega^{\mathbf{G}} : \mathcal{L}(\mathbf{G}) \to 2^X$, as follows upon the occurrence of $s \in \mathcal{L}(\mathbf{G})$:

$$\mathcal{E}_\Omega^{\mathbf{G}}(s) := \{x \in X : \exists t \in \mathcal{L}(\mathbf{G}) \text{ s.t. } P_\Omega(s) = P_\Omega(t) \wedge \delta(x_0, t) = x\}$$

By a simple induction (see, e.g., [22]), we can show that, for any $s \in \mathcal{L}(\mathbf{G})$, we have $I(f(P_\Omega(s))) = \mathcal{E}_\Omega^{\mathbf{G}}(s)$, i.e., the state components of the observer state reached upon $P_\Omega(s)$ is the state estimator value after s.

5.3 Problem Formulation

As was explained in the introduction, in a given problem domain (control, diagnosis, and so forth), the sensor activation policy must satisfy some problem-dependent *property* (observability, diagnosability, and so forth). For the sake of generality, we define a property φ as a function $\varphi : \Omega \to \{0, 1\}$ and for any sensor activation policy Ω, we write $\varphi(\Omega) = 1$ to mean that Ω satisfies property φ. The properties of interest are typically defined in a language-based manner rather than a state-based manner. Hereafter, similar to the supervisor synthesis problem, we also consider a special class of properties called *information-state-based (IS-based) properties*. These are properties whose verification can be performed over information states (i.e., sets of states) of the (possibly refined) system state space. We formalize this notion next.

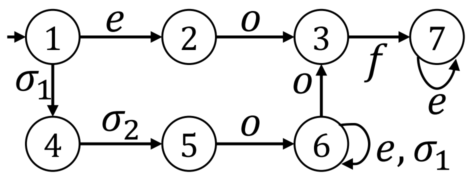


Figure 5.1: System $\mathbf{G}$ with $\Sigma_o = \{o\}$, $\Sigma_s = \{\sigma_1, \sigma_2\}$, and $\Sigma_{uo} = \{e, f\}$

We still define an *information state* to be a subset of states in X and denote by $I = 2^X$ the set of information states. Roughly speaking, an IS-based property is a property that only depends on the *current* knowledge of the system, as provided by the state estimator function $\mathcal{E}_\Omega^{\mathbf{G}}$ under a given sensor activation policy Ω. In particular, the property should not depend on information about the *future* behavior of the system. We will show later that most of the important properties in the DES literature can be formulated as IS-based properties, possibly after suitable state space refinements of the original model $\mathbf{G}$. First, we present the formal definition of the IS-based property.

Definition 5.3.1. *(IS-based Property). Let $\boldsymbol{G} = (X, \Sigma, \delta, x_0)$ be the system automaton and $\Omega : \mathcal{L}(\boldsymbol{G}) \to \Theta$ be a sensor activation policy. An IS-based property w.r.t. $\boldsymbol{G}$ is a function $\varphi : 2^X \to \{0, 1\}$. We say that Ω satisfies φ w.r.t. $\boldsymbol{G}$, denoted by $\Omega \models_{\boldsymbol{G}} \varphi$, if $\forall s \in \mathcal{L}(\boldsymbol{G}) : \varphi(\mathcal{E}_\Omega^{\boldsymbol{G}}(s)) = 1$.*

Example 5.3.1. *Consider the system $\boldsymbol{G}$ in Figure 5.1. Let $\varphi : 2^X \to \{0, 1\}$ be an IS-based property defined as follow:*

$$\forall i \in 2^X : [\varphi(i) = 1] \Leftrightarrow [\nexists x \in \{1, 4, 5, 6\} : \{3, x\} \subseteq i] \tag{5.9}$$

This IS-based property φ requires that we should never confuse state 3 with any state in $\{1, 4, 5, 6\}$.

Let us consider the information mapping Ω defined by $\forall s \in \mathcal{L}(\boldsymbol{G}) : \Omega(s) = \{o\}$. By taking $eo \in \mathcal{L}(\boldsymbol{G})$, we know that $\mathcal{E}_\Omega^{\boldsymbol{G}}(eo) = \{3, 6\}$. Therefore, $\Omega \not\models_{\boldsymbol{G}} \varphi$.

As was mentioned earlier, the objective of this chapter is to synthesize a sensor activation policy such that some given property provably holds. Since turning sensors on/off can be costly (for some given cost function on power, bandwidth, or switching for instance), we define the *Minimal Sensor Activation Problem for IS-Based Properties* as follows.

Problem 5. *(Minimal Sensor Activation Problem for IS-Based Properties). Let $\boldsymbol{G} = (X, \Sigma, \delta, x_0)$ be the system automaton and $\varphi : 2^X \to \{0,1\}$ be an IS-based property w.r.t. $\boldsymbol{G}$. Find a sensor activation policy Ω such that:*

(i) $\Omega \models_{\boldsymbol{G}} \varphi$;

(ii) $\nexists \Omega'$ *such that* $\Omega' \models_{\boldsymbol{G}} \varphi$ *and* $\Omega' < \Omega$.

In some contexts, we may be interested in the dual version of the Minimal Sensor Activation Problem, the Maximal Sensor Activation Problem for IS-Based Properties. *Its definition is analogous, with "<" replaced by ">" in condition (ii). As is well known for the main properties of interest in control or diagnosis of partially-observed DES, Problem 5 does not have a unique "globally optimal" solution, and many incomparable "locally optimal" solutions may exist in the logical setting under consideration. This explains the manner in which condition (ii) is stated. (See Remark 5.5.1.)*

Remark 5.3.1. In [102], the *state disambiguation problem* is defined. Formally, $T_{spec} \subseteq X \times X$ is the set of state pairs that need to be distinguished and the goal is to find a minimal Ω such that $(\forall s \in \mathcal{L}(\mathbf{G}))(\forall x_1, x_2 \in \mathcal{E}_{\Omega}^{\mathbf{G}}(s))[(x_1, x_2) \notin T_{spec}]$. Clearly, the state disambiguation problem is a special case of the minimal sensor activation problem for IS-based properties, since given T_{spec}, we can always define an IS-based property $\varphi_{spec} : 2^X \to \{0,1\}$ by: $\forall i \in 2^X : [\varphi_{spec}(i) = 0] \Leftrightarrow [\exists x_1, x_2 \in i : (x_1, x_2) \in T_{spec}]$. Therefore, the problem we consider here is more general than the state disambiguation problem.

Remark 5.3.2. In many cases, the system is not only monitored by its internal controller, but it may also be monitored by an *external* observer that is potentially malicious. Therefore, instead of disambiguating states, the objective is to *confuse* the external observer so that it may not infer a given secret about the system. In such a scenario, the "disablement" of sensors can be costly, since we need to spend some additional effort, e.g., adding a dynamic mask, to hide the occurrences of the corresponding events. In this regard, the optimal dynamic mask synthesis problem investigated in the literature (see, e.g., [15]) is essentially the maximal sensor activation problem defined above. This justifies our earlier assertion that the problem considered in this chapter is very general and covers many earlier works in different problem domains. We further elaborate on this issue in Section 5.6.

5.4 A General Most Permissive Observer

In this section, we first discuss the evolution of the available information during the execution of the system under dynamic observations. Then we define the notion of bipartite dynamic observer, which is similar to the bipartite transition system for the supervisor synthesis problem. Finally, we define the generalized most permissive observer that embeds all valid sensor activation policies in its structure.

5.4.1 Information State Dynamics

A sensor activation policy Ω works dynamically as follows. Initially, a sensing decision θ_0 is issued. Then, upon the occurrence of (monitored) event $\sigma_1 \in \theta_0$, a new decision θ_1 is made and so forth. We call such a sequence in the form of $\theta_0\sigma_1\theta_1\sigma_2\ldots$, where $\theta_i \in \Theta, \sigma_{i+1} \in \theta_i, \forall i \geq 0$, a *run*. For any $s \in \mathcal{L}(\mathbf{G})$, suppose that $s = \xi_0\sigma_1\xi_1\sigma_2\ldots\xi_{n-1}\sigma_n\xi_n$, where $\xi_i \in (\Sigma \setminus \Omega(\xi_0\sigma_1\ldots\xi_{i-1}\sigma_i))^*, \forall i \geq 0$ and $\sigma_i \in \Omega(\xi_0\sigma_1\ldots\sigma_{i-1}\xi_{i-1}), \forall i \geq 1$, i.e., $P_\Omega(s) = \sigma_1\ldots\sigma_n$. Intuitively, ξ_i is just an unobserved string and σ_i is a monitored event. Then the information available to the sensor activation module upon the occurrence of s is, in fact, the run

$$\mathcal{R}_\Omega(s) := \theta_0\sigma_1\theta_1 \dots \theta_{n-1}\sigma_n\theta_n \tag{5.10}$$

where $\theta_i = \Omega(\xi_0\sigma_1 \dots \xi_{i-1}\sigma_i\xi_i), \forall i \geq 0$.

To capture the alternating nature of sensing decisions and observations of monitored events, similar to the BTS, we define two analogous kinds of states, also termed Y-states and Z-states, respectively. A Y-state y is an information state from which a sensing decision is made and $Y \subseteq I$ denotes the set of Y-states. A Z-state z is an information state augmented with a sensing decision from which observations of monitored events occur. $Z \subseteq I \times \Theta$ denotes the set of Z-states and we write $z = (I(z), \Theta(z))$ for any $z \in Z$. Next, we define the transition function from Y-states to Z-states, $\hbar_{YZ} : Y \times \Theta \to Z$, and the transition function from Z-states to Y-states, $\hbar_{ZY} : Z \times \Sigma \to Y$. For any $y \in I, z \in I \times \Theta, \sigma \in \Sigma$ and $\theta \in \Theta$,

- $z = \hbar_{YZ}(y, \theta)$ if and only if
  $I(z) = \{x \in X : \exists x' \in y, \exists s \in (\Sigma \setminus \theta)^* \text{ s.t. } \delta(x', s) = x\}$ and $\Theta(z) = \theta$
- $y = \hbar_{ZY}(z, \sigma)$ if and only if
  $\sigma \in \Theta(z)$ and $y = \{x \in X : \exists x' \in I(z) \text{ s.t. } \delta(x', \sigma) = x\}$

For simplicity hereafter, we write $y \xrightarrow{\theta} z$ if $z = \hbar_{YZ}(y, \theta)$ and $z \xrightarrow{\sigma} y$ if $z = \hbar_{ZY}(z, \sigma)$. Intuitively, $y \xrightarrow{\theta} z$ simply represents the *unobserved reach* under sensing decision θ and it remembers the sensing decision that leads to it. On the other hand, $z \xrightarrow{\sigma} y$ represents the set of states the system can reach *immediately after* the occurrence of event σ. We require that $\sigma \in \Theta(z)$, since σ must be monitored.

Now, let $s \in \mathcal{L}(\mathbf{G})$ be a string and $\mathcal{R}_\Omega(s) = \theta_0\sigma_1\theta_1 \dots \theta_{n-1}\sigma_n\theta_n$ be the corresponding run defined in Equation (5.10). Let $y_0 = \{x_0\}$ be the initial Y-state. Then occurrence of the run $\theta_0\sigma_1\theta_1 \dots \theta_{n-1}\sigma_n\theta_n$ will reach an alternating sequence of $Y-$

and Z-states

$$y_0 \xrightarrow{\theta_0} z_0 \xrightarrow{\sigma_1} y_1 \xrightarrow{\theta_1} \dots \xrightarrow{\theta_{n-1}} z_{n-1} \xrightarrow{\sigma_n} y_n \xrightarrow{\theta_n} z_n \tag{5.11}$$

We denote by $\mathcal{I}_\Omega^Y(s)$ and $\mathcal{I}_\Omega^Z(s)$, the last Y-state and Z-state in $y_0 z_0 y_1 z_2 \dots z_{n-1} y_n z_n$, respectively, i.e., $\mathcal{I}_\Omega^Y(s) = y_n$ and $\mathcal{I}_\Omega^Z(s) = z_n$. By induction on the length of $P_\Omega(s)$, it can be verified (see, e.g., the proof of Lemma III.1 in [22]) that

$$I(\mathcal{I}_\Omega^Z(s)) = \mathcal{E}_\Omega^{\mathbf{G}}(s) \tag{5.12}$$

which essentially says that the information state component of $\mathcal{I}_\Omega^Z(s)$ is the state estimator of s.

Example 5.4.1. *Let us return to the system $\boldsymbol{G}$ in Figure 5.1. Consider the sensor activation policy Ω that monitors event σ_1 only when nothing has been observed so far and monitors noting after the first event is observed. (Note that o is always monitored by default.) Let us consider the string $s = \sigma_1\sigma_2$. The corresponding run of s is*

$$\mathcal{R}_\Omega(\sigma_1\sigma_2) = \{o, \sigma_1\}\sigma_1\{o\} \tag{5.13}$$

and the corresponding sequence of Y- and Z-states is

$$\{1\} \xrightarrow{\{o,\sigma_1\}} (\{1,2\}, \{o, \sigma_1\}) \xrightarrow{\sigma_1} \{4\} \xrightarrow{\{o\}} (\{4,5\}, \{o\}) \tag{5.14}$$

So $\mathcal{I}_\Omega^Y(\sigma_1\sigma_2) = \{4\}$, $\mathcal{I}_\Omega^Z(\sigma_1\sigma_2) = (\{4,5\}, \{o\})$ and $\mathcal{E}_\Omega^{\boldsymbol{G}}(\sigma_1\sigma_2) = I(\mathcal{I}_\Omega^Z(\sigma_1\sigma_2)) = \{4,5\}$.

$\square$

5.4.2 Bipartite Dynamic Observer

Recall that the sensor activation policy Ω is defined as an automaton A with a function L. In the following, we define the structure of Bipartite Dynamic Observer (BDO) that also provide a way to realize a (set of) sensor activation policy(ies).

Definition 5.4.1. *A bipartite dynamic observer $\mathcal{O}$ is a 7-tuple*

$$\mathcal{O} = (Q_Y^{\mathcal{O}}, Q_Z^{\mathcal{O}}, \hbar_{YZ}^{\mathcal{O}}, \hbar_{ZY}^{\mathcal{O}}, \Sigma, \Theta, y_0) \tag{5.15}$$

where, $Q_Y^{\mathcal{O}} \subseteq I$ is a set of Y-states, $Q_Z^{\mathcal{O}} \subseteq I \times \Theta$ is a set of Z-states, $\hbar_{YZ}^{\mathcal{O}} : Q_Y^{\mathcal{O}} \times \Theta \rightarrow Q_Z^{\mathcal{O}}$ and $\hbar_{ZY}^{\mathcal{O}} : Q_Z^{\mathcal{O}} \times \Sigma \rightarrow Q_Y^{\mathcal{O}}$ are partial transition functions such that for any $z \in Q_Z^{\mathcal{O}}, y \in Q_Y^{\mathcal{O}}, \theta \in \Theta$ and $\sigma \in \Sigma$, the following conditions hold

C1. $\hbar_{ZY}^{\mathcal{O}}(z, \sigma) = y \Leftrightarrow \hbar_{ZY}(z, \sigma) = y$;

C2. $\hbar_{YZ}^{\mathcal{O}}(y, \theta) = z \Rightarrow \hbar_{YZ}(y, \theta) = z$;

C3. $\forall y \in Q_Y^{\mathcal{O}}, \exists \theta \in \Theta : \hbar_{YZ}^{\mathcal{O}}(y, \theta)!$.

Σ is the set of events of $\boldsymbol{G}$, Θ is the set of admissible sensing decisions, and $y_0 = \{x_0\}$ is the initial Y-state. For brevity, we only consider the accessible part of a BDO.

The three conditions in the above definition are interpreted as follows. Condition C1 says that the transition function $\hbar_{ZY}^{\mathcal{O}}$ in $\mathcal{O}$ is identical to $\hbar_{ZY}$. Therefore, for any $z \in Q_Z^{\mathcal{O}}$, $\hbar_{ZY}^{\mathcal{O}}(z, \sigma)$ is defined for any possible observation $\sigma \in \Theta(z)$ by the definition of $\hbar_{ZY}$. This is due to the fact that we cannot decide which monitored event will occur once we make a sensing decision. Conditions C2 says that for the transition function $\hbar_{YZ}^{\mathcal{O}}$, we have either $\hbar_{YZ}^{\mathcal{O}}(y, \theta) = \hbar_{YZ}(y, \theta)$ or it is undefined. Condition C3 requires that for any Y-state $y \in Q_Y^{\mathcal{O}}$, there exists at least one $\theta \in \Theta$ such that $\hbar_{YZ}^{\mathcal{O}}(y, \theta)$ is defined. This is because a sensor activation policy is defined for all strings in $\mathcal{L}(\mathbf{G})$ and we must make a sensing decision at all accessible Y-states.

Definition 5.4.2. *Given a BDO $\mathcal{O}$, we say that a sensor activation policy Ω is* allowed *by $\mathcal{O}$ if*

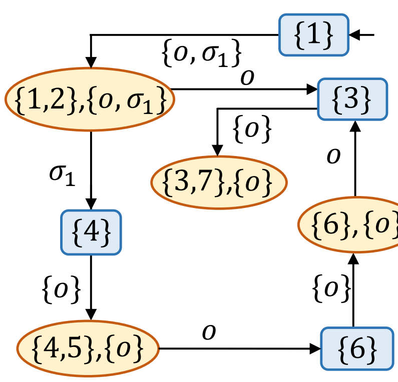
$$\forall s \in \mathcal{L}(\boldsymbol{G}) : \hbar_{YZ}^{\mathcal{O}}(\mathcal{I}_{\Omega}^{Y}(s), \Omega(s))! \tag{5.16}$$

With a slight abuse of notation, we write that $\Omega \in \mathcal{O}$ whenever Ω is allowed by $\mathcal{O}$.

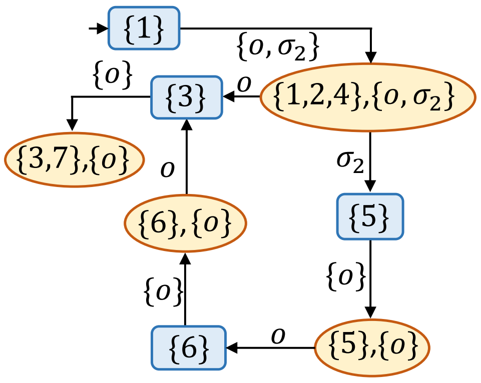
Note that, given a BDO $\mathcal{O}$, the set of sensor activation policies allowed by $\mathcal{O}$ may not be a singleton, since for each Y-state there may be multiple sensing decisions to choose from. Moreover, the domain of a sensor activation policy in a BDO need not be finite since different sensing decisions may be chosen on different visits to the same Y-state. We say that a BDO $\mathcal{O}$ is *deterministic* if, for any $y \in Q_Y^{\mathcal{O}}$, there exists only one $\theta \in \Theta$ such that $\hbar_{YZ}^{\mathcal{O}}(y, \theta)!$. It is clear that a deterministic BDO $\mathcal{O}$ allows a unique sensor activation policy; we denote it by $\Omega_{\mathcal{O}}$. More specifically, $\Omega_{\mathcal{O}} = (A_{\mathcal{O}}, L_{\mathcal{O}})$, where $A_{\mathcal{O}} = (X_A, \delta_A, \Sigma_o \cup \Sigma_s, x_{0,A})$, is defined by: $X_A = Q_Y^{\mathcal{O}}$, $x_{0,A} = y_0$ and for any $x_1, x_2 \in X_A$, we have $f_A(x_1, \sigma) = x_2 \Leftrightarrow \hbar_{ZY}^{\mathcal{O}}(\hbar_{YZ}^{\mathcal{O}}(x_1, \theta_{x_1}), \sigma) = x_2$ and $L(x_1) = \Theta(\hbar_{YZ}^{\mathcal{O}}(x_1, \theta_{x_1}))$, where θ_{x_1} is the unique sensing decision defined at x_1 in $\mathcal{O}$. Note that, in general we may need infinite memory to realize a sensor activation policy. Therefore, a deterministic BDO can only represent a sensor activation policy that has at most 2^X states. However, we will show later that this memory is always sufficient for the purpose of synthesis.

Example 5.4.2. *Consider again the system $\boldsymbol{G}$ in Figure 5.1. Figure 5.2(a) provides an example of a deterministic BDO. For the initial Y-state $y_0 = \{1\}$, by making sensing decision $\theta = \{o, \sigma_1\}$, we will reach Z-state $z = \hbar_{YZ}(y_0, \theta) = (\{1, 2\}, \{o, \sigma_1\})$. From z, only monitored events o and σ_1 can be observed. If σ_1 is observed, then the next Y-state is $y_1 = \hbar_{ZY}(z, \sigma_1) = \{4\}$. We can verify that the sensor activation policy Ω defined in Example 5.4.1 is allowed by $\mathcal{O}_1$; moreover, it is the only one allowed by $\mathcal{O}_1$ since this BDO is deterministic. Similarly, the BDO $\mathcal{O}_2$ shown in Figure 5.2(b) is also deterministic. However, the BDO shown in Figure 5.3 is not a deterministic BDO, since there are two sensing decisions $\{o, \sigma_1\}$ and $\{o, \sigma_2\}$ defined at Y-state $\{1\}$.*

$\square$



(a) Solution $\mathcal{O}_1$



(b) Solution $\mathcal{O}_2$

Figure 5.2: Examples of BDOs that represent two incomparable minimal solutions; [blue] rectangular states and [yellow] oval states represent, respectively, Y-states and Z-states.

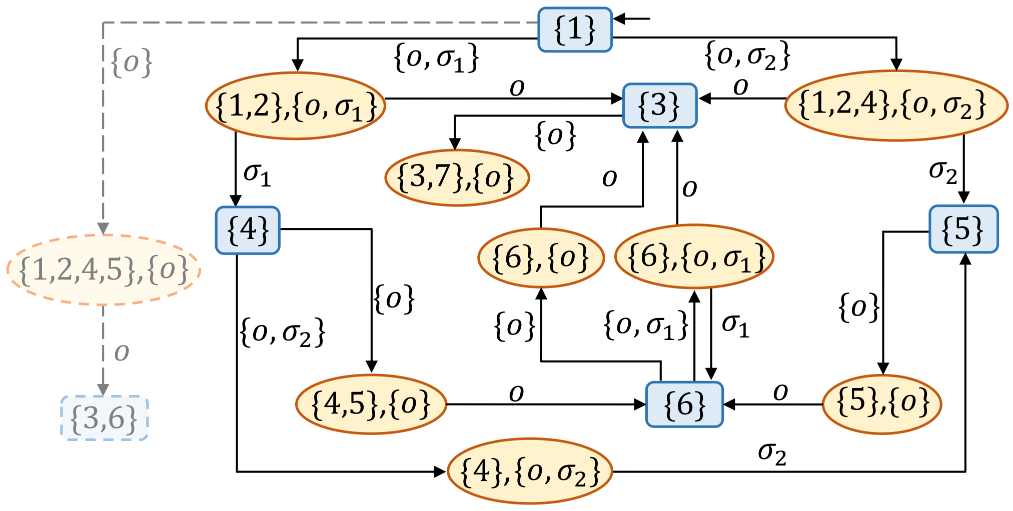


Figure 5.3: Example of MPO

5.4.3 Generalized MPO and its Properties

We return to the sensor action problem for IS-based properties formulation in Section 5.3. By condition (i) in Problem 1, we must find an Ω such that $\forall s \in \mathcal{L}(\mathbf{G}) : \varphi(\mathcal{E}_{\Omega}^{\mathbf{G}}(s)) = 1$. However, for any BDO, we know that $\forall s \in \mathcal{L}(\mathbf{G}) : I(\mathcal{I}_{\Omega}^{Z}(s)) = \mathcal{E}_{\Omega}^{\mathbf{G}}(s)$ and $\mathcal{I}_{\Omega}^{Z}(s)$ is indeed the Z-state reached by the run $\mathcal{R}_{\Omega}(s)$ in the BDO. Therefore, if we construct a BDO $\mathcal{O}$ such that

$$\forall z \in Q_Z^{\mathcal{O}} : \varphi(I(z)) = 1 \tag{5.17}$$

and such that $\mathcal{O}$ is "as large as possible", then the resulting structure will contain all sensor activation policies that satisfy φ. The property of such a BDO being as large as possible is actually well defined: if $\mathcal{O}_1$ and $\mathcal{O}_2$ are two BDOs that both satisfy Equation (5.17), then their union, in the sense of graph merger, is a BDO that satisfies Equation (5.17). For example, the BDOs $\mathcal{O}_1$ and $\mathcal{O}_2$ shown in Figure 5.2(a) and Figure 5.2(b), respectively, both satisfy Equation (5.17). Their union, which is a sub-graph of the BDO in Figure 5.3, is also a BDO satisfying Equation (5.17). This observation leads to the definition of the most permissive observer.

Definition 5.4.3. *(Most Permissive Observer). Let $\boldsymbol{G} = (X, \Sigma, \delta, x_0)$ be the system and let $\varphi : 2^X \to \{0,1\}$ be the IS-based property under consideration. The Most Permissive Observer for φ is the BDO*

$$\mathcal{MPO}_{\varphi} = (Q_Y^{MPO}, Q_Z^{MPO}, \hbar_{YZ}^{MPO}, \hbar_{ZY}^{MPO}, \Sigma, \Theta, y_0)$$

defined as the largest BDO such that $\forall z \in Q_Z^{MPO} : \varphi(I(z)) = 1$.

The following theorem reveals the correctness of the MPO defined above, namely, the MPO embeds all sensor activation policies satisfying φ in its structure.

Theorem V.1. $\Omega \models_{\boldsymbol{G}} \varphi$ *if and only if* $\Omega \in \mathcal{MPO}_{\varphi}$.

Proof. ($\Leftarrow$) Suppose that Ω is allowed by the MPO. Then we know that for any $s \in \mathcal{L}(\mathbf{G})$ we have $\mathcal{I}_\Omega^Z(s) \in Q_Z^{MPO}$ by definition. Since $\forall z \in Q_Z^{MPO} : \varphi(I(z)) = 1$, by Equation (5.12), we know that $\forall s \in \mathcal{L}(\mathbf{G}) : \varphi(\mathcal{E}_\Omega^{\mathbf{G}}(s)) = 1$. Therefore, $\Omega \models_{\mathbf{G}} \varphi$.

($\Rightarrow$) Suppose that $\Omega \models_{\mathbf{G}} \varphi$. Then there exists a BDO $\mathcal{O}$ such that: 1) $\Omega \in \mathcal{O}$; and 2) $\forall z \in Q_Z^{\mathcal{O}} : \varphi(I(z)) = 1$. Specifically, $\mathcal{O}$ is obtained by: $Q_Y^{\mathcal{O}} = \{y \in Y : \exists s \in \mathcal{L}(\mathbf{G}) \text{ s.t. } y = \mathcal{I}_\Omega^Y(s)\}$, $Q_Z^{\mathcal{O}} = \{z \in Z : \exists s \in \mathcal{L}(\mathbf{G}) \text{ s.t. } z = \mathcal{I}_\Omega^Z(s)\}$ and $\forall y \in Q_Y^{\mathcal{O}}, \forall \theta \in \Theta : [\hbar_{YZ}^{\mathcal{O}}(y, \theta)!] \Leftrightarrow [\exists s \in \mathcal{L}(\mathbf{G}) : y = \mathcal{I}_\Omega^Y(s) \wedge \theta = \Omega(s)]$. Now let us assume that Ω is not allowed by the MPO. Then we know that $\exists s \in \mathcal{L}(\mathbf{G}) : \hbar_{YZ}^{MPO}(\mathcal{I}_\Omega^Y(s), \Omega(s))$ is not defined. However, this implies that the union of $\mathcal{O}$ and $\mathcal{MPO}_\varphi$ is strictly larger than $\mathcal{MPO}_\varphi$, since $\Omega(s)$ is defined at $\mathcal{I}_\Omega^Y(s)$ in $\mathcal{O}$ but not in $\mathcal{MPO}_\varphi$. This contradicts to the fact the MPO is the largest BDO satisfying Equation (5.17). $\square$

Algorithm 1 provides a procedure for the construction of the MPO. The steps of Algorithm 1 follow direction from the definition of the MPO. First, we search through the state space of Y-states and Z-states until a Z-state that violates the IS-based property φ is encountered. This step is done by Procedure DoDFS, which is simply a depth-first search. However, this may lead to the situation where there is a Y-state that has no successors. Recall that this situation is illegal by the definition of the BDO. Therefore, we need to go back to prune such a Y-state and the corresponding Z-states that lead to this state, until the structure converge. This step is done by the while loop, which will stop in a finite number of steps. The worst-case time complexity of the construction of the MPO is exponential in both $|X|$ and $|\Sigma_s|$.

Example 5.4.3. *We return to system* $\mathbf{G}$ *in Figure 5.1 and IS-based property* φ *defined by Equation (5.9). The corresponding MPO is shown in Figure 5.3. At initial* Y*-state* $\{1\}$*, if we make sensing decision* $\{o\}$*, then* Y*-state* $\{3, 6\}$ *will be reached upon the occurrence of monitored event* o *(see the dashed lines). However, at state* $\{3, 6\}$*, no matter what sensing decision we make, a* Z*-state that contains both state* 3 *and* 6 *will be reached, which violates the IS-based property* φ*. Therefore, we need to go back*

Algorithm 8: The construction of the MPO

```
Data: $\mathbf{G}$ and $\varphi$
Result: $\mathcal{MPO}_\varphi$
 1  $Q_Y^{MPO} \leftarrow y_0 = \{x_0\}$ and $Q_Z^{MPO} \leftarrow \emptyset$;
 2  DoDFS($y_0, \mathcal{MPO}_\varphi, \varphi$);
 3  while $\exists y \in Q_Y^{MPO} : \nexists \theta \in \Theta \ s.t. \ \hbar_{YZ}^{MPO}(y, \theta)!$ do
 4      $Q_Y^{MPO} \leftarrow Q_Y^{MPO} \setminus \{y\}$;
 5      remove all $Z$-states $z \in Q_Z^{MPO}$ such that $\hbar_{ZY}^{MPO}(z, \sigma) = y$ for some $\sigma \in \Sigma$;
 6      take the accessible part of $\mathcal{MPO}_\varphi$;
    procedure DoDFS($y, \mathcal{MPO}_\varphi, \varphi$);
 7  for $\theta \in \Theta$ do
 8      $z \leftarrow \hbar_{YZ}(y, \theta)$;
 9      if $\varphi(I(z)) = 1$ then
10          add transition $y \xrightarrow{\theta} z$ to $\hbar_{YZ}^{MPO}$;
11          if $z \notin Q_Z^{MPO}$ then
12              $Q_Z^{MPO} \leftarrow Q_Z^{MPO} \cup \{z\}$;
13              for $\sigma \in \Sigma \ s.t. \ \hbar_{ZY}(z, \sigma)!$ do
14                  $y' \leftarrow \hbar_{ZY}(z, \sigma)$;
15                  add transition $z \xrightarrow{\sigma} y'$ to $\hbar_{ZY}^{MPO}$;
16                  if $y' \notin Q_Y^{MPO}$ then
17                      $Q_Y^{MPO} \leftarrow Q_Y^{MPO} \cup \{y'\}$;
18                      DoDFS($y', \mathcal{MPO}_\varphi, \varphi$);
```

to prune Y-state $\{3,6\}$ and its predecessor Z-state $(\{1,2,4,5\},\{o\})$. This is why we cannot make sensing decision $\{o\}$ at the initial state. □

Remark 5.4.1. In Figure 5.3, we can also make sensing decision $\{o,\sigma_1,\sigma_2\}$ at the initial Y-state. However, σ_2 cannot be observed before the next sensing decision is issued, which will occur when either o or σ_1 is observed. Therefore, σ_2 is a "redundant" event in the sensing decision, since it has no effect on future states in the MPO. In this chapter, we adopt the following convention. We *remove* all redundant events from sensing decisions in the MPO when solving the *minimal* sensor activation problem. Similarly, we *include* all redundant events to the sensing decisions in the MPO when solving the *maximal* sensor activation problem. Clearly, these conventions will not affect the properties of the MPO.

5.5 Synthesis of Optimal Sensor Activation Policies

In this section, we show how to synthesize from the MPO an optimal sensor activation policy Ω that solves Problem 5. Specifically, we require that Ω satisfy the minimality criterion (ii) of Problem 5 (or the maximality criterion for the dual version of Problem 5). Moreover, we shall also require that Ω be defined over a finite domain, so that it can be effectively implemented. As was mentioned earlier, a sensor activation policy allowed by the MPO need not be finitely realizable, since we can select different sensing decisions upon different visits to the same Y-state. Therefore, we define a special class of sensor activation policies that are represented by subgraphs of the MPO and thus have finite realizations.

Definition 5.5.1. *(IS-based Sensor Activation Policy). A sensor activation policy Ω is said to be Information-State-based (or IS-based) if*

$$\forall s,t \in \mathcal{L}(\boldsymbol{G}) : \mathcal{I}^Y_\Omega(s) = \mathcal{I}^Y_\Omega(t) \Rightarrow \Omega(s) = \Omega(t) \tag{5.18}$$

Clearly, if Ω is IS-based, then Ω can always be represented by a deterministic BDO that is a subgraph of the MPO.

Definition 5.5.2. *(Greedy Optimal Sensor Activation Policy). Suppose that Ω is a sensor activation policy such that $\Omega \models_{\boldsymbol{G}} \varphi$. We say that Ω is* greedy minimal *if*

$$\forall s \in \mathcal{L}(\boldsymbol{G}), \forall \theta \in \Theta : \hbar_{YZ}^{MPO}(\mathcal{I}_{\Omega}^{Y}(s), \theta)! \Rightarrow \theta \not\subset \Omega(s) \tag{5.19}$$

The notion of greedy maximality *is defined analogously.*

The following theorem says that a greedy minimal (respectively, maximal) solution is a minimal (respectively, maximal) solution.

Theorem V.1. *Let Ω be a sensor activation policy such that $\Omega \models_{\boldsymbol{G}} \varphi$. Then Ω is minimal (respectively, maximal) if it is greedy minimal (respectively, greedy maximal).*

Proof. We prove minimality by contradiction; maximality can be proved analogously. Suppose that Ω is greedy minimal and assume that it is not minimal. This implies that there exists another Ω' such that $\Omega' < \Omega$ and $\Omega' \models_{\mathbf{G}} \varphi$. Then we know that there exists a string $t \in \mathcal{L}(\mathbf{G})$ such that: 1) $\Omega'(t) \subset \Omega(t)$; and 2) $\forall t' \in \overline{\{t\}} \setminus \{t\}$: $\Omega'(t') = \Omega(t')$. Then we know that $\mathcal{I}_{\Omega'}^{Y}(t) = \mathcal{I}_{\Omega}^{Y}(t)$ and we call this Y-state y. Since $\Omega' \models_{\mathbf{G}} \varphi$, by Theorem V.1, we know that Ω' is also allowed by the MPO, which means that $\hbar_{YZ}^{MPO}(y, \Omega'(s))!$. However, by the fact that Ω is greedy minimal, we know that $[\forall \theta \in \Theta : \hbar_{YZ}^{MPO}(y, \theta)!]\theta \not\subset \Omega(s)$. This contradicts $\Omega'(t) \subset \Omega(t)$. □

By Theorem V.1, it is clear that if we synthesize an IS-based greedy optimal sensor activation policy, then we will have obtained a solution to Problem 5, which was our objective. (Of course, not all solutions to Problem 5 need be IS-based or greedy.) An IS-based greedy optimal sensor activation policy can be obtained by a depth-first search over the state space of the MPO that picks *one* greedy optimal sensing decision at each Y-state and then picks *all* observations for each Z-state.

The resulting structure will be a deterministic BDO that represents the solution. We illustrate this synthesis procedure by an example.

Example 5.5.1. *We return to the MPO shown in Figure 5.3. To synthesize a minimal sensor activation policy for φ, we can pick decision $\{o, \sigma_1\}$, which is greedy minimal, at the initial Y-state. Then, upon the occurrence of monitored event σ_1, the new Y-state $\{4\}$ is reached. At that state, we pick the unique greedy minimal decision $\{o\}$, and so forth. These choices result in deterministic BDO $\mathcal{O}_1$ shown in Figure 5.2(a) that allows the unique sensor activation policy $\Omega_{\mathcal{O}_1}$, which is provably minimal. We see that $\Omega_{\mathcal{O}_1}$ is, in fact, the sensor activation policy Ω defined by Equation (5.4.1).*

Remark 5.5.1. In the synthesis step in the previous example, we could have selected sensing decision $\{o, \sigma_2\}$ at the initial Y-state, which yields the minimal solution shown in Figure 5.2(b). Interestingly, we see that the intersection of the two valid decisions $\{o, \sigma_1\}$ and $\{o, \sigma_2\}$, i.e, $\{o\}$ is not a valid decision, since $\{o\}$ is not defined at Y-state $\{1\}$ in the MPO. This illustrates the earlier claim that Problem 1 may not have an infimal (respectively, supremal) solution in general, but instead several incomparable minimal (respectively maximal) solutions. This phenomenon is similar to the supervisory control problem under partial observation studied in Chapters II and III, in which supremal solutions do not exist in general and only locally maximal solutions exist.

5.6 Applications of the Generalized MPO

5.6.1 Application to Control and Diagnosis

Observability and diagnosability are two key properties of interest in control and diagnosis of DES. It is shown in [102] that the problem of sensor activation for observability can be formulated as a state-disambiguation problem. Similarly, it is shown in [22] that the problem of sensor activation for K-diagnosability can be formulated as a state-disambiguation problem. Therefore, as was discussed in Remark 5.3.1, both of these sensor activation problems can be solved by the generalized MPO approach that we have presented. In fact, the most permissive observer for K-diagnosability [16,22] is a special case of the MPO defined in this chapter. On the other hand, the notion of an MPO for the property of observability has never been considered in the literature. The generalized MPO therefore provides a new approach for solving sensor activation for enforcement of observability. The reader is referred to [22, 102] to see how observability and K-diagnosability can be formulated as IS-based properties. Another property of interest in sensor activation is detectability [89]; it relates to state reconstruction. By using the same approach that is used for the reformulation of K-diagnosability in [22], we can show that strong K-detectability can also be formulated as an IS-based property and thereby our solution procedure also applies to that property.

5.6.2 Application to Fault Prediction

As a specific example of how the methodology presented in this chapter can be used to solve problems that have not yet been addressed in the literature, we consider the problem of sensor activation for the enforcement of *predictability*, a notion introduced in [29]. Let $e_d \in \Sigma$ be the fault event to be predicted. Recall that $\Psi(e_d) := \{se_d \in \mathcal{L}(\mathbf{G}) : s \in \Sigma^*\}$ is the set of strings that end with e_d and we write $e_d \in s$ if $\overline{s} \cap \Psi(e_d) \neq \emptyset$. We recall the definition of predictability from [29].

Definition 5.6.1. *(Predictability). A live language $\mathcal{L}(\boldsymbol{G})$ is said to be predictable w.r.t. $e_d \in \Sigma$ and Ω if*

$$\begin{aligned}
&(\forall s \in \Psi(e_d))(\exists t \in \overline{\{s\}} : e_d \notin t)(\forall u \in \mathcal{L}(\boldsymbol{G}) : e_d \notin u \wedge P_\Omega(u) = P_\Omega(t)) \\
&(\exists n \in \mathbb{N})(\forall v \in \mathcal{L}(\boldsymbol{G})/u)[|v| \geq n \Rightarrow e_d \in v] \qquad (5.20)
\end{aligned}$$

The above definition requires that the fault event e_d should be predicted unambiguously before its the occurrence. Note that the liveness assumption here is w.l.o.g., since we can always add unobservable self-loops at terminal states in $\mathbf{G}$.

To proceed further, we assume that state space of $\mathbf{G}$ is partitioned into two disjoint sets $X = X_Y \dot{\cup} X_N$, such that

- $\forall s \in \mathcal{L}(\mathbf{G}) : \delta(x_0, s) \in X_Y \Rightarrow e_d \in s$; and
- $\forall s \in \mathcal{L}(\mathbf{G}) : \delta(x_0, s) \in X_N \Rightarrow e_d \notin s$.

That is, X_Y is the set of faulty states and X_N is the set of non-faulty states. This assumption is also w.l.o.g., since we can always refine the state space of $\mathbf{G}$ by taking the parallel composition of $\mathbf{G}$ with an automaton with two states that captures the occurrence of e_d. Next, similarly to the notions of boundary strings and indicator strings in [47], we define the two following sets:

- Boundary states, $\partial_X = \{x \in X : \delta(q, e_d)!\}$; and
- Non-indicator states, $\mathcal{N}_X = \{x \in X_N : \forall n \in \mathbb{N}, \exists s \in \mathcal{L}(\mathbf{G}, x) \text{ s.t. } |s| > n \wedge e_d \notin s\}$.

A boundary state is a state from which the fault event can occur and a non-indicator state is a state from which an arbitrary long non-faulty string can occur. Note that ∂_X and $\mathcal{N}_X$ need not be disjoint in general.

With the above notions, we define the IS-based property $\varphi_{pre} : 2^X \to \{0, 1\}$ by:

$$\forall i \in 2^X : [\varphi_{pre}(i) = 0] \Leftrightarrow [\exists x, x' \in i : x \in \partial_X \wedge x' \in \mathcal{N}_X] \tag{5.21}$$

The following result says that predictability is equivalent to the IS-based property φ_{pre}.

Theorem V.1. *Let φ_{pre} be the IS-based property defined by Equation (5.21). For any sensor activation policy Ω, $\mathcal{L}(\boldsymbol{G})$ is predictable w.r.t. e_d and Ω if and only if $\Omega \models_{\boldsymbol{G}} \varphi_{pre}$.*

Proof. ($\Rightarrow$) By contrapositive. Suppose that $\Omega \not\models_{\mathbf{G}} \varphi_{pre}$. We know that $\exists s, t \in \mathcal{L}(\mathbf{G}) : P_\Omega(s) = P_\Omega(t) \wedge \delta(x_0, s) \in \partial_Q \wedge \delta(x_0, t) \in \mathcal{N}_Q$. Considering the above s and t, we also know that $(\forall v \in \overline{\{s\}})(\exists u \in \overline{\{t\}})[P_\Omega(v) = P_\Omega(u)]$. By definition, $\delta(x_0, t) \in \mathcal{N}_Q$ implies that $\forall u \in \overline{\{t\}} : \delta(x_0, u) \in \mathcal{N}_Q$. Then we have $(\exists s e_d \in \Psi(e_d))(\forall v \in \overline{\{s\}} : e_d \notin v)(\exists u \in \mathcal{L}(\mathbf{G}) : e_d \notin u \wedge P_\Omega(v) = P_\Omega(u))(\forall n \in \mathbb{N})(\exists w \in \mathcal{L}(\mathbf{G})/u)[|w| \geq n \wedge e_d \notin w]$. Thus, $\mathcal{L}(\mathbf{G})$ is not predictable.

($\Leftarrow$) By contrapositive. Suppose that $\mathcal{L}(\mathbf{G})$ is not predictable. By Definition 5.6.1, we know that $(\exists s e_d \in \Psi(e_d))(\forall t \in \overline{\{s\}} : e_d \notin t)(\exists u \in \mathcal{L}(\mathbf{G}) : e_d \notin u \wedge P_\Omega(u) = P_\Omega(t))(\forall n \in \mathbb{N})(\exists v \in \mathcal{L}(\mathbf{G})/u)[|v| \geq n \wedge e_d \notin v]$. For the above s, let us consider the string $\alpha \in \overline{\{s\}}$ such that $e_d \notin \alpha$ and $\alpha e_d \in \overline{\{s\}}$, i.e., α is the longest non-faulty prefix of s. By definition, we know that $\delta(x_0, \alpha) \in \partial_Q$. Also, we know that $\exists u \in \mathcal{L}(\mathbf{G})$ such that $P_\Omega(u) = P_\Omega(\alpha)$ and $\delta(x_0, \alpha) \in \mathcal{N}_Q$. This implies that $\exists \alpha \in \mathcal{L}(\mathbf{G}), \exists x, x' \in \mathcal{E}_\Omega^{\mathbf{G}}(\alpha) : x \in \partial_Q \wedge x' \in \mathcal{N}_Q$. Therefore, $\Omega \not\models_{\mathbf{G}} \varphi_{pre}$. □

The above theorem implies that to synthesize a minimal sensor activation policy for the purpose of prediction, it suffices to solve Problem 5 by taking φ_{pre} into account. We illustrate this result by the following example.

Example 5.6.1. *Let us return to the system* $\mathbf{G}$ *in Figure 5.1. Suppose that* f *is the fault event that we want to predict.* $\mathbf{G}$ *already satisfies the state partition assumption* $X = X_Y \dot\cup X_N$*, where* $X_N = \{1, 2, 3, 4, 5, 6\}$ *and* $X_Y = \{7\}$*. Also, we know that state* 3 *is the only boundary state and states* 1, 4, 5 *and* 6 *are non-indicator states. For example, from state* 6*, the arbitrary long non-faulty behavior* $e^n, n \in \mathbb{N}$*, can occur. However, states* 2 *and* 3 *are not non-indicator states, since from either of these two states, we know for sure that the fault event will occur in a finite number of steps. Therefore, we have* $\partial_X = \{3\}$ *and* $\mathcal{N}_X = \{1, 4, 5, 6\}$*. In fact, we see that the IS-based property defined by Equation (5.9) that we considered in the previous examples is indeed the IS-based property* φ_{pre} *for this example. Therefore, the solutions* $\mathcal{O}_1$ *and* $\mathcal{O}_2$ *shown in Figure 5.2 that we obtained previously are two (incomparable) minimal* *sensor activation policies that guarantee predictability.*

5.6.3 Application to Cyber-Security

As was discussed earlier in Remark 5.3.2, in some cases, the system may also be monitored by an *external* observer that is potentially malicious. Therefore, for security purposes, one may want the information mapping not to release some crucial information to this external observer. We recall an important security property called opacity.

Definition 5.6.2. *Secret $X_S \subseteq X$ is current-state opaque w.r.t. $\boldsymbol{G}$ and Ω if $\forall s \in \mathcal{L}(\boldsymbol{G}) : \mathcal{E}_{\Omega}^{\boldsymbol{G}}(s) \not\subseteq X_S$.*

In the above definition, the secret of the system is defined in terms of the current-state estimator. Some other notions of opacity, e.g., initial-state opacity and language-based opacity, have also been studied in the literature. However, since all of these notions can be mapped to one another (see [107]), the study of current-state opacity here is without essential loss of generality. Current-state opacity is clearly an IS-based property. Therefore, the most permissive dynamic mask studied in [15] is also a special case of the generalized MPO and the problem of synthesizing a maximal sensor activation policy (or dynamic mask) can also be solved by the approach presented in this chapter.

Moreover, the same approach can be applied to other user defined properties. For example, consider the IS-based property $\varphi : 2^X \to \{0, 1\}$ defined by

$$\forall i \in 2^X : \varphi(i) = 0 \Leftrightarrow |i| = 1 \tag{5.22}$$

This property is related to 1-*anonymity* studied in the computer security literature [84, 90]. Intuitively, it requires that the observer should never determine the current-state of the system precisely. We can also synthesize a sensor activation policy for it by applying the generalized MPO approach.

5.7 Conclusion

In this chapter, we presented a new approach to the problem of synthesizing an optimal sensor activation policy that guarantees some observation property in problems of control, diagnosis, prediction, or other types in the context of partially-observed discrete event systems. To this end, we defined a novel information structure called the generalized Most Permissive Observer that is applicable to a wide class of properties called information-state-based properties. The generalized MPO embeds all valid sensor activation policies in its structure. We presented an algorithm for the construction of the MPO and a procedure for synthesizing an optimal sensor activation policy based on the MPO. Our approach generalizes the previous works on the MPO, which pertain to specific properties such as opacity or K-diagnosability. Our approach is applicable to a wide class of user-defined properties. In particular, we showed how the problem of optimal sensor activation for the purpose of fault prediction, not previously considered in the literature, can be solved by the generalized MPO.

CHAPTER VI

Sensor Activation in Decentralized Decision Making

6.1 Introduction

In this chapter, we investigate the problem of decentralized decision making in DES that operates under dynamic observations. In this context, the system is monitored by *a set of agents* that act as a team to make global decisions. Each agent makes observations online through its sensors; these sensors can be turned on/off online dynamically during the evolution of the system according to a sensor activation policy that depends on the agent's observations. Due to energy, bandwidth, or security constraints, sensors activations are "costly". Therefore, in order to reduce sensor-related costs, it is of interest to minimize, in some technical sense, the sensor activations of each agent while maintaining some desired observational property.

However, for the decentralized sensor activation problem, there are very few results in the literature. In [101], the problem of dynamic sensor activation for decentralized diagnosis is studied. Specifically, a "window-based partition" approach is proposed in order to obtain a solution. The main drawback of this approach is that the solution obtained is only optimal with respect to a finite (restricted) solution space and may not be language-based optimal in general. In other words, by enlarging the solution space by refining the state space of the system model, better solutions could be obtained in principle. Similarly, in [104], the problem of dynamic sensor activation for decentralized control is also studied, where the solution obtained is again optimal w.r.t. a finite solution space. To the best of our knowledge, the problem of *language-based* sensor optimization for *decentralized* diagnosis or control has remained an open problem, as is mentioned in the recent survey [82].

One important reason for the lack of results for the decentralized sensor activation problem is that decentralized multi-player decision problems are conceptually much more difficult to solve than their corresponding centralized versions. In general, these types of problems have been discussed in the framework of team decision theory [32]. In the DES literature, it is well-known that many problems that are decidable in the centralized setting become undecidable (e.g., the problem of synthesizing safe and non-blocking supervisors [98]) or open (e.g., the problem of synthesizing maximally permissive safe supervisors [59]) in the decentralized case, even when only two agents are involved.

In this chapter, we propose a new approach to tackle the problem of dynamic sensor activation for the purpose of decentralized decision-making. The main contributions of this chapter are as follows. First, we formulate a general class of decentralized decision-making problems called the *decentralized state disambiguation problem*. We propose the notion of *decentralized distinguishability*, which covers coobservability, K-codiagnosability and coprognosability. Second, to solve the dynamic sensor activation problem, we adopt a *person-by-person* approach (see, e.g., [100] and the references therein) to decompose the decentralized minimization problem to two consecutive centralized minimization problems. We first minimize the sensor activation policy for Agent 1 by keeping the policy of Agent 2 fixed. Then, we fix Agent 1's sensor activation policy to the one obtained and solve the same minimization problem but for Agent 2. Essentially, we solve two centralized *constrained* minimization problems, since we need to take the other agent's information into account when we minimize the decisions of an agent. A novel approach is also proposed to reduce each centralized constrained minimization problem to a problem that we solved in Chapter V.. Moreover, we prove that the solution obtained by our procedure is minimal with respect to the system language (i.e., over an infinite set in general), in contrast to the works reviewed above where minimality was with respect to a finite solution space. As special cases of the proposed framework, *language-based* sensor optimizations for decentralized diagnosis and decentralized control, which were previously open, are solved. Finally, we show that the proposed framework is applicable to both the disjunctive architecture and the conjunctive architecture.

In general, a person-by-person approach in team decision problems may not terminate in a finite number of steps, since we may need to iterate between the two constrained minimization problems. However, we show that for the problem under consideration in this chapter, such iterations are not required due to a certain type of *monotonicity* that arises. Moreover, we prove that the solution obtained by our procedure is minimal w.r.t. the system language (i.e., over an infinite set in general), in contrast to the works reviewed above where minimality was with respect to a finite solution space. In the DES literature, the person-by-person approach has also been applied to the decentralized control problem [59] and to the decentralized communication problem [7, 66, 75]. However, to the best of our knowledge, it has not been applied so far to decentralized sensor activation.

The remainder of this chapter is organized as follows. Section 6.2 we formulate the decentralized state disambiguation problem and the decentralized minimization problem that we solve in this chapter. In Section 6.3, shows how to solve the centralized constrained minimization problem by reducing it to a fully centralized problem. In Section 6.4, we present our algorithm for synthesizing a minimal decentralized solution. In Section 6.5, we show how specific problems, e.g., sensor activation for decentralized diagnosis/control/prognosis can be solved by the proposed framework. We also extend our results to the conjunctive architecture in Section 6.6. Finally, we conclude this chapter in Section 6.7.

6.2 Problem Formulation and Solution Overview

6.2.1 Decentralized Distinguishability

In the decentralized decision-making problem, at each instant, each local agent sends highly compressed information, i.e., a local decision, to the coordinator based on its local (dynamic) observation. Then the coordinator makes a global decision based on the information received from each local agent. Let $\mathcal{I}$ be the index set of local agents. For each agent $i \in \mathcal{I}$, we denote by Ω_i its sensor activation policy and by $\Sigma_{o,i}$ the set of events that can be monitored in Ω_i. For the sake of simplicity, we develop all results hereafter for the case of two agents, i.e., $\mathcal{I} = \{1, 2\}$. The principle can be extended to an arbitrary number of agents. We define the pair of sensor activation policies as $\bar{\Omega} = [\Omega_1, \Omega_2]$.

In order to formulate the decentralized decision-making problem, we need to specify the following three ingredients:

- What requirement the global decision has to fulfill?
- What information each local agent can send to the coordinator?
- What rule the coordinator uses to calculate a global decision based on the local decisions?

Hereafter, we refer to the first ingredient as the *specification* of the decentralized decision-making problem. The last two ingredients are referred to as the *architecture* of the decentralized decision-making problem.

Several different specifications have been studied separately in the literature for decentralized decision-making problems, e.g., to diagnose every occurrence of fault events [24,62], to predict every occurrence of fault events [47] or to control the system [69, 128]. In this chapter, we do not study a specific specification. Instead, we define a general class of specifications called *decentralized state disambiguation.* We show will later in Section 6.5 that many existing decentralized decision-making problems are special cases of the decentralized disambiguation problem. Formally, we define a specification as a pair of state sets

$$T = X_A^T \times X_B^T \subseteq X \times X \tag{6.1}$$

Intuitively, specification T is used to capture the following requirement. State set X_A^T represents the set of states at which the global system *must* take some desired action associated to T and state set X_B^T represents the set of states at which the global system *should not* take such an action. Then the system must be able to distinguish between states in X_A^T and states in X_B^T (under certain decentralized architecture, which will be specified later) when a state in X_A^T is reached; otherwise, the desired action associated to T cannot be taken safely.

Regarding the architecture of the decentralized decision-making problem, here we consider the following mechanism, which is widely used in the literature for many different problems [24, 47, 62, 69, 128]. We assume that communication between each agent and the coordinator is costly and only a binary decision is allowed for each agent at each instant. That is, each local agent can only send to the coordinator a highly compressed decision "1" or "0", which correspond to "take the action" and "do not take the action", respectively. Then, the coordinator has two possible *fusion rules* to obtain a global decision from local decisions:

- the disjunctive rule: issues "1" globally, if and only if, *one* local agent issues "1".

- the conjunctive rule: issues “1” globally, if and only if, *all* local agents issue “1”.

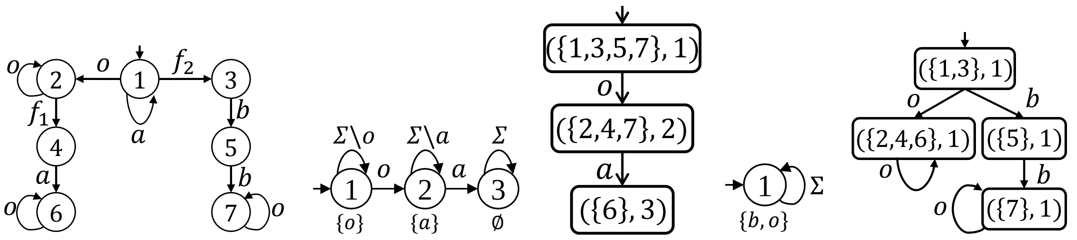
Hereafter, we will develop the main results based on the disjunctive rule. We will discuss how to extend our results to the conjunctive case in Section 6.6.

In general, the system may have multiple distinct objectives, i.e., it needs to distinguish different states pairs for different purposes. For the sake of generality, we consider m specifications and denote by $\mathbb{T} = \{T_1, \dots, T_m\}$ the set of specifications, where $T_k = X_A^{T_k} \times X_B^{T_k} \subseteq X \times X, T_k \in \mathbb{T}$. Also, for the sake of generality, for each $T_k \in \mathbb{T}$, we define $\mathcal{I}_{T_k} \subseteq \mathcal{I}$ as the non-empty set of agents that can contribute to the decision associated to T_k. If $\mathcal{I}_{T_k}$ is a singleton, then the global decision will be “1” if the unique agent in $\mathcal{I}_{T_k}$ issues “1”. However, in the case that $|\mathcal{I}_{T_k}| > 1$, since we consider the disjunctive architecture, the global decision will be “1” if *one* agent in $\mathcal{I}_{T_k}$ issues “1”. Therefore, an agent must be able to distinguish any states pair in T_k *unambiguously* when it issues “1”; otherwise a wrong global decision may be made. This observation leads to the following definition of *decentralized distinguishability.*

Definition 6.2.1. *(Decentralized Distinguishability). Let $\boldsymbol{G}$ be the system, $\mathbb{T} = \{T_1, \dots, T_m\}$ be a set of specifications and $\bar{\Omega} = [\Omega_1, \Omega_2]$ be a pair of sensor activation policies. We say that $\boldsymbol{G}$ is decentralized distinguishable w.r.t. $\bar{\Omega}$ and $\mathbb{T}$ if*

$$(\forall T_k \in \mathbb{T})(\forall s \in \mathcal{L}(\boldsymbol{G}) : \delta(s) \in X_A^{T_k})(\exists i \in \mathcal{I}_{T_k})[\mathcal{E}_{\Omega_i}^{\boldsymbol{G}}(s) \cap X_B^{T_k} = \emptyset] \tag{6.2}$$

Intuitively, the above definition says the following. For any specification $T_k \in \mathbb{T}$, for any string that goes to a state in $X_A^{T_k}$, i.e., a state at which we must take the action associated to T_k, there must exist at least one local agent in $\mathcal{I}_{T_k}$ that knows *for sure* that we can take such an action. Note that, in our setting, only $X_B^{T_k}$ are the set of states at which we cannot take the action associated to T_k. In other words, there is no harm in taking the action if the system is in $X \setminus (X_A^{T_k} \cup X_B^{T_k})$. This is why we require $\mathcal{E}_{\Omega_i}^{\mathbf{G}}(s) \cap X_B^{T_k} = \emptyset$ rather than $\mathcal{E}_{\Omega_i}^{\mathbf{G}}(s) \subseteq X_A^{T_k}$. We will show later



(a) System **G** (b) Ω_1 (c) $Obs_{\Omega_1}(\mathbf{G})$ (d) Ω_2 (e) $Obs_{\Omega_2}(\mathbf{G})$

Figure 6.1: Examples of sensor activation policies and observers

in Section 6.5 that K-codiagnosability, coobservability and coprognosability are all instances of decentralized distinguishability. Note that, if $X_A^{T_k} \cap X_B^{T_k} \neq \emptyset$ for some $T_k \in \mathbb{T}$, then $\mathbf{G}$ will not be decentralized distinguishable for any sensor activation policies $\bar{\Omega}$. This phenomenon may occur in the fault prognosis problem as we will discuss later in Section 6.5.3.

Example 6.2.1. *We consider the system $\boldsymbol{G}$ in Figure 6.1(a) and $\Sigma_{s,1} = \{o, a\}, \Sigma_{s,2} = \{o, b\}$ and $\Sigma_{o,1} = \Sigma_{o,2} = \emptyset$ are two sets of observable events. We assume that the observations are static, i.e., $\Omega_1 = \Omega_{\Sigma_{s,1}}$ and $\Omega_2 = \Omega_{\Sigma_{s,2}}$. Let us consider the following set of specifications $\mathbb{T} = \{T_1, T_2\}$, where*

$$T_1 = X_A^{T_1} \times X_B^{T_1} = \{6\} \times \{1, 2, 3, 5, 7\}$$

$$T_2 = X_A^{T_2} \times X_B^{T_2} = \{5, 7\} \times \{1, 2, 4, 6\}$$

and $\mathcal{I}_{T_1} = \mathcal{I}_{T_2} = \{1, 2\}$. We can verify that $\boldsymbol{G}$ is decentralized distinguishable w.r.t. $\{T_1, T_2\}$ and $[\Omega_{\Sigma_{s,1}}, \Omega_{\Sigma_{s,2}}]$. For example, for specification T_1 and string of_1a such that $\delta(of_1a) = 6 \in X_A^{T_1}$, we have $1 \in \mathcal{I}_{T_1}$ and $\mathcal{E}^{\boldsymbol{G}}_{\Omega_{\Sigma_{s,1}}}(of_1a) \cap X_B^{T_1} = \{6\} \cap \{1, 2, 3, 5, 7\} = \emptyset$. However, if we add another specification $T_3 = \{4\} \times \{1, 2\}$ to $\{T_1, T_2\}$, then $\boldsymbol{G}$ will not be decentralized distinguishable. For example, for $\delta(of_1) = 4 \in X_A^{T_3}$, we have $\mathcal{E}^{\boldsymbol{G}}_{\Omega_{\Sigma_{s,1}}}(of_1) \cap X_B^{T_3} = \{2, 4\} \cap \{1, 2\} \neq \emptyset$ and $\mathcal{E}^{\boldsymbol{G}}_{\Omega_{\Sigma_{s,2}}}(of_1) \cap X_B^{T_3} = \{2, 4, 6\} \cap \{1, 2\} \neq \emptyset$, i.e., none of the agents can distinguish specification T_3.

Remark 6.2.1. The state disambiguation problem and its sensor activation have been studied in the literature in the centralized setting; see, e.g., [22, 81, 102]. Compared to its centralized counterpart, the decentralized disambiguation problem has the following important difference. In the centralized setting, specification $X_A \times X_B$ and specification $X_B \times X_A$ are equivalent in the sense that if the system can distinguish state x_1 from state x_2, the it can also distinguish x_2 from x_1. However, it is not the case in the decentralized setting and we cannot swap X_A and X_B arbitrarily. One can easily verify that $\mathbf{G}$ is decentralized distinguishable w.r.t. $X_A \times X_B$ does not necessarily imply that it is decentralized distinguishable w.r.t. $X_B \times X_A$. Moreover, our procedure for solving the sensor activation problem in the decentralized setting is completely different from those in the centralized case.

6.2.2 Problem Formulation and Solution Overview

Let $\mathbb{T}$ be the set of specifications. Then the goal of the sensor activation problem is to find an *optimal* pair of sensor activation policies $\bar{\Omega} = [\Omega_1, \Omega_2]$ such that the system is decentralized distinguishable w.r.t. $\bar{\Omega}$ and $\mathbb{T}$. In this chapter, we consider the logical optimal criterion that is widely used in the literature [82, 101, 104]. Specifically, for any $\bar{\Omega} = [\Omega_1, \Omega_2]$ and $\bar{\Omega}' = [\Omega'_1, \Omega'_2]$, the inclusion $\bar{\Omega}' \subseteq \bar{\Omega}$ means that

$$\forall i \in \mathcal{I} : \Omega'_i \subseteq \Omega_i \tag{6.3}$$

and the strict inclusion $\bar{\Omega}' \subset \bar{\Omega}$ means that

$$[\bar{\Omega}' \subseteq \bar{\Omega}] \wedge [\exists i \in \mathcal{I} : \Omega'_i \subset \Omega_i] \tag{6.4}$$

We are now ready to formulate the problem of minimal sensor activation for decentralized state disambiguation.

Problem 6. *Let $\boldsymbol{G}$ be the system and $\mathbb{T} = \{T_1, \ldots, T_m\}$ be a set of specifications. For each agent $i \in \{1, 2\}$, let $\Sigma_{o,i} \subseteq \Sigma$ be the set of observable events. Find sensor* *activation policies* $\bar{\Omega}^* = [\Omega_1^*, \Omega_2^*]$ *such that:*

C1. $\boldsymbol{G}$ *is decentralized distinguishable w.r.t.* $\bar{\Omega}^*$ *and* $\mathbb{T}$*.*

C2. $\bar{\Omega}^*$ *is minimal, i.e., there does not exist another* $\bar{\Omega}' \subset \bar{\Omega}^*$ *that satisfies (C1).*

Remark 6.2.2. In [101, 104], "sub-optimal" solutions to two special cases of Problem 1, the decentralized control problem and the decentralized diagnosis problem, are provided, in the sense that the solutions found therein are minimal among all solutions over *given* finite restricted solution spaces. In principle, the solutions found in [101, 104] could be improved by employing finer partitions and repeating the optimization procedure. In this chapter, we are aiming for a *language-based* minimal solution, in the sense that the notion of strict inclusion of sensor activation policies is defined in terms of the strings in $\mathcal{L}(\mathbf{G})$ (see Equations (5.4) and (6.4)). In other words, we do not impose, a priori, any constraints on the solution space of each Ω_i. Hence, no better solution can be obtained by refining the state space of $\mathbf{G}$ and repeating the solution procedure. To the best of our knowledge, such a language-based optimal solution to the decentralized sensor activation problem has never been reported in the literature. Moreover, Problem 1 is more general than the problems studied in [101, 104].

Before we formally tackle Problem 6, let us first provide a brief overview of our solution approach. We adopt the person-by-person approach that has been widely used in decentralized optimization problems. Specifically, we decompose the decentralized minimization problem to a set of centralized constrained minimization problems and for each such problem, we only attempt to minimize one agent's sensor activation policy while the other one is fixed. However, the following questions arise. First, by taking the person-by-person approach, iterations involving minimization for each agent may be required in general, and such iterations may not terminate in a finite number of steps. We will show that in our particular problem such iterations are not required. This is due of the so-called *monotonicity property* that arises in dynamic sensor activation problems. The second question of interest is how to minimize the sensor activation policy of one agent when the policy of the other agent is fixed. This problem is different from the fully centralized minimization problem, since we should not only consider the information of the agent whose sensor activation policy we are minimizing, but we must also take into account the information available to the other agent, whose sensor activation policy is fixed. Therefore, the true information state for this minimization problem consist of (i) the knowledge of the agent whose sensor activation policy is being minimized; and (ii) *this agent's inference of the other agent's potential knowledge of the system based on that agent's own information.* To resolve this information dependency, we develop a novel approach by which we encode the second agent's knowledge into the system model. This is discussed in the next section.

6.3 Constrained Minimization Problem

In this section, we tackle problem of minimizing the sensor activation policy for one agent when the sensor activation policy of the other one is fixed. This problem is also referred to as the *centralized constrained minimization problem* herafter. Throughout this section, $i \in \{1, 2\}$ denotes the agent whose sensor activation policy is being minimized while $j \in \{1, 2\}, j \neq i$ denotes the other agent whose sensor activation policy is fixed.

6.3.1 Constrained Minimization Problem

Problem 7. *(Centralized Constrained Minimization Problem). Let* $\boldsymbol{G}$ *be the system and* $\mathbb{T} = \{T_1, \ldots, T_m\}$ *be a set of specifications. Let* $i, j \in \{1, 2\}, i \neq j$ *be two agents. Suppose that the sensor activation policy* Ω_j *for Agent* j *is fixed. Find a sensor activation policy* Ω_i *for Agent* i *such that:*

C1. $\boldsymbol{G}$ *is decentralized distinguishable w.r.t.* $[\Omega_1, \Omega_2]$ *and* $\mathbb{T}$*.*

C2. For any Ω_i' satisfying (C1), we have $\Omega_i' \not\subset \Omega_i$.

The above problem is different from both the centralized and decentralized minimization problems. In the centralized minimization problem, where only one agent is involved, to maintain distinguishability, we need to require that

$$\forall T_k \in \mathbb{T}, \forall s \in \mathcal{L}(\mathbf{G}) : (\mathcal{E}_{\Omega}^{\mathbf{G}}(s) \times \mathcal{E}_{\Omega}^{\mathbf{G}}(s)) \cap T_k = \emptyset$$

where Ω is the centralized sensor activation policy. In other words, the agent should always be able to distinguish states in $X_A^{T_k}$ from states in $X_B^{T_k}$ for any $T_k \in \mathbb{T}$. However, in the decentralized disambiguation problem, it is possible that there exists a string $s \in \mathcal{L}(\mathbf{G}), \delta(s) \in X_A^{T_k}$ such that $\mathcal{E}_{\Omega_i}^{\mathbf{G}}(s) \cap X_B^{T_k} \neq \emptyset$, but $\mathcal{E}_{\Omega_j}^{\mathbf{G}}(s) \cap X_B^{T_k} = \emptyset$, where $j \in \mathcal{I}_{T_k}$. Therefore, Agent j may "help" Agent i to resolve the ambiguity. In other words, to solve the constrained minimization problem for one agent, we must take the other agent's sensor activation policy into account.

6.3.2 Problem Reduction

Recall that, in Chapter IV, we have solved a general class of fully centralized sensor activation problems. Therefore, if we can reduce Problem 7 to Problem 5, then it means that Problem 7 can also be solved effectively and the solution will be finitely realizable. We now show that such a reduction is possible by using automata $\mathbf{V}$ and $\tilde{\mathbf{V}}$, which are defined next.

Let $\mathbf{G}$ be the system and Ω_j be the fixed sensor activation policy, where $\Omega_j = (R_j, L_j)$ and $R_j = (X_R^j, \Sigma, \delta_R^j, x_{0,R}^j)$. We define a new automaton

$$\mathbf{V} = (X_V, \delta_V, \Sigma_V, x_{0,V}) \tag{6.5}$$

where

- $X_V \subseteq X \times X_R^j \times X \times X_R^j$ is the set of states;
- $\Sigma_V = (\Sigma \cup \{\epsilon\}) \times (\Sigma \cup \{\epsilon\})$ is the set of events;
- $x_{0,V} = (x_0, x_{0,R}^j, x_0, x_{0,R}^j)$ is the initial state;

The transition function $\delta_V : X_V \times \Sigma_V \to X_V$ is defined by: for any (x_1, x_1^R, x_2, x_2^R) and $\sigma \in \Sigma$, the following transitions are defined:

- If $\sigma \in L_j(x_1^R)$ and $\sigma \in L_j(x_2^R)$, then

$$\delta_V((x_1, x_1^R, x_2, x_2^R), (\sigma, \sigma)) = (\delta(x_1, \sigma), \delta_R^j(x_1^R, \sigma), \delta(x_2, \sigma), \delta_R^j(x_2^R, \sigma))$$

- If $\sigma \in L_j(x_1^R)$ and $\sigma \notin L_j(x_2^R)$, then

$$\delta_V((x_1, x_1^R, x_2, x_2^R), (\epsilon, \sigma)) = (x_1, x_1^R, \delta(x_2, \sigma), \delta_R^j(x_2^R, \sigma))$$

- If $\sigma \notin L_j(x_1^R)$ and $\sigma \in L_j(x_2^R)$, then

$$\delta_V((x_1, x_1^R, x_2, x_2^R), (\sigma, \epsilon)) = (\delta(x_1, \sigma), \delta_R^j(x_1^R, \sigma), x_2, x_2^R)$$

- If $\sigma \notin L_j(x_1^R)$ and $\sigma \notin L_j(x_2^R)$, then

$$\delta_V((x_1, x_1^R, x_2, x_2^R), (\sigma, \epsilon)) = (\delta(x_1, \sigma), \delta_R^j(x_1^R, \sigma), x_2, x_2^R)$$
$$\delta_V((x_1, x_1^R, x_2, x_2^R), (\epsilon, \sigma)) = (x_1, x_1^R, \delta(x_2, \sigma), \delta_R^j(x_2^R, \sigma))$$

The above construction follows the well-known $\mathcal{M}$-machine (or twin-plant) construction that was originally used for the verification of (co)observability [33, 67, 99]; but we generalize it to the dynamic observation setting. Essentially, V tracks a pair of strings that look the same for Agent j under Ω_j. Specifically, the first two components are used to track a string in the original system and the last two components are used to track a string that looks the same as the first string. Since we are considering the dynamic observation setting, we also need to track states in the sensor activation policy in order to determine the set of monitored events; this is why the second (respectively, fourth) component always moves together with the first (respectively, third) component. Therefore, for any $(s_1, s_2) \in \mathcal{L}(\mathbf{V})$, we have that $P_{\Omega_j}(s_1) = P_{\Omega_j}(s_2)$. Similarly, for any $t, w \in \mathcal{L}(\mathbf{G})$ such that $P_{\Omega_j}(t) = P_{\Omega_j}(w)$, there exists $(s_1, s_2) \in \mathcal{L}(\mathbf{V})$ such that $s_1 = t$ and $s_2 = w$, i.e., state $(\delta(t), \delta_R^j(t), \delta(w), \delta_R^j(w))$ is reachable in $\mathbf{V}$.

Next, we modify $\mathbf{V}$ as follows. For each transition in $\mathbf{V}$,

- if the event is in the form of (σ, σ) or (σ, ϵ), then we replace the event by σ;
- if the event is in the form of (ϵ, σ), then we replace the event by ϵ.

We denote by $\tilde{\mathbf{V}} = (Q_{\tilde{V}}, \delta_{\tilde{V}}, \Sigma_{\tilde{V}}, x_{0,\tilde{V}})$ the modified automaton. Similar modification was also used in [127, 131] in the static observation setting for different purpose. Intuitively, $\tilde{\mathbf{V}}$ only keeps the first component of the event of each transition in V, since this part corresponds to the transition in the real system. Note that $\tilde{\mathbf{V}}$ is a non-deterministic automaton, since ϵ-transition is allowed. Therefore, $\delta_{\tilde{V}}(s)$ is the *set* of states that can be reached from $x_{0,\tilde{V}}$ via s.

The modified automaton $\tilde{\mathbf{V}}$ has the following properties. First, we have that $\mathcal{L}(\tilde{\mathbf{V}}) = \mathcal{L}(\mathbf{G})$. Clearly, $\mathcal{L}(\tilde{\mathbf{V}}) \subseteq \mathcal{L}(\mathbf{G})$ since a transition in $\tilde{\mathbf{V}}$ is defined only when the corresponding transition in $\mathbf{G}$ is defined. Also, for any string $s \in \mathcal{L}(\mathbf{G})$, we know that $(s, s) \in \mathcal{L}(\mathbf{V})$, which implies that $s \in \mathcal{L}(\tilde{\mathbf{V}})$. Second, for any $s \in \mathcal{L}(\tilde{\mathbf{V}}) = \mathcal{L}(\mathbf{G})$, we know that

$$\begin{aligned} \delta_{\tilde{V}}(s) =& \{(\delta(s), \delta_R^j(s), \delta(t), \delta_R^j(t)) \in X_{\tilde{V}} : (s, t) \in \mathcal{L}(V)\} \\ =& \{(\delta(s), \delta_R^j(s), \delta(t), \delta_R^j(t)) \in X_{\tilde{V}} : t \in \mathcal{L}(\mathbf{G}) \wedge P_{\Omega_j}(s) = P_{\Omega_j}(t)\} \end{aligned} \tag{6.6}$$

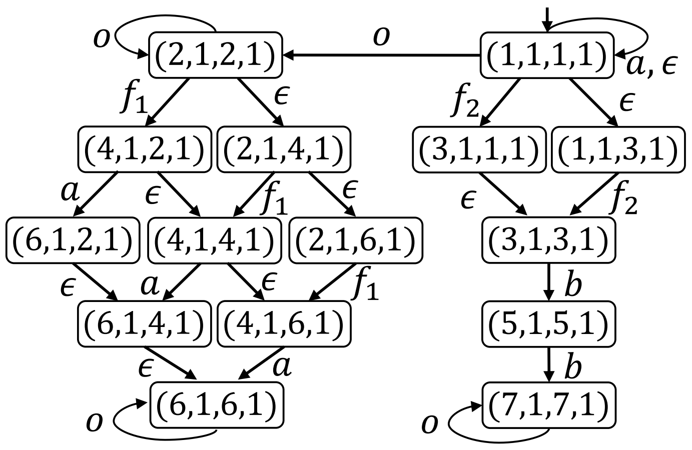


Figure 6.2: Automaton $\tilde{\mathbf{V}}$.

Therefore, for any string $s \in \mathcal{L}(\mathbf{G}) = \mathcal{L}(\tilde{\mathbf{V}})$, if $(x_1, x_1^R, x_2, x_2^R) \in \delta_{\tilde{V}}(s)$, then it implies that $\delta(s) = x_1$ and state x_2 cannot be distinguished from x_1 under Ω_j. For any $q \in 2^{X_{\tilde{V}}}$, we denote by $I_1(q) = \{x_1 \in X : (x_1, x_1^R, x_2, x_2^R) \in q\}$ the set of states in the first component of q. Then, for any sensor activation policy Ω, by Equation (6.6), we have $\mathcal{E}_{\Omega}^{\mathbf{G}}(s) = I_1(\mathcal{E}_{\Omega}^{\tilde{\mathbf{V}}}(s))$ for any $s \in \mathcal{L}(\mathbf{G})$.

Example 6.3.1. *Let us still consider the system **G** shown in Figure 6.1(a). Suppose that the fixed Ω_j is the sensor activation policy Ω_2 shown in Figure 6.1(d), i.e., Ω_j always monitors o and b. Then automaton $\tilde{\boldsymbol{V}}$ constructed from **G** and Ω_j is shown in Figure 6.2. Clearly, we see that $\mathcal{L}(\tilde{\boldsymbol{V}}) = \mathcal{L}(\boldsymbol{G})$. For string $of_1a \in \mathcal{L}(\tilde{\boldsymbol{V}}) = \mathcal{L}(\boldsymbol{G})$, we have that $\delta_{\tilde{V}}(of_1a) = \{(6,1,2,1),(6,1,4,1),(6,1,6,1)\}$ and $I_1(\mathcal{E}_{\Omega_j}^{\tilde{\boldsymbol{V}}}(of_1a)) = I_1(\{(2,1,2,1),(4,1,2,1),(6,1,2,1),(2,1,4,1),(4,1,4,1),(2,1,6,1),(4,1,6,1),$ $(6,1,6,1)\}) = \{2,4,6\} = \mathcal{E}_{\Omega_j}^{\boldsymbol{G}}(of_1a)$.*

Now, let us show how to use $\tilde{\mathbf{V}}$ to reduce the constraint minimization problem, i.e., Problem 7, to a fully centralized minimization problem, i.e., Problem 5. First, we define the distinguishability function $DF : 2^{X_{\tilde{V}}} \to \{0,1\}$ as follows: for each $q \in 2^{X_{\tilde{V}}}$,

$$DF(q) = \begin{cases} 1, & \text{if } \forall T_k \in \mathbb{T} : \text{(c-i) or (c-ii) holds} \\ 0, & \text{otherwise} \end{cases} \tag{6.7}$$

where conditions (c-i) and (c-ii) are defined by:

(c-i) $i \in \mathcal{I}_{T_k}$ and $(I_1(q) \times I_1(q)) \cap T_k = \emptyset$.

(c-ii) $j \in \mathcal{I}_{T_k}$ and $\forall x_1 \in I_1(q) \cap X_A^{T_k}, \forall (x_1, x_1^R, x_2, x_2^R) \in q : (x_1, x_2) \notin T_k$.

Let us explain the intuition of the above two conditions in function DF. Suppose that Ω_i is the sensor activation policy to be synthesized for Agent i. Let $s \in \mathcal{L}(\mathbf{G})$ be a string such that $\delta(s) \in X_A^{T_k}$, i.e., the coordinator must take the action associated to T_k when s is executed. Then $\mathcal{E}_{\Omega_i}^{\tilde{\mathbf{V}}}(s)$ is the state estimate w.r.t. the state space of $\tilde{\mathbf{V}}$ under Ω_i. Essentially, function DF evaluates whether or not decentralized distinguishability is fulfilled by checking whether or not $q := \mathcal{E}_{\Omega_i}^{\tilde{\mathbf{V}}}(s)$ satisfies conditions (c-i) and (c-ii), which can be interpreted as follows.

- If (c-i) holds, then we know that Agent i can contribute to the global decision associated to T_k, since $i \in \mathcal{I}_{T_k}$. Moreover, it can contribute the right decision since it knows for sure that the action associated to T_k has to be taken, since $(\mathcal{E}_{\Omega_i}^{\mathbf{G}}(s) \cap \mathcal{E}_{\Omega_i}^{\mathbf{G}}(s)) \cap T_k = \emptyset$. Therefore, the disambiguation requirement is fulfilled even without looking at Agent j.

- If (c-i) does not hold, then we know that either Agent i cannot contribute to to the global decision associated to T_k or Agent i cannot make a right decision due to states ambiguity, i.e., $\exists x_1, x_2 \in \mathcal{E}_{\Omega_i}^{\mathbf{G}}(s) = I_1(\mathcal{E}_{\Omega_i}^{\tilde{\mathbf{V}}}(s)) : (x_1, x_2) \in T_k$. In order to issue the right global decision, Agent j must be able to help Agent i to distinguish those ambiguous strings, i.e., condition (c-ii) needs to hold. First, Agent j should be able to contribute to the global decision associated to T_k, i.e., $j \in \mathcal{I}_{T_k}$. Then, for any string t that looks the same as s for Agent i and leads to a state in $X_A^{T_k}$, there should not exist another string w that looks the same as t for Agent j and leads to a state in $X_B^{T_k}$. Recall that $\tilde{\mathbf{V}}$ is constructed by tracking all states that cannot be distinguished from x_1 by Agent j. Therefore, Agent i can *infer* which states Agent j cannot distinguish by using $\tilde{\mathbf{V}}$. Specifically, if for any $(x_1, x_1^R, x_2, x_2^R) \in q : (x_1, x_2) \notin T_k$, then we know that there is no such a string w that can confuse Agent j for some string t, i.e., Agent j can make a right decision associated to T_k.

Finally, we would like to remark that, although specification $\mathbb{T}$ is defined over the state space of $\mathbf{G}$, the distinguishability function DF is defined over the state space of $\tilde{\mathbf{V}}$, i.e., we need to solve Problem 3 for the modified system $\tilde{\mathbf{V}}$. However, this is not a problem, since the first component of a state $\tilde{\mathbf{V}}$ exactly carries the same state information in $\mathbf{G}$, i.e., $I_1(\mathcal{E}^{\tilde{\mathbf{V}}}_{\Omega_i}(s)) = \mathcal{E}^{\mathbf{G}}_{\Omega_i}(s)$ for any Ω_i. Moreover, since $\mathcal{L}(\tilde{\mathbf{V}}) = \mathcal{L}(\mathbf{G})$, we know that $\tilde{\mathbf{V}}$ and $\mathbf{G}$ have the same observable behavior under any sensor activation policy. Therefore, we can first use $\tilde{\mathbf{V}}$ to synthesize a sensor activation policy and then use it to monitor $\mathbf{G}$.

We summarize the above discussions by the following theorem.

Theorem VI.1. *Let $\boldsymbol{G}$ be the system and $\mathbb{T} = \{T_1, \ldots, T_m\}$ be a set of specifications. Let $\tilde{\boldsymbol{V}}$ be the automaton constructed based on Ω_j. Then, $\boldsymbol{G}$ is decentralized distinguishable w.r.t. $[\Omega_1, \Omega_2]$ and $\mathbb{T}$ if and only if*

$$\forall s \in \mathcal{L}(\boldsymbol{G}) : DF(\mathcal{E}^{\tilde{\boldsymbol{V}}}_{\Omega_i}(s)) = 1 \tag{6.8}$$

Proof. ($\Leftarrow$) By contraposition. Suppose that $\mathcal{L}(\mathbf{G})$ is not decentralized distinguishable. Then we know that, there exists $T_k \in \mathbb{T}$, such that

$$(\exists s \in \mathcal{L}(\mathbf{G}) : \delta(s) \in X_A^{T_k})(\forall p \in \mathcal{I}_{T_k})[\mathcal{E}^{\mathbf{G}}_{\Omega_p}(s) \cap X_B^{T_k} \neq \emptyset] \tag{6.9}$$

Let us consider the following three cases for $\mathcal{I}_{T_k}$.

Case 1: $\mathcal{I}_{T_k} = \{i\}$.

Let us consider (c-i), since (c-ii) is violated directly. By Equation (6.9), since $\delta(s) \in$

$X_A^{T_k} \cap \mathcal{E}_{\Omega_i}^{\mathbf{G}}(s)$ and $X_B^{T_k} \cap \mathcal{E}_{\Omega_i}^{\mathbf{G}}(s) \neq \emptyset$, we know that

$$(I_1(\mathcal{E}_{\Omega_i}^{\tilde{\mathbf{V}}}(s)) \times I_1(\mathcal{E}_{\Omega_i}^{\tilde{\mathbf{V}}}(s))) \cap T_k = (\mathcal{E}_{\Omega_i}^{\mathbf{G}}(s) \times \mathcal{E}_{\Omega_i}^{\mathbf{G}}(s)) \cap T_k \neq \emptyset$$

Therefore, (c-i) is also violated and $DF(\mathcal{E}_{\Omega_i}^{\tilde{\mathbf{V}}}(s)) = 0$.

Case 2: $\mathcal{I}_{T_k} = \{j\}$.

Let us consider (c-ii), since (c-i) is violated directly. We still consider string s in Equation (6.9). We have $\delta(s) \in \mathcal{E}_{\Omega_i}^{\mathbf{G}}(s) \cap X_A^{T_k} = I_1(\mathcal{E}_{\Omega_i}^{\tilde{\mathbf{V}}}(s)) \cap X_A^{T_k}$. Since $\mathcal{E}_{\Omega_j}^{\mathbf{G}}(s) \cap X_B^{T_k} \neq \emptyset$, we know that there exists a string $t \in \mathcal{L}(\mathbf{G})$ such that $P_{\Omega_j}(s) = P_{\Omega_j}(t)$ and $\delta(t) \in X_B^{T_k}$. This implies that $(\delta(s), \delta(t)) \in T_k$. Since $P_{\Omega_j}(s) = P_{\Omega_j}(t)$, by the construction of $\tilde{\mathbf{V}}$, we know that $(\delta(s), \delta_R^j(s), \delta(t), \delta_R^j(t)) \in \delta_{\tilde{V}}(s) \subseteq \mathcal{E}_{\Omega_i}^{\tilde{\mathbf{V}}}(s)$. Therefore, we know that (c-ii) is also violated and $DF(\mathcal{E}_{\Omega_i}^{\tilde{\mathbf{V}}}(s)) = 0$.

Case 3: $\mathcal{I}_{T_k} = \{1, 2\}$.

For string s in Equation (6.9), since $\mathcal{E}_{\Omega_i}^{\mathbf{G}}(s) \cap X_B^{T_k} \neq \emptyset$, by the same argument in Case 1, we know that (c-i) does not hold. Also, since $\mathcal{E}_{\Omega_j}^{\mathbf{G}}(s) \cap X_B^{T_k} \neq \emptyset$, by the same argument in Case 2, we know that (c-ii) also does not hold. Therefore, $DF(\mathcal{E}_{\Omega_i}^{\tilde{\mathbf{V}}}(s)) = 1$.

Overall, for each case, $\exists s \in \mathcal{L}(\mathbf{G}) : DF(\mathcal{E}_{\Omega_i}^{\tilde{\mathbf{V}}}(s)) = 1$, which completes the contrapositive proof.

($\Rightarrow$) Still by contrapositive. Suppose that $\exists s \in \mathcal{L}(\mathbf{G}) : DF(\mathcal{E}_{\Omega_i}^{\tilde{\mathbf{V}}}(s)) = 0$. Then we know that, there exists $T_k \in \mathbb{T}$ such that none of (c-i) and (c-ii) holds for $\mathcal{E}_{\Omega_i}^{\tilde{\mathbf{V}}}(s)$. Next, we still consider the following three cases for $\mathcal{I}_{T_k}$.

Case 1: $\mathcal{I}_{T_k} = \{i\}$.

Since (c-i) does not hold, we know that $(\mathcal{E}_{\Omega_i}^{\mathbf{G}}(s) \times \mathcal{E}_{\Omega_i}^{\mathbf{G}}(s)) \cap T_k = (I_1(\mathcal{E}_{\Omega_i}^{\tilde{\mathbf{V}}}(s)) \times I_1(\mathcal{E}_{\Omega_i}^{\tilde{\mathbf{V}}}(s))) \cap T_k \neq \emptyset$. This implies that $\exists w \in \mathcal{L}(\mathbf{G})$ such that $\delta(w) \in X_A^{T_k}$, $P_{\Omega_i}(s) = P_{\Omega_i}(w)$ and $\mathcal{E}_{\Omega_i}^{\mathbf{G}}(w) \cap X_B^{T_k} = \mathcal{E}_{\Omega_i}^{\mathbf{G}}(s) \cap X_B^{T_k} \neq \emptyset$. Therefore, for $T_k \in \mathbb{T}$, we have $(\exists w \in \mathcal{L}(\mathbf{G}) : \delta(t) \in X_A^{T_k})[\mathcal{E}_{\Omega_i}^{\mathbf{G}}(w) \cap X_B^{T_k} \neq \emptyset]$, i.e., $\mathbf{G}$ is not decentralized distinguishable.

Case 2: $\mathcal{I}_{T_k} = \{j\}$.

Since (c-ii) does not hold, we know that

$$\exists x_1 \in \mathcal{E}^{\mathbf{G}}_{\Omega_i}(s) \cap X^{T_k}_A, \exists (x_1, x^R_1, x_2, x^R_2) \in \mathcal{E}^{\tilde{\mathbf{V}}}_{\Omega_i}(s) : (x_1, x_2) \in T_k$$

Since $(x_1, x^R_1, x_2, x^R_2) \in \mathcal{E}^{\tilde{\mathbf{V}}}_{\Omega_i}(s)$, we know that there exists a string $t \in \mathcal{L}(\tilde{\mathbf{V}}) = \mathcal{L}(\mathbf{G})$, such that $P_{\Omega_i}(s) = P_{\Omega_i}(t)$ and $(x_1, x^R_1, x_2, x^R_2) \in \delta_{\tilde{V}}(t)$, which further implies that $x_1 = \delta(t)$ and there exists $w \in \mathcal{L}(\mathbf{G})$ such that $x_2 = \delta(w)$ and $P_{\Omega_j}(t) = P_{\Omega_j}(w)$. Therefore, $\{x_1, x_2\} \subseteq \mathcal{E}^{\mathbf{G}}_{\Omega_j}(t) = \mathcal{E}^{\mathbf{G}}_{\Omega_j}(w)$. Since $(x_1, x_2) \in T_k$, we know that $x_1 \in X^{T_k}_A$ and $x_2 \in X^{T_k}_B$. Overall, for $T_k \in \mathbb{T}$, we have $(\exists t \in \mathcal{L}(\mathbf{G}) : \delta(t) \in X^{T_k}_A)[\mathcal{E}^{\mathbf{G}}_{\Omega_j}(t) \cap X^{T_k}_B \neq \emptyset]$, i.e., $\mathbf{G}$ is not decentralized distinguishable.

Case 3: $\mathcal{I}_{T_k} = \{1, 2\}$.

Since (c-ii) does not hold, by the same argument in Case 2, we know that there exists $\exists t \in \mathcal{L}(\mathbf{G})$ such that $P_{\Omega_i}(s) = P_{\Omega_i}(t)$, $\delta(t) \in X^{T_k}_A$ and $\mathcal{E}^{\mathbf{G}}_{\Omega_j}(t) \cap X^{T_k}_B \neq \emptyset$. Since (c-i) does not hold, by the same argument in Case 1, we know that $\mathcal{E}^{\mathbf{G}}_{\Omega_i}(s) \cap X^{T_k}_B \neq \emptyset$. Since $P_{\Omega_i}(s) = P_{\Omega_i}(t)$, we have $\mathcal{E}^{\mathbf{G}}_{\Omega_i}(t) \cap X^{T_k}_B \neq \emptyset$. Therefore, we know that

$$(\exists t \in \mathcal{L}(\mathbf{G}) : \delta(t) \in X^{T_k}_A)(\forall p \in \mathcal{I}_{T_k})[\mathcal{E}^{\mathbf{G}}_{\Omega_p}(t) \cap X^{T_k}_B \neq \emptyset]$$

i.e., $\mathbf{G}$ is not decentralized distinguishable.

Overall, $\mathbf{G}$ is not decentralized distinguishable for each case. This completes the contrapositive proof. $\square$

In the above development, the essence of using $\tilde{\mathbf{V}}$ is that we can encode Agent j's information, i.e., Ω_j, into the system model in order to reduce the constrained minimization problem for Agent i to a fully centralized minimization problem. That is, $\tilde{\mathbf{V}}$ is a non-deterministic refinement of $\mathbf{G}$ that carries both the original state information in $\mathbf{G}$ and some useful information of Ω_j. Once $\tilde{\mathbf{V}}$ is constructed, we will not use Ω_j anymore, since all useful information, i.e., which pairs of states Agent j cannot distinguish, has been encoded in $\tilde{\mathbf{V}}$. Finally, using Theorem VI.1, we have the following result.

Theorem VI.2. *Problem 7 can be effectively solved.*

Proof. By Theorem VI.1, it is clear that Problem 7 is a special case of Problem 5 by considering system $\tilde{\mathbf{V}}$ and setting φ to be $DF : 2^{X_{\tilde{V}}} \to \{0,1\}$. Since Problem 5 can be effectively solved, Problem 7 can also be effectively solved. $\square$

Example 6.3.2. *We return to the system $\boldsymbol{G}$ in Figure 6.1(a) with $\Sigma_{s,1} = \{o, a\}$ and $\Sigma_{s,2} = \{o, b\}$. We still consider specifications $\mathbb{T} = \{T_1, T_2\}$ defined in Example 6.2.1. We assume that sensor activation policy Ω_2 shown in Figure 6.1(d) is fixed for Agent 2 and the corresponding automaton $\tilde{\boldsymbol{V}}$ has been shown in Figure 6.2. Now, we want to synthesize sensor activation policy Ω_1 such that $\boldsymbol{G}$ is decentralized distinguishable. By defining function DF for $\tilde{\boldsymbol{V}}$ and applying the synthesis algorithm in [112], we obtain a minimal sensor activation policy Ω_1^* shown in Figure 6.3(a).*

For example, for specification T_1, we consider string of_1a such that $\delta(of_1a) = 6 \in X_A^{T_k}$. Then we have $q = \mathcal{E}_{\Omega_1^*}^{\tilde{\boldsymbol{V}}}(of_1a) = \{(6,1,6,1)\}$, i.e., $I_1(q) = \{6\}$. Therefore, condition (c-i) holds for q and we have $DF(q) = 1$. For specification T_2, let us consider string f_2b such that $\delta(f_2b) = 5 \in X_A^{T_2}$. Then we have $q = \mathcal{E}_{\Omega_1^*}^{\tilde{\boldsymbol{V}}}(f_2b) = \{(1,1,1,1),(3,1,1,1,),(1,1,3,1),(3,1,3,1),(5,1,5,1),(7,1,7,1)\}$, i.e., $I_1(q) = \{1,3,5,7\}$. For this case, condition (c-i) does not hold for q since $1 \in I_1(q) \cap X_B^{T_2}$. However, for $5 \in I_1(q) \cap X_A^{T_2} = \{5,7\}$, $(5,1,5,1)$ is the only state in q whose first component is 5 and $(5,5) \notin T_2$. Similarly, for $7 \in I_1(q) \cap X_A^{T_2}$, $(7,1,7,1)$ is the only state in q whose first component is 7 and $(7,7) \notin T_2$. Therefore, condition (c-ii) holds and we still have $DF(q) = 1$.

6.4 Synthesis Algorithm

In this section, we first present an algorithm that solves the decentralized sensor activation problem by using the results we developed so far. Then we prove the

Algorithm 9: D-MIN-ACT

```
input  : $G, \mathbb{T}, \Sigma_{s,1}, \Sigma_{o,1}, \Sigma_{s,2}, \Sigma_{o,2}$
output: $\bar{\Omega}^*$
1  $\Omega_1^* \leftarrow \Omega_{\Sigma_{o,1}\cup\Sigma_{s,1}}$ and $\Omega_2^* \leftarrow \Omega_{\Sigma_{o,2}\cup\Sigma_{s,2}}$
2  for $i \in \{1,2\}$ do
3      $j \in \{1,2\} \setminus \{i\}$
4      Fix $\Omega_j^*$. Construct automaton $\tilde{\mathbf{V}}$ w.r.t. $\Omega_j^*$ and define function $DF$.
5      Obtain minimal $\Omega_i'$ by solving Problem 5 w.r.t. system $\tilde{\mathbf{V}}$ and function $DF$.
6      $\Omega_i^* \leftarrow \Omega_i'$.
7  $\bar{\Omega}^* \leftarrow [\Omega_1^*, \Omega_2^*]$
```

correctness of the algorithm.

Our synthesis algorithm is formally presented in Algorithm D-MIN-ACT. Essentially, Algorithm D-MIN-ACT solves two centralized constrained minimization problems. First, we set Agent 2's sensor activation policy to be $\Omega_{\Sigma_{o,2}} \cup \Sigma_{s,2}$, i.e., the most conservative one, and solve the constrained minimization problem for Agent 1. Then we fix the obtained sensor activation policy for Agent 1 and solve the constrained minimization problem for Agent 2. However, the following question arises: *"After the above procedure, do we need to fix Agent 2's new sensor activation policy and go back to minimize Agent 1's sensor activation policy again?"* In other words, we need to answer whether or not iterations between two centralized constrained minimization problems are required in order to obtain a decentralized minimal solution. Hereafter, we show that such iterations are not necessary for our problem and Algorithm D-MIN-ACT indeed yields a decentralized minimal solution in the above two steps. This is because of the following monotonicity property.

Theorem VI.1. *(Monotonicity Property). Let $\boldsymbol{G}$ be the system, $\mathbb{T}$ be a set of specifications and $\bar{\Omega} = [\Omega_1, \Omega_2]$ and $\bar{\Omega}' = [\Omega_1', \Omega_2']$ be two sensor activation policies such that $\bar{\Omega}' \subseteq \bar{\Omega}$. Then $\boldsymbol{G}$ decentralized distinguishable w.r.t. $\bar{\Omega}'$ and $\mathbb{T}$ implies that $\boldsymbol{G}$ is decentralized distinguishable w.r.t. $\bar{\Omega}$ and $\mathbb{T}$.*

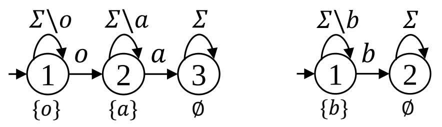
Proof. By contradition. Assume that $\mathbf{G}$ is not decentralized distinguishable w.r.t. $\bar{\Omega}$, i.e., $\exists T_k \in \mathbb{T}, \exists s \in \mathcal{L}(\mathbf{G}) : \delta(s) \in X_A^{T_k}$ such that $\forall i \in \{1,2\} : \mathcal{E}_{\Omega_i}^{\mathbf{G}}(s) \cap X_B^{T_k} \neq \emptyset$. Since $\bar{\Omega}' \subseteq \bar{\Omega}$, we know that $\forall i \in \{1,2\} : \Omega_i' \subseteq \Omega_i$, which implies that $\mathcal{E}_{\Omega_i}^{\mathbf{G}}(s) \subseteq \mathcal{E}_{\Omega_i'}^{\mathbf{G}}(s)$ for any $s \in \mathcal{L}(\mathbf{G})$. Therefore, for the same T_k and s, we also have that $\forall i \in \{1,2\} : \mathcal{E}_{\Omega_i'}^{\mathbf{G}}(s) \cap X_B^{T_k} \neq \emptyset$. However, this contradicts the fact that $\mathbf{G}$ is decentralized distinguishable w.r.t. $\bar{\Omega}'$. Therefore, $\mathbf{G}$ must be decentralized distinguishable w.r.t. $\bar{\Omega}$. □

We are now ready to prove the correctness of Algorithm D-MIN-ACT.

Theorem VI.2. *Let $\bar{\Omega}^*$ be the output of Algorithm D-MIN-ACT. Then $\bar{\Omega}^*$ solves Problem 6.*

Proof. It is clear that $\mathbf{G}$ is decentralized distinguishable w.r.t. $\bar{\Omega}^*$ and $\mathbb{T}$, since decentralized distinguishability is guaranteed in each centralized constrained minimization problem. It remains to show that $\bar{\Omega}^*$ is minimal; we proceed by contradiction. Let us assume that there exists another sensor activation policy $\bar{\Omega}' = [\Omega_1', \Omega_2']$ such that $\mathbf{G}$ is decentralized distinguishable w.r.t. $\bar{\Omega}'$ and $\mathbb{T}$; and (ii) $\bar{\Omega}' \subset \bar{\Omega}^*$. The second condition means that $\exists i, j \in \{1,2\}, i \neq j$ such that (iii) $\Omega_i' \subset \Omega_i^*$; and (iv) $\Omega_j' \subseteq \Omega_j^*$. Suppose that $i=1$ and $j=2$. Then we know that Ω_1^* is obtained by fixing Agent 2's sensor activation policy to be $\Omega_{\Sigma_{o,2} \cup \Sigma_{s,2}}$, where $\Omega_2' \subseteq \Omega_2^* \subseteq \Omega_{\Sigma_{o,2} \cup \Sigma_{s,2}}$. By Theorem VI.1, we know that $\mathbf{G}$ is decentralized distinguishable w.r.t. $[\Omega_1', \Omega_2']$ implies that $\mathbf{G}$ is decentralized distinguishable w.r.t. $[\Omega_1', \Omega_{\Sigma_{o,2} \cup \Sigma_{s,2}}]$. However, since $\Omega_1' \subset \Omega_1^*$, this contradicts to the fact that Ω_1^* is a solution to Problem 7. Similarly, suppose that $i=2$ and $j=1$. Then we know that Ω_2^* is obtained by fixing Agent 1's sensor activation policy to be Ω_1^*, where $\Omega_1' \subseteq \Omega_1^*$. By Theorem VI.1, we know that $\mathbf{G}$ is decentralized distinguishable w.r.t. $[\Omega_1', \Omega_2']$ implies that $\mathbf{G}$ is decentralized distinguishable w.r.t. $[\Omega_1^*, \Omega_2']$. However, since $\Omega_2' \subset \Omega_2^*$, it again contradicts the fact that Ω_2^* is a solution to Problem 7. □

We illustrate Algorithm D-MIN-ACT by an example.



(a) Ω_1^* (b) Ω_2^*

Figure 6.3: Decentralized minimal solutions

Example 6.4.1. *Again, consider the system $\boldsymbol{G}$ in Figure 6.1(a) and specifications $\mathbb{T} = \{T_1, T_2\}$ defined in Example 6.2.1. Let $\Sigma_{s,1} = \{o, a\}$ and $\Sigma_{s,2} = \{s, b\}$, respectively, be the set of observable events for Agent 1 and Agent 2. Initially, we set $\Omega_2 = \Omega_{\Sigma_{s,2}}$ and solve the constrained minimization problem for Agent 1; this has been solved in Example 6.3.2 and we obtained Ω_1^* shown in Figure 6.3(a). Next, we fix Ω_1^* for Agent 1 and solve the constrained minimization problem for Agent 2. Then we obtain the sensor activation policy Ω_2^* as shown in Figure 6.3(b). We see that Ω_2^* turns all sensors off after b is observed, since once b occurs, Agent 2 will know for sure that the system is in state* 5 *or* 7 *and there is no need to monitor any event. Therefore, $[\Omega_1^*, \Omega_2^*]$ is a minimal pair of sensor activation policies that ensure decentralized distinguishability.*

Remark 6.4.1. In general, the minimal solution to Problem 6 is not unique due to the following reasons. First, for each centralized constraint minimization problem involved in Algorithm D-MIN-ACT, the minimal solution is not unique in general [112]. There may exist two incomparable centralized minimal solutions to Problem 2 or 3. Second, the decentralized minimal solution obtained by Algorithm D-MIN-ACT also depends on the order of the centralized constraint minimization problems. In general, fixing Agent 1 first and fixing Agent 2 first may result in different minimal solutions. However, in any case, solution $\bar{\Omega}^*$ returned by Algorithm D-MIN-ACT is *guaranteed* to be minimal in the sense that other minimal solutions must be *incomparable* with $\bar{\Omega}^*$.

Remark 6.4.2. We conclude this section by discussing the complexity of synthesis algorithm. Suppose that we first fix Agent 2. Initially, $\Omega_2 = \Omega_{\Sigma_{s,2}\cup\Sigma_{o,2}}$ and its automaton only contains a single state. To solve the constraint optimization problem when Ω_2 is fixed, first, we need to construct $\tilde{\mathbf{V}}$, which is polynomial in the size of $\mathbf{G}$ and Ω_2. However, since an observer-like constructed is exploited, the algorithm in [112] requires exponential complexity w.r.t the size of the system, i.e., $\tilde{\mathbf{V}}$, and the size of the solution Ω_1^* is also exponential in the size of $\tilde{\mathbf{V}}$. Again, constructing $\tilde{\mathbf{V}}$ when Agent 1 is fixed only requires polynomial complexity w.r.t. Ω_1^*, but synthesizing Ω_2^* requires exponential complexity again. Therefore, the overall complexity is doubly-exponential w.r.t. the size of $\mathbf{G}$. Such a doubly-exponential complexity arises in many synthesis problems where two *incomparable* observations are involved; see, e.g., [26, 59].

6.5 Application of the Decentralized State Disambiguation Problem

In this section, we show that the notions of K-codiagnosability, coobservability and coprognosability are instances of decentralized distinguishability. Therefore, the proposed framework is applicable for solving the dynamic sensor activation problems for the purposes of decentralized fault diagnosis, decentralized control and decentralized fault prognosis.

6.5.1 Decentralized Fault Diagnosis

In the decentralized fault diagnosis problem, the local agents need to work as a team such that any fault be diagnosed within a bounded number of steps. Formally, we denote by $\Sigma_F \subseteq \Sigma_{uo}$ the set of fault events. We assume that Σ_F is partitioned into m *fault types*: $\Sigma_F = \Sigma_{F_1}\dot{\cup}\dots\dot{\cup}\Sigma_{F_m}$; we denote by Π the partition and by $\mathcal{F} = \{1,\dots,m\}$ the index set of the fault types. For any $k \in \mathcal{F}$, we define $\Psi(E_{F_k}) = \{sf \in \mathcal{L}(\mathbf{G}) : f \in E_{F_k}\}$ to be the set of strings that end with a fault event of type

k. We write $E_{F_k} \in s$, if $\overline{\{s\}} \cap \Psi(E_{F_k}) \neq \emptyset$. The notion of K-codiagnosability was proposed in the literature to capture whether or not any fault can be diagnosed within K steps [24, 62].

Definition 6.5.1. *(K-Codiagnosability). Let $K \in \mathbb{N}$. We say that live language $\mathcal{L}(\boldsymbol{G})$ is K-codiagnosable w.r.t. $\bar{\Omega}$, Σ_F and Π if*

$$\begin{aligned} &(\forall k \in \mathcal{F})(\forall s \in \Psi(\Sigma_{F_k}))(\forall t \in \mathcal{L}(\boldsymbol{G})/s : |t| \geq K) \\ &(\exists i \in \{1,2\})(\forall w \in \mathcal{L}(\boldsymbol{G}))[P_{\Omega_i}(w) = P_{\Omega_i}(st) \Rightarrow \Sigma_{F_k} \in w]. \end{aligned} \tag{6.10}$$

To show that K-codiagnosability can be formulated as decentralized distinguishability, following similar constructions in [22, 111], we first refine the state space of $\mathbf{G}$ by defining a new automaton $\tilde{\mathbf{G}} = (\tilde{X}, \Sigma, \tilde{\delta}, \tilde{x}_0)$, where $\tilde{X} \subseteq X \times \{-1, 0, 1 \ldots, K\}^m$, $\tilde{x}_0 = (x_0, -1, \ldots, -1)$ and the partial transition function $\tilde{\delta} : \tilde{X} \times \Sigma \to \tilde{X}$ is defined by: for any $(x, n_1, \ldots, n_m) \in \tilde{X}$ and $\sigma \in \Sigma$, we have

$$\tilde{\delta}((x, n_1, \ldots, n_m), \sigma) = (\delta(x, \sigma), n_1 + \Delta_1, \ldots, n_m + \Delta_m)$$

where for each $i \in \{1, \ldots, m\}$, Δ_i is defined by

$$\Delta_i = \begin{cases} 0, & \text{if } [n_i = K] \text{ or } [n_i = -1 \wedge \sigma \notin \Sigma_{F_i}] \\ 1 & \text{if } [0 \leq n_i < K] \text{ or } [n_i = -1 \wedge \sigma \in \Sigma_{F_i}] \end{cases}$$

Intuitively, $\tilde{\mathbf{G}}$ simply unfolds $\mathbf{G}$ by "counting" the number of steps since each type of fault has occurred. Since $\mathcal{L}(\tilde{\mathbf{G}}) = \mathcal{L}(\mathbf{G})$, we can synthesize a sensor activation policy for $\mathbf{G}$ based on $\tilde{\mathbf{G}}$. For any state $\tilde{x} = (x, n_1, \ldots, n_m) \in \tilde{\mathbf{G}}$, we denote by $[\tilde{x}]_i$ its $(i+1)^{\text{th}}$ component, i.e., n_i.

Based on $\tilde{\mathbf{G}}$, we define a set of specifications $\mathbb{T}_{diag} = \{T_1, T_2, \ldots, T_m\}$ as follows:

for each $T_k \in \mathbb{T}$, we have

$$X_A^{T_k} = \{x \in \tilde{X} : [x]_k = K\} \text{ and } X_B^{T_k} = \{x \in \tilde{X} : [x]_k = -1\}$$

The following result reveals that, to enforce K-codiagnosability, it suffices to enforce decentralized distinguishability for $\mathbb{T}_{diag}$.

Theorem VI.1. *A live language $\mathcal{L}(\boldsymbol{G})$ is K-codiagnosable w.r.t. $\bar{\Omega}$, Σ_F and Π, if and only if, $\tilde{\boldsymbol{G}}$ is decentralized distinguishable w.r.t. $\bar{\Omega}$ and $\mathbb{T}_{diag}$.*

Proof. ($\Rightarrow$) By contraposition. Suppose that $\tilde{\mathbf{G}}$ is not decentralized distinguishable. Then we know that there exist $k \in \{1, \ldots, m\}$ and a string $s \in \mathcal{L}(\mathbf{G})$ such that $x := \tilde{\delta}(s) \in X_A^{T_k}$ and for each $i \in \{1, 2\}$, there exists $x_i \in \mathcal{E}_{\Omega_i}^{\tilde{\mathbf{G}}}(s)$ such that $x_i \in X_B^{T_k}$. Then we know that, for each $i \in \{1, 2\}$, there exists a string $s_i \in \mathcal{L}(\mathbf{G})$ such that $\tilde{\delta}(s_i) = x_i$ and $P_{\Omega_i}(s) = P_{\Omega_i}(s_i)$. By the definition of T_k, $x \in X_A^{T_k}$ implies that $[x]_k = K$. According to the construction of $\tilde{\mathbf{G}}$, $\tilde{\delta}(s) = x$ implies that we can write $s = uv$ such that $u \in \Psi(\Sigma_{F_k})$ and $|v| \geq K$. For each $i \in \{1, 2\}$, since $x_i \in X_B^{T_k}$, we know that $[x_i]_k = -1$, which implies that $\Sigma_{F_k} \notin s_i$. Overall, we know that

$$\begin{aligned} &(\exists k \in \mathcal{F})(\exists u \in \Psi(\Sigma_{F_k}))(\exists v \in \mathcal{L}(\mathbf{G})/u : |v| \geq K) \\ &(\forall i \in \{1, 2\})(\exists s_i \in \mathcal{L}(\mathbf{G}))[P_{\Omega_i}(uv) = P_{\Omega_i}(s_i) \wedge \Sigma_{F_i} \notin s_i] \end{aligned} \tag{6.11}$$

i.e., $\mathcal{L}(\mathbf{G})$ is not K-codiagnosable.

($\Rightarrow$) Still by contraposition. Suppose that $\tilde{\mathbf{G}}$ is not K-codiagnosable, i.e., Equation (6.11) holds. Let $x := \tilde{\delta}(uv)$, $x_1 := \tilde{\delta}(s_1)$ and $x_2 := \tilde{\delta}(s_2)$. Then, according to the definition of $\tilde{\mathbf{G}}$, we know that $[x]_k = K$, $[x_1]_k = [x_2]_k = -1$, which implies that $x \in X_A^{T_k}$ and $x_1, x_2 \in X_B^{T_k}$. Moreover, since for each $i = 1, 2$, $P_{\Omega_i}(uv) = P_{\Omega_i}(s_i)$, we know that $x_i \in \mathcal{E}_{\Omega_i}^{\tilde{\mathbf{G}}}(s_i) = \mathcal{E}_{\Omega_i}^{\tilde{\mathbf{G}}}(uv)$, i.e., $\mathcal{E}_{\Omega_i}^{\tilde{\mathbf{G}}}(uv) \cap X_B^{T_k} \neq \emptyset$. Overall, we know that $(\exists T_k \in \mathbb{T})(\exists uv \in \mathcal{L}(\tilde{\mathbf{G}}) : \tilde{\delta}(uv) \in X_A^{T_k})(\forall i \in \{1, 2\})[\mathcal{E}_{\Omega_i}^{\tilde{\mathbf{G}}}(uv) \cap X_B^{T_k} \neq \emptyset]$, i.e., $\tilde{\mathbf{G}}$ is not

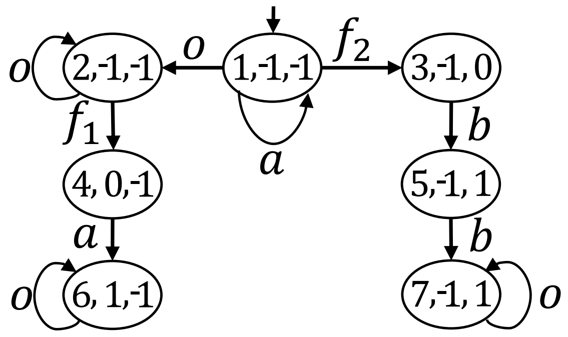


Figure 6.4: Augmented system $\tilde{\mathbf{G}}$

decentralized distinguishable w.r.t. $\mathbb{T}_{diag}$. □

Example 6.5.1. *Let us consider again system $\boldsymbol{G}$ shown in Figure 6.1(a). Suppose that $\Sigma_F = \Sigma_{F_1} \dot{\cup} \Sigma_{F_2} = \{f_1\} \dot{\cup} \{f_2\}$. Let us consider $K = 1$. Then the refined automaton $\tilde{\boldsymbol{G}}$ is shown in Figure 6.4. For example, state $x = (6, 1, -1)$ means that: (i) the system is at state 6 in $\boldsymbol{G}$; (ii) f_1 has occurred for more than one step (since $[x]_1 = K$); and (iii) f_2 has not occurred (since $[x]_2 = -1$). Then $\mathbb{T}_{diag} = \{T_1, T_2\}$ is defined by $T_1 = \{(6, 1, -1)\} \times \{(1, -1, -1), (2, -1, -1), (3, -1, 0), (5, -1, 1), (7, -1, 1)\}$ and $T_2 = \{(5, -1, 1), (7, -1, 1)\} \times \{(1, -1, -1), (2, -1, -1), (4, 0, -1), (6, 1, -1)\}$. Since $\tilde{\boldsymbol{G}}$ and $\boldsymbol{G}$ are isomorphic for this specific example, we see that $\mathbb{T}_{diag}$ is indeed the same specification $\mathbb{T}$ defined in Example 6.2.1. Therefore, the solution we obtained in Example 6.4.1 has solved the sensor activation problem for 1-codiagnosability.*

6.5.2 Decentralized Supervisory Control

Another important decentralized decision-making problem is the decentralized supervisory control problem [69, 128]. In this problem, each local agent $i \in \mathcal{I}$ can disable events in $\Sigma_{c,i} \subseteq \Sigma$ dynamically based on its local observation Ω_i. We define $\Sigma_c = \cup_{i \in \mathcal{I}} \Sigma_{c,i}$ as the set of all controllable events and for each $\sigma \in \Sigma_c$, we define $\mathcal{I}^c(\sigma) = \{i \in \mathcal{I} : \sigma \in \Sigma_{c,i}\}$ as the set of agents that can disable σ. The control objective is to make sure that the closed-loop system achieves a desired language

$\mathcal{L}(\mathbf{H}) \subseteq \mathcal{L}(\mathbf{G})$. The key property regarding the decentralized information in this problem is the notion of *coobservability*; it together with the notion of controllability provide the necessary and sufficient conditions for exactly achieving a given specification language. We recall its definition from [69].

Definition 6.5.2. *(Coobservability). We say that $\mathcal{L}(\boldsymbol{G})$ is coobservable w.r.t. $\mathcal{L}(\boldsymbol{H})$, $\Sigma_{c,1}, \Sigma_{c,2}$ and $\bar{\Omega}$ if*

$$(\forall s \in \mathcal{L}(\boldsymbol{H}))(\forall \sigma \in \Sigma_c : s\sigma \in \mathcal{L}(\boldsymbol{G}) \setminus \mathcal{L}(\boldsymbol{H}))(\exists i \in \mathcal{I}^c(\sigma_k))[P_{\Omega_i}^{-1}(P_{\Omega_i}(s))\{\sigma\} \cap \mathcal{L}(\boldsymbol{H}) = \emptyset]$$

Hereafter, we assume w.o.l.g. that $H = (X_H, \Sigma, \delta_H, x_{0,H})$ is a strict sub-automaton of $\mathbf{G}$. Now, suppose that $\Sigma_c = \{\sigma_1, \ldots, \sigma_m\}$ is the set of controllable events. We define a set of specifications $\mathbb{T}_{cont} = \{T_1, T_2, \ldots, T_m\}$ as follows: for each $T_k \in \mathbb{T}$, we have

$$\begin{aligned} X_A^{T_k} =& \{x \in X_H : \delta(q, \sigma_k)! \wedge \delta_H(q, \sigma_k)\neg!\} \\ X_B^{T_k} =& \{x \in X_H : \delta_H(q, \sigma_k)!\} \end{aligned}$$

with $\mathcal{I}_{T_k} = \mathcal{I}^c(\sigma_k)$, where "$\neg!$" means "is not defined".

Intuitively, for each controllable event $\sigma_k \in \Sigma_c$, $X_A^{T_k}$ is the set of states at which σ_k must be disabled for safety purposes, while $X_A^{T_k}$ is the set of states at which σ_k must be enabled to achieve $\mathcal{L}(\mathbf{H})$. The following result reveals that coobservability is also a special case of decentralized distinguishability with $\mathbb{T}_{cont}$.

Theorem VI.2. *Let $\boldsymbol{G}$ be the system and H be the specification automaton. Then $\mathcal{L}(\boldsymbol{G})$ is coobservable w.r.t. $\mathcal{L}(\boldsymbol{H})$, $\Sigma_{c,1}, \Sigma_{c,2}$ and $\bar{\Omega}$, if and only if, $\boldsymbol{G}$ is decentralized distinguishable w.r.t. $\bar{\Omega}$ and $\mathbb{T}_{cont}$.*

Proof. ($\Rightarrow$) By contraposition. Suppose that $\mathbf{G}$ is not decentralized distinguishable. Then we know that there exist $T_k \in \mathbb{T}, s \in \mathcal{L}(\mathbf{G}) : \delta(s) \in X_A^{T_k}$ such that for each $i \in \mathcal{I}_{T_k}$, there exists $t_i \in \mathcal{L}(\mathbf{G})$ such that $P_{\Omega_i}(s) = P_{\Omega_i}(t_i)$ and $\delta(t_i) \in X_B^{T_k}$. Let

$\sigma_k \in \Sigma_c$ be the controllable event associated to T_k. Then $\delta(s) \in X_A^{T_k}$ implies that $s \in \mathcal{L}(\mathbf{H})$, $s\sigma_k \in \mathcal{L}(\mathbf{G}) \setminus \mathcal{L}(\mathbf{H})$ and $\delta(t_i) \in X_B^{T_k}$ implies that $t_i\sigma_k \in \mathcal{L}(\mathbf{H})$. Moreover, $\mathcal{I}_{T_k} = \mathcal{I}^c(\sigma_k)$. Overall, we know that $\exists s \in \mathcal{L}(\mathbf{H}), \sigma_k \in \Sigma_c$ such that $s\sigma_k \in \mathcal{L}(\mathbf{G})\setminus\mathcal{L}(\mathbf{H})$ and for each $i \in \mathcal{I}^c(\sigma_k)$, $t_i\sigma_k \in P_{\Omega_i}^{-1}(P_{\Omega_i}(s))\{\sigma_k\} \cap \mathcal{L}(\mathbf{H}) \neq \emptyset$, i.e., $\mathcal{L}(\mathbf{G})$ is not coobservable.

($\Leftrightarrow$) By contraposition. Suppose that $\mathcal{L}(\mathbf{G})$ is not coobservable. Then we know that $\exists s \in \mathcal{L}(\mathbf{H}), \sigma_k \in \Sigma_c : s\sigma_k \in \mathcal{L}(\mathbf{G}) \setminus \mathcal{L}(\mathbf{H})$ such that for each $i \in \mathcal{I}^c(\sigma_k)$, there exists $t_i \in \mathcal{L}(\mathbf{G})$ such that $t_i\sigma_k \in \mathcal{L}(\mathbf{H})$ and $P_{\Omega_i}(s) = P_{\Omega_i}(t_i)$. For the above s and t_i, we know that $\delta(s) \in X_A^{T_k}$ and $\delta(t_i) \in X_B^{T_k}$. Therefore, for s and σ_k, we know that for each $i \in \mathcal{I}_{T_k} = \mathcal{I}^c(\sigma_k)$, $\delta(t_i) \in \mathcal{E}_{\Omega_i}^{\mathbf{G}}(t_i) \cap X_B^{T_k} = \mathcal{E}_{\Omega_i}^{\mathbf{G}}(s) \cap X_B^{T_k} \neq \emptyset$, i.e., $\mathbf{G}$ is not decentralized distinguishable. $\square$

6.5.3 Decentralized Fault Prediction

In some safety-critical systems, we may not only want to diagnose any fault after its occurrence, but also want to *predict* any fault before it occurs [29]. In [47], the notion of coprognosability was proposed to capture whether or not any fault occurrence can be predicted in a decentralized system. The definition is reviewed as follows.

Definition 6.5.3. *(Coprognosability). We say that language $\mathcal{L}(\boldsymbol{G})$ is coprognosable w.r.t. $\bar{\Omega}$ and Σ_F if*

$$(\forall s \in \Psi(\Sigma_F))(\exists t \in \overline{\{s\}} : \Sigma_F \notin t)(\exists i \in \{1,2\})(\forall u \in P_{\Omega_i}^{-1}(P_{\Omega_i}(t)) : \Sigma_F \notin u)$$
$$(\exists K \in \mathbb{N})(\forall v \in \mathcal{L}(\boldsymbol{G})/u)[|v| \geq K \Rightarrow \Sigma_F \in uv]$$

Still, we assume that the state space of $\mathbf{G}$ is partitioned as $X = X_N \dot{\cup} X_F$ such that $\forall s \in \mathcal{L}(\mathbf{G}) : \delta(s) \in X_N \Leftrightarrow \Sigma_F \notin s$; and $\forall s \in \mathcal{L}(\mathbf{G}) : \delta(s) \in X_F \Leftrightarrow \Sigma_F \in s$. And we still denote by $\mathcal{N}_X$ and ∂_X the set of non-indicator states and the set of boundary states, respectively.

With these notions, we define a simple specification $\mathbb{T}_{pre} := \{T_1\}$, where $X_A^{T_1} = \partial_X$ and $X_B^{T_1} = \mathcal{N}_X$ with $\mathcal{I}_{T_1} = \mathcal{I}$. The following result reveals that, to enforce coprognosability, it suffices to enforce decentralized distinguishability with $\mathbb{T}_{pre}$.

Theorem VI.3. *$\mathcal{L}(\boldsymbol{G})$ is coprognosable w.r.t. $\bar{\Omega}$ and Σ_F, if and only if, $\boldsymbol{G}$ is decentralized distinguishable w.r.t. $\bar{\Omega}$ and $\mathbb{T}_{pre}$.*

Proof. ($\Rightarrow$) By contraposition. Suppose that $\mathbf{G}$ is not decentralized distinguishable. Then we know that there exists $s \in \mathcal{L}(\mathbf{G})$ such that $x := \delta(s) \in \partial_X$ and for each $i \in \{1, 2\}$, there exists $x_i \in \mathcal{E}^{\mathbf{G}}_{\Omega_i}(s)$ such that $x_i \in \mathcal{N}_X$, i.e., there exists a string $s_i \in \mathcal{L}(\mathbf{G})$ such that $\Sigma_F \notin s_i$, $\delta(s_i) = x_i$ and $P_{\Omega_i}(s) = P_{\Omega_i}(s_i)$. Since $x \in \partial_X$, we know that $\exists f \in \Sigma_F : sf \in \Psi(\Sigma_F)$. Let $t \in \overline{\{s\}}$ be an arbitrary prefix of s such that $\Sigma_F \notin t$. Then for each $i \in \{1, 2\}$, since $P_{\Omega_i}(s) = P_{\Omega_i}(s_i)$, we know that

$$\forall t \in \overline{\{s\}}, \exists t_i \in \overline{\{s_i\}} : P_{\Omega_i}(t) = P_{\Omega_i}(t_i) \wedge \Sigma_F \notin t_i \tag{6.12}$$

Moreover, since $x_i \in \mathcal{N}_X$ which is reachable from $\delta(t_i)$, we know that, for any $K \in \mathbb{N}$, there exists a string w_i such that $t_i w_i \in \mathcal{L}(\mathbf{G})$, $\Sigma_F \notin t_i w_i$ and $|w_i| \geq K$. Overall, we know that

$$\begin{aligned}&(\exists sf \in \Psi(\Sigma_F))(\forall t \in \overline{\{sf\}} : \Sigma_F \notin t)(\forall i \in \{1, 2\})(\exists t_i \in P^{-1}_{\Omega_i}(P_{\Omega_i}(t)) : \Sigma_F \notin t_i) \\ &(\forall K \in \mathbb{N})(\exists w_i \in \mathcal{L}(\mathbf{G})/t_i)[|w_i| \geq K \wedge \Sigma_F \notin t_i w_i]\end{aligned} \tag{6.13}$$

i.e., $\mathbf{G}$ is not coprognosable w.r.t. $\bar{\Omega}$ and Σ_F.

($\Leftarrow$) Suppose that $\mathbf{G}$ is not coprognosable, i.e., Equation (6.13) holds. Let sf be a string satisfying Equation (6.13). Let t a prefix of s such that $\Sigma_F \notin t$ and $tf' \in \overline{\{s\}}$ for some $f' \in \Sigma_F$. Then we know that $x := \delta(t) \in \partial_X$. According to Equation (6.13) , we know that, for each agent $i \in \{1, 2\}$, there exists a string $t_i \in \mathcal{L}(\mathbf{G})$ such that 1) $\Sigma_F \notin t_i$; and 2) $(\forall K \in \mathbb{N})(\exists w_i \in \mathcal{L}(\mathbf{G})/t_i)[|w_i| \geq K \wedge \Sigma_F \notin t_i w_i]$; and 3)

$P_{\Omega_i}(t_i) = P_{\Omega_i}(t)$. The first two conditions imply that $x_i := \delta(t_i) \in \mathcal{N}_X$. Moreover, the last condition implies that $\{x, x_i\} \subseteq \mathcal{E}^{\mathbf{G}}_{\Omega_i}(t)$. Overall, we know that $(\exists t \in \mathcal{L}(\mathbf{G}) : \delta(t) \in \partial_X)(\forall i \in \{1,2\})[\mathcal{E}^{\mathbf{G}}_{\Omega_i}(t) \cap \mathcal{N}_X = \emptyset]$, i.e., $\mathbf{G}$ is not decentralized distinguishable. $\square$

Remark 6.5.1. Note that ∂_X and $\mathcal{N}_X$ need not be disjoint. By the above theorem, the system will not be coprognosable under any sensor activation policies if $\partial_X \cap \mathcal{N}_X \neq \emptyset$.

6.6 Extension to the Conjunctive Architecture

In Section 6.5, we have shown that K-codiagnosability, coobservability and coprognosability are special cases of decentralized distinguishability. As we mentioned earlier, all results in this chapter are developed based on the disjunctive architecture, i.e., the coordinator issues "1" globally, if and only if, *one* local agent issues "1". Alternatively, one may also use the *conjunctive* rule to obtain a global decision, i.e., the coordinator issues "0" globally, if and only if, *one* local agent issues "0". In this case, suppose that a string leading to a state in $X_B^{T_k}$ is executed and a global decision "0" has to be made. Then, a local agent must know that the system is not in $X_A^{T_k}$ *unambiguously* when it issues "0"; otherwise a wrong global decision may be made at some state in $X_A^{T_k}$. Therefore, we need to require that

$$(\forall s \in \mathcal{L}(\mathbf{G}) : \delta(s) \in X_B^{T_k})(\exists i \in \mathcal{I}_{T_k})[\mathcal{E}^{\mathbf{G}}_{\Omega_i}(s) \cap X_A^{T_k} = \emptyset]$$

By comparing the above requirement with decentralized distinguishability, which is defined in terms of the disjunctive architecture, we see that this requirement is indeed the same as decentralized distinguishability by swapping $X_A^{T_k}$ and $X_B^{T_k}$. Therefore, there is no need to define a conjunctive version of decentralized distinguishability; it is just a matter of how the specification T_k is defined.

For example, in [128], the notion of D&A-coobservability was proposed as a complement of coobservability[1]. We recall its definition.

Definition 6.6.1. *(D&A-Coobservability). We say that $\mathcal{L}(\boldsymbol{G})$ is D&A-coobservable w.r.t. $\mathcal{L}(\boldsymbol{H})$, $\Sigma_{c,1}, \Sigma_{c,2}$ and $\bar{\Omega}$ if*

$$(\forall s \in \mathcal{L}(\boldsymbol{H}))(\forall \sigma \in \Sigma_c : s\sigma \in \mathcal{L}(\boldsymbol{H}))(\exists i \in \mathcal{I}^c(\sigma_k))[P_{\Omega_i}^{-1}(P_{\Omega_i}(s))\{\sigma\} \cap \mathcal{L}(\boldsymbol{G}) \subseteq \mathcal{L}(\boldsymbol{H})]$$

Intuitively, D&A-coobservability requires that for any string for which σ has to be enabled, there exists at least one agent that knows for sure that σ should not be disabled. We can also formulate D&A-coobservability as an instance of decentralized distinguishability by defining $\mathbb{T}_{cont}^{CJ} = \{T_1^{CJ}, T_2^{CJ}, \ldots, T_m^{CJ}\}$, where for each $T_k^{CJ} \in \mathbb{T}$, we have

$$\begin{aligned} X_A^{T_k^{CJ}} =& \{x \in X_H : \delta_H(q, \sigma_k)!\} \\ X_B^{T_k^{CJ}} =& \{x \in X_H : \delta(q, \sigma_k)! \wedge \delta_H(q, \sigma_k)\neg!\} \end{aligned}$$

with $\mathcal{I}_{T_k} = \mathcal{I}^c(\sigma_k)$. The proof of the correctness of $\mathbb{T}_{cont}^{CJ}$ is omitted since it is similar to the proof of Theorem VI.2.

Similarly, one can also show that conjunctive K-codiagnosability [105, 108] and conjunctive coprognosability [40, 126] are instances of decentralized distinguishabiity; we just need to define new specifications $\mathbb{T}_{diag}^{CJ}$ and $\mathbb{T}_{pre}^{CJ}$ by swapping each $X_A^{T_k}$ and $X_B^{T_k}$ in $\mathbb{T}_{diag}$ and $\mathbb{T}_{pre}$, respectively.

[1]Here, "D&A" stands for "Disjunctive & Anti-permissive". Also, coobservability in Definition 6.5.2 is referred to as C&P-coobservability, where "C&A" standards for "Conjunctive & Permissive". The reason why C&P-coobservability corresponds to decentralized distinguishability in the disjunctive architecture is that, [69] considers the conjunction of enablements, while $\mathbb{T}_{cont}$ captures the disjunction of disablements; they are essentially equivalent.

6.7 Conclusion

We presented a novel approach for solving the problem of decentralized sensor activation for a class of properties. We proposed the notion of decentralized distinguishability, which covers coobservability, K-codiagnosability and coprognosability. To enforce decentralized distinguishability, we first adopted a person-by-person approach to decompose the decentralized minimization problem to two consecutive centralized constrained minimization problems. Then, a novel approach was proposed to reduce each centralized constrained minimization problem to a fully centralized sensor activation that is solved effectively in the literature. Finally, we showed that the decentralized solution obtained by our methodology is language-based minimal.

CHAPTER VII

Conclusion and Future Work

7.1 Conclusion

In this dissertation, we developed novel approaches for solving the control problem and the sensor activation problem for the purpose of property enforcement in Cyber-Physical Systems that are modeled as Discrete Event Systems. The results developed in this dissertation provide new theoretical foundations and computational algorithms for synthesizing higher-level control logic in cyber-physical systems that is provably correct in terms of a set of qualitative requirements. Moreover, the novel approaches developed can also significantly reduce the system integration, verification and validation cycle (and therefore time-to-market) for important classes of societal systems.

More specifically, for the supervisory control problem, a uniform framework for supervisory control under partial observation was developed. The uniform framework is based on the newly proposed structures called All Enforcement Structure (AES) and Non-blocking All Enforcement Structure (NB-AES). Based on the AES and the NB-AES, algorithms for synthesizing maximally-permissive non-blocking supervisors for the enforcement of a large class of properties were proposed. The proposed framework not only handles the enforcement of many important properties in the DES literature in a uniform manner, but it also can handle the issue of non-blockingness. As a special case of the general framework, the problem of synthesizing a maximally-permissive safe and non-blocking supervisor, that had remained open for more than 25 years, was solved. An algorithm was also proposed to solve the range control problem for the case of prefix-closed specifications. This results in a safe and locally maximal supervisor that provably contains a given desired behavior.

For the dynamic sensor activation problem, a uniform framework for synthesizing sensor activation policies for a wide class of properties was also developed. A generalized version of the Most Permissive Observer (MPO) was defined to solve the synthesis problem. Based on the MPO, we presented an algorithm for the synthesis of optimal sensor activation policies under a logical performance objective. We also investigated the problem of minimizing sensor activations for decentralized fault diagnosis. A person-by-person approach together with the centralized uniform framework were used to solve decentralized minimization problem. This provided the first algorithm that solves the language-based sensor optimization for decentralized DES.

7.2 Future Work

The framework developed in this dissertation opens several future research directions. First, in Chapter II, the non-blockingness condition considered requires that the system is always able to reach a marked state. However, this does not guarantee that a marked state is eventually reached. One interesting future direction is to consider a stronger version of non-blockingness, which requires that a marked state can *always eventually* be reached. In some applications, different markings may represent different types of targets and one may be interested in synthesizing a supervisor such that different marked states can be reached in a certain order. This problem has been studied in the fully-observed setting in the context of multitasking supervisory control [63]. Generalizing the multitasking supervisory control framework to the partially-observed setting is also an interesting future direction.

In the range control problem studied in Chapter IV, we only considered prefix-closed safety specifications, i.e., the issue of non-blockingness is not considered. One important future direction is to generalize the results in Chapter IV to include the non-blockingness requirement, given any IS-based property. Also, in Chapters II, III and IV, we only investigated the centralized supervisory control problem. One future direction is to study the supervisor synthesis problem in the decentralized setting [69,128]. It has been shown in [95,98] that the problem of synthesizing a decentralized non-blocking supervisor is undecidable. However, to the best of our knowledge, how to synthesize a locally maximal decentralized supervisor is still an open problem. Investigating the decidability of this problem is an interesting future direction.

In this dissertation, only qualitative requirements were considered in the control synthesis problem. It would be of interest to develop theoretical foundations and computational algorithms for synthesizing higher-level control logic in cyber-physical systems that is provably correct in terms of a set of qualitative requirements (in the form of safety and liveness properties) and, at the same time, satisfies a set of quantitative energy or resource constraints imposed on the system. More specifically, one interesting future direction is to investigate how to guarantee that the accumulated cost of any execution of the controlled system is non-negative. This essentially requires to solve an *energy game* over the AES or the MPO structure. Another future direction is to investigate how to minimize the average cost in addition to the accumulated cost. For this direction, we may need to solve a *mean-payoff game* over the AES or the MPO structure.

Also, in this dissertation, the supervisor synthesis problem and the sensor activation policy synthesis problem were addressed *separately*. One interesting future direction is to investigate the *joint* synthesis of supervisory control strategies and sensor activation strategies and associated design trade-offs. To this end, we may need to define a new transition structure that combines the AES and the MPO. The development of this new structure would allow the study the dependency between the control strategy and the sensing strategy in this joint synthesis problem.

Regarding the decentralized sensor activation problem investigated in Chapter VI, we only considered so-called "non-conditional" decision fusion architectures. We may also wish to extend the results in Chapter VI to the conditional architectures defined in [105, 130], the inference architecture of [45, 46], and the intersection-based architecture recently investigated in [115].

Finally, to further mitigate the computational challenges associated with the growth of the number of states in the system, it would be of interest to use symbolic methods to improve the scalability of the approaches developed in this dissertation. In this regard, it wold be of interest to investigate how to symbolically represent the BTS/MPO structures and their associated operators. Symbolic methods have been recently applied to the basic supervisor synthesis problem in the theory of supervisory control; see [28, 39, 55–57, 65]. However, to the best of our knowledge, all existing works focus on the full observation case. The case of partially-observed systems has not been addressed yet; neither has the problem of synthesizing sensor activation policies.

It is our thesis that the novel methodologies developed in this dissertation provide sound theoretical foundations and adaptable algorithmic procedures for investigating the above-discussed problems.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] A. Arnold, A. Vincent, and I. Walukiewicz. Games for synthesis of controllers with partial observation. *Theoretical Computer Science*, 303(1):7–34, 2003.

[2] E. Badouel, M. Bednarczyk, A. Borzyszkowski, B. Caillaud, and P. Darondeau. Concurrent secrets. *Discrete Event Dynamic Systems: Theory & Applications*, 17(4):425–446, 2007.

[3] G. Barrett and S. Lafortune. On the separation of estimation and control in discrete-event systems. In *Proceedings of the 39th IEEE Conference on Decision and Control*, pages 2258–2259, 2000.

[4] N. Ben Hadj-Alouane, S. Lafortune, and F. Lin. Centralized and distributed algorithms for on-line synthesis of maximal control policies under partial observation. *Discrete Event Dynamic Systems: Theory & Applications*, 6(4):379–427, 1996.

[5] M. Ben-Kalefa and F. Lin. Supervisory control for opacity of discrete event systems. In *49th IEEE Annual Allerton Conference on Communication, Control, and Computing*, pages 1113–1119, 2011.

[6] G. Benattar, F. Cassez, D. Lime, and O.H. Roux. Control and synthesis of non-interferent timed systems. *International Journal of Control*, 88(2):217–236, 2015.

[7] R. Boel and J.H. van Schuppen. Decentralized failure diagnosis for discrete-event systems with costly communication between diagnosers. In *Proc. 6th International Workshop on Discrete Event Systems*, pages 175–181, 2002.

[8] R.D. Brandt, V. Garg, R. Kumar, F. Lin, S.I. Marcus, and W.M. Wonham. Formulas for calculating supremal controllable and normal sublanguages. *Systems & Control Letters*, 15(2):111–117, 1990.

[9] Y. Brave and M. Heymann. Stabilization of discrete-event processes. *International Journal of Control*, 51(5):1101–1117, 1990.

[10] J.W. Bryans, M. Koutny, L. Mazaré, and P. Y. Ryan. Opacity generalised to transition systems. *International Journal of Information Security*, 7(6):421–435, 2008.

[11] K. Cai, R. Zhang, and W.M. Wonham. Relative observability of discrete-event systems and its supremal sublanguages. *IEEE Transactions on Automatic Control*, 60(3):659–670, 2015.

[12] C.G. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems*. Springer, 2nd edition, 2008.

[13] F. Cassez. Dynamic observers for fault diagnosis of timed systems. In *49th IEEE Conf. Decision and Control*, pages 4359–4364, 2010.

[14] F. Cassez. The complexity of codiagnosability for discrete event and timed systems. *IEEE Transactions on Automatic Control*, 57(7):1752–1764, 2012.

[15] F. Cassez, J. Dubreil, and H. Marchand. Synthesis of opaque systems with static and dynamic masks. *Formal Methods in System Design*, 40(1):88–115, 2012.

[16] F. Cassez and S. Tripakis. Fault diagnosis with static and dynamic observers. *Fundamenta Informaticae*, 88(4):497–540, 2008.

[17] K. Chatterjee, L. Doyen, T.A. Henzinger, and J.-F. Raskin. Algorithms for omega-regular games with imperfect information. In *Computer Science Logic*, pages 287–302. Springer, 2006.

[18] H. Cho and S.I. Marcus. On supremal languages of classes of sublanguages that arise in supervisor synthesis problems with partial observation. *Mathematics of Control, Signals, and Systems*, 2(1):47–69, 1989.

[19] H. Cho and S.I. Marcus. Supremal and maximal sublanguages arising in supervisor synthesis problems with partial observations. *Mathematical Systems Theory*, 22(1):177–211, 1989.

[20] R. Cieslak, C. Desclaux, A.S. Fawaz, and P. Varaiya. Supervisory control of discrete-event processes with partial observations. *IEEE Transactions on Automatic Control*, 33(3):249–260, 1988.

[21] E.M. Clarke, O. Grumberg, and D. Peled. *Model checking*. MIT press, 1999.

[22] E. Dallal and S. Lafortune. On most permissive observers in dynamic sensor activation problems. *IEEE Transactions on Automatic Control*, 59(4):966–981, 2014.

[23] P. Darondeau, H. Marchand, and L. Ricker. Enforcing opacity of regular predicates on modal transition systems. *Discrete Event Dynamic Systems: Theory & Applications*, 2014.

[24] R. Debouk, S. Lafortune, and D. Teneketzis. Coordinated decentralized protocols for failure diagnosis of discrete event systems. *Discrete Event Dynamic Systems: Theory & Applications*, 10(1-2):33–86, 2000.

[25] R. Debouk, S. Lafortune, and D. Teneketzis. On an optimization problem in sensor selection. *Discrete Event Dynamic Systems: Theory & Applications*, 12(4):417–445, 2002.

[26] J. Dubreil, P. Darondeau, and H. Marchand. Supervisory control for opacity. *IEEE Transactions on Automatic Control*, 55(5):1089–1100, 2010.

[27] J. Fa, X. Yang, and Y. Zheng. Formulas for a class of controllable and observable sublanguages larger than the supremal controllable and normal sublanguage. *Systems & Control Letters*, 20(1):11–18, 1993.

[28] Z. Fei, S. Reveliotis, S. Miremadi, and K. Åkesson. A bdd-based approach for designing maximally permissive deadlock avoidance policies for complex resource allocation systems. *IEEE Transactions on Automation Science and Engineering*, 12(3):990–1006, 2015.

[29] S. Genc and S. Lafortune. Predictability of event occurrences in partially-observed discrete-event systems. *Automatica*, 45(2):301–311, 2009.

[30] A. Haji-Valizadeh and K. Loparo. Minimizing the cardinality of an events set for supervisors of discrete-event dynamical systems. *IEEE Transactions on Automatic Control*, 41(11):1579–1593, 1996.

[31] M. Heymann and F. Lin. On-line control of partially observed discrete event systems. *Discrete Event Dynamic Systems: Theory & Applications*, 4(3):221–236, 1994.

[32] Y.-C. Ho. Team decision theory and information structures. *Proceedings of the IEEE*, 68(6):644–654, 1980.

[33] Y. Huang, K. Rudie, and F. Lin. Decentralized control of discrete-event systems when supervisors observe particular event occurrences. *IEEE Trans. Autom. Contr.*, 53(1):384–388, 2008.

[34] K. Inan. Nondeterministic supervision under partial observations. In *11th International Conference on Analysis and Optimization of Systems: Discrete Event Systems*, pages 39–48. Springer, 1994.

[35] M. Iordache and P.J. Antsaklis. *Supervisory control of concurrent systems: a Petri net structural approach.* Springer Science & Business Media, 2007.

[36] R. Jacob, J.-J. Lesage, and J.-M. Faure. Overview of discrete event systems opacity: Models, validation, and quantification. *Annual Reviews in Control*, 41:135–146, 2016.

[37] S. Jiang, R. Kumar, and H. Garcia. Optimal sensor selection for discrete-event systems with partial observation. *IEEE Transactions on Automatic Control*, 48(3):369–381, 2003.

[38] S. Jiang, R. Kumar, S. Takai, and W. Qiu. Decentralized control of discrete-event systems with multiple local specifications. *IEEE Transactions on Autom. Sci. Eng.*, 7(3):512–522, 2010.

[39] G. Kalyon, T. Le Gall, H. Marchand, and T. Massart. Symbolic supervisory control of distributed systems with communications. *IEEE Transactions on Automatic Control*, 59(2):396–408, 2014.

[40] A. Khoumsi and H. Chakib. Conjunctive and disjunctive architectures for decentralized prognosis of failures in discrete-event systems. *IEEE Trans. Autom. Sci. Eng.*, 9(2):412–417, 2012.

[41] J. Komenda and J.H. van Schuppen. Control of discrete-event systems with partial observations using coalgebra and coinduction. *Disctre Event Dynamic Systems.: Theory & Application*, 15(3):257–315, 2005.

[42] R. Kumar and V.K. Garg. *Modeling and control of logical discrete event systems.* Kluwer, 1995.

[43] R. Kumar, S. Jiang, C. Zhou, and W. Qiu. Polynomial synthesis of supervisor for partially observed discrete-event systems by allowing nondeterminism in control. *IEEE Transactions on Automatic Control*, 50(4):463–475, 2005.

[44] R. Kumar and M. Shayman. Formulae relating controllability, observability, and co-observability. *Automatica*, 34(2):211–215, 1998.

[45] R. Kumar and S. Takai. Inference-based ambiguity management in decentralized decision-making: Decentralized control of discrete event systems. *IEEE Transactions on Automatic Control*, 52(10):1783–1794, 2007.

[46] R. Kumar and S. Takai. Inference-based ambiguity management in decentralized decision-making: Decentralized diagnosis of discrete-event systems. *IEEE Transactions on Automation Science and Engineering*, 6(3):479–491, 2009.

[47] R. Kumar and S. Takai. Decentralized prognosis of failures in discrete event systems. *IEEE Transactions on Automatic Control*, 55(1):48–59, 2010.

[48] O. Kupferman and M. Vardi. Synthesis with incomplete informatio. In *Advances in Temporal Logic*, pages 109–127. Springer, 2000.

[49] Z. Li, M. Zhou, and N. Wu. A survey and comparison of petri net-based deadlock prevention policies for flexible manufacturing systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 38(2):173–188, 2008.

[50] H. Liao, S. Lafortune, S. Reveliotis, Y. Wang, and S. Mahlke. Optimal liveness-enforcing control for a class of petri nets arising in multithreaded software. *IEEE Transactions on Automatic Control*, 58(5):1123–1138, 2013.

[51] F. Lin. Opacity of discrete event systems and its applications. *Automatica*, 47(3):496–503, 2011.

[52] F. Lin and W.M. Wonham. On observability of discrete-event systems. *Inform. Sciences*, 44(3):173–198, 1988.

[53] F. Lin and W.M. Wonham. Decentralized control and coordination of discrete-event systems with partial observation. *IEEE Transactions on Automatic Control*, 35(12):1330–1337, 1990.

[54] F. Lin and W.M. Wonham. Supervisory control of timed discrete-event systems under partial observation. *IEEE Transactions on Automatic Control*, 40(3):558–562, 1995.

[55] C. Ma and W.M. Wonham. Nonblocking supervisory control of state tree structures. *IEEE Transactions on Automatic Control*, 51(5):782–793, 2006.

[56] S. Miremadi, Z. Fei, K. Åkesson, and B. Lennartson. Symbolic supervisory control of timed discrete event systems. *IEEE Trans. Control Systems Technology*, 23(2):584–597, 2015.

[57] S. Miremadi, B. Lennartson, and K. Akesson. A BDD-based approach for modeling plant and supervisor by extended finite automata. *IEEE Trans. Control Systems Technology*, 20(6):1421–1435, 2012.

[58] T. Moor and K.W. Schmidt. Fault-tolerant control of discrete-event systems with lower-bound specifications. In *5th International Workshop on Dependable Control of Discrete Systems*, pages 161–166, 2015.

[59] A. Overkamp and J.H. van Schuppen. Maximal solutions in decentralized supervisory control. *SIAM Journal on Control and Optimization*, 39(2):492–511, 2000.

[60] C.M. Özveren, A.S. Willsky, and P.J. Antsaklis. Stability and stabilizability of discrete event dynamic systems. *Journal of the ACM*, 38(3):729–751, 1991.

[61] S. Pinchinat and S. Riedweg. You can always compute maximally permissive controllers under partial observation when they exist. In *American Control Conference*, pages 2287–2292, 2005.

[62] W. Qiu and R. Kumar. Decentralized failure diagnosis of discrete event systems. *IEEE Transactions on Systems, Man and Cybernetics, Part A*, 36(2):384–395, 2006.

[63] M.H. Queiroz, J.E.R. Cury, and W.M. Wonham. Multitasking supervisory control of discrete-event systems. *Discrete Event Dyn. Sys.: Theory & Appl.*, 15(4):375–395, 2005.

[64] P.J. Ramadge and W.M. Wonham. Supervisory control of a class of discrete event processes. *SIAM Journal on Control and Optimization*, 25(1):206–230, 1987.

[65] B.C. Rawlings. *Discrete Dynamics in Chemical Process Control and Automation*. PhD thesis, Carnegie Mellon University, 2016.

[66] K. Rudie, S. Lafortune, and F. Lin. Minimal communication in a distributed discrete-event system. *IEEE Transactions on Automatic Control*, 48(6):957–975, 2003.

[67] K. Rudie and J.C. Willems. The computational complexity of decentralized discrete-event control problems. *IEEE Transactions on Automatic Control*, 40(7):1313–1319, 1995.

[68] K. Rudie and W.M. Wonham. The infimal prefix-closed and observable superlanguange of a given language. *Syst. Control Letters*, 15(5):361–371, 1990.

[69] K. Rudie and W.M. Wonham. Think globally, act locally: Decentralized supervisory control. *IEEE Transactions on Automatic Control*, 37(11):1692–1708, 1992.

[70] A. Saboori and C.N. Hadjicostis. Notions of security and opacity in discrete event systems. In *46th IEEE Conference on Decision and Control*, pages 5056–5061, 2007.

[71] A. Saboori and C.N. Hadjicostis. Verification of k-step opacity and analysis of its complexity. *IEEE Transactions on Automation Science and Engineering*, 8(3):549–559, 2011.

[72] A. Saboori and C.N. Hadjicostis. Opacity-enforcing supervisory strategies via a state estimator constructions. *IEEE Transactions on Automatic Control*, 57(5):1155–1165, 2012.

[73] A. Saboori and C.N. Hadjicostis. Verification of infinite-step opacity and complexity considerations. *IEEE Transactions on Automatic Control*, 57(5):1265–1269, 2012.

[74] A. Saboori and C.N. Hadjicostis. Verification of initial-state opacity in security applications of discrete event systems. *Information Sciences*, 246:115–132, 2013.

[75] W. Sadid, S.L. Ricker, and S. Hashtrudi-Zad. Nash equilibrium for communication protocols in decentralized discrete-event systems. In *American Control Conference*, pages 3384–3389, 2010.

[76] M. Sampath, S. Lafortune, and D. Teneketzis. Active diagnosis of discrete-event systems. *IEEE Transactions on Automatic Control*, 43(7):908–929, 1998.

[77] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis. Diagnosability of discrete-event systems. *IEEE Transactions on Automatic Control*, 40(9):1555–1575, 1995.

[78] K.W. Schmidt and C. Breindl. A framework for state attraction of discrete event systems under partial observation. *Information Sciences*, 281(10):265–280, 2014.

[79] D. Sears and K. Rudie. Computing sensor activation decisions from state equivalence classes in discrete-event systems. In *52nd IEEE Conference on Decision and Control*, pages 6972–6977, 2013.

[80] D. Sears and K. Rudie. Efficient computation of sensor activation decisions in discrete-event systems. In *52nd IEEE Conference on Decision and Control*, pages 6966–6971, 2013.

[81] D. Sears and K. Rudie. On computing indistinguishable states of nondeterministic finite automata with partially observable transitions. In *53rd IEEE Conf. Decision and Control*, pages 6731–6736, 2014.

[82] D. Sears and K. Rudie. Minimal sensor activation and minimal communication in discrete-event systems. *Disctre Event Dynamic Systems.: Theory & Application*, 26(2):295–349, 2016.

[83] C. Seatzu, M. Silva, and J.H. Van Schuppen. *Control of Discrete-Event Systems. Automata and Petri net Perspectives.* Springer London, 2013.

[84] K. Shin, X. Ju, Z. Chen, and X. Hu. Privacy protection for users of location-based services. *IEEE Wireless Communications*, 19(1):30–39, 2012.

[85] S. Shu, Z. Huang, and F. Lin. Online sensor activation for detectability of discrete event systems. *IEEE Transactions on Automation Science and Engineering*, 10(2):457–461, 2013.

[86] S. Shu and F. Lin. Detectability of discrete event systems with dynamic event observation. *Systems & Control Letters*, 59(1):9–17, 2010.

[87] S. Shu and F. Lin. Enforcing detectability in controlled discrete event systems. *IEEE Transactions on Automatic Control*, 58(8):2125–2130, 2013.

[88] S. Shu and F. Lin. I-detectability of discrete-event systems. *IEEE Transactions on Automation Science and Engineering*, 10(1):187–196, 2013.

[89] S. Shu, F. Lin, and H. Ying. Detectability of discrete event systems. *IEEE Transactions on Automatic Control*, 52(12):2356–2359, 2007.

[90] L. Sweeney. K-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness & Knowledge-Based Systems*, 10(05):557–570, 2002.

[91] P. Tabuada. *Verification and Control of Hybrid Systems: A Symbolic Approach*. Springer, 2009.

[92] S. Takai and Y. Oka. A formula for the supremal controllable and opaque sublanguage arising in supervisory control. *SICE Journal of Control, Measurement, and System Integration*, 1(4):307–311, 2008.

[93] S. Takai and T. Ushio. Effective computation of an $L_m(G)$-closed, controllable, and observable sublanguage arising in supervisory control. *Systems & Control Letters*, 49(3):191–200, 2003.

[94] A. Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific J. Math.*, 5(2):285–309, 1955.

[95] J.G. Thistle. Undecidability in decentralized supervision. *Systems & Control Letters*, 54(5):503–509, 2005.

[96] J.G. Thistle and H.M. Lamouchi. Effective control synthesis for partially observed discrete-event systems. *SIAM Journal on Control and Optimization*, 48(3):1858–1887, 2009.

[97] D. Thorsley and D. Teneketzis. Active acquisition of information for diagnosis and supervisory control of discrete event systems. *Disctre Event Dynamic Systems.: Theory & Application*, 17(4):531–583, 2007.

[98] S. Tripakis. Undecidable problems of decentralized observation and control on regular languages. *Information Processing Letters*, 90(1):21–28, 2004.

[99] J.N. Tsitsiklis. On the control of discrete-event dynamical systems. *Mathematics of Control, Signals, and Systems*, 2(2):95–107, 1989.

[100] J.H. van Schuppen. Control of distributed stochastic systems-introduction, problems, and approaches. In *Proc. 18th IFAC World Congress*, volume 2011, 2011.

[101] W. Wang, S. Lafortune, A.R. Girard, and F. Lin. Optimal sensor activation for diagnosing discrete event systems. *Automatica*, 46(7):1165–1175, 2010.

[102] W. Wang, S. Lafortune, and F. Lin. An algorithm for calculating indistinguishable states and clusters in finite-state automata with partially observable transitions. *Systems & Control Letters*, 56(9):656–661, 2007.

[103] W. Wang, S. Lafortune, F. Lin, and A.R. Girard. An online algorithm for minimal sensor activation in discrete event systems. In *48th IEEE Conference on Decision and Control*, pages 2242–2247, 2009.

[104] W. Wang, S. Lafortune, F. Lin, and A.R. Girard. Minimization of dynamic sensor activation in discrete event systems for the purpose of control. *IEEE Transactions on Automatic Control*, 55(11):2447–2461, 2010.

[105] Y. Wang, T.-S. Yoo, and S. Lafortune. Diagnosis of discrete event systems using decentralized architectures. *Discrete Event Dynamic Systems: Theory & Applications*, 17(2):233–263, 2007.

[106] W. M. Wonham. *Supervisory control of discrete-event systems.* Systems Control Group, ECE Dept, University of Toronto, 2014.

[107] Y.-C. Wu and S. Lafortune. Comparative analysis of related notions of opacity in centralized and coordinated architectures. *Discrete Event Dynamic Systems*, 23(3):307–339, 2013.

[108] T. Yamamoto and S. Takai. Conjunctive decentralized diagnosis of discrete event systems. In *4th IFAC Workshop on Dependable Control of Discrete Systems*, pages 67–72, 2013.

[109] X. Yin and S. Lafortune. A general approach for synthesis of supervisors for partially-observed discrete-event systems. In *Proceedings of the 19th IFAC World Congress*, pages 2422–2428, 2014.

[110] X. Yin and S. Lafortune. Synthesis of maximally permissive non-blocking supervisors for partially observed discrete event systems. In *Proceedings of 53rd IEEE Conference on Decision and Control*, pages 5156–5162, 2014.

[111] X. Yin and S. Lafortune. Codiagnosability and coobservability under dynamic observations: Transformation and verification. *Automatica*, 61:241–252, 2015.

[112] X. Yin and S. Lafortune. A general approach for solving dynamic sensor activation problems for a class of properties. In *Proceedings of the 54th IEEE Conference on Decision and Control*, pages 3610–3615, 2015.

[113] X. Yin and S. Lafortune. Minimization of sensor activation in decentralized fault diagnosis of discrete event systems. In *Proceedings of the 54th IEEE Conference on Decision and Control*, pages 1014–1019, 2015.

[114] X. Yin and S. Lafortune. A new approach for synthesizing opacity-enforcing supervisors for partially-observed discrete-event systems. In *Proceedings of the 2015 American Control Conference*, pages 377–383, 2015.

[115] X. Yin and S. Lafortune. Decentralized supervisory control with intersection-based architecture. *IEEE Transactions on Automatic Control*, 61(11):3644–3650, 2016.

[116] X. Yin and S. Lafortune. A general approach for optimizing dynamic sensor activations for discrete event systems. *Automatica*, 2016. in preparation.

[117] X. Yin and S. Lafortune. On maximal permissiveness in partially-observed discrete event systems: Verification and synthesis. In *13th International Workshop on Discrete Event Systems*, pages 1–7, 2016.

[118] X. Yin and S. Lafortune. On the maximally-permissive range control problem in partially-observed discrete event systems. In *Proceedings of the 55th IEEE Conference on Decision and Control*, pages 3923–3928, 2016.

[119] X. Yin and S. Lafortune. On two-way observer and its application to the verification of infinite-step and K-step opacity. In *13th International Workshop on Discrete Event Systems*, pages 361–366, 2016.

[120] X. Yin and S. Lafortune. Synthesis of maximally permissive supervisors for partially observed discrete event systems. *IEEE Transactions on Automatic Control*, 61(5):1239–1254, 2016.

[121] X. Yin and S. Lafortune. A uniform approach for synthesizing property-enforcing supervisors for partially-observed discrete-event systems. *IEEE Transactions on Automatic Control*, 61(8):2140–2154, 2016.

[122] X. Yin and S. Lafortune. DPO-SYNT: Discrete control synthesis for partially-observed systems. In *20th IFAC World Congress*, 2017. accepted.

[123] X. Yin and S. Lafortune. Minimization of sensor activation in decentralized discrete event systems. *IEEE Transactions on Automatic Control*, 2017. under review.

[124] X. Yin and S. Lafortune. A new approach for the verification of infinite-step and k-step opacity using two-way observers. *Automatica*, 2017. In Print, DOI: 10.1016/j.automatica.2017.02.037.

[125] X. Yin and S. Lafortune. Synthesis of maximally-permissive supervisors for the range control problem. *IEEE Transactions on Automatic Control*, 2017. In Print, DOI: 10.1109/TAC.2016.2644867.

[126] X. Yin and Z.-J. Li. Decentralized fault prognosis of discrete event systems with guaranteed performance bound. *Automatica*, 69:375–379, 2016.

[127] S. Yokota, T. Yamamoto, and S. Takai. Computation of the delay bounds and synthesis of diagnosers for decentralized diagnosis with conditional decisions. *Discrete Event Dyn. Sys.: Theory & Appl.*, 21(1):45–84, 2017.

[128] T.-S. Yoo and S. Lafortune. A general architecture for decentralized supervisory control of discrete-event systems. *Disctre Event Dynamic Systems.: Theory & Application*, 12(3):335–377, 2002.

[129] T.-S. Yoo and S. Lafortune. NP-completeness of sensor selection problems arising in partially observed discrete-event systems. *IEEE Transactions on Automatic Control*, 47(9):1495–1499, 2002.

[130] T.-S. Yoo and S. Lafortune. Decentralized supervisory control with conditional decisions: Supervisor existence. *IEEE Transactions on Automatic Control*, 49(11):1886–1904, 2004.

[131] T.-S. Yoo and S. Lafortune. Decentralized supervisory control with conditional decisions: Supervisor realization. *IEEE Transactions on Automatic Control*, 50(8):1205, 2005.

[132] T.-S. Yoo and S. Lafortune. Solvability of centralized supervisory control under partial observation. *Discrete Event Dynamic Systems: Theory & Applications*, 16(4):527–553, 2006.

[133] J. Zaytoon and S. Lafortune. Overview of fault diagnosis methods for discrete event systems. *Annual Reviews in Control*, 37(2):308–320, 2013.