VERIFICATION AND ENFORCEMENT OF STATE-BASED NOTIONS OF OPACITY IN DISCRETE EVENT SYSTEMS

BY

ANOOSHIRAVAN SABOORI

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2010

Urbana, Illinois

Doctoral Committee:

Associate Professor Christoforos N. Hadjicostis, Chair
Professor Tamer Başar
Assistant Professor Todd P. Coleman
Professor P. R. Kumar
Associate Professor Ramavarapu S. Sreenivas

ABSTRACT

Motivated by security and privacy considerations in applications of discrete event systems, we describe and analyze the complexity of verifying various state-based notions of opacity in systems that are modeled as (possibly non-deterministic) finite automata with partial observation on their transitions. Assuming that the intruder observes system activity through some projection map and has complete knowledge of the system model, we define three notions of opacity with respect to a set of *secret states*: (i) **initial-state opacity** is a notion that requires the membership of the system true *initial state* to the set of secret states remain opaque (i.e., uncertain) to the intruder; (ii) K**-step opacity** is a notion that requires that *at any specific point in time within the last* K *observations*, the entrance of the system state to the given set of secret states remain opaque to the intruder; (iii) **infinite-step opacity** is a notion that requires the entrance of the system state *at any particular instant* to the set of secret states remain opaque, for the *length* of the system operation, to the intruder. As illustrated via examples in the thesis, the above state-based notions of opacity can be used to characterize the security requirements in many applications, including encryption using pseudo-random generators, coverage properties in sensor networks, and anonymity requirements in protocols for web transactions.

In order to model the intruder capabilities regarding initial-state opacity, we address the initial-state estimation problem in a non-deterministic finite automaton under partial observations on its transitions via the construction of an *initial-state estimator*. We analyze the properties and complexity of the initial-state estimator, and show how the complexity of the verification method can be greatly reduced in the special case when the set of secret states is *invariant* (i.e., it does not change over time). We also establish that the verification of initial-state opacity is a PSPACE-complete problem.

In order to verify K-step opacity, we introduce the K-delay state estimator

which constructs the estimate of the state of the system K observations ago (K-*delayed state estimates*) for a given non-deterministic finite automaton under partial observation on its transitions. We provide two methods for constructing K-delay state estimators, and hence two methods for verifying K-step opacity, and analyze the computational complexity of both. In the process, we also establish that the verification of K-step opacity is an NP-hard problem. We also investigate the role of the delay K in K-step opacity and show that there exists a delay K^* such that K-step opacity implies K'-step opacity for any K and K' such that $K' > K \geq K^*$. This is not true for arbitrary $K' > K$ though the converse holds trivially.

Infinite-step opacity can be verified via the construction of a *current-state estimator* and a bank of appropriate initial-state estimators. The verification of infinite-step opacity is also shown to be a PSPACE-hard problem.

Finally, we tackle the problem of constructing a minimally restrictive opacity-enforcing *supervisor* (MOES) which limits the system's behavior within some pre-specified legal behavior while enforcing opacity requirements. We characterize the solution to MOES, under some mild assumptions, in terms of the supremal element of certain *controllable*, *normal*, and *opaque* languages. We also show that this supremal element always exists and that it can be implemented using state estimators. The result is a supervisor that achieves conformance to the pre-specified legal behavior while enforcing opacity by disabling, at any given time, a subset of the controllable system events, in a way that minimally restricts the range of allowable system behavior.

To Anna, the love of my life.
To my parents, for their love and support.

ACKNOWLEDGMENTS

I am heartily thankful to my adviser, Professor Christoforos Hadjicostis, whose encouragement, supervision and support from the preliminary to the final stages of my doctoral research enabled me to develop an understanding of the subject. It would have been next to impossible to write this dissertation without his help and guidance.

I also would like to thank my parents for supporting me throughout my graduate studies. They were always there when I needed them.

Last, but not least, I would like to thank my wife, Anna, whose support made it possible for me to walk this long path.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Motivation and Overview

Motivated by the increased reliance on shared cyber-infrastructures in many application areas (ranging from defense and banking to health care and power distribution systems), various notions of *security and privacy* have received considerable attention from researchers. The work pursued so far can be roughly classified into two main approaches: the first approach focuses on carefully characterizing the intruder's capabilities, whereas the second one focuses on the *information flow* from the system to the intruder [1,2]. *Opacity* is a security notion that falls in the second category and aims at determining whether a given system's *secret* behavior (i.e., a subset of the behavior of the system that is considered critical and is usually represented by a predicate) is kept opaque to outsiders [3, 4]. More specifically, this requires that the intruder (modeled as a passive observer of the system's behavior) never be able to establish the truth of the predicate.

In this thesis, we study various notions of opacity with respect to predicates that are state-based. More specifically, we consider a scenario where we are given a discrete event system (DES) that can be modeled as a non-deterministic finite automaton with partial observation on its transitions. The intruder is assumed to have full knowledge of the system model and be able to track the occurrence of the observable transitions in the system. Assuming that the initial state of the system is (partially) unknown, we define, analyze, and describe three state-based notions of opacity as described below.

(i) **Initial-state opacity:** The secret behavior of the system is defined as the membership of its *initial* state to a set of secret states S [5]. This notion requires that the intruder never be certain that the initial state of the system

belonged to the set of secret states S, regardless of the activity that takes place in the system.

(ii) **K-step opacity ($K \geq 0$):** The secret behavior of the system is defined as the *evolution* of the system's state to a set of secret states S [4, 6] at any given time within the past K observations. Specifically, K-step opacity requires that, at any given time, the intruder cannot determine with certainty that the state of the system $0, 1, \ldots,$ or K observations ago belonged to the secret set of states S. The notion of K-step opacity is suitable for situations where the secrecy of some states becomes unimportant after the occurrence of a certain number of events (e.g., the passage of time).

(iii) **Infinite-step opacity:** The secret behavior of the system is defined as the *evolution* of the system's state to a set of secret states S [7] within the past observations (including the latest observation). Specifically, infinite-step opacity requires that, at any given time, the intruder cannot determine with certainty that the state of the system $0, 1, 2, \ldots,$ observations ago belonged to the secret set of states S. This is essentially the extension of K-step opacity as K approaches infinity.

There are many areas where state-based notions of opacity can be used to characterize security requirements. For example, state-based notions of opacity can be used to study conditions under which the key sequence generated by a pseudo-random generator in a cryptographic protocol can become compromised (e.g., because its initial state or its state at a particular point in time is revealed). Another example can be found in the context of coverage analysis of a mobile agent in a terrain equipped with cameras (that provide some partial coverage). Here, one can use the notion of K-step opacity to characterize paths that a mobile agent can follow without exposing the exact time(s) (measured with respect to the snap-shots provided by the cameras) at which the object goes through certain strategic (secret) areas. Some of these applications are discussed in more detail Chapter 3 and in Chapter 8.

As another example, consider the communication protocols for a bank transaction where the user communicates important account information in certain states and dummy information in others. This mechanism prohibits an eavesdropper from intercepting the packets that include important information — which can be used for *replay attack*. A replay attack is a network attack where valid data transmission is maliciously or fraudulently repeated or delayed [8]. The notion of infinite-step opacity can be used to verify

whether the given communication protocol indeed hides such important information from the eavesdropper.

Due to the *state-based* nature of the notions of opacity introduced in this thesis, one way to verify them is to construct appropriate *state estimators*. The problem of *state estimation* in discrete event systems along with its applications to fault diagnosis and control, has attracted significant research interest over the last two decades [9]–[16]. The main problem that has been studied consists of reconstructing all possible states that a known system can be in based on possibly limited knowledge of its initial state and partial knowledge of the sequence of events that occur in the system. The state estimation problem has found applications in diverse areas, including stabilizing supervisory control [15],[17], fault diagnosis [11],[16], interface design [13], and discrete event system inversion [18]. The notions of opacity introduced in this thesis require that the truth of a certain predicate on the system state cannot be determined by an outside observer for the duration of a certain time window (or for all times in the case of infinite-step opacity). Depending on the notion of opacity that is used, this predicate can be defined for states visited in the past (with no bound on how far into the past) or for states which have been visited a fixed number of observations in the past. In either case, existing state estimation techniques cannot verify these properties since they are tracking the current state but not the state trajectory or the previous states.

Motivated by such limitations, we study the problem of *state estimation* in discrete event systems. The basic state estimation setting we consider is the following: we are given a finite non-deterministic automaton with (partially) unknown initial state and partial *event* observation (without loss of generality we assume that no *state* observation is explicitly available — we can always incorporate any state information that is available by appropriately enhancing the underlying automaton [19]). We formally define the problems of initial-state estimation and K-delayed state estimation, and introduce the initial-state estimator and K-delay state estimator as respective solutions to these problems.

In order to show how state estimators can be used to verify opacity, we start in Chapter 5 by showing that a system is initial-state opaque if and only if all initial-state estimates (in its initial-state estimator) contain at least one state outside the set of secret states S. Initial-state opacity verification using

the initial-state estimator is shown to require space complexity that in the worst case can be exponential in the square of the number of states of the given finite automaton. For a fixed set of secret states S, we also propose a more efficient method for verifying initial-state opacity which requires space complexity that is exponential in the number of states of the given finite automaton (but is specific to the secret set of states S and, unlike the approach that uses the initial-state estimator, has to be repeated for a different set of secret states S). We also describe how the approach based on the initial-state estimator can be extended in certain settings where the set of secret states S is time-varying. The exponential complexity of the methods for verifying initial-state opacity is not desirable for implementation purposes; however, since in Chapter 6 we establish that the verification of initial-state opacity problem is PSPACE-complete for $|\Sigma_{obs}| > 1$ (where $|\Sigma_{obs}|$ denotes the number of observable events in the system), it is unlikely that the notion of initial-state opacity can be verified via a polynomial-time algorithm.

In Chapter 5, we also show that a system is K-step opaque if and only if all K-delayed state estimates (in its K-delay state estimator) contain at least one state outside the set of secret states S. K-step opacity verification using the K-delay state estimator is shown to require space complexity that in the worst case can be exponential in the square of the number of states of the given finite automaton and K. Again, the exponential complexity of this algorithm is not desirable; however, we show in Chapter 6 that the verification of the K-step opacity problem is NP-hard for $|\Sigma_{obs}| > 1$.

Finally, in oder to verify infinite-step opacity, we show in Chapter 5 that for any $K \geq 2^{N^2} - 1$ (where N is the number of states of the discrete event system) K-step opacity and infinite-step opacity become equivalent; hence one can construct the K-delay state estimator, with $K = 2^{N^2} - 1$, to verify infinite-step opacity. We also introduce a reduced-complexity method to verify infinite-step opacity using the *current-state estimator* and a bank of initial-state estimators. In Chapter 6, we establish that the verification of the infinite-step opacity problem is PSPACE-hard for $|\Sigma_{obs}| > 1$.

In order to show the applicability of the theoretical developments of this thesis to real-sized problems, we also employ existing tools and appropriate transformations to implement the algorithm for verifying infinite-step opacity using a current-state estimator and a bank of initial-state estimators. We use this implementation in Chapter 8 to analyze tracking problems in some

representative sensor networks.

A natural question that follows the verification problem is the enforcement problem: in case the given system is not opaque, are there ways to enforce opaqueness? There are many ways one can answer such questions depending on the available *control* (or enforcement) mechanisms that the supervisor can use to remove behaviors from the system that violate opacity. Examples of such control mechanisms include: (i) disabling *controllable* events (which are essentially a subset of system events that can be controlled by the *supervisor*); (ii) changing the observability of events (where the supervisor can force some sequences of events in the system to look alike so that the secret behavior is concealed). In this thesis, we assume that the available control mechanism is capable of disabling controllable events and study how it can be used to enforce various notions of opacity. More specifically, in Chapter 7, we consider the problem of designing a (minimally restrictive) supervisor which (i) limits the system's behavior within some pre-specified legal behavior, and (ii) enforces (either initial-state or infinite-step) opacity requirements by disabling, at any given time, the least possible number of events. For enforcing initial-state opacity, we establish that the set of solutions can be characterized as the intersection of controllable, normal, and *opaque* languages. Using this characterization, we then show that the solution to our problem is the supremal element of such languages. We argue that, under some mild assumptions, the supremal element exists, and we derive a formulation for it. Moreover, assuming that the given legal behavior is *regular* (i.e., it can be described via a finite automaton), we show that the supremal element is also regular. Finally, we propose a procedure that uses an appropriate state estimator to implement this supremal element, effectively integrating the verification and control problems. For enforcing infinite-step opacity, we leverage the results on initial-state opacity and build a finite bank of supervisors that implement minimally restrictive supervisory strategies.

1.2 Related Work

The developments in this thesis are related to existing security work in the area of DESs. In particular, [20] and [21] focus on finite Petri nets and define opacity with respect to state-based predicates; our work here essentially (i)

introduces the notions of K-step and infinite-step opacity (not present in either [3], [20], or [21]) and (ii) studies and solves these problems (as well as the problem of initial-state opacity) for the case of finite automata.

The authors of [22] consider multiple intruders modeled as observers with different observation capabilities (namely different natural projection maps) and require that no intruder be able to determine that the actual trajectory of the system belongs to the secret language assigned to that intruder. Assuming that the supervisor can observe/control all events, the authors establish sufficient conditions for the existence of a supervisor with a finite number of states (i.e., a regular supervisor) that enforces this opacity requirement for all observers of the system. In particular, it is shown that the optimal supervisor in this setting always exists but is not guaranteed to be regular. The assumptions on the full controllability and observability of events by the supervisor are partially relaxed in [23] where the authors consider a single intruder that might observe different events than the ones observed/controlled by the supervisor. Under these assumptions, [23] establishes that a minimally restrictive supervisor always exists, but its regularity depends on the relationship between the set of events observed by the intruder and the sets of events observed/controlled by the supervisor. In contrast to [22] and [23], opacity in our framework assumes that the states of the system can be partitioned into *secret* and *non-secret* ones; this state-based formulation is what enables us to use various state estimators to verify opacity. Also, note that the notions of opacity introduced here are not considered in [22] and [23] and (as explained in more details in Chapter 3 of the thesis) they cannot be readily captured by the framework of [22, 23]. The supervisor introduced in this work to enforce initial-state opacity can certainly be formulated in terms of the language framework of [23]. However, our approach in Chapter 7 of this thesis leads to a closed form expression for the optimal solution which is not present in [23] and also allows us to use a state estimator to synthesize the supervisor.

Related to our work here is also the work in [24] where the authors partition the event set into public level and private level events, and consider the verification of *intransitive non-interference*, a property that captures the allowed information flow (e.g., the occurrence of certain events) from private level events to public level events through a downgrading process. Moreover, the authors of [25] consider the problem of designing a minimally restric-

tive supervisor which enforces non-interference for automata when there is no downgrading process. Our model of the intruder's capability (in terms of observability power) is different from [24] and [25] which makes the two frameworks incomparable. As we show in Chapter 3, when there is no downgrading process, the notion of non-interference can be translated to an instance of 0-step opacity (*current-state opacity*); however, in general, one cannot formulate the state-based notions of opacity in the framework of [24] or [25]. Since the results of Chapter 7 of this thesis on designing supervisory control can be easily extended to design minimally restrictive supervisors that enforce 0-step opacity (see Chapter 7), the synthesis of a supervisor that enforces non-interference in [25] can also be generated using the framework of this thesis.

The authors of [26] model the intruder as a non-passive entity which can override the decision of the supervisor to disable certain events; they are concerned with the problem of intrusion detection and derive sufficient conditions under which a supervisor can detect the presence of an intrusion. Compared to [26], the intruder in our framework is passive (modeled as an observer) and cannot change the system configuration or model.[1] Also note that, unlike [26], we are not seeking to *avoid* states as long as we can guarantee that entrance to these states retains opacity.

[1]Note that the intruder in our framework can actually be allowed to interfere with uncontrollable events, without having to adjust any of the developments we present. In fact, as long as the intruder is capable of only disabling events that are enabled by the supervisor (but not enabling events that are disabled by the supervisor), the development that we present here still guarantees the enforcement of opacity.

CHAPTER 2

BACKGROUND

2.1 Languages and Automata

Let Σ be an alphabet and denote by Σ^* the set of all finite-length strings of elements of Σ, including the empty string ϵ. For any string t, $|t|$ denotes the length of t (with $|\epsilon|$ taken to be zero). A language $L \subseteq \Sigma^*$ is a subset of finite-length strings from strings in Σ^*. A language is finite if it contains only a finite number of strings. We say that a finite language L is of length K if the maximum length of the strings in L is K. For a string ω, $\overline{\omega}$ denotes the *prefix-closure* of ω and is defined as $\bar{\omega} = \{t \in \Sigma^* | \ \exists s \in \Sigma^* \{ts = \omega\}\}$, where ts denotes the concatenation of strings t and s. The prefix closure $\overline{L}$ of language L is the union of all prefix closures of all strings in L. A language is prefix-closed if $L = \overline{L}$. The post-string ω/t of ω after $t \in \bar{\omega}$ is defined as $\omega/t = \{s \in \Sigma^* | \ ts = \omega\}$. The concatenation $L_1 L_2$ of two languages L_1 and L_2 is defined as $L_1 L_2 = \{ts | t \in L_1, s \in L_2\}$ [27].

A DES is modeled in this thesis as a non-deterministic finite automaton $G = (X, \Sigma, \delta, X_0)$, where $X = \{0, 1, \ldots, N-1\}$ is the set of states, Σ is the set of events, $\delta : X \times \Sigma \rightarrow 2^X$ (where 2^X is the power set of X) is the non-deterministic state transition function, and $X_0 \subseteq X$ is the set of initial states. The function δ can be extended from the domain $X \times \Sigma$ to the domain $X \times \Sigma^*$ in the routine recursive manner: $\delta(i, ts) := \bigcup_{j \in \delta(i,t)} \delta(j, s)$, for $t \in \Sigma$ and $s \in \Sigma^*$ with $\delta(i, \epsilon) := i$. The behavior of DES G is captured by $L(G) := \{s \in \Sigma^* \mid \exists i \in X_0 \{\delta(i, s) \neq \emptyset\}\}$. We use $L(G, i)$ to denote the set of all traces that originate from state i of G (so that $L(G) = \bigcup_{i \in X_0} L(G, i)$). The prefix-closed language E is regular if there exists a finite automaton G such that $L(G) = E$ [19, 28].

The product of two non-deterministic automata $G_1 = (X_1, \Sigma_1, \delta_1, X_{01})$ and $G_2 = (X_2, \Sigma_2, \delta_2, X_{02})$ is the automaton $G_1 \times G_2 := AC(X_1 \times X_2, \Sigma_1 \cap$

$\Sigma_2, \delta_{1\times 2}, X_{01} \times X_{02})$ where $\delta_{1\times 2}((i_1, i_2), \alpha) := \delta_1(i_1, \alpha) \times \delta_2(i_2, \alpha)$ for $\alpha \in \Sigma_1 \cap \Sigma_2$, and AC denotes the accessible part of the automaton (i.e., the set of states reachable from the set of initial states via some string $s \in \Sigma^*$ where $\Sigma = \Sigma_1 \cap \Sigma_2$). The construction of the product automaton implies that $L(G_1 \times G_2) = L(G_1) \cap L(G_2)$ [27].

In general, only a subset Σ_{obs} of the events can be observed. Typically, one assumes that Σ can be partitioned into two sets, the set of observable events Σ_{obs} and the set of unobservable events Σ_{uo} (so that $\Sigma_{obs} \cap \Sigma_{uo} = \emptyset$ and $\Sigma_{obs} \cup \Sigma_{uo} = \Sigma$). The natural projection $P : \Sigma^* \to \Sigma_{obs}^*$ can be used to map any trace executed in the system to the sequence of observations associated with it. This projection is defined recursively as $P(ts) = P(t)P(s)$, $t \in \Sigma, s \in \Sigma^*$, with

$$P(t) = \begin{cases} t, & \text{if} \quad t \in \Sigma_{obs}, \\ \epsilon, & \text{if} \quad t \in \Sigma_{uo} \cup \{\epsilon\}. \end{cases}$$

More general projections of the form $P : \Sigma \to \Delta \cup \{\epsilon\}$ that may map multiple events to a label in the set $\Delta \cup \{\epsilon\}$ can also be incorporated in our development in a straightforward manner. To keep notation simple we only discuss the natural projection.

2.2 State Mappings

Given a non-deterministic finite automaton $G = (X, \Sigma, \delta, X_0)$, X^K $(K \geq 2)$ denotes the set of K-tuples of states of DES G, i.e., $X^K := X \times X \times \ldots \times X = \{(j_1, \ldots, j_K) | j_k \in X\}$, $1 \leq k \leq K$. We call $m \subseteq X^K$ a K-dimensional state mapping.

The set of states included as the first (last) component in a K-dimensional state mapping m is called the set of starting (ending) states of m and is denoted by $m(K-1)$ (by $m(0)$). We also denote by $m(k), 0 < k < K-1$, the set of intermediate states in the K-tuple, i.e.,

$$m(k) = \{j_{K-k} | (j_1, \ldots, j_K) \in m\}.$$

A 2-dimensional state mapping is referred to as a state mapping.We say that

state mapping m_1 *refines* m_2 if the set of starting states $m_1(1)$ is a subset of the set of starting states $m_2(1)$, i.e., $m_1(1) \subseteq m_2(1)$. Moreover, if mapping m_1 refines m_2 and mapping m_2 refines m_1, then we say that mapping m_1 is *consistent* with m_2 (in this case, $m_1(1) = m_2(1)$). For $m = \emptyset$, we define $m(1) = m(0) = \emptyset$.

We define the *shift* operator $>>$: $2^{X^K} \times 2^{X^2} \rightarrow 2^{X^K}$ for a K-dimensional state mapping $m_1 \in 2^{X^K}$ and a state mapping $m_2 \in 2^{X^2}$ as

$$m_1 >> m_2 := \{(j_2, \ldots, j_K, j_{K+1}) | (j_1, j_2, \ldots, j_K) \in m_1, (j_K, j_{K+1}) \in m_2\}.$$

We also define the composition operator $\circ : 2^{X^2} \times 2^{X^2} \rightarrow 2^{X^2}$ for state mappings $m_1, m_2 \in 2^{X^2}$ as

$$m_1 \circ m_2 := \{(j_1, j_3) | \exists j_2 \in X \{(j_1, j_2) \in m_1, (j_2, j_3) \in m_2\}\}.$$

Note that the shift operator takes as input two sets of tuples: the first set involves K-tuples $K \geq 2$, and the second set involves tuples of size two; for each K-tuple of the first set, all 2-tuples in the second set whose first element is the same as the last element of the K-tuple from the first set are used to produce an output K-tuple by using the 2^{nd}, 3^{rd}, ..., K^{th} elements of the first K-tuple, and the second element of the second 2-tuple. The composition operator, on the other hand, takes as input two sets of 2-tuples and produces as output another set of 2-tuples by including all 2-tuples with the first element borrowed from a 2-tuple in the first input set and the second element borrowed from a 2-tuple in the second set, as long as these two 2-tuples share the same second/first element (as done in the case of the shift operator). Note that the composition operator is only defined for tuples of size two. For any $Z \subseteq X$ and $K \geq 2$, we define the operator $\odot_K : 2^X \rightarrow 2^{X^K}$ as $\odot_K(Z) = \{(i, i, \ldots, i) | i \in Z\}$ where the tuples involve K identical elements.

Given a non-deterministic finite automaton $G = (X, \Sigma, \delta, X_0)$ and a natural projection map P with respect to the set of observable events Σ_{obs} ($\Sigma_{obs} \subseteq \Sigma$), we can map any sequence of observations $\omega \in \Sigma_{obs}^*$ (of finite but arbitrary length) in DES G to a state mapping by using the mapping $M : \Sigma_{obs}^* \rightarrow 2^{X^2}$ such that the 2-tuple $(i, j) \in M(\omega)$ if and only if there exists a sequence of events that starts from i and ends in j, and produces observation ω. We call

$M(\omega)$ the ω-induced state mapping and define it formally below.

Definition 2.2.1 (ω-Induced State Mapping)**.** *Given a non-deterministic finite automaton $G = (X, \Sigma, \delta, X_0)$ and a natural projection map P with respect to the set of observable events Σ_{obs} ($\Sigma_{obs} \subseteq \Sigma$), the ω-induced state mapping after observing the sequence of observations $\omega \in \Sigma_{obs}^*$ is defined as*

$$M(\omega) = \{(i,j) | i, j \in X, \exists t \in \Sigma^* \{P(t) = \omega, j \in \delta(i,t)\}\}.$$

We define $M(\epsilon) = \odot_2(X)$. Note that $M(\omega) = \emptyset$ denotes the fact that the sequence of observations ω is not feasible in DES G. ■

The following proposition can be easily shown.

Proposition 2.2.1. *Given a non-deterministic finite automaton $G = (X, \Sigma, \delta, X_0)$ and a natural projection map P with respect to the set of observable events Σ_{obs} ($\Sigma_{obs} \subseteq \Sigma$), the ω-induced state mapping satisfies the following for $\omega_1, \omega_2 \in \Sigma_{obs}^*$: $M(\omega_1\omega_2) = M(\omega_1) \circ M(\omega_2)$.* ■

Generalizing the notion of the ω-induced state-mapping, we map a sequence of observation $\omega = \alpha_0\alpha_1 \ldots \alpha_{|\omega|-1}$ in DES G to a $(|\omega|+1)$-dimensional state mapping via the mapping $T_{|\omega|} : \Sigma_{obs}^{|\omega|} \to 2^{X^{|\omega|+1}}$ as

$$T_{|\omega|}(\omega) = \{(j_0, j_1, \ldots, j_{|\omega|}) \in X^{|\omega|+1} | \forall l (0 \leq l \leq |\omega| - 1) \exists t_l \in \Sigma^* \{P(t_l) = \alpha_l, \\ j_{l+1} \in \delta(j_l, t_l)\}\}$$

which we call the ω-induced $(|\omega| + 1)$-dimensional state mapping. Note that $T_{|\omega|}(\omega) = \emptyset$ denotes the fact that the sequence of observations ω is not feasible in DES G.

CHAPTER 3

STATE-BASED NOTIONS OF OPACITY

In this chapter, we consider discrete event systems that are modeled as nondeterministic finite automata under a natural projection map P, and we define and motivate three state-based notions of opacity: initial-state opacity, K-step opacity, and infinite-step opacity. We motivate these notions by considering representing applications in the security/privacy domains. For instance, we show that the conditions under which the key sequence generated by a pseudo-random generator in a cryptographic protocol becomes compromised (e.g., because its initial state or its state at a particular point in time is revealed) can be formulated and analyzed using state-based notions of opacity. We also consider tracking problems in sensor networks and motivate the application of state-based notions of opacity in such contexts. At the end of this chapter, we also discuss the relation between the state-based notions of opacity introduced in this chapter and other related notions in discrete event systems, including non-interference, diagnosability, and unique input/output sequences.

3.1 Initial-State Opacity

Initial-state opacity requires that the membership of the system initial state to the set of secret states (denoted by S) remains opaque to an intruder (outside observer) who is observing the events that occur in the system through a natural projection map P. In other words, for a system to be initial-state opaque, we require that the intruder be unable to determine with certainty that the initial state of the system belonged to the set of secret states S. Note that our definition of initial-state opacity allows the intruder to deter-

mine that the system started from a non-secret initial state.[1] The following definition describes initial-state opacity formally.

Definition 3.1.1 (Initial-State Opacity)**.** *Given a non-deterministic finite automaton $G = (X, \Sigma, \delta, X_0)$, a natural projection map P with respect to the set of observable events Σ_{obs} ($\Sigma_{obs} \subseteq \Sigma$), and a set of secret states $S \subseteq X$, automaton G is initial-state opaque with respect to S and P (or (S, P, ∞) initial-state opaque), if for all $i \in X_0 \cap S$ and for all $t \in L(G, i)$ we have*

$$\exists j \in X_0 - S, \exists s \in L(G, j)\{P(s) = P(t)\}. \tag{3.1}$$

■

According to Definition 3.1.1, DES G is (S, P, ∞) initial-state opaque if for every string t that originates from a system initial state in the set of secret states S, there exists a string s that originates from a system initial state outside the set of secret states S and has the same projection as t.

3.2 K-Step Opacity

The notion of K-step opacity is suitable for cases when, following K observations, one does not mind if an intruder can infer information about secret states (e.g., because the secret transaction has completed or because the intrusion will be detected). The formal definition of K-step opacity (refer to Figure 3.1) is provided below.

Definition 3.2.1 (K-Step Opacity)**.** *Given a non-deterministic finite automaton $G = (X, \Sigma, \delta, X_0)$, a natural projection map P with respect to the set of observable events Σ_{obs} ($\Sigma_{obs} \subseteq \Sigma$), and a set of secret states $S \subseteq X$, automaton G is K-step opaque (for a nonnegative integer K) with respect to S and P (or (S, P, K)-opaque), if for all $t \in \Sigma^*$, $t' \in \bar{t}$, and $i \in X_0$,*

$$\{|P(t)/P(t')| \leq K, \exists j \in S\{j \in \delta(i, t'), \delta(j, t/t') \neq \emptyset\}\}$$

[1] This is because non-secret initial-states do not contain secret information; thus, revealing that the system started from a non-secret state does not expose any critical information to the intruder. Note, however, that one can easily address the case when exposing that the system started from a non-secret state is also considered critical information.

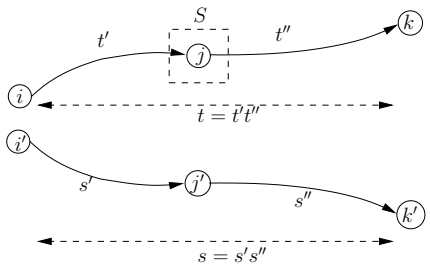

Figure 3.1: Graphical representation of the notation used in Definition 3.2.1 ($|P(t'')| = |P(s'')| \leq K$).

$$\Rightarrow \{\exists s \in \Sigma^*, \exists s' \in \bar{s}, \exists i' \in X_0, \exists j' \in \delta(i', s')\{P(s) = P(t),$$

$$P(s') = P(t'), j' \in X - S, \delta(j', s/s') \neq \emptyset\}\}.$$

■

Note that 0-step opacity is a special case of of the above definition when $K = 0$. In this thesis, we use 0-step opacity and *current-state opacity* interchangeably. For $t, s \in L(G)$ with $P(s) = P(t)$ we say that t passes through state j when s passes through state j' if there exists $t' \in \bar{t}$, $s' \in \bar{s}$, and $i, i' \in X_0$ such that $j \in \delta(i, t')$, $j' \in \delta(i', s')$ while (i) $P(t') = P(s')$ and (ii) t/t' and s/s' have continuations from states j and j', respectively. According to Definition 3.2.1, DES G is (S, P, K)-opaque if for every string t in $L(G)$ that visits a state j in S within the past K observations (and has a continuation from j), there exists a string s in $L(G)$ with $P(s) = P(t)$ such that when string t passes through the state j in S, string s passes through a state j' in $X - S$ (and has a continuation from j'). Note that s could be the same as t, in which case t would be passing through both secret and non-secret states.

3.3 Infinite-Step Opacity

K-step opacity requires opacity for only K observations since the last entrance of the system to the set of secret states S. This notion is suitable for cases where there is a bounded delay, after which one does not care if the intruder can infer information about the behavior (states) that was (were) considered previously secret. However, in many applications the existence of such bound might not be viable. For this reason, we extend the definition of

(S, P, K)-opacity to cases where K approaches infinity. We call this notion infinite-step opacity and define it formally in the following definition.

Definition 3.3.1 (Infinite-Step Opacity)**.** *Given a non-deterministic finite automaton $G = (X, \Sigma, \delta, X_0)$, a natural projection map P with respect to the set of observable events Σ_{obs} ($\Sigma_{obs} \subseteq \Sigma$), and a set of secret states $S \subseteq X$, automaton G is infinite-step opaque with respect to S and P (or (S, P, ∞)-opaque), if for all $t \in \Sigma^*$, $t' \in \bar{t}$, and $i \in X_0$,*

$$\{\exists j \in S\{j \in \delta(i, t'), \delta(j, t/t') \neq \emptyset\}\}$$

$$\Rightarrow \{\exists s \in \Sigma^*, \exists s' \in \bar{s}, \exists i' \in X_0, \exists j' \in \delta(i', s')\{P(s) = P(t),$$

$$P(s') = P(t'), j' \in X - S, \delta(j', s/s') \neq \emptyset\}\}.$$

■

According to Definition 3.3.1, DES G is (S, P, ∞)-opaque if for every string t in $L(G)$ that visits a state j in S (and has a continuation from j), there exists a string s in $L(G)$ with $P(s) = P(t)$ such that when string t passes through the state j in S, string s passes through a state j' in $X - S$ (and has a continuation from j'). Note that s could be the same as t, in which case t would be passing through both secret and non-secret states.

3.4 Motivational Examples

The state-based notions of opacity introduced in this chapter can be used to describe various desirable properties in security applications where vital information is associated with the states of the system and needs be kept secret from the intruder for the duration of system operation. Applications include encryption using key strings provided by pseudo-random generators and coverage properties of mobile agents in sensor networks. In this section, we motivate the aforementioned state-based notions of opacity using examples from these contexts.

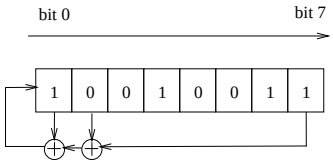

Figure 3.2: A conventional 8-bit LFSR with tapped bits 0,1,7 and seed (initial state) 10010011.

3.4.1 Encryption using Pseudorandom Generators

In cryptography, a symmetric cipher combines plain text bits (original information) with a pseudo-random bit stream (key-stream), typically using a bitwise XOR operation. For example, message 1010 XOR-ed with key-stream 0100 results in the encrypted message 1110. Knowledge of the encrypted message does not reveal the plain text unless the key-stream is compromised. To create the key-stream, one often uses a linear feedback shift register (LFSR) or some other type of pseudo-random number generator. An LFSR (Figure 3.2) is an autonomous shift register whose input bit (on the left in Figure 3.2) is obtained by XOR-ing some predefined combination of the bits that are stored in the shift register (this implies that the input bit is a linear function — in $GF(2)$ — of the LFSR's previous state). The initial state of the LFSR is called the seed, and the bits that affect the next input bit (and thus the next state) are called the tapped bits. The taps are XOR-ed sequentially and then fed back into the register as the input bit. Figure 3.2 shows an 8-bit LFSR with tapped bits 0, 1, 7, and seed 10010011. The output of the LFSR is usually the last bit shifted out from the shift register.

Because the operation of the register is deterministic, the sequence of values produced by the register (which is used to generate the key-stream for the stream cipher) is completely determined by its seed. For example, given that the seed (initial state) of the LFSR in Figure 3.2 is 10010011, the next output is 1 (i.e., the rightmost bit shifted out) and the next state of the LFSR becomes 01001001 (because the incoming bit is given by $1 \oplus 0 \oplus 1 = 0$, and the rest of the bits are the seven leftmost bits of 10010011 with the rightmost bit shifted out). Note that the register has a finite number of possible states (2^8 states in the example of Figure 3.2), so it must eventually enter a repeating cycle. An LFSR with a well-chosen feedback function (taps) and initial state can have a very long cycle and can produce a sequence

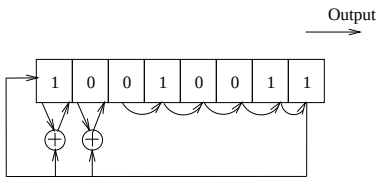

Figure 3.3: Operation of LFSR when the clock mechanism is active.

of bits which appears[2] random. Alternative structures to the conventional LFSR do exist; for example, the LFSR structure proposed in [29] contains a clock mechanism (Figure 3.3). When the system is clocked, bits that are not tapped are shifted as normal to the next flip-flop. The tapped bits, on the other hand, are XOR-ed with the new output before the resulting bits are shifted. For example, in Figure 3.3, the value of the shift register, before the clock is enabled, is 10010011; upon clocking, the output of the shift register becomes 1, which is the value of the rightmost bit that is shifted out; after that, the value of the shift register becomes 10101001. Note that bits 2, 3, 4, 5, 6, and 7 are shifted to the right as in the conventional LFSR (bit 7 simply becomes the output bit); bits 0 and 1 are XOR-ed with the output of the LFSR (which is 1) before they are shifted (to take positions 1 and 2). Also note that bit 0 in this case is simply the rightmost bit before clocking. When the clock is not active, the shifting mechanism remains identical to that of the conventional LFSR. For example, in the previous scenario, if the clock was not active, the next state of the shift register would have been 01001001 (since the tapped bits are bits 0, 1, and 7). Clearly, by manipulating the activation of the clock mechanism, one can create more complex behavior than that of an unclocked LFSR.

An intruder can interact with protocols that use (clocked) LFSRs by inserting some plain text and by observing the ciphered text in order to find the seed. Note that finding the seed is equivalent to finding the stream of the keys used to encrypt all previous messages. Hence, if the intruder records all of the (encrypted) conversation, after finding the seed, she/he can go back and decrypt them using the key-stream. As we will see, many of the security concerns about this protocol can be recast in the framework of this thesis; for example, the question of whether there is a seed for which there exists

[2]Clearly, initial state 00000000 would not be a good choice in the example of Figure 3.2, regardless of the choice of tapped bits.

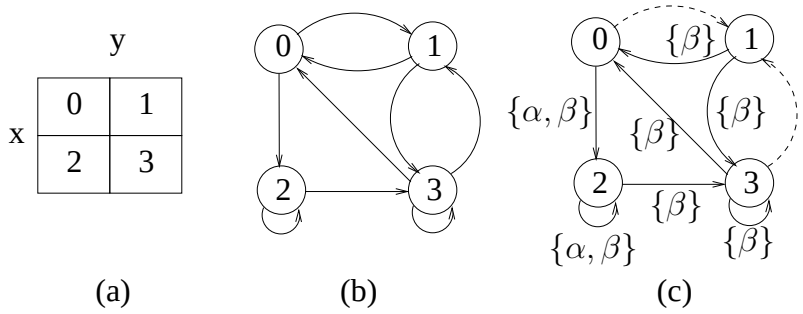

Figure 3.4: (a) 2-dimensional grid in which a vehicle can move; (b) kinematic model for a vehicle in the grid in (a); (c) automaton model of the vehicle kinematic model and the corresponding sensor readings (there are two sensors, α and β, with coverage areas within cell 2, and within cells 0, 2 and 3, respectively).

a sequence of inputs that reveals that seed, can be answered by formulating the problem as a set of initial-state opacity problems, each with a different set of secret initial states (as we will see in Chapter 5, this set of problems can be solved via the construction of a single initial-state estimator). Note that if there is such a seed, one might be interested in how long (in terms of input size) it takes for the intruder to detect it. An answer to this question can be obtained via a problem formulation that involves K-step opacity.

3.4.2 Tracking of Mobile Agents using Sensor Networks

Consider a vehicle capable of moving in a space modelled as a two-dimensional array of cells (grid). In Figure 3.4-a we provide a toy example of a 2×2 grid. The state of the vehicle corresponds to the coordinates (x, y) (or cell number) of its location in this grid, and any trajectory that the vehicle follows corresponds to a sequence of states. Clearly, the origin of the trajectory is captured by the initial state of the vehicle.

In order to capture vehicle movement limitations (due to physical obstacles or constraints in the vehicle motion), we assume that the vehicle possible movements are available via a kinematic model, i.e., a finite state machine whose states are associated with the state (position) of the vehicle and whose transitions correspond to the possible movements of the vehicle at each position (up, right, diagonal, etc.). Figure 3.4-b depicts an example of a kinematic model H for a vehicle that moves in the grid of Figure 3.4-a. This model

captures the various restrictions on the vehicle movement: for example, the kinematic model H indicates that the vehicle can go from cell 0 to cell 2 and potentially park itself there (i.e., stay in cell 2 for consecutive time instants), but it cannot go directly from cell 2 to cell 0. Similarly, the vehicle can commute between cells 0 and 1 indefinitely, but it cannot park in any of these cells (it cannot remain in the same cell for consecutive instants).

We assume that a number of sensors are deployed in the grid. Typically, the sensor network will not capture all movements of the vehicle and hence the observation of movements will be partial. Each sensor detects the presence of the vehicle in a cell or in some aggregation of cells, and it emits a signal to indicate that a vehicle passes through a cell within its coverage. However, sensors cannot determine the exact cell at which the vehicle resides within their coverage area. In order to model sensor readings, we can enhance the kinematic model by assigning label α to all transitions that end in a cell within the coverage area of sensor α. Since sensor coverage may overlap, the label of transitions ending in areas which are covered by more than one sensor can be chosen to be a special label that indicates the set of all sensors covering that location. In Figure 3.4-c, assuming that the coverage area of sensor α only includes cell 2, and that the coverage area of sensor β includes cells 0, 2 and 3, we depict the (non-deterministic) automaton G that models both the kinematic model of the vehicle and the corresponding sensor readings. Dotted arrows correspond to (unobservable) transitions to locations that are not covered by any sensor.

One of the questions that might arise in the above context is that of characterizing all the trajectories (sequences of states) that a vehicle can follow such that the passage of each trajectory from specific locations remains ambiguous to the sensor network. These trajectories can be of interest for a variety of reasons. For example, they can be employed to hide the origin of a trajectory (initial-state) from an intruder who is employing the sensor network (i.e., who is observing the labels in Figure 3.4-c) trying to identify whether the vehicle originated from a set of secret (strategically important) locations or whether the vehicle passed from this particular set of locations at some instant of time. Questions regarding the *passage* of a trajectory from specific locations, in general, can be answered using the infinite-step opacity framework, whereas questions regarding the *origin* of a trajectory can be answered using the initial-state opacity framework of this thesis.

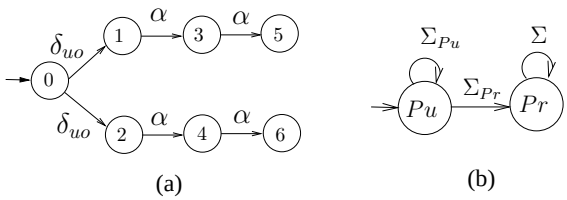

Figure 3.5: (a) DES G in Sections 3.5.1 and 3.5.3; (b) DES H in Section 3.5.2.

3.5 Existing Related Notions

3.5.1 Trajectory-Based K-Step Opacity

The DES G in Figure 3.5-a is 2-step opaque with respect to $S = \{1, 6\}$; however, upon observing $\alpha\alpha$, the intruder is certain that, regardless of the state sequence that has occurred, the system has visited a secret state within the last 2 observations (although one cannot determine exactly when this happened). This system can be considered as insecure if the attacker is only interested in determining whether the system has reached secret states at any point during the last $K = 2$ observations. We refer to a system for which this scenario does not occur as a *trajectory-based K-step opaque* system.

It can be easily seen that a system that is trajectory-based K-step opaque is also K-step opaque; however, as the preceding example demonstrated, the converse is not necessarily true. In other words, K-step opacity is a weaker condition than trajectory-based K-step opacity. Note that the essential difference between K-step opacity and trajectory-based K-step opacity is the time at which the state of the system is exposed. Depending on the application, K-step opacity might be a more suitable requirement than trajectory-based K-step opacity for characterizing security requirements. For instance, suppose the DES G in Figure 3.5-a is a communication protocol for a bank transaction where a user has two options: communicate important account information while at state 1 (secret state) and dummy information while at states 3 and 5 (non-secret states), or communicate dummy information at states 2 and 4 (non-secret states) and important account information while

at state 6 (secret state). If an eavesdropper does not know which of the two options the user has followed (due to the unobservable event δ_{uo}), then (even though she/he knows that important account information has been communicated) she/he does not know when this was done. Therefore, the fact that the system is 2-step opaque is critical (despite the fact that the system is not trajectory-based 2-step opaque).

As another example, consider the pseudo-random generator introduced in Section 3.4.1 that is used for generating a key string in encryption applications. As mentioned earlier, such a pseudo-random generator is usually implemented as an autonomous finite machine that cycles through a large number of states. In such case, knowing that the system was in a particular state at a specific point in the past (as captured by K-step opacity) is indeed important because this exposes the subsequent sequence of states and thus the key string used for encryption. On the other hand, knowing that the system has been in a particular state in the recent past (as captured by trajectory-based K-step opacity) offers little information (in fact, it offers zero information if K is larger than the number of states of the pseudo-random generator).

Another example can be found in the context of coverage analysis of a mobile agent in a terrain equipped with cameras (that provide some partial coverage) which was discussed in Section 3.4.2. Here, one can use the notion of K-step opacity to characterize paths that a mobile agent can follow without exposing the exact time(s) (measured with respect to the snap-shots provided by the cameras) at which the object goes through certain strategic (secret) areas. Again, depending on the underlying application, knowledge that the agent has been through a secret state might not be important if the exact timing is not precisely known.

3.5.2 Non-Interference

The notion of non-interference is defined assuming that the set of events Σ can be partitioned into a finite number of security levels [24,25]. When there are only two levels of security, i.e., public events Σ_{Pu} and private events Σ_{Pr} (so that $\Sigma = \Sigma_{Pu} \cup \Sigma_{Pr}$ and $\Sigma_{Pu} \cap \Sigma_{Pr} = \emptyset$), non-interference requires that a user who only observes public events cannot determine the occurrence of any

private event(s) [24, 25]. More specifically, in a *non-interferent* system that is typically defined as a deterministic finite automaton $G = (X, \Sigma, \delta, X_0)$, for every string in $L(G)$ containing private events, there exists another string in $L(G)$ with the same substring of public events and without private events. The notion of non-interference for this scenario can be easily translated to an instance of current-state opacity as follows: we first obtain the product $\hat{G} = G \times H$ where the deterministic automaton H is depicted in Figure 3.5-b (this product essentially splits and annotates the states of G with label Pr if they are reached from a state with label Pr or via a private event, and with label Pu, otherwise); if we define the initial state of $\hat{G}$ to be $X_0 \times \{Pu\}$ and the set of secret states S to be the set of states in $\hat{G}$ with label Pr, i.e., $S = \{(x, Pr) | x \in X\}$, then it is not hard to argue that G is non-interferent if and only if $\hat{G}$ is $(S, P, 0)$-opaque (i.e., current-state opaque) where the projection map P is with respect to the set of observable events $\Sigma_{obs} = \Sigma_{Pu}$. Note that, in general, the reverse is not possible: we cannot translate each instance of a K-step opacity (or even a current-state opacity) problem to an instance of a non-interference problem.

3.5.3 Language-Based Opacity

In the framework considered in [22] and [23], the set of strings $L(G)$ in the system is partitioned into secret strings E and non-secret strings $L(G) - E$, and the system is opaque if $P(E) \subseteq P(L(G) - E)$. The notion of initial-state opacity introduced in this chapter can be translated to a version of opacity as studied in [22] and [23] by defining the secret language to be

$$E = \bigcup_{i \in S \cap X_0} L(G, i).$$

Using this approach, DES G is (S, P, ∞) initial-state opaque if and only if it is opaque with respect to E.

The notion of K-step opacity cannot be easily translated to a version of language-based opacity as studied in [22] because in the framework we consider here, each string in the system can serve as both secret (when it visits secret states during the past K observations) and non-secret (when it does not visit any secret state during the past K observations). To clarify

this issue consider the DES G in Figure 3.5-a and assume that $X_0 = \{0\}$, $S = \{1, 6\}$, and $\Sigma_{obs} = \{\alpha\}$. In order to characterize 2-step opacity using the framework of [22], one might be tempted to define this language as the set of all strings that can be generated by the system and visit a secret state within the past 2 observations. Hence, $E = \{\delta_{uo}, \delta_{uo}\alpha, \delta_{uo}\alpha\alpha\}$ which implies that $L(G) - E = \{\epsilon\}$. It is not hard to see that $P(E) \not\subseteq P(L(G) - E)$ although the system is 2-step opaque. The problem is that string $\delta_{uo}\alpha\alpha$ can be considered as secret when it passes through the set of secret states (i.e., after δ_{uo} or $\delta_{uo}\alpha\alpha$) and non-secret when it does not pass through the set of secret states (i.e., after $\delta_{uo}\alpha$). A similar argument can be made for $\delta_{uo}\alpha$. Note that it might be possible to enlarge the state space (i.e., consider a larger automaton) and capture K-step opacity of the original automaton via a language formulation in this larger automaton. This, however, will be at the cost of higher complexity.

We can extend the definition trajectory-based K-step opacity (see Section 3.5.1) to trajectory-based infinite-step opacity by removing the restriction on the length K of the sequence of observations. Note that trajectory-based infinite-step opacity coincides with the language-based notion of opacity considered in [22]. If we define the language $E \subseteq L(G)$ to be the set of strings in G that visit at least one secret state, then $L(G) - E$ is the set of strings in G that only visit non-secret states. It is not hard to see that G is trajectory-based infinite-step opaque if and only if $P(E) \subseteq P(L(G) - E)$.

3.5.4 Diagnosability

Diagnosability (in its simplest form) is defined assuming that the states of the system can be partitioned into two sets: faulty states (denoted by the set F, $F \subseteq X$) and normal states (denoted by $X - F$) [11]. The system is assumed to enter a faulty state in the set F after the occurrence of a fault and remain within the set F afterwards. The system is diagnosable if there exists a finite integer $K \geq 0$ such that a failure, i.e., an unobservable event (that causes the system to enter states in the set F and remain there), is guaranteed to be detected and isolated after the occurrence of at most K events following the failure.

One major difference between diagnosability and K-step opacity, is the

assumption that faulty states are δ-invariant, i.e., once the system evolves to a faulty state it remains in the set F, whereas in opacity the set S is not necessarily δ-invariant. Another — perhaps not so important — difference is that diagnosability focuses on whether a fault can be detected/diagnosed after a finite number of events (and not observations) which results in the (rather standard) requirement that cycles of unobservable events are absent from the given system.

Assuming that: (i) S is δ-invariant and, (ii) S comprises the set of faulty states, then G being diagnosable implies that the system is not (S, P, K)-opaque (or (S, P, ∞)-opaque) for some large enough K; the reason for this is that diagnosability implies that the entrance of the system to states in S will be detected after at most n_0 events. It is important to point out that K-step opacity is not the inverse of diagnosability: for a system *not* to be diagnosable, there must exist at least *two* infinite traces with the same projection such that one enters the set of faulty (secret) states and one does not. On the other hand, for K-step opacity we need this to be true for *each* trace that enters the faulty (secret) states: each such trace needs to have a corresponding trace that does not enter faulty (secret) states and has identical projection. This difference perhaps explains why it is unlikely that there exists an algorithm that can verify K-step opacity with polynomial-time complexity (the problem is shown to be NP-hard in Chapter 6) while polynomial-time tests for verifying diagnosability exist[30, 31].

3.5.5 Initial-State Verification

The authors of [32] study the initial-state verification problem, i.e., the existence of a unique input/output (UIO) sequence for a state i of a given discrete event system which can be modeled as a *Mealy* machine.[3] A UIO sequence for state i is an input sequence x such that the output sequence generated by the machine in response to x from initial state i is different from the response to x from any other initial state. While the notion of initial-state opacity is related to the notion of a UIO sequence, these two notions were developed for completely different purposes: with initial-state opacity we are trying to hide the membership of the initial state of the system to the set of secret

[3] A Mealy machine is a finite state machine that generates an output based on its current state and input [19].

states while with a UIO sequence, we are trying to demonstrate that the system started from a given initial state (or set of initial states).

CHAPTER 4

DELAYED STATE ESTIMATION IN DISCRETE EVENT SYSTEMS

Initial-state opacity, K-step opacity, and infinite-step opacity are predicates that can be defined for states visited in the past (at a fixed point in time or at a fixed number of observations in the past). Existing state estimation techniques cannot verify these properties since they are not tracking the sequence of states and the time at which there were visited. For this reason, in this chapter, we introduce the problem of *delayed estimation* and construct delay state estimators which are capable of capturing additional information about state estimates and can be used to verify opacity notions of interest.

The basic state estimation setting we consider is the following: we are given a finite automaton with (partially) unknown initial state and partial *event* observation. For this type of DES, we consider two estimation problems: initial-state estimation and K-delayed state estimation. The former requires the estimate of the initial state of the system following a sequence of observations, whereas the latter requires the estimate of the state the system was in when it generated the K^{th} to last output (i.e., the state of the system K observations ago). To solve these two problems we construct appropriate state estimators. Specifically, we construct (i) an *initial-state estimator* (that can be used to capture all the information that is relevant to initial-state estimation and is contained in any sequence of observations of finite but arbitrary length); (ii) a *K-delay state estimator* (that can be shown to contain, after observing any sequence of observations, the information that is necessary to deduce the state the system was in K observations ago). Note that the initial-state estimator and the K-delay state estimator are not comparable since they aim to capture different information.

4.1 Initial-State Estimates

Given a sequence of observations $\omega \in \Sigma_{obs}^*$, the initial-state estimation problem requires the enumeration of all states that belong to the set of initial states X_0, and which could have generated this observed sequence of labels. We call this estimate the *initial-state estimate* and define it formally as follows.

Definition 4.1.1 (Initial-State Estimate)**.** *Given a non-deterministic finite automaton $G = (X, \Sigma, \delta, X_0)$ and a natural projection map P with respect to the set of observable events Σ_{obs} ($\Sigma_{obs} \subseteq \Sigma$), the initial-state estimate after observing the sequence of observations $\omega \in \Sigma_{obs}^*$ is defined as*

$$\hat{X}_0(\omega) = \{i | i \in X_0, \exists t \in \Sigma^* \{P(t) = \omega, \delta(i, t) \neq \emptyset\}\}.$$

■

Remark 4.1.1. *In the area of testing of finite state machines (refer to [33] for a detailed survey), there are two problems that relate to the initial-state estimation problem. (i) The initial state identification problem where we seek to identify (if possible) a sequence of inputs (called distinguishing sequence) that uniquely identifies the unknown initial state of the system. (ii) The initial-state verification problem where we seek to identify (if possible) a sequence of inputs (called unique input/output (UIO) sequence) that can be used to verify that the system starts from a known initial state. The partial observation and the non-deterministic nature of the underlying automaton in our framework make the problem of initial-state estimation that we consider here more general than the problems in [33].* ■

4.2 Initial-State Estimator

In order to capture initial-state estimates for a given non-deterministic finite automaton $G = (X, \Sigma, \delta, X_0)$, a natural projection map P with respect to the set of observable events Σ_{obs} ($\Sigma_{obs} \subseteq \Sigma$), and a set of secret states $S \subseteq X$, the first method that comes to mind is to use the sequence of observations to obtain the current-state estimate (using standard techniques described in Section 4.4) and then back-propagate the state trajectory using the system

model (obtaining in this way all system initial states that could have generated the sequence of observations). Though straightforward, this method requires storage of the sequence of observations, which is not feasible since the system might generate a sequence of arbitrary length (e.g., in an online monitoring application). Next we introduce an algorithm which generates the initial-state estimator $G_{\infty,obs}$, a deterministic finite automaton that is driven by observable events and whose states are associated with distinct state mappings.

To construct the initial-state estimator, we associate its initial state m_0 with the state mapping $\odot_2(X_0)$, where X_0 is the set of initial states of the system; with a slight abuse of notation we denote this by $m_0 = \odot_2(X_0)$. When observation $\alpha \in \Sigma_{obs}$ is made, the initial state mapping m_0 is composed with the induced state mapping $M(\alpha)$ corresponding to observation α, resulting in a state mapping m_1 that associates with the next state of the estimator, i.e., $m_1 = m_0 \circ M(\alpha)$. Similarly, for each subsequent observation $\beta \in \Sigma_{obs}$, the current state of the ISE that is associated with a state mapping m transitions into the state associated with the state mapping $m' = m \circ M(\beta)$. From the structure of the state mappings and the nature of the composition operator, we can establish that, at any given time step, each state mapping associated with each state of $G_{\infty,obs}$ (other than the initial state m_0) includes all pairs of one starting state (from X_0) and one ending state (from X) such that the ending state can be reached from the starting state via a sequence of events that generates the sequence of observations seen so far. This is all the information we need in order to update the state of the estimator as more labels are observed: at any given time, the state mapping provides information about the possible initial- and current-state estimates (and the connections between them) through its pairs of starting and ending states. Note that this structure (which will be described formally and shown to be well-defined shortly) is guaranteed to be finite and has at most 2^{N^2} states (actually $2^{|X_0|\times N}$ states[1]) where N is the number of states of the finite automaton G (and $|X_0|$ is the number of possible initial states of the system).

The initial-state estimator, when considered as a finite automaton, summarizes the effect of any sequence of observations on the estimate of the initial and current states. This summary is independent of the observation length

[1]In the sequel, for the sake of easier presentation, we use $X \times X$ instead of $X_0 \times X$ to denote the pairs of possible initial and current states.

and allows multiple sequences of observations (possibly of different lengths) to be mapped to the same state in the initial-state estimator, as long as they impose identical constraints on the initial/current-state estimates and the possible paths between them. Thus, state mappings are important as a tool for compressing information and performing initial-state estimation recursively using finite memory. We can think of state mappings as a way to partition the set of all sequences of observations, of arbitrary but finite length, into a finite number of equivalence classes: two strings belong to the same equivalence class if and only if they induce the same state mapping. The following algorithm formally describes the construction of the initial-state estimator.

Definition 4.2.1 (Initial-State Estimator (ISE))**.** *Given a non-deterministic finite automaton $G = (X, \Sigma, \delta, X_0)$ and a natural projection map P with respect to the set of observable events Σ_{obs} ($\Sigma_{obs} \subseteq \Sigma$), the initial-state estimator is the deterministic automaton $G_{\infty,obs} = AC(2^{X\times X}, \Sigma_{obs}, \delta_{\infty,obs}, X_{\infty,0})$ with set of states $2^{X\times X}$ (power set of $X \times X$), event set Σ_{obs}, initial state $X_{\infty,0} = \odot_2(X_0)$, and state transition function $\delta_{\infty,obs} : 2^{X\times X} \times \Sigma_{obs} \to 2^{X\times X}$ defined for $\alpha \in \Sigma_{obs}$ as*

$$m' = \delta_{\infty,obs}(m, \alpha) := m \circ M(\alpha),$$

where $m, m' \in 2^{X\times X}$. Recall that $M(\alpha)$ denotes the state mapping that is induced by observing α and AC denotes the states of this automaton that are accessible starting from state $X_{\infty,0}$. If we let $X_{\infty,obs} \subseteq 2^{X\times X}$ be the reachable states from the initial state $X_{\infty,0}$ under $\delta_{\infty,obs}$, then $G_{\infty,obs} = (X_{\infty,obs}, \Sigma_{obs}, \delta_{\infty,obs}, X_{\infty,0})$. ■

Remark 4.2.1. *The authors of [34] also consider the problem of initial-state estimation and provide two formulae for obtaining initial-state estimates: the first one is a simple set intersection expression and the second one has a predictor-corrector recursive form. They also introduce an initial-state estimator which generates initial-state estimates using pairs of states of the form (initial-state, current-state) which is similar to our state mappings. Under the assumption that the system is "initial-state observable" (which guarantees that singleton state mappings are always reachable within a finite number of observations), the transition function of the initial-state estimator in [34] is*

defined using a recursive algorithm based on stacks. At each point in time, the stack contains the list of state mappings for which the transition function should be defined. At each level, the top state mapping is popped out and the next state mapping is obtained (for all possible observation sequences of length one). All of the non-singleton state mappings (i.e., state mappings with more than one pair) that are obtained in this way are pushed back into the end of the stack and the current state mapping is permanently popped out. This iterative algorithm continues until all reachable state mappings are singleton. The authors of [34] also prove that if the underlying system is deterministic, the state complexity of the corresponding initial-state estimator is $O(2^N)$ *where* $N = |X|$ *is the number of states of the system. Moreover, it is shown that the number of different initial-state estimates in the pairs (initial-state, current-state) is* $2N - 1$.

Although the construction of the state transition function using stacks is more effective from a programming perspective, it is not efficient since it may calculate the transition function more than once for the same state mapping. The method we introduce here for constructing the initial-state estimator does not suffer from this problem (and does not require the system to be "initial-state observable"). Also, while suited for online initial-state estimation, the results of [34] cannot be used for verifying initial-state opacity since we need to find the effect of "all" possible observations on the initial-state estimate. This is not possible using the initial-state estimator of [34], if the system is not initial-state observable. The ISE construction, on the other hand, achieves this purpose creating in the process a finite structure.

Another difference between our work and that of [34] is the underlying model: in this chapter, we assume that the system can be modeled as a non-deterministic automaton with event observations, whereas the system in [34] is modeled as a deterministic finite automaton with (possibly partial) state observations. This makes our framework more general compared to that of [34]. Also, since our system is non-deterministic, none of the bounds obtained in [34] on the number of states of its initial-state estimator is applicable in our setting. ■

Remark 4.2.2. *In [35], a finite tree, called unique input-output (UIO) tree, is constructed using the concepts of path vector and vector perturbation which are special cases of a state mapping. In a path vector, each element in the*

set of starting states is mapped to exactly one element in the set of ending states (this follows from the assumption of full-observation in [35]), whereas in a state mapping this association can be one-to-many. ■

In the following lemma, we show that the state mapping associated with the state of the ISE $G_{\infty,obs}$ reached via a string ω consists exactly of the pairs of a starting/ending state from which a sequence of events that generates the sequence of observations ω could have originated/ended.

Theorem 4.2.1. *Consider a non-deterministic finite automaton* $G = (X, \Sigma, \delta, X_0)$ *and a natural projection map* P *with respect to the set of observable events* Σ_{obs} *(*$\Sigma_{obs} \subseteq \Sigma$*), and its ISE* $G_{\infty,obs} = (X_{\infty,obs}, \Sigma_{obs}, \delta_{\infty,obs}, X_{\infty,0})$ *constructed as in Definition 4.2.1. If ISE state* m *is reachable from ISE initial state* $X_{\infty,0} = \odot_2(X_0)$ *via a finite-length string* ω *(*$\omega \neq \epsilon$*), then* m *is associated with a state mapping that satisfies*

$$m = \{(i,j) | i \in X_0, j \in X, \exists t \in \Sigma^* \{P(t) = \omega, j \in \delta(i,t)\}\}.$$

■

Proof. Assume $\omega = \alpha_0 \dots \alpha_n$ and denote the sequence of ISE states visited via ω (starting from the ISE initial state $m_0 = \odot_2(X_0)$) by $m_0, m_1, \dots, m_{n+1}$. We prove the result by induction: for $\omega = \alpha_0$, the statement can easily be seen to be true: $m_1 = m_0 \circ M(\alpha_0)$ is simply the set of pairs of an ending state (in X) that can be reached from the starting state (in X_0) via a string with projection α_0. Now assuming that the lemma holds for $\omega' = \alpha_0 \alpha_1 \dots \alpha_{n-1}$, we prove it for $\omega = \alpha_0 \dots \alpha_n$. Recall that m_{n+1} is the ISE state that is reachable from the ISE initial state with string ω (in the lemma, state m_{n+1}

is denoted by m). By construction, we have

$$
\begin{aligned}
m_{n+1} &= m_n \circ M(\alpha_n) \\
&= \{(i,k)|\exists j \in X\{(i,j) \in m_n, (j,k) \in M(\alpha_n)\}\} && (4.1) \\
&= \{(i,k)|\exists j \in X, \exists i \in X_0, \exists k \in X, \exists t^{n-1} \in \Sigma^*\{P(t^{n-1}) = \\
&\quad \alpha_0\alpha_1 \ldots \alpha_{n-1}, j \in \delta(i,t^{n-1}), (j,k) \in M(\alpha_n)\}\} && (4.2) \\
&= \{(i,k)|\exists j \in X, \exists i \in X_0, \exists k \in X, \exists t^{n-1} \in \Sigma^*\{P(t^{n-1}) = \\
&\quad \alpha_0\alpha_1 \ldots \alpha_{n-1}, j \in \delta(i,t^{n-1}), \exists t_n \in \Sigma^* \\
&\quad \{P(t_n) = \alpha_n, k \in \delta(j,t_n)\}\}\} && (4.3) \\
&= \{(i,k)|i \in X_0, k \in X, \exists t^n \in \Sigma^*\{P(t^n) = \alpha_0\alpha_1 \ldots \alpha_n, \\
&\quad k \in \delta(i,t^n)\}\},
\end{aligned}
$$

where (4.1) follows from definition of the $\circ$ operator, (4.2) follows from the induction hypothesis, and (4.3) follows from definition of $M(\alpha)$. Note that in the last line, we use $t^n = t^{n-1}t_n$. If we rename k to j, t^n to t, and replace $\alpha_0\alpha_1 \ldots \alpha_n$ with ω, the proof is completed. □

Next we prove that the estimate of the system initial state after observing a nonempty string ω (according to Definition 4.1.1) is the set of starting states in the state mapping associated with the ISE state reached via string ω.

Corollary 4.2.1. *Consider a non-deterministic finite automaton* $G = (X, \Sigma, \delta, X_0)$ *with a natural projection map* P *with respect to the set of observable events* Σ_{obs} *(*$\Sigma_{obs} \subseteq \Sigma$*), and its ISE* $G_{\infty,obs} = (X_{\infty,obs}, \Sigma_{obs}, \delta_{\infty,obs}, X_{\infty,0})$ *constructed as in Definition 4.2.1. The initial-state estimate* $\hat{X}_0(\omega)$ *after observing the sequence of observations* ω, $\omega \neq \epsilon$, *can be captured using the ISE as* $\hat{X}_0(\omega) = m(1)$ *where* $m = \delta_{\infty,obs}(X_{\infty,0}, \omega)$. ■

Proof. By Theorem 4.2.1,

$$m = \delta_{\infty,obs}(X_{\infty,0}, \omega) = \{(i,j)|i \in X_0, j \in X, \exists t \in \Sigma^*\{P(t) = \omega, j \in \delta(i,t)\}\}.$$

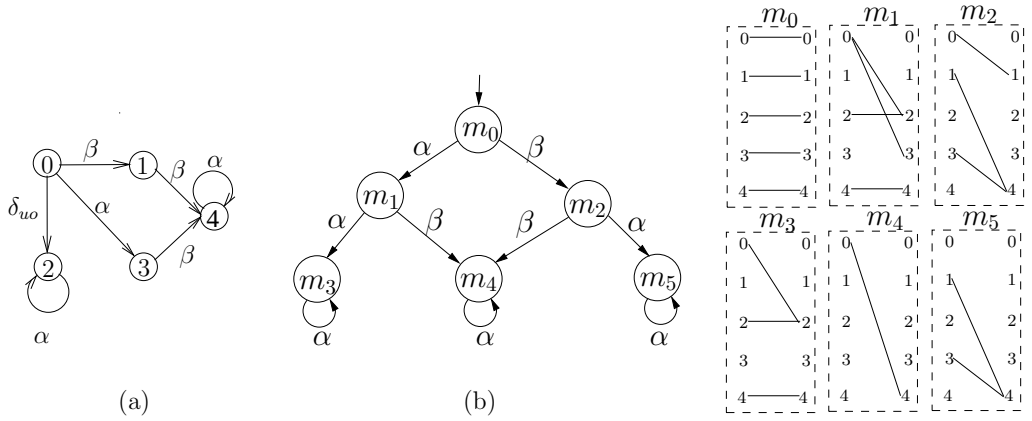

Figure 4.1: (a) G; (b) initial-state estimator $G_{\infty,obs}$.

Therefore,

$$\begin{aligned} m(1) &= \{i|(i,j) \in m\} \\ &= \{i|i \in X_0, \exists j \in X, \exists t \in \Sigma^*\{P(t) = \omega, j \in \delta(i,t)\}\} \\ &= \{i|i \in X_0, \exists t \in \Sigma^*\{P(t) = \omega, \delta(i,t) \neq \emptyset\}\} \\ &= \hat{X}_0(\omega), \end{aligned}$$

which completes the proof. □

The following example clarifies the ISE construction.

Example 4.2.1. *Consider the automaton G of Figure 4.1-a with $\Sigma_{obs} = \{\alpha, \beta\}$. Figure 4.1-b shows the ISE for this system. The initial uncertainty is assumed to be equal to the state space and, hence, the initial state of the ISE is the state mapping $m_0 = \{(0,0), (1,1), (2,2), (3,3), (4,4)\}$. The state mapping $M(\alpha)$ induced by observing α is $\{(0,2), (0,3), (2,2), (4,4)\}$ which implies that α can be observed only from states 0, 2 and 4. Moreover, if the initial state was 0, the current state can only be one of the states in $\{2,3\}$; however, if the initial state was 2, the current state could only be 2; finally, if the initial state was 4, the current state would be 4. Following observation α, the next state m' in the ISE can be constructed as $m' = m_0 \circ M(\alpha) = \{(0,2), (0,3), (2,2), (4,4)\} \equiv m_1$. Mapping m_1 summarizes starting/ending state information with its pairs (on the right of Figure 4.1-b we use a graphical way to describe the pairs associated with the ISE; for example, after observing α we know that one possibility is that we started at state 0 and ended in state*

2 or state 3).

Similarly, the state mapping $M(\beta)$ induced by observing β is $M(\beta) = \{(0,1),(1,4),(3,4)\}$ implying that β can be observed from states 0, 1, and 3 and that the current state can be either $\{1\}$, $\{4\}$, or $\{4\}$, respectively. To take into account the observation α followed by observation β, we need to compose state mapping m_1 and $M(\beta)$ which results in $\{(0,4)\} \equiv m_4$.

Using this approach for all possible observations (from each state), the ISE construction can be completed as shown in Figure 4.1-b. Note that this figure does not include the state that corresponds to the empty state mapping and is reached via sequences of observations that cannot be generated by G; we can (and will) safely ignore this state since it cannot be reached via sequences of observations that are generated by underlying activity in system G. ■

Remark 4.2.3. *Note that all estimates of the system initial state that are associated with an ISE state (state mapping) m' that is reachable from a given ISE state (state mapping) m in $G_{\infty,obs}$ are refinements of the estimates of the system initial state associated with m. In other words, the initial-state estimate, if changed, can only get more accurate; this is a straightforward consequence of the definition of the initial-state estimate and implies that a sequence of observations that result in the traversal of a cycle in $G_{\infty,obs}$ does not yield extra information about the system initial state. In other words, if we consider a cycle $m_1 \xrightarrow{\alpha_1} m_2 \xrightarrow{\alpha_2} \ldots \xrightarrow{\alpha_{n-1}} m_n \xrightarrow{\alpha_n} m_1$ in $G_{\infty,obs}$ (constructed as in Definition 4.2.1), then $m_1, m_2, \ldots, m_n$ are consistent (refer to Section 2 for the definition). Moreover, if m_1 in the aforementioned cycle is reached via string ω in $G_{\infty,obs}$, i.e., $\delta_{\infty,obs}(X_{\infty,0}, \omega) = m_1$, then observing ω, $\omega\alpha_1 \ldots \alpha_n$ or $\omega(\alpha_1 \ldots \alpha_n)^*$ yields the same information about the initial state. This implies that if one is given these choices, then one can simply use ω as a representative of this class of observation sequences and discard all other observation sequences without loosing any information about the initial state. The use of state mappings systematically accomplishes such reductions and hence provides us with a practical way to summarize observations (using finite storage).* ■

Remark 4.2.4. *In $G_{\infty,obs}$ there exist cycles of states such that, once reached, there are no traces outside these cycles. We refer to the ISE states in these cycles as ergodic states. The ergodic states in the ISE are associated with estimates of the system initial state that cannot be further refined. For ex-*

ample, the self-loop at ISE state m_5 in Figure 4.1-b denotes one such cycle (consisting of a single state). Note that if a sequence of observations reaches an ergodic ISE state, the estimate of the initial state cannot be improved with future observations. For example, if we reach ISE state m_5 via $\beta\alpha\alpha^*$ there is no reason to wait for more observations to refine the initial-state estimate. Since the ISE has at most 2^{N^2} states, we know that among all strings of length $2^{N^2}-1$, at least one is guaranteed to reach any given (ergodic or otherwise) ISE state. One can therefore argue that the set of all strings of length smaller than or equal to K (for some $K \geq 2^{N^2}-1$) reveals the same information about the system initial state as any other set of strings of length smaller than or equal to K' with $K' > K$: specifically, for any string of length smaller than or equal to K', there exists an "equivalent string" (as far as initial-state estimation is concerned) of length smaller than or equal to K. In Chapter 5, we use this fact to show that K-step opacity and K'-step opacity are equivalent for $K' > K \geq 2^{N^2}-1$. This implies that infinite-step opacity is equivalent to K-step opacity for $K \geq 2^{N^2}-1$. ■

4.3 Current-State Estimates

Upon observing some string ω (sequence of observations), the state of the system might not be identifiable uniquely due to the lack of knowledge of the initial state, the partial observation of events, and/or the non-deterministic behavior of the system. We denote the set of states that the system might reside in *given that ω was observed* as the current-state estimate $\hat{X}_{|\omega|}(\omega)$ and define it formally below.

Definition 4.3.1 (Current-State Estimate)**.** *Given a non-deterministic finite automaton $G = (X, \Sigma, \delta, X_0)$ and a natural projection map P with respect to the set of observable events Σ_{obs} ($\Sigma_{obs} \subseteq \Sigma$), the current-state estimate after observing string ω is defined as*

$$\hat{X}_{|\omega|}(\omega) := \{j \in X | \exists t \in \Sigma^*, \exists i \in X_0 \{j \in \delta(i,t), P(t) = \omega\}\}.$$

■

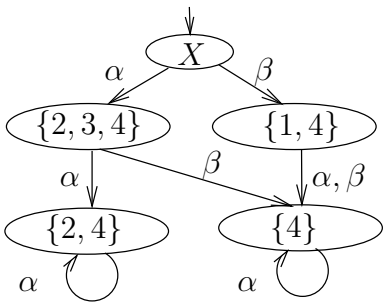

Figure 4.2: Current-state estimator $G_{0,obs}$.

4.4 Current-State Estimator

Given a non-deterministic finite automaton $G = (X, \Sigma, \delta, X_0)$ and a natural projection map P with respect to the set of observable events Σ_{obs} ($\Sigma_{obs} \subseteq \Sigma$), the *current-state estimator* is a deterministic automaton $G_{0,obs}$ which captures current-state estimates and can be constructed as follows [34]. Each state of $G_{0,obs}$ is associated with a unique subset of states of the original DES G (so that there are at most $2^{|X|} = 2^N$ states). The initial state of $G_{0,obs}$ is associated with X_0, representing the fact that the initial state could be any state in X_0. At any state Z of the estimator ($Z \subseteq X$), the next state upon observing an event $\alpha \in \Sigma_{obs}$ is the unique state of $G_{0,obs}$ associated with the set of states that can be reached in G from (one or more of) the states in Z with a string of events that generates the observation α. We denote this automaton by $G_{0,obs} = AC(2^X, \Sigma_{obs}, \delta_{obs}, X_0)$ where 2^X (power set of X) is the state set, Σ_{obs} is the set of observable events, δ_{obs} is the state transition function, X_0 is the initial state, and AC is the part of the automaton that is accessible from initial state X_0. We also define $X_{obs} \subseteq 2^X$ to be the reachable states from X_0 under δ_{obs}, so that $G_{0,obs} = AC(2^X, \Sigma_{obs}, \delta_{obs}, X_0) = (X_{obs}, \Sigma_{obs}, \delta_{obs}, X_0)$. The following example clarifies this construction. More details can be found in [34].

Example 4.4.1. *Consider the DES G in Figure 4.1-a with initial state $X_0 = X$. Assuming that $\Sigma_{obs} = \{\alpha, \beta\}$, then the current-state estimator $G_{0,obs}$ in Figure 4.2 is constructed as follows. Starting from the initial state X_0 and observing α, the current state is any of the states in $\{2, 3, 4\}$; at this new state, the set of possible transitions is the union of all possible transitions for each of the states in $\{2, 3, 4\}$. Following this procedure, $G_{0,obs}$ can be completed as in Figure 4.2. Note that the state of $G_{0,obs}$ that is associated*

with the empty set of states (and that is reached via strings $\omega \in \Sigma^*_{obs}$ for which $P^{-1}(\omega) \cap L(G) = \emptyset$) is not drawn in Figure 4.2 for clarity purposes. In fact, we can (and will) safely ignore such states since they can never be reached via sequences of observations that are generated by underlying activity in system G. ■

4.5 K-Delayed State Estimates

Similar to the current-state estimates, we denote the set of states that the system was possibly in when it generated the K^{th} to last output (i.e., the state of the system K observations ago) following a sequence of observations $\omega = \alpha_0\alpha_1 \ldots \alpha_n$ $(n \geq K)$ as the K-delayed state estimate $\hat{X}_{|\omega|-K}(\omega)$ and define it formally below.[2] Note that the current-state estimate can also be seen as the 0-delayed state estimate.

Definition 4.5.1 (K-Delayed State Estimate)**.** *Given a non-deterministic finite automaton $G = (X, \Sigma, \delta, X_0)$ and a natural projection map P with respect to the set of observable events Σ_{obs} ($\Sigma_{obs} \subseteq \Sigma$), the K-delayed state estimate after observing string $\omega = \alpha_0\alpha_1 \ldots \alpha_n$ $(n \geq K)$ is defined as*

$$\hat{X}_{|\omega|-K}(\omega) := \{j \in X | \exists t', t'' \in \Sigma^*, \exists i \in X_0 \{j \in \delta(i, t'), \delta(j, t'') \neq \emptyset, \\ P(t') = \alpha_0\alpha_1 \ldots \alpha_{n-K}, P(t'') = \alpha_{n-K+1} \ldots \alpha_n\}\}.$$

■

Based on Definition 4.5.1, the K-delayed state estimate $\hat{X}_{|\omega|-K}(\omega)$ after observing $\omega = \alpha_0\alpha_1 \ldots \alpha_n$ $(n \geq K)$ is the set of all states that (i) are reachable in G from (at least one pair of) initial state i and string t' with projection $P(t')$ equal to the first $n - K$ observable events in ω (in the same order) and (ii) for which there exists at least one continuation t'' with projection $P(t'')$ equal to the last K observable events in ω (in the same order). Note that the set of states reachable in G via a string t' with projection $P(t') = \alpha_0\alpha_1 \ldots \alpha_{n-K} \equiv \omega'$ is the current-state estimate that was obtained after observing ω' but before observing $P(t'') = \alpha_{n-K+1} \ldots \alpha_n \equiv \omega''$;

[2] K-delayed state estimation for discrete event systems is related to *fixed-lag smoothing* for discrete-time systems [36].

thus, $\hat{X}_{|\omega|-K}(\omega) \subseteq \hat{X}_{|\omega'|}(\omega')$ and the K-delayed state estimate can be considered as the subset of states in $\hat{X}_{|\omega'|}(\omega')$ from which the post K observations $\alpha_{n-K+1} \ldots \alpha_n$ are possible. Note that Definition 4.5.1 implies that if $\omega \in \Sigma_{obs}^*$ is not a valid sequence of observations in G, then $\hat{X}_{|\omega|-K}(\omega) = \emptyset$. Also, by convention, $\hat{X}_{|\omega|-K}(\omega)$ is taken to be $\hat{X}_0(\omega)$ for $|\omega| < K$.

4.6 K-Delay State Estimator

The *K-delay state estimator* (KDE) is a deterministic finite automaton that reconstructs the k-delayed state estimates ($0 \leq k \leq K$) associated with a given sequence of observations ω. In the sequel, we introduce two methods for constructing K-delay state estimators: (i) by storing the possible sequences of the last $(K+1)$-visited states via $(K+1)$-dimensional state mappings, (ii) by storing the k-delayed state estimates, $0 \leq k \leq K$, and remembering the sequence of the last K observations.

4.6.1 State Mapping-Based K-Delay State Estimator (SM-KDE)

The SM-KDE utilizes $(K+1)$-dimensional state mappings to capture the K-delayed state estimates as follows: each state of the SM-KDE is associated with a unique $(K+1)$-dimensional state mapping, with the initial state m_0 of the SM-KDE associated with the $(K+1)$-dimensional state mapping $\odot_{K+1}(X_0)$. With a slight abuse of notation, we denote this by $m_0 = \odot_{K+1}(X_0)$. When observation $\alpha \in \Sigma_{obs}$ is made, this initial $(K+1)$-dimensional state mapping m_0 is shifted with the induced state mapping $M(\alpha)$ corresponding to observation α, resulting in a $(K+1)$-dimensional state mapping m_1 that associates with the next state of the state estimator, i.e., $m_1 = m_0 >> M(\alpha)$. Similarly, for each subsequent observation $\beta \in \Sigma_{obs}$, the current state of the SM-KDE that is associated with a $(K+1)$-dimensional state mapping m transitions into the state associated with the $(K+1)$-dimensional state mapping $m' = m >> M(\beta)$. From the structure of $(K+1)$-dimensional state mappings and the nature of the shift operator, we can establish that a sequence of observations causes the SM-KDE to transition through a sequence of $(K+1)$-dimensional state mappings such that, at

any given time step, the set of states in the associated $(K+1)$-dimensional state mapping m correspond to delayed state estimates. More specifically, the set of ending states $m(0)$ corresponds to zero-delayed state estimates (current-state estimates), the set of intermediate states $m(k), 1 < k < K$, corresponds to k-delayed state estimates, and the set of starting states $m(K)$ corresponds to K-delayed state estimates. In this manner, we can build a structure which, at any time following a given sequence of observations, maintains information about the 0-delayed, 1-delayed, ..., and K-delayed state estimates through the $(K+1)$-dimensional state mappings associated with each of its states. In fact, the estimator also contains complete information about the possible system state trajectories during the last K observations.

Definition 4.6.1 (State Mapping-Based K-Delay State Estimator (SM-KDE))**.** *Given a non-deterministic finite automaton $G = (X, \Sigma, \delta, X_0)$ and a natural projection map P with respect to the set of observable events Σ_{obs} ($\Sigma_{obs} \subseteq \Sigma$), we define the K-delay state estimator as the deterministic automaton $G_{K,obs} = AC(2^{X^{(K+1)}}, \Sigma_{obs}, \delta_{K,obs}, X_{K,0})$ with state set $2^{X^{(K+1)}}$, event set Σ_{obs}, initial state $X_{K,0} = \odot_{K+1}(X_0)$, and state transition function $\delta_{K,obs} : 2^{X^{(K+1)}} \times \Sigma_{obs} \to 2^{X^{(K+1)}}$ defined for $\alpha \in \Sigma_{obs}$ as*

$$m' = \delta_{K,obs}(m, \alpha) := m >> M(\alpha),$$

where $m, m' \in 2^{X^{(K+1)}}$. Recall that $M(\alpha)$ denotes the state mapping that is induced by observing α and AC denotes the states of this automaton that are accessible starting from state $X_{K,0}$. If we let $X_{K,obs} \subseteq 2^{X^{(K+1)}}$ be the reachable states from the initial state $X_{K,0}$ under $\delta_{K,obs}$, then $G_{K,obs} = (X_{K,obs}, \Sigma_{obs}, \delta_{K,obs}, X_{K,0})$. ■

Example 4.6.1. *Consider the DES G in Figure 4.1-a with $X_0 = \{0, 1, 2, 3, 4\}$ and $\Sigma_{obs} = \{\alpha, \beta\}$. For this system, the 2-delay state estimator is represented in Figure 4.3 along with the 3-dimensional state mappings $m_0, m_1, \ldots, m_{10}$ needed in the construction. The initial state of the system is $X_0 = X$ and the initial state of the 2-delay state estimator captures this in m_0 via a 3-dimensional state mapping that maps each system state to itself as $\odot_3(X) = \{(0,0,0), (1,1,1), (2,2,2), (3,3,3), (4,4,4)\} \equiv m_0$.*

Starting from the initial state, assume that we observe α. As described in Example 4.2.1, the state mapping $M(\alpha)$ induced by observing α is $\{(0,2), (0,3),$

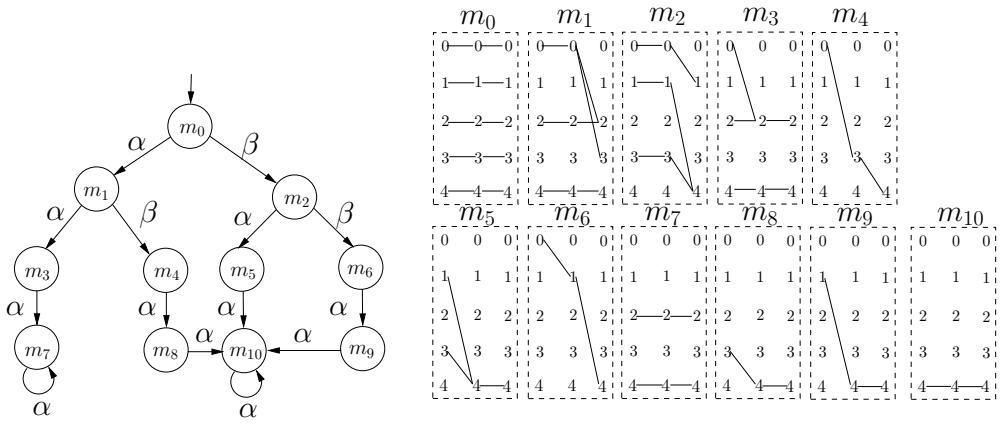

Figure 4.3: State mapping-based 2-delay state estimator corresponding to DES G.

$(2,2),(4,4)\}$. *Following observation* α*, the next state* m' *in the 2-delay state estimator can be constructed as* $m' = \delta_{K,obs}(m_0, \alpha) = m_0 >> M(\alpha) = \{(0,0,2),(0,0,3),(2,2,2),(4,4,4)\} \equiv m_1$.

Next, consider the case when following observation α *we observe* β*. As the induced state mapping* $M(\beta) = \{(0,1),(1,4),(3,4)\}$*, we have* $m' = \delta_{K,obs}(m_1, \beta) = m_1 >> M(\beta) = \{(0,3,4)\} \equiv m_4$*. This implies that* $\alpha\beta$ *can only be observed if the system follows the state trajectory* $0 \to 3 \to 4$*. Using this approach for all possible observations (from each state), the 2-delay state estimator construction can be completed as shown in Figure 4.3. Note that we have not included the state that corresponds to the all empty state mapping (and any transitions from/to it) to avoid cluttering the diagram.* ■

Remark 4.6.1. *On the right of Figure 4.3, we use 3-dimensional trellis diagrams to describe the triples associated with states of the 2-delay state estimator. In general, we can graphically represent an induced* K*-dimensional state mapping* m *using a* K*-dimensional trellis diagram, i.e., a* K*-partite graph where the nodes in the state set* X *are replicated* K *times and ordered into* K *vertical slices ranging from slice 0 to slice* $K-1$ *(hence a* K*-dimensional trellis diagram has* $K \cdot N$ *nodes with* $N = |X|$*). Each node at slice* k $(1 \leq k \leq K-2)$ *is either isolated or connected to (at least) one node at slice* $k-1$ *and (at least) one node at slice* $k+1$*. The nodes at slice 0* $(K-1)$ *are either isolated or connected to (at least) one node at slice 1* $(K-2)$*.* ■

In the following theorem, we show that the SM-KDE state m that is

reached via a sequence of observations ω is associated with a $(K+1)$-dimensional state mapping such that the first $|\omega|-K$ observations would have taken the system to the starting states of the $(K+1)$-dimensional state mapping and, in addition, the last K observations could have taken place from these starting states, visiting in the process the intermediate states in the tuple, in the order captured by the elements of the $(K+1)$-dimensional state mapping.

Theorem 4.6.1. *Consider a non-deterministic finite automaton* $G = (X, \Sigma, \delta, X_0)$ *with a natural projection map* P *with respect to the set of observable events* Σ_{obs} *(*$\Sigma_{obs} \subseteq \Sigma$*), and its SM-KDE* $G_{K,obs} = (X_{K,obs}, \Sigma_{obs}, \delta_{K,obs}, X_{K,0})$ *constructed as in Definition 4.6.1. Suppose SM-KDE state* m *is reachable from the SM-KDE initial state* $X_{K,0} = \odot_{K+1}(X_0)$ *via the string* $\omega = \alpha_0\alpha_1 \ldots \alpha_n$, $\omega \neq \epsilon$. *Then, SM-KDE state* m *can be characterized as follows:*
(i) $|\omega| \leq K$*:* $m = \{(j_0, j_1, \ldots, j_K) \in X_0 \times X^K | (0 \leq l \leq |\omega|-1, 0 \leq d \leq K-|\omega|-1) : j_{d+1} = j_d, \exists t_l \in \Sigma^*\{P(t_l) = \alpha_l, j_{K-|\omega|+l+1} \in \delta(j_{K-|\omega|+l}, t_l)\}\}$.
(ii) $|\omega| \geq K$*:* $m = \{(j_0, j_1, \ldots, j_K) \in X^{K+1} | (0 \leq l \leq K-1) : \exists t_l \in \Sigma^*\{P(t_l) = \alpha_{n-K+1+l}, j_{l+1} \in \delta(j_l, t_l)\}, \exists i \in X_0, \exists t' \in \Sigma^*\{P(t') = \alpha_0\alpha_1 \ldots \alpha_{n-K}, j_0 \in \delta(i, t')\}\}$.
(iii) $m = \emptyset$ *when there is no* $t \in L(G)$ *such that* $P(t) = \omega$. ■

Proof. (i) When $|\omega| \leq K$ (before K observations are made), the states in the ω-induced $(|\omega|+1)$-dimensional state mapping replace the rightmost states in the current $(K+1)$-tuple leaving the first $K-|\omega|+1$ leftmost states identical. This implies that $m = \{(j_0, j_1, \ldots, j_K) \in X_0 \times X^K | (0 \leq l \leq |\omega|-1, 0 \leq d \leq K-|\omega|-1) : j_{d+1} = j_d, \exists t_l \in \Sigma^*\{P(t_l) = \alpha_l, j_{K-|\omega|+l+1} \in \delta(j_{K-|\omega|+l}, t_l)\}\}$ which completes the proof of part (i).
(ii) Denote the sequence of SM-KDE states visited via ω by $m_0, \ldots, m_n, m_{n+1}$. We prove the result by induction: for $\omega = \alpha_0\alpha_1 \ldots \alpha_{K-1}$, $(|\omega| = K)$, the statement is true from Part (i). Now assuming that the lemma holds for $\omega = \alpha_0\alpha_1 \ldots \alpha_{n-1}$ $(n \geq K)$, we prove it holds for $\omega = \alpha_0\alpha_1 \ldots \alpha_n$. Recall that m_{n+1} is the SM-KDE state that is reachable with $\omega = \alpha_0\alpha_1 \ldots \alpha_n$ (which

in the lemma is denoted by m). By construction we have

$$
\begin{aligned}
m_{n+1} &= m_n >> M(\alpha_n) \\
&= \{(j_1, \ldots, j_{K+1}) | (j_0, j_1, \ldots, j_K) \in m_n, (j_K, j_{K+1}) \in M(\alpha_n)\} \\
&\qquad\qquad \text{(by definition of } >> \text{ operator)} \\
&= \{(j_1, \ldots, j_{K+1}) \in X^{K+1} | (0 \leq l \leq K-1) : \exists t_l \in \Sigma^* \{P(t_l) = \alpha_{n-K+l}, \\
&j_{l+1} \in \delta(j_l, t_l)\}, \exists i \in X_0, \exists t \in \Sigma^* \{P(t) = \alpha_0 \alpha_1 \ldots \alpha_{n-K-1}, j_0 \in \delta(i, t)\}\} \\
&(j_K, j_{K+1}) \in M(\alpha_n)\} \quad \text{(by induction hypothesis)} \\
&= \{(j_1, \ldots, j_{K+1}) \in X^{K+1} | (0 \leq l \leq K-1) : \\
&\exists t_l \in \Sigma^* \{P(t_l) = \alpha_{n-K+l}, j_{l+1} \in \delta(j_l, t_l)\} \\
&\exists i \in X_0 \{\exists t \in \Sigma^* \{P(t) = \alpha_0 \alpha_1 \ldots \alpha_{n-K-1}\}, j_0 \in \delta(i, t)\} \\
&\exists t_K \in \Sigma^* \{P(t_K) = \alpha_n, j_{K+1} \in \delta(j_K, t_K)\}\} \text{ (by definition of } M(\alpha_n)) \\
&= \{(j_1, \ldots, j_{K+1}) \in X^{K+1} | (1 \leq l \leq K) : \\
&\exists t_l \in \Sigma^* \{P(t_l) = \alpha_{n-K+1+l}, j_{l+1} \in \delta(j_l, t_l)\} \\
&\exists i \in X_0, \exists t' \in \Sigma^* \{P(t') = \alpha_0 \alpha_1 \ldots \alpha_{n-K}, j_1 \in \delta(i, t')\}\}\}.
\end{aligned}
$$

Note that in the last line, we define $t' = tt_1$ from the previous line. The proof is completed by decreasing the indices by one.
(iii) Follows from the definition of the shift operator. □

Corollary 4.6.1. *Consider a non-deterministic finite automaton* $G = (X, \Sigma, \delta, X_0)$ *with a natural projection map* P *with respect to the set of observable events* Σ_{obs} *(*$\Sigma_{obs} \subseteq \Sigma$*), and its SM-KDE* $G_{K,obs} = (X_{K,obs}, \Sigma_{obs}, \delta_{K,obs}, X_{K,0})$ *constructed as in Definition 4.6.1. The* k*-delayed state estimate* $\hat{X}_{|\omega|-k}(\omega)$, $0 \leq k \leq \min(K, |\omega|)$, *after observing* $\omega = \alpha_0 \alpha_1 \ldots \alpha_n$ *can be captured as follows:* $\hat{X}_{|\omega|-k}(\omega) = m(k)$ *where* $\delta_{K,obs}(X_{K,0}, \omega) = m$. ■

Proof. First, assume that $|\omega| \geq K$. From Theorem 4.6.1 we have for $0 \leq$

$k \leq K$

$$
\begin{aligned}
m(k) &= \{j_{K-k} | (j_0, j_1, \ldots, j_K) \in m\} \\
&= \{j_{K-k} | (j_0, j_1, \ldots, j_K) \in X^{K+1}, 0 \leq l \leq K-1 : \\
&\exists t_l \in \Sigma^* \{P(t_l) = \alpha_{n-K+l+1}, j_{l+1} \in \delta(j_l, t_l)\} \\
&\exists i \in X_0, \exists t' \in \Sigma^* \{P(t') = \alpha_0 \alpha_1 \ldots \alpha_{n-k}, j_0 \in \delta(i, t')\}\}\} \\
&= \{j_{K-k} | \exists s', s'' \in \Sigma^*, \exists i \in X_0 \{j_{K-k} \in \delta(i, s'), \\
&\delta(j_{K-k}, s'') \neq \emptyset, P(s') = \alpha_0 \alpha_1 \ldots \alpha_{n-k}, P(s'') = \alpha_{n-k+1} \ldots \alpha_n\}\} \\
&= \hat{X}_{|\omega|-k}(\omega),
\end{aligned}
$$

where the third equation follows from the second equation with $s' = t' t_0 t_1 \ldots t_{K-k-1}$ and $s'' = t_{K-k} t_{K-k+1} \ldots t_{K-1}$. The proof for $|\omega| \leq K$ is similar to the previous case and is omitted. □

4.6.2 Observation Sequence-Based K-Delay State Estimator (OS-KDE)

In this section, we introduce the construction of automaton $G_{K,obs}^{observation}$ which captures K-delayed state estimates by remembering the sequence of the last K observations (this should be contrasted to $G_{K,obs}$ which captures the compatible sequences of the last K-visited states via $(K+1)$-dimensional state mappings). At each state of $G_{K,obs}^{observation}$, we store a $(K+2)$-tuple $Q \in \Sigma_{obs,\epsilon}^K \times 2^X \times \ldots \times 2^X$ consisting of the following information: *(i)* the last K observations ($\Sigma_{obs,\epsilon}$ denotes the set $\Sigma_{obs} \cup \{\epsilon\}$), and *(ii)* all the k-delayed state estimates for $k = 0, 1, \ldots, K$. Upon observing a new event, all k-delayed state estimates are updated to ensure that estimates that are not consistent with the last observation are removed. Finally, the string that stores the last K observations is updated by adding the last observation to the end of it and by removing the first one. The main difference here is that $G_{K,obs}^{observation}$ only remembers the *sets* of states that are possible 0, 1, …, K observations ago, but does not explicitly record the sequences of states that are possible; however, knowledge of the last K observations (together with the underlying system model) allows one to reconstruct these sequences if required. We now discuss the systematic construction of $G_{K,obs}^{observation}$. For brevity, we define the function $\delta_o : X \times \Sigma_{obs} \to 2^X$ for any $i \in X$ and $\alpha \in \Sigma_{obs}$

as

$$\delta_o(i,\alpha) = \{j \in X | \exists s \in \Sigma^* \{P(s) = \alpha, j \in \delta(i,s)\}\}. \tag{4.4}$$

The function δ_0 can be extended from the domain $X \times \Sigma_{obs}$ to the domain $X \times \Sigma_{obs}^*$ in the routine recursive manner: for $t \in \Sigma_{obs}$ and $s \in \Sigma_{obs}^*$,

$$\delta_o(i, ts) := \bigcup_{j \in \delta_o(i,t)} \delta_o(j, s),$$

with $\delta_o(i, \epsilon) := i$. With a slight abuse of notation, we use $\delta_o : 2^X \times \Sigma_{obs} \to 2^X$ to also denote its extension from states to sets of states as follows: for all $Z \subseteq X$ define

$$\delta_o(Z, \alpha) = \bigcup_{z \in Z} \delta_o(z, \alpha).$$

For clarity, we also introduce the following notation: let $\omega = \alpha'_0 \dots \alpha'_n$ with $n \geq K$ denote the sequence of observations that have been seen so far (from the initialization of the system); we will rename the last K observations $\alpha'_{n-K+1} \dots \alpha'_n$ to $\alpha_{-K+1} \dots \alpha_0$ (i.e., $\alpha'_{n-i} = \alpha_{-i}$ for $i = 0, 1, \dots, K-1$) to make the discussion independent of n (the total number of observations seen so far). Also note that in the following definition strings $\alpha_{-k}\alpha_{-k+1} \dots \alpha_0$ of length less than K are represented as $\epsilon^{K-k-1}\alpha_{-k}\alpha_{-k+1} \dots \alpha_0$.

Definition 4.6.2 (Observation Sequence-Based K-Delay State Estimator (OS-KDE))**.** *Given a non-deterministic finite automaton $G = (X, \Sigma, \delta, X_0)$ and a natural projection map P with respect to the set of observable events Σ_{obs} ($\Sigma_{obs} \subseteq \Sigma$), we define the observation sequence-based K-delay state estimator as the deterministic automaton $G_{K,obs}^{observation} = AC((\Sigma_{obs,\epsilon}^K \times 2^X \times \dots \times 2^X), \Sigma_{obs}, \delta_{K,obs}^{observation}, X_{K,0}^{observation})$ with*
(i) set of states $\Sigma_{obs,\epsilon}^K \times 2^X \times \dots \times 2^X$, where each state is a $(K+2)$-tuple which consists of one string of length K or less, and $K+1$ subsets of X;
(ii) event set Σ_{obs};
(iii) initial state $X_{K,0}^{observation} = (\epsilon, X_0, \dots, X_0)$; and
(iv) state transition function $\delta_{K,obs}^{observation} : (\Sigma_{obs,\epsilon}^K \times 2^X \times \dots \times 2^X) \times \Sigma_{obs} \to (\Sigma_{obs,\epsilon}^K \times 2^X \times \dots \times 2^X)$ defined as follows: if the current state is the $(K+2)$-tuple $Q = (\Omega, Z_K, \dots, Z_0) \in \Sigma_{obs,\epsilon}^K \times 2^X \times \dots \times 2^X$, where $\Omega = \alpha_{-K} \dots \alpha_{-1} \in \Sigma_{obs,\epsilon}^K$ and $Z_k \in 2^X, 0 \leq k \leq K$, then the next state for $\alpha_0 \in \Sigma_{obs}$ is

$$\hat{Q} = \delta_{K,obs}^{observation}(Q, \alpha_0) = (\hat{\Omega}, \hat{Z}_K, \dots, \hat{Z}_0)$$

$Q_0 = (\epsilon, X, X, X)$
$Q_1 = (\alpha, \{0,2,4\}, \{0,2,4\}, \{2,3,4\})$
$Q_2 = (\beta, \{0,1,3\}, \{0,1,3\}, \{1,4\})$
$Q_3 = (\alpha\alpha, \{0,2,4\}, \{2,4\}, \{2,4\})$
$Q_4 = (\alpha\beta, \{0\}, \{3\}, \{4\})$
$Q_5 = (\beta\alpha, \{1,3\}, \{4\}, \{4\})$
$Q_6 = (\beta\beta, \{0\}, \{1\}, \{4\})$
$Q_7 = (\alpha\alpha, \{2,4\}, \{2,4\}, \{2,4\})$
$Q_8 = (\beta\alpha, \{3\}, \{4\}, \{4\})$
$Q_9 = (\beta\alpha, \{1\}, \{4\}, \{4\})$
$Q_{10} = (\alpha\alpha, \{4\}, \{4\}, \{4\})$

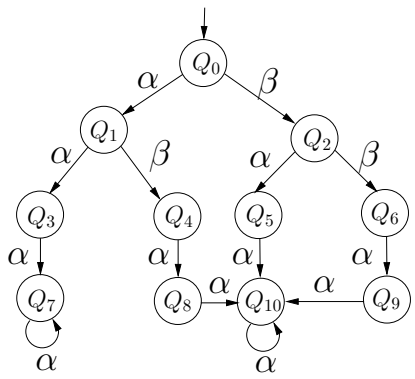

Figure 4.4: Observation sequence-based 2-delay state estimator corresponding to DES G in Figure 4.1-a.

where $\hat{\Omega} = \Omega\alpha_0/\alpha_{-K}$, *and the sets* $\hat{Z}_k$ *are defined recursively as*

$$\hat{Z}_k = \{z | z \in Z_{k-1}, \exists \hat{z} \in \hat{Z}_{k-1} : \hat{z} \in \delta_o(z, \alpha_{-k+1})\}$$

for $k = 1, 2, \ldots, K$ *with* $\hat{Z}_0 = \delta_o(Z_0, \alpha_0)$. *Recall that AC denotes the states of this automaton that are accessible starting from state* $X_{K,0}^{observation}$. *If we let* $X_{K,obs}^{observation} \subseteq (\Sigma_{obs,\epsilon}^K \times 2^X \times \ldots \times 2^X)$ *be the reachable states from the initial state* $X_{K,0}^{observation}$ *under* $\delta_{K,obs}^{observation}$, *then* $G_{K,obs}^{observation} = (X_{K,obs}^{observation}, \Sigma_{obs}, \delta_{K,obs}^{observation}, X_{K,0}^{observation})$. ■

Remark 4.6.2. *The OS-KDE introduced in Definition 4.6.2 is related to the inverter with delay that was introduced in [18]. Assuming that the system is invertible with delay, the inverter in [18] acts as an online algorithm which, for a given time index, stores the K subsequent observations (where K is the fixed delay in the definition of invertibility with delay) in order to be able to refine the state estimate at this time index (using back propagation). The refined state estimate that is obtained is used along with the plant model to reconstruct the executed sequence of events. The OS-KDE is a finite structure that captures all estimates with delay for any observation sequence. In other words, what we do here can be seen as an offline approach for refining the current-state estimate using any possible sequence of observations and K future observations. This is necessary when trying to verify system properties that depend on delayed state estimates (such as K-step opacity).* ■

Example 4.6.2. *Consider the DES in Figure 4.1-a. For this system, the observation sequence-based 2-delayed state estimator* $G_{2,obs}^{observation}$ *is represented*

in Figure 4.4. The initial state $X_0 = X$ and hence the OS-KDE initial-state $X_{2,0}^{observation}$ becomes $(\epsilon, X, X, X) \equiv Q_0$ which, using the notation in Definition 4.6.2, implies that $Z_0 = X$, $Z_1 = X$, $Z_2 = X$, $\Omega = \epsilon$ and $\alpha_{-1} = \alpha_{-2} = \epsilon$. Upon observing α ($\alpha_0 = \alpha$), the current (system) state estimate becomes $\hat{Z}_0 = \{2, 3, 4\}$ and since α can only be observed from $\{0, 2, 4\}$, $\hat{Z}_1 = \{0, 2, 4\}$. Also, since only one observation has been made, $\hat{Z}_2 = \hat{Z}_1 = \{0, 2, 4\}$; finally, $\hat{\omega} = \omega\alpha/\alpha_{-2} = \epsilon\alpha/\epsilon$, so that the next OS-KDE state upon observing α becomes $(\alpha, \{0, 2, 4\}, \{0, 2, 4\}, \{2, 3, 4\}) \equiv Q_1$.

Note that at OS-KDE state Q_1, $Z_0 = \{2, 3, 4\}$, $Z_1 = \{0, 2, 4\}$, $Z_2 = \{0, 2, 4\}$, $\Omega = \alpha$ and hence $\alpha_{-1} = \alpha$ and $\alpha_{-2} = \epsilon$. If β is observed at OS-KDE state Q_1, i.e., $\alpha_0 = \beta$, the next OS-KDE state $\hat{Q} = (\hat{\Omega}, \hat{Z}_2, \hat{Z}_1, \hat{Z}_0) \equiv Q_4$ can be obtained via

$$
\begin{aligned}
\hat{Z}_0 &= \delta_o(Z_0, \alpha_0) \\
&= \delta_o(\{2, 3, 4\}, \beta) \\
&= \{4\} \\
\hat{Z}_1 &= \{z \in Z_0 |\ \exists \hat{z} \in \hat{Z}_0 : \hat{z} \in \delta_o(z, \alpha_0)\} \\
&= \{z \in \{2, 3, 4\} |\ \exists \hat{z} \in \{4\} : \hat{z} \in \delta_o(z, \beta)\} \\
&= \{3\} \\
\hat{Z}_2 &= \{z \in Z_1 |\ \exists \hat{z} \in \hat{Z}_1 : \hat{z} \in \delta_o(z, \alpha_{-1})\} \\
&= \{z \in \{0, 2, 4\} |\ \exists \hat{z} \in \{3\} : \hat{z} \in \delta_o(z, \alpha)\} \\
&= \{0\},
\end{aligned}
$$

with $\hat{\Omega}{=}\Omega\alpha_0/\alpha_{-2}{=}\alpha\beta/\epsilon{=}\alpha\beta$.

OS-KDE state $Q_4 \equiv (\alpha\beta, \{0\}, \{3\}, \{4\})$ conveys the following information: the current-state estimate is $\{4\}$, the previous state estimate is $\{3\}$, and the estimate of the system state two observations ago is $\{0\}$. Also $\alpha\beta$ captures the last two observations (observed in that order). Using this approach for the remaining possible observations, the observation sequence-based 2-delayed state estimator can be completed as shown in Figure 4.4. Note that the states associated with empty state estimates (i.e., of the form $(\Omega, \emptyset, \emptyset, \ldots, \emptyset)$) and transitions from/to these state have not been included. Note that the SM-KDE and OS-KDE in Figures 4.3 and 4.4 respectively are identical automata but, as clarified later on, this will not necessarily be the case in general. ■

In the sequel, we obtain a characterization of each set of states Z_k, $0 \leq k \leq K$, in the OS-KDE state $Q = (\Omega, Z_K, \ldots, Z_0)$ reached via a sequence of observations ω. Specifically, we show that $j \in Z_k$, $0 \leq k \leq K$, if and only if there exists a string t in $L(G)$ that has projection ω, $|\omega| \geq k$, and visits state j exactly k observations ago. If there does not exist $t \in L(G)$ such that $P(t) = \omega$, then $Z_k = \emptyset$, $0 \leq k \leq K$. Furthermore, if $|\omega| < k$, then Z_k, $0 \leq k \leq |\omega|$, is as described above and $Z_k = Z_{|\omega|}$ for $|\omega| + 1 \leq k \leq K$. The following theorem states this formally.

Theorem 4.6.2. *Consider a non-deterministic finite automaton* $G = (X, \Sigma, \delta, X_0)$ *and a natural projection map* P *with respect to the set of observable events* Σ_{obs} *(*$\Sigma_{obs} \subseteq \Sigma$*), and its OS-KDE* $G_{K,obs}^{observation} = (X_{K,obs}^{observation}, \Sigma_{obs}, \delta_{K,obs}^{observation}, X_{K,0}^{observation})$ *constructed as in Definition 4.6.2. Suppose that state* Q *is reachable from the OS-KDE initial state* $X_{K,0}^{observation} = (\epsilon, X_0, X_0, \ldots, X_0)$ *via the string* $\omega = \alpha_0\alpha_1 \ldots \alpha_n$*. Then, the OS-KDE state* Q *can be characterized as follows:*
(i) $|\omega| < K$*:* $Q = (\Omega, Z_K, \ldots, Z_0) \in (\Sigma_{obs,\epsilon}^K, 2^X, \ldots, 2^X)$ *with*

1. $\Omega = \alpha_0 \ldots \alpha_n$,
2. $Z_0 = \{j \in X | \exists t \in \Sigma^*, \exists i \in X_0 \{P(t) = \omega, j \in \delta(i,t)\}\}$,
3. *for* $1 \leq k \leq |\omega|$, $Z_k = \{j \in X | \exists t \in \Sigma^*, \exists t' \in \bar{t}, \exists i \in X_0 \{P(t) = \omega, P(t)/P(t') = \alpha_{n-k+1} \ldots \alpha_n, j \in \delta(i,t'), \delta(j, t/t') \neq \emptyset\}\}$, *and*
4. *for* $|\omega| + 1 \leq k \leq K$, $Z_k = Z_{|\omega|}$.

(ii) $|\omega| \geq K$*:* $Q = (\Omega, Z_K, \ldots, Z_0) \in (\Sigma_{obs,\epsilon}^K, 2^X, \ldots, 2^X)$ *with*

1. $\Omega = \alpha_{n-K+1} \ldots \alpha_n$,
2. $Z_0 = \{j \in X | \exists t \in \Sigma^*, \exists i \in X_0 \{P(t) = \omega, j \in \delta(i,t)\}\}$,
3. *for* $1 \leq k \leq K$, $Z_k = \{j \in X | \exists t \in \Sigma^*, \exists t' \in \bar{t}, \exists i \in X_0 \{P(t) = \omega, P(t)/P(t') = \alpha_{n-k+1} \ldots \alpha_n, j \in \delta(i,t'), \delta(j, t/t') \neq \emptyset\}\}$.

(iii) $Z_k = \emptyset$, $0 \leq k \leq K$, *when there is no* $t \in L(G)$ *such that* $P(t) = \omega$. ■

Proof. (i) Denote the sequence of OS-KDE states visited via $\omega = \alpha_0\alpha_1 \ldots \alpha_n$ by $Q_0, Q_1 \ldots, Q_n, Q_{n+1}$ where $Q_0 = (\epsilon, X_0, X_0, \ldots, X_0)$. We prove the result by using induction on $|\omega|$: first assume that $\omega = \alpha_0$, *i.e.*, $|\omega| = 1$, and that

state $Q_1 = (\alpha_0, Z_K, \ldots, Z_1, Z_0)$ is reachable in OS-KDE via ω. Characterization of Z_0 follows from (4.4). For Z_1 by the update rule of OS-KDE we have

$$\begin{aligned} Z_1 &= \{z \in X_0 | \exists \hat{z} \in Z_0 \{\hat{z} \in \delta_o(z, \alpha_0)\}\} \\ &= \{z \in X_0 | \exists s \in \Sigma^* \{P(s) = \alpha_0, \hat{z} \in \delta(z, s)\}\}. \end{aligned}$$

Define $t = s$, $t' = \epsilon$, and $i = j = z$, and the characterization is complete. Note that for $2 \leq k \leq K$, $Z_k = Z_{|\omega|}$ since the system has not generated enough observations. Now assuming that the lemma holds for $\omega = \alpha_0\alpha_1 \ldots \alpha_{n-1}$ ($n < K$), we prove it holds for $\omega = \alpha_0\alpha_1 \ldots \alpha_{n-1}\alpha_n$ with $\alpha_n \in \Sigma_{obs}$. Let $Q_{n+1} = (\alpha_0\alpha_1 \ldots \alpha_n, \hat{Z}_K, \hat{Z}_{K-1}, \ldots, \hat{Z}_0)$ be the OS-KDE state that is reachable with $\omega = \alpha_0\alpha_1 \ldots \alpha_n$ (which in the lemma is denoted by Q). Also assume that $Q_n = (\alpha_0\alpha_1 \ldots \alpha_{n-1}, Z_K, Z_{K-1}, \ldots, Z_0)$. The characterization of $\hat{Z}_0$, $\hat{Z}_1$, and $\hat{Z}_k$, $|\omega| + 1 \leq k \leq K$, follows from the previous part. Next we show that the characterization given in the theorem holds for $\hat{Z}_k$, $1 \leq k \leq |\omega|$, using induction on k: we have already established that the characterization for $\hat{Z}_1$ follows from the previous part; assuming that the characterization holds for $k-1$, $2 < k < |\omega| + 1$, we show that it also holds for k. By the update rule of OS-KDE, we have

$$\begin{aligned} \hat{Z}_k &= \{z \in Z_{k-1} | \exists \hat{z} \in \hat{Z}_{k-1} \{\hat{z} \in \delta_o(z, \alpha_{n-k+1})\}\} \\ &= \{z \in X | \exists t \in \Sigma^*, \exists t' \in \bar{t}, \exists i \in X_0, \exists \hat{z} \in \hat{Z}_{k-1} \{P(t) = \alpha_0 \ldots \alpha_{n-1}, \\ &P(t)/P(t') = \alpha_{n-k+1} \ldots \alpha_{n-1}, z \in \delta(i, t'), \delta(z, t/t') \neq \emptyset, \hat{z} \in \delta_o(z, \alpha_{n-k+1})\} \qquad (4.5) \\ &= \{z \in X | \exists s, t \in \Sigma^*, \exists t' \in \bar{t}, \exists i \in X_0, \exists \hat{z} \in \hat{Z}_{k-1} \{P(t) = \alpha_0 \ldots \alpha_{n-1}, \\ &P(t)/P(t') = \alpha_{n-k+1} \ldots \alpha_{n-1}, z \in \delta(i, t'), \delta(z, t/t') \neq \emptyset, P(s) = \alpha_{n-k+1}, \\ &\hat{z} \in \delta(z, s)\}, \qquad (4.6) \end{aligned}$$

where (4.5) follows from the induction hypothesis on n and (4.6) follows from (4.4). By induction hypothesis on k, we have

$$\begin{aligned} \hat{z} \in \hat{Z}_{k-1} \Leftrightarrow \exists r \in \Sigma^*, \exists r' \in \bar{r}, \exists \hat{i} \in X_0 \{P(r) = \alpha_0 \ldots \alpha_n, \\ P(r)/P(r') = \alpha_{n-k+2} \ldots \alpha_n, \hat{z} \in \delta(\hat{i}, r'), \delta(\hat{z}, r/r') \neq \emptyset\}. \qquad (4.7) \end{aligned}$$

Define $p' = t'$ and $p = p's(r/r')$. Then, by (4.6) and (4.7), we have

$$\hat{Z}_k = \{z \in X | \exists p \in \Sigma^*, \exists p' \in \bar{p}, \exists i \in X_0 \{P(p) = \alpha_0 \dots \alpha_n, \\ P(p)/P(p') = \alpha_{n-k+1} \dots \alpha_n, z \in \delta(i, p'), \delta(z, p/p') \neq \emptyset\}\}.$$

This completes the proof for part (i).
(ii) Similar to part (i).
(iii) If the sequence of observations ω is not feasible in the original system, we assume (without generality) that the sequence of observations ω becomes infeasible due to the last observation α. Then by (4.4), $\delta_o(i, \alpha) = \emptyset$ for all $i \in Z_0$ since, by assumption, it is not possible to observe α from any state i in Z_0. This implies that $\hat{Z}_0 = \emptyset$ which then implies that $\hat{Z}_k = \emptyset$, $1 \leq k \leq K$, by the update rule of OS-KDE. This completes the proof for part (iii). □

Corollary 4.6.2. *Consider a non-deterministic finite automaton $G = (X, \Sigma, \delta, X_0)$ and a natural projection map P with respect to the set of observable events Σ_{obs} ($\Sigma_{obs} \subseteq \Sigma$), and its OS-KDE $G_{K,obs}^{observation} = (X_{K,obs}^{observation}, \Sigma_{obs}, \delta_{K,obs}^{observation}, X_{K,0}^{observation})$ constructed as in Definition 4.6.2. The k-delayed state estimate $\hat{X}_{|\omega|-k}(\omega)$, $0 \leq k \leq \min(K, |\omega|)$, after observing $\omega = \alpha_0\alpha_1 \dots \alpha_n$ can be captured as follows: $\hat{X}_{|\omega|-k}(\omega) = Z_k$ where $\delta_{K,obs}^{observation}(X_{K,0}^{observation}, \omega) = (\Omega, Z_K, \dots, Z_0)$.* ■

Proof. Follows from Theorem 4.6.2 and Definition 4.5.1 with $t'' \equiv t/t'$. □

4.6.3 Analysis of State-Space Complexity for K-Delay State Estimators

State Complexity of OS-KDE

The construction of $G_{K,obs}^{observation}$ suggests that its number of states could be as high as $(|\Sigma_{obs}| + 1)^K \times (2^N)^{(K+1)}$, where N denotes the number of states of the given automaton G. However, as we argue next, its state complexity is $O((|\Sigma_{obs}| + 1)^K \times 2^N)$ which is significantly lower.

Theorem 4.6.3. *Given a non-deterministic finite automaton $G = (X, \Sigma, \delta, X_0)$ and a natural projection map P with respect to the set of observable events Σ_{obs} ($\Sigma_{obs} \subseteq \Sigma$), the state complexity of $G_{K,obs}^{observation}$ (constructed ac-*

cording to Definition 4.6.2) is $O((|\Sigma_{obs}|+1)^K \times 2^N)$*, where* $N = |X|$ *denotes the number of states of the given automaton* G*.* ■

Proof. We establish the state space complexity of $G_{K,obs}^{observation}$ by observing that given (i) the sequence of the past K observations $\Omega = \alpha_{-K+1} \dots \alpha_0$, and (ii) the K-delayed state estimate Z_K, the intermediate k-delayed state estimates Z_k $(0 \leq k < K)$ can be reconstructed uniquely using our knowledge of the plant model and the past K observations. We can accomplish this in two steps:
(1) construct K sets of states X_k $(0 \leq k < K)$ as the set of states reachable in G, from states in Z_K via a string that produces the sequence of observations $\alpha_{-K+1} \dots \alpha_{-k}$. Following the notation in (4.4), we have $X_k = \delta_o(Z_K, \alpha_{-K+1} \dots \alpha_{-k})$.
(2) Update all X_k with their post observations $\alpha_{-k+1} \dots \alpha_0$ to construct the k-delayed state estimate Z_k. To accomplish this, we can use an approach similar to the recursive state transition function introduced in Definition 4.6.2: by definition, X_0 is the same as Z_0. Next, we start from X_1 and remove all those estimates that are not consistent with observing α_0 (i.e., their transitions do not generate observation α_0 or do not result in a state in Z_0); this way, we obtain Z_1. Then, we consider X_2 and remove all states y in X_2 from which α_{-2} cannot occur (i.e., state y from which $\delta_o(y, \alpha_{-2}) = \emptyset$) or states y for which α_{-2} leads to a state outside Z_1 (i.e., states which have been removed in previous steps). We can repeat this procedure for $X_3, \dots, X_{K-1}$ and $\alpha_{-3}, \dots, \alpha_{-K+1}$. Therefore, using only Ω and Z_K, we can construct all intermediate k-delayed state estimates Z_k that were explicitly stored at each state of the K-delayed estimator in our earlier construction.

To understand why the reconstruction above is correct, consider an observation sequence-based 2-delayed state estimator and a state $\hat{Q} = (\hat{Z}_2 \xrightarrow{\alpha_{-1}} \hat{Z}_1 \xrightarrow{\alpha_0} \hat{Z}_0)$ (a graphical way to represent $(\alpha_{-1}\alpha_0, \hat{Z}_2, \hat{Z}_1, \hat{Z}_0)$) that is reachable in that estimator via a string ω. Now consider the last successor of the OS-KDE state Q namely state $Q = (Z_2 \xrightarrow{\alpha_{-2}} Z_1 \xrightarrow{\alpha_{-1}} Z_0)$. Based on our notation, Z_0 is the last estimate made before observing α_0 (or reaching $(\hat{Z}_2 \xrightarrow{\alpha_{-1}} \hat{Z}_1 \xrightarrow{\alpha_0} \hat{Z}_0)$) and thus can be denoted by X_1. By construction, the unit-delayed state estimate $\hat{Z}_1$ is the subset of Z_0, or X_1, from which α_0 can occur. This illustrates how the proposed method constructs the intermediate delayed state estimates. Note that for the case when less than K observations

are available, i.e., when $|\Omega| < K$, a similar approach can be taken to construct the intermediate k-delayed state estimates, $0 \leq k < |\Omega|$. In this case, keeping the past $|\Omega|$ observations Ω along with the $|\Omega|$-delayed state estimate suffices to find the intermediate k-delayed state estimates, $0 \leq k < |\Omega|$. □

Example 4.6.3. *For the DES G in Example 4.6.2 with the sequence-based 2-delayed state estimator in Figure 4.4, the state $(\alpha\alpha, \{0,2,4\}, \{2,4\}, \{2,4\})$ can simply be represented by $(\alpha\alpha, \{0,2,4\})$. Specifically, we can obtain the missing unit-delayed and current-state estimates as described above. Note that based on the above notation, in state $(\alpha\alpha, \{0,2,4\}, \{2,4\}, \{2,4\})$, we have $Z_2 = \{0,2,4\}$ and $\alpha_{-2}\alpha_{-1} = \alpha\alpha$. Using this, we can obtain*

$$\begin{aligned} X_1 &= \delta_o(Z_2, \alpha_{-2}) \\ &= \delta_o(\{0,2,4\}, \alpha) \\ &= \{2,3,4\}, \end{aligned}$$

and

$$\begin{aligned} X_0 &= \delta_o(Z_2, \alpha_{-2}\alpha_{-1}) \\ &= \delta_o(\{0,2,4\}, \alpha\alpha) \\ &= \{2,4\}. \end{aligned}$$

We can then set $Z_0 = X_0$ and reflect the post observation ($\alpha_{-1} = \alpha$) made after the previous estimation, which updates X_1 to $Z_1 = \{2,4\}$. ■

The above discussion not only demonstrates that storing the intermediate state estimates is not necessary (as long as the plant model is readily available and one is willing to do some computation) but also implies that keeping this information as part of the state label does not generate new states (even if the plant model is unavailable). In other words, $G_{K,obs}^{observation}$ with reduced state labels (constructed without explicitly storing the intermediate delayed state estimates as described above) is isomorphic to the one that stores them explicitly (described in Definition 4.6.2).

State Complexity of SM-KDE

Each state of the SM-KDE $G_{K,obs}$ is associated with a $(K+1)$-dimensional state mapping. This suggests that the state complexity of this automaton is

bounded by $2^{N^{K+1}}$ since there are N^{K+1} $(K+1)$-dimensional state mappings over the N states of the given automaton (each SM-KDE state is associated with a subset of these state mappings). In this section, we use the results on the state complexity of $G_{K,obs}^{observation}$ that we established in the beginning of this section to prove that the state complexity of $G_{K,obs}$ is $O((|\Sigma_{obs}|+1)^K \times 2^N)$. More specifically, we introduce a function which maps each state of $G_{K,obs}^{observation}$ to a state in $G_{K,obs}$ and then show that the range of this function covers all states of $G_{K,obs}$. This implies that the number of states of $G_{K,obs}$ is less than or equal to the number of states of $G_{K,obs}^{observation}$ and hence establishes that the state space complexity of $G_{K,obs}$ is also $O((|\Sigma_{obs}|+1)^K \times 2^N)$. The following theorem states and proves this formally.

Theorem 4.6.4. *Given a non-deterministic finite automaton $G = (X, \Sigma, \delta, X_0)$ and a natural projection map P with respect to the set of observable events Σ_{obs} ($\Sigma_{obs} \subseteq \Sigma$), the state complexity of $G_{K,obs}$ (constructed according to Definition 4.6.1) is $O((|\Sigma_{obs}|+1)^K \times 2^N)$, where $N = |X|$ denotes the number of states of the given automaton G.* ■

Proof. To prove that the set of states of the SM-KDE, denoted by $X_{K,obs}$, has cardinality equal to or less than the set of states of the OS-KDE, denoted by $X_{K,obs}^{observation}$, we define a function $f : X_{K,obs}^{observation} \rightarrow 2^{X^{(K+1)}}$ and show that for all $Q \in X_{K,obs}^{observation}$:
(a) $f(Q) \in X_{K,obs}$, and
(b) for each $m \in X_{K,obs}$, there exists at least one $Q \in X_{K,obs}^{observation}$ such that $f(Q) = m$.
The establishment of these two properties proves that the number of elements in the set $X_{K,obs}$ is less than or equal to the number of elements in the set $X_{K,obs}^{observation}$.

We define the mapping $f : X_{K,obs}^{observation} \rightarrow 2^{X^{(K+1)}}$ as follows:
(i) For $\Omega = \alpha_0\alpha_1 \ldots \alpha_n$, $|\Omega| < K$, $Q = (\Omega, Z_{|\Omega|}, \ldots, Z_{|\Omega|}, Z_{|\Omega|}, Z_{|\Omega|-1}, \ldots, Z_0) \in X_{K,obs}^{observation}$:

$$\begin{aligned} f(Q) \equiv &\{(j_0, j_1, \ldots, j_K) | (K - |\Omega| \leq k \leq K, \\ &0 \leq l \leq |\Omega| - 1, 0 \leq w \leq K - |\Omega| - 1) : j_k \in Z_{K-k}, \\ &j_{w+1} = j_w, \exists t_l \in \Sigma^* \{P(t_l) = \alpha_l, j_{K-|\Omega|+l+1} \in \delta(j_{K-|\Omega|+l}, t_l)\}\}. \end{aligned} \quad (4.8)$$

(ii) For $\Omega = \alpha_{n-K+1}\alpha_{n-K+2}\ldots\alpha_n, |\Omega| = K, Q = (\Omega, Z_K, \ldots, Z_0) \in X_{K,obs}^{observation}$:

$$f(Q) \equiv \{(j_0, j_1, \ldots, j_K) | (0 \leq k \leq K, 0 \leq l \leq K-1) : \\ j_k \in Z_{K-k}, \exists t_l \in \Sigma^* \{P(t_l) = \alpha_{n-K+1+l}, j_{l+1} \in \delta(j_l, t_l)\}\}. \quad (4.9)$$

Function f maps an OS-KDE state Q to a set of $(K+1)$-tuples of states $(j_0, \ldots, j_K)$ such that each state j_k in this tuple is chosen from the corresponding delayed state estimate Z_{K-k} and also such that the system can visit the sequence of states $j_0, \ldots, j_K$ and produce the sequence of observations Ω.

To prove (a), we first consider the case where $|\Omega| < K$ with $\Omega = \alpha_0\alpha_1\ldots\alpha_n$. Applying the characterization of Z_k, $0 \leq k \leq |\Omega|$, from part (i) of Theorem 4.6.2 to the definition of mapping f in (4.8), we have

$$f(Q) = \{(j_0, j_1, \ldots, j_K) | (1 \leq k \leq K, 0 \leq l \leq |\Omega| - 1, 0 \leq w \leq K - |\Omega| - 1) : \\ j_0 \in X_0, j_k \in X, j_{w+1} = j_w, \exists t_l \in \Sigma^* \{P(t_l) = \alpha_l, j_{K-|\Omega|+l+1} \in \delta(j_{K-|\Omega|+l}, t_l)\}\}$$

By part (i) of Theorem 4.6.1, $f(Q)$ is the SM-KDE state m which is reachable from initial state $X_{K,0} = \odot_{K+1}(X)$ via string Ω.

Next, we consider the case where $|\Omega| = K$. Assume that OS-KDE state $Q = (\Omega, Z_K, \ldots, Z_0)$ is reachable in $G_{K,obs}^{observation}$ from its initial state $X_{K,0}^{observation}$ via string $\omega = \alpha_0\alpha_1\ldots\alpha_n$ with $|\omega| \geq K$ and $\Omega = \alpha_{n-K+1}\ldots\alpha_n$. Applying the characterization of Z_k, $0 \leq k \leq K$, from part (ii) of Theorem 4.6.2 to the definition of mapping f in (4.8), we have

$$f(Q) = \{(j_0, j_1, \ldots, j_K) | (0 \leq k \leq K, 0 \leq l \leq K-1) : \exists t_l \in \Sigma^* \{ \\ j_k \in X, P(t_l) = \alpha_{n-K+1+l}, j_{l+1} \in \delta(j_l, t_l), \exists t' \in \Sigma^*, \exists i \in X_0 \{j_0 \in \delta(i, t')\}\}.$$

By part (ii) of Theorem 4.6.1, $f(Q)$ is the SM-KDE state m which is reachable from initial state $X_{K,0} = \odot_{K+1}(X)$ via string ω (state m has the additional restriction that $P(t') = \alpha_0\alpha_1\ldots\alpha_{n-K}$). This completes the proof for part (a).

To prove (b), consider state m in $G_{K,obs}$ which is reachable from initial state $X_{K,0} = \odot_{K+1}(X)$ via string $\omega = \alpha_0\alpha_1\ldots\alpha_n$. We consider two cases depending on whether $|\omega| < K$ or $|\omega| \geq K$.

First suppose that $|\omega| \leq K$ and define $Z_k := m(k)$, for $0 \leq k \leq K$.

Next, we characterize Z_k, $0 \leq k \leq |\omega|$. We use part (i) of Theorem 4.6.1 to characterize m as $m = \{(j_0, j_1, \ldots, j_K) \in X_0 \times X^K | (0 \leq l \leq |\omega| - 1, 0 \leq w \leq K - |\omega| - 1) : j_{w+1} = j_w, \exists t_l \in \Sigma^* \{P(t_l) = \alpha_l, j_{K-|\omega|+l+1} \in \delta(j_{K-|\omega|+l}, t_l)\}\}$. Any $j \in Z_0$ maps to the j_K in an $(j_0, j_1, \ldots, j_K) \in m$. Hence, take $t = t_0 t_1 \ldots t_{|\omega|-1}$, $i = j_0$ and use the above characteristic to prove that for all $j \in Z_0$, $\exists t \in \Sigma^*, \exists i \in X_0$ such that $P(t) = \omega, j \in \delta(i, t)$. Similarly, any $j \in Z_k$ for $1 \leq k \leq |\omega|$, maps to a j_{K-k} in $(j_0, j_1, \ldots, j_{K-k}, \ldots, j_K) \in m$. Hence, we can take $t' = t_0 t_1 \ldots t_{n-k}$, and argue that for all $j \in Z_k$, $1 \leq k \leq |\omega|$, there exists $t \in \Sigma^*$, $t' \in \bar{t}$, and $i \in X_0$ such that: $P(t) = \omega$, $P(t)/P(t') = \alpha_{n-k+1} \ldots \alpha_n$, $j \in \delta(i, t')$, and $\delta(j, t/t') \neq \emptyset$. Therefore, by part (i) of Theorem 4.6.2, Q is a state in $G_{K,obs}^{observation}$ reachable from the initial state $X_{K,0}^{observation}$. It follows from the definition of f that $f(Q) = m$.

We can use a similar approach for $|\omega| > K$. Defining $\Omega = \alpha_{n-K+1}\alpha_{n-K+2} \ldots \alpha_n$ and $Z_k := m(k)$, $0 \leq k \leq K$, we show that there exists a state $Q \equiv (\Omega, Z_K, \ldots, Z_0)$ in $G_{K,obs}^{observation}$ that is reachable from its initial state $X_{K,0}^{observation}$ via string ω and satisfies $f(Q) = m$. Using part (ii) of Theorem 4.6.1, we have $m = \{(j_0, j_1, \ldots, j_K) \in X^{K+1} | (0 \leq l \leq K - 1) : \exists t_l \in \Sigma^* \{P(t_l) = \alpha_{n-K+l+1}, j_{l+1} \in \delta(j_l, t_l), \exists i \in X_0, \exists t' \in \Sigma^* \{P(t') = \alpha_0 \alpha_1 \ldots \alpha_{n-K}, j_0 \in \delta(i, t')\}\}$. Any $j \in Z_0$ maps to an j_K in $(j_0, j_1, \ldots, j_K) \in m$. Hence, take $t = t' t_0 t_1 \ldots t_{K-1}$ and use the above characteristic to prove that $\forall j \in Z_0 : \exists t \in \Sigma^*, \exists i \in X_0$ such that $P(t) = \omega$ and $j \in \delta(i, t)$. Similarly, any $j \in Z_k$ for $0 \leq k \leq K$, maps to an j_{K-k} in $(j_0, j_1, \ldots, j_{K-k}, \ldots, j_K) \in m$. Hence, take $t = t' t_0 t_1 \ldots t_{K-1}$, and use the characterization of m to prove that for all $j \in Z_k$, $1 \leq k \leq K$, $\exists t \in \Sigma^*, \exists t' \in \bar{t}, \exists i \in X_0 \{P(t) = \omega, P(t)/P(t') = \alpha_{n-k+1} \ldots \alpha_n, j \in \delta(i, t'), \delta(j, t/t') \neq \emptyset\}$. Therefore, Q is a state in $G_{K,obs}^{observation}$ reachable from the initial state $X_{K,0}^{observation}$ via string ω. Clearly, $f(Q) = m$ which shows that any state in $X_{K,obs}$ is the image of mapping f of at least one state in $X_{K,obs}^{observation}$. Thus, $G_{K,obs}$ has no more states than $G_{K,obs}^{observation}$, which completes the proof. □

Theorem 4.6.4 proves that it is more state space efficient to construct the SM-KDE than to construct the OS-KDE; however, implementing the OS-KDE is easier than implementing the SM-KDE since at each state of the OS-KDE fewer elements need to be stored and the update rule for states is simpler. Therefore, the preference between ease of programming and state space efficiency is the deciding factor for choosing between these two imple-

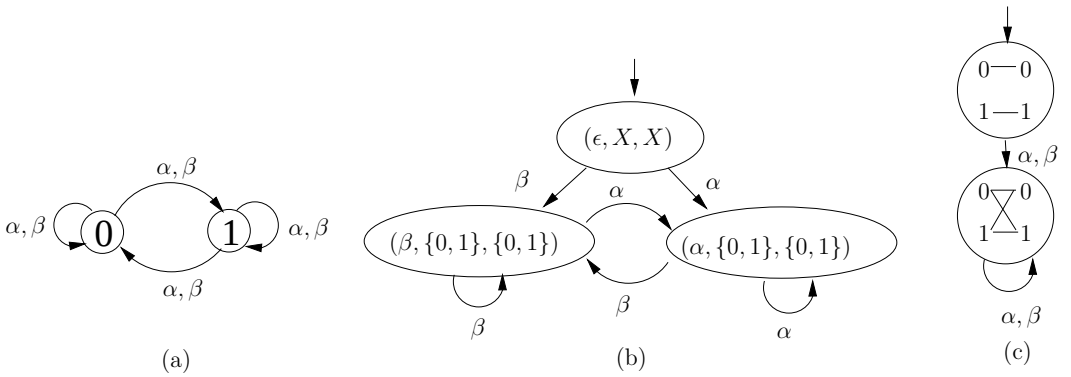

Figure 4.5: Example demonstrating that the number of states in $G_{K,obs}$ can be less than the number of states in $G_{K,obs}^{observation}$: (a) G; (b) $G_{1,obs}^{observation}$; (c) $G_{1,obs}$.

mentations.

Note that for DES G in Figure 4.1-a, the number of states for both estimators is the same (indeed they are isomorphic) which shows that the introduced mapping between the states of these two estimators can be one-to-one. Note, however, that this is not necessarily the case. Consider, for example, the DES G in Figure 4.5-a with initial state set $X_0 = \{0, 1\} = X$. Figures 4.5-b and 4.5-c depict $G_{1,obs}^{observation}$ and $G_{1,obs}$, respectively. As can be seen, storing the last observation results in creating more states compared to storing the 2-dimensional state mapping corresponding to the last two states visited. This can be explained as follows: sequences of observations $\alpha\alpha$, $\beta\alpha$, $\beta\beta$, and $\alpha\beta$ correspond to the same sequence of states visited and hence all states reachable in $G_{1,obs}^{observation}$ via these sequences of observations are mapped to the (unique) 2-dimensional state mapping in $G_{1,obs}$ reachable via any of these sequences. This demonstrates that the mapping introduced in the proof of Theorem 4.6.4 between the states of $G_{1,obs}^{observation}$ and $G_{1,obs}$ can be many-to-one.

Remark 4.6.3. *Theorem 4.6.4 can be also interpreted as follows: in order to construct k-delayed state estimates, $0 \leq k \leq K$, it is more state space efficient to store the set of sequences of the last K visited states (that generate the last K observations) instead of storing the sequence of the last K observations. The intuition behind this is the fact that multiple sequences of observations can be mapped to the same set of state sequences.* ■

CHAPTER 5

VERIFICATION OF STATE-BASED NOTIONS OF OPACITY

In this chapter, we show how state estimators can be used to verify opacity. We start by showing that a system is initial-state opaque if and only if all initial-state estimates (in its initial-state estimator) contain at least one state outside the set of secret states S; therefore, one can construct the initial-state estimator of Definition 4.2.1 to verify initial-state opacity. Verification using the initial-state estimator is shown to require space and time complexity that, in the worst case, can be exponential in the square of the number of states of the given finite automaton. A more efficient method for verifying initial-state opacity for a fixed set of secret states S is also proposed and analyzed. This method requires space (and thus time) complexity that is exponential in the number of states of the given finite automaton but is specific to the secret set of states S and (unlike the approach using ISE) has to be repeated for a different set of secret states. Before closing our discussion on the verification of initial-state opacity, we also describe how our approach can be extended in certain settings where the set of secret states S is time-varying.

Next, we show that a system is K-step opaque if and only if all k-delayed state estimates, $0 \leq k \leq K$, in its K-delay state estimator, contain at least one state outside the set of secret states S; therefore, one can use a K-delay state estimator to verify K-step opacity. Verification using the K-delay state estimator is shown to require space complexity that in the worst case can be exponential in K and exponential in the square of the number of states of the given finite automaton.

Finally, we show that for any $K \geq 2^{N^2} - 1$ (where N is the number of states of the discrete event system) K-step opacity and infinite-step opacity become equivalent, and hence one can construct the K-delay state estimator, with $K = 2^{N^2} - 1$, and verify infinite-step opacity. We also introduce a reduced-complexity method to verify infinite-step opacity using the current-state estimator and a bank of initial-state estimators.

5.1 Verifying Initial-State Opacity

Initial-state opacity requires that regardless of the string that might be generated by the system (and regardless of the corresponding sequence of observations) no definite information about the membership of the initial state of the system to the set of secret states S can be inferred. In the following lemma, we show that this is equivalent to the fact that none of the possible initial-state estimates $\hat{X}_0(\omega)$ associated with any possible sequence of observations ω in the system is a subset of the set of secret states, unless the sequence of observations ω cannot be generated by G (in which case $\hat{X}_0(\omega) = \emptyset$).

Theorem 5.1.1. *Given a non-deterministic finite automaton $G = (X, \Sigma, \delta, X_0)$, a natural projection map P with respect to the set of observable events Σ_{obs} ($\Sigma_{obs} \subseteq \Sigma$), and a set of secret states $S \subseteq X$, automaton G is (S, P, ∞) initial-state opaque if and only if for all $\omega \in \Sigma_{obs}^*$*

$$\hat{X}_0(\omega) \not\subseteq S \text{ or } \hat{X}_0(\omega) = \emptyset,$$

where $\hat{X}_0(\omega)$ is the initial-state estimate after observing the sequence of observations ω. ■

Proof. First we consider sequences of observations ω such that $\hat{X}_0(\omega) = \emptyset$. Note that $\hat{X}_0(\omega) = \emptyset$ if and only if there is no string t in $L(G)$ such that $P(t) = \omega$; in other words, $\omega \notin P(L(G))$. Therefore, the sequence of observations ω cannot violate initial-state opacity because Definition 3.1.1 requires that the sequences of observations ω be generated by DES G, i.e., $\omega \in P(L(G))$.

Next we consider sequences of observations ω such that $\hat{X}_0(\omega) \neq \emptyset$. Using Definition 4.1.1, we have

$$\begin{aligned}
&\{\forall \omega \in \Sigma_{obs}^* \{\hat{X}_0(\omega) \not\subseteq S\}\} \Leftrightarrow \\
&\{\forall \omega \in \Sigma_{obs}^* \{\exists j \in X_0 - S, \exists s \in \Sigma^* \{P(s) = \omega, \delta(j, s) \neq \emptyset\}\}\} \Leftrightarrow \\
&\{\forall \omega \in \Sigma_{obs}^* \{\exists j \in X_0 - S, \exists s \in L(G, j) \{P(s) = \omega\}\}\} \Leftrightarrow \\
&\{\forall i \in X_0, \forall t \in L(G, i) \{\exists j \in X_0 - S, \exists s \in L(G, j) \{P(s) = P(t)\}\}\},
\end{aligned}$$

which is equivalent to initial-state opacity of G. This completes the proof. □

5.1.1 Verifying Initial-State Opacity using Initial-State Estimator

In this section we discuss how the ISE construction can be used to verify initial-state opacity. Recall that in Theorem 5.1.1, we show that the given DES G is initial-state opaque if and only if none of the (non-empty) initial-state estimates associated with a sequence of observations is a subset of the set of secret states. Also, from Corollary 4.2.1, we know that the ISE construction captures the set of all initial-state estimates for all possible sequences of observations (of any nonzero length) in the system, and hence by Theorem 5.1.1, it can be used for verifying initial-state opacity. We formalize this discussion in the following theorem.

Theorem 5.1.2. *Consider a non-deterministic finite automaton* $G = (X, \Sigma, \delta, X_0)$ *with a natural projection map* P *with respect to the set of observable events* Σ_{obs} *(*$\Sigma_{obs} \subseteq \Sigma$*), and its ISE* $G_{\infty,obs} = (X_{\infty,obs}, \Sigma_{obs}, \delta_{\infty,obs}, X_{\infty,0})$ *constructed as in Definition 4.2.1. Automaton* G *is* (S, P, ∞) *initial-state opaque if and only if for all* $m \in X_{\infty,obs}$*,*

$$m(1) \not\subseteq S \text{ or } m(1) = \emptyset.$$

■

Proof. Follows from Theorem 5.1.1 and Theorem 4.2.1. Note that $m(1) = \emptyset$ implies that this state is reached only via sequences of observations that cannot be generated by the original system. □

Example 5.1.1. *Consider the DES* G *in Figure 4.1-a and the corresponding ISE in Figure 4.1-b. This system is not* $(\{0\}, P, \infty)$ *initial-state opaque due to the existence of ISE state* m_4*, which can be reached via sequences of the form* $(\beta\beta + \alpha\beta)\alpha^*$*. Since* $m_4 = \{(0, 4)\}$*, its only possible starting state is* $\{0\}$ *which falls completely within* S *(i.e.,* $m_4(1) \subseteq S$*). This means that observing any string of the form* $(\beta\beta + \alpha\beta)\alpha^*$ *positively determines the system initial state as state 0, which falls completely within the set of secret states and hence violates initial-state opacity.* ■

In order to verify initial-state opacity using Theorem 5.1.2, we need to construct the ISE and verify that each (non-empty) set of starting states in the state mappings associated with ISE states contains an element outside the

set of secret states S. As a result, checking for initial-state opacity has space and time complexity $O(2^{N^2})$, where $N = |X|$ is the number of states of the given automaton. This exponential complexity is not desirable for implementation purposes. As we show in Chapter 6, however, verifying initial-state opacity is a PSPACE-complete problem and hence it is unlikely that any algorithm can verify this property in polynomial-time [37]; nevertheless, a reduced complexity verification method (that can be used when the secret set S is fixed and invariant over time) is presented in the next section.

5.1.2 Verifying Initial-State Opacity for a Fixed Set of Secret States

Verifying initial-state opacity using the initial-state estimator (ISE) as described in Section 5.1.1 is not tailored to a particular S (i.e., the ISE can be used even if the set of secret states is modified or is to be designed). If S is invariant (fixed over time), we can potentially simplify the verification method since we do not necessarily need the exact estimate of the system initial states but only knowledge of whether the current system state(s) is (are) reachable from secret/non-secret initial states. In this section we show that, for a given (invariant) set of secret states S, the complexity of the verification method can be reduced from exponential in the square of the number of states ($O(2^{N^2})$) to exponential in the number of states ($O(4^N)$), where $N = |X|$ denotes the number of states of the underlying automaton G. Note that the price paid for this reduction is that each set of secret states S would require a separate ISE construction.

Consider a non-deterministic finite automaton $G = (X, \Sigma, \delta, X_0)$ with set of states $X = \{0, 1, \ldots, N-1\}$, a natural projection map P with respect to the set of observable events Σ_{obs} ($\Sigma_{obs} \subseteq \Sigma$), and a set of secret states $S \subseteq X$. We start with the observation that in order to verify initial-state opacity, we only need to know whether (following a sequence of observations $\omega \in \Sigma_{obs}^*$) a possible current state is reachable from secret initial states (in S) or not (outside S). Therefore, instead of associating a set of starting states with each ending state (as done with state mappings), we can simply assign to each ending state one of four labels to indicate whether this state is (i) reachable exclusively from secret initial states (label $\mathbb{S}$), (ii) reachable

exclusively from non-secret ones (label $\mathbb{NS}$), (iii) reachable from a mixture of both types of states (label $\mathbb{M}$), or (iv) not reachable from any initial state (label $\mathbb{NR}$). We capture this via a set of N pairs of the type $(\mathbb{Y}(i), i)$ where $i \in X$ and $N = |X|$. This set of pairs is called a state-status mapping q which is of the form

$$q = \{(\mathbb{Y}(0), 0), (\mathbb{Y}(1), 1), \ldots, (\mathbb{Y}(N-1), N-1) | \mathbb{Y}(i) \in \{\mathbb{S}, \mathbb{NS}, \mathbb{M}, \mathbb{NR}\} \forall i \in X\}.$$

The set of all state-status mappings is denoted by $\{\mathbb{S}, \mathbb{NS}, \mathbb{M}, \mathbb{NR}\}^X \equiv \Delta^X$ and has cardinality 4^N.

Each time a new observation is made, the label $\mathbb{Y}$ associated with (ending) states in the pairs of state-status mapping q can be easily propagated along with the ending state estimates according to the following simple rule: when composing the current state-status mapping with the state mapping induced by the new observation, if two or more (ending) state estimates with two different labels from the set $\{\mathbb{S}, \mathbb{NS}, \mathbb{M}\}$ merge to an identical ending state estimate, we assign to this new (ending) state estimate the label ($\mathbb{M}$); this indicates that the new ending state can be reached from at least one secret and at least one non-secret initial state. If, on the other hand, the merging involves one or more states with a single label from the set $\{\mathbb{S}, \mathbb{NS}, \mathbb{M}\}$ and zero or more states with label $\mathbb{NR}$, then the label $\mathbb{S}$, $\mathbb{NS}$, or $\mathbb{M}$ propagates intact. Finally, if the last observation cannot occur from any of the ending states or the merging involves only states with label $\mathbb{NR}$, we assign label $\mathbb{NR}$ to it. We formalize the above idea via the composition operator $\otimes : \Delta^X \times 2^{X^2} \rightarrow \Delta^X$ defined for a state-status mapping $q \in \Delta^X$ and a state mapping $m \in 2^{X^2}$ as:

$$q \otimes m = \{(\mathbb{Y}(i), i) | i \in X\},$$

where

$$\mathbb{Y}(i) = \begin{cases} \mathbb{S}, & \text{if (i) } \exists i' \in X\{(i',i) \in m\}, \text{ and (ii) } \forall i' \in X\{(i',i) \in m \Rightarrow \\ & \{(\mathbb{S},i') \in q \text{ or } (\mathbb{NR},i') \in q\}, \\ \mathbb{NS}, & \text{if (i) } \exists i' \in X\{(i',i) \in m\}, \text{ and (ii) } \forall i' \in X\{(i',i) \in m \Rightarrow \\ & \{(\mathbb{NS},i') \in q \text{ or } (\mathbb{NR},i') \in q\}, \\ \mathbb{M}, & \text{if either: (i) } \exists i' \in X\{(i',i) \in m \text{ and } (\mathbb{M},i') \in q\} \text{ or} \\ & \text{(ii) } \exists i', i'' \in X\{(i',i),(i'',i) \in m \text{ and } (\mathbb{S},i'),(\mathbb{NS},i'') \in q\}, \\ \mathbb{NR}, & \text{if either: (i) } \nexists i' \in X\{(i',i) \in m\} \text{ or} \\ & \text{(ii) } \forall i' \in X\{(i',i) \in m \Rightarrow (\mathbb{NR},i') \in q\}. \end{cases}$$

In order to verify initial-state opacity, we construct a deterministic finite automaton $G_{verifier}$, called *verifier*. Under a sequence of observations $\omega \in \Sigma_{obs}^*$, $\omega \neq \epsilon$, the verifier reaches a state associated with a state-status mapping for which each state i is associated with a unique label $\mathbb{Y}(i)$ from the set $\Delta \equiv \{\mathbb{S}, \mathbb{NS}, \mathbb{M}, \mathbb{NR}\}$. Label $\mathbb{Y}(i)$ indicates whether i is reachable in G via sequences of events (with projection ω) that start exclusively from secret initial states ($\mathbb{Y}(i) = \mathbb{S}$), or exclusively from non-secret initial states ($\mathbb{Y}(i) = \mathbb{NS}$), or from a mixture of secret and non-secret initial states ($\mathbb{Y}(i) = \mathbb{M}$), or not reachable at all ($\mathbb{Y}(i) = \mathbb{NR}$). The state-status mapping associated with the verifier initial state is $q_0 = \{(\mathbb{Y}_0(i), i) | i \in X\}$ where

$$\mathbb{Y}_0(i) = \begin{cases} \mathbb{S}, & \text{for } i \in X_0 \cap S, \\ \mathbb{NS}, & \text{for } i \in X_0 - S, \\ \mathbb{NR}, & \text{for } i \in X - X_0. \end{cases}$$

Upon observing an event $\alpha \in \Sigma_{obs}$, both components in each pair of the state-status mapping associated with the verifier current state need to be updated; this is accomplished by defining the next state under input α to be the state associated with state-status mapping $q_0 \otimes M(\alpha)$, where $M(\alpha)$ is the state mapping induced by observation α. The construction of the deterministic automaton $G_{verifier}$ continues in this way as stated formally below.

Definition 5.1.1 (Verifier)**.** *Consider a non-deterministic finite automaton $G = (X, \Sigma, \delta, X_0)$, with set of states $X = \{0, 1, \ldots, N-1\}$, a natural projection map P with respect to the set of observable events Σ_{obs} ($\Sigma_{obs} \subseteq \Sigma$), and*

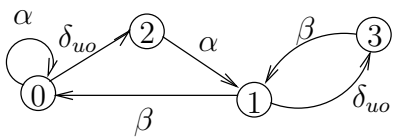

Figure 5.1: G with $\Sigma_{obs} = \{\alpha, \beta\}$.

a set of secret states $S \subseteq X$. We define the verifier as the deterministic automaton $G_{verifier} = AC(\Delta^X, \Sigma_{obs}, \delta_{verifier}, X_{verifier,0})$ with set of labels $\Delta = \{\mathbb{S}, \mathbb{NS}, \mathbb{M}, \mathbb{NR}\}$, set of states Δ^X, event set Σ_{obs}, initial state $X_{verifier,0} = \{(\mathbb{Y}_0(i), i) | i \in X\}$, and state transition function $\delta_{verifier} : \Delta^X \times \Sigma_{obs} \to \Delta^X$ defined for $\alpha \in \Sigma_{obs}$ as

$$\delta_{verifier}(q, \alpha) := q \otimes M(\alpha).$$

Recall that $M(\alpha)$ denotes the state mapping that is induced by observing α and AC denotes the accessible part of this automaton starting from state $X_{verifier,0}$. If we let $X_{verifier} \subseteq \Delta^X$ be the set of verifier states reachable from the verifier initial state $X_{verifier,0}$ under the state transition mapping $\delta_{verifier}$, then $G_{verifier} = (X_{verifier}, \Sigma_{obs}, \delta_{verifier}, X_{verifier,0})$. ■

Example 5.1.2. *Consider the DES G in Figure 5.1 with $X_0 = X$ and $S = \{2\}$. The state-status mapping q_0 associated with the initial state of the verifier $G_{verifier}$ for G equals $\{(\mathbb{NS}, 0), (\mathbb{NS}, 1), (\mathbb{S}, 2), (\mathbb{NS}, 3)\}$. Upon observing α, following the rules of the operator $\otimes$, the next state of the verifier becomes*

$$q_1 \equiv q_0 \otimes M(\alpha) = \{(\mathbb{NS}, 0), (\mathbb{M}, 1), (\mathbb{NS}, 2), (\mathbb{M}, 3)\}.$$

To better understand this, observe that system states 0 and 2 are only reachable from the non-secret initial state 0 via the observation α and therefore the label associated with these states is $\mathbb{NS}$; system states 1 and 3 are reachable from both the secret initial state 2 and the non-secret initial state 0 via observation α. Therefore, states 1 and 3 are associated with label $\mathbb{M}$ to denote the fact that they are reachable from both secret and non-secret initial states. Note that q_1 is the state-status mapping associated with the verifier state reachable via α from the verifier initial state q_0. Upon observing another α, the state-status mapping is updated according to the rules of the operator $\otimes$

$$q_2 \equiv q_1 \otimes M(\alpha) = \{(\mathbb{NS}, 0), (\mathbb{NS}, 1), (\mathbb{NS}, 2), (\mathbb{NS}, 3)\}.$$

The construction of the verifier can be continued in this fashion until no more states can be constructed. ■

A system is initial-state opaque if a viable sequence of observations in the system can originate from at least one non-secret initial state. In the following theorem, we prove that the verifier described in Definition 5.1.1, contains this information (for all possible observations) in the state-status mappings associated with its states.

Theorem 5.1.3. *Consider a non-deterministic finite automaton* $G = (X, \Sigma, \delta, X_0)$ *with set of states* $X = \{0, 1, \ldots, N-1)$*, a natural projection map* P *with respect to the set of observable events* Σ_{obs} *(*$\Sigma_{obs} \subseteq \Sigma$*), a set of secret states* $S \subseteq X$*, and the verifier* $G_{verifier} = (X_{verifier}, \Sigma_{obs}, \delta_{verifier}, X_{verifier,0})$ *constructed as in Definition 5.1.1. The verifier state* q *that is reachable from the verifier initial state* $X_{verifier,0}$ *via a finite-length string* $\omega \in \Sigma^*_{obs}$*,* $\omega \neq \epsilon$*, is associated with a state-status mapping* $q \in \Delta^X$ *such that*

(i) $\{(\mathbb{Y}(i), i) \in q, \mathbb{Y}(i) \in \{\mathbb{S}, \mathbb{NS}, \mathbb{M}\}\} \Leftrightarrow \{\exists i' \in X_0, \exists s \in \Sigma^*\{P(s) = \omega, i \in \delta(i', s)\}\}$*; moreover,* q *additionally satisfies the following:*

  (a) $(\mathbb{S}, i) \in q \Leftrightarrow \{\forall i' \in X_0\{\exists s \in \Sigma^*\{P(s) = \omega, i \in \delta(i', s)\}\} \Rightarrow \{i' \in S\}\}$*;*

  (b) $(\mathbb{NS}, i) \in q \Leftrightarrow \{\forall i' \in X_0\{\exists s \in \Sigma^*\{P(s) = \omega, i \in \delta(i', s)\}\} \Rightarrow \{i' \in X_0 - S\}\}$*;*

  (c) $(\mathbb{M}, i) \in q \Leftrightarrow \{\exists i' \in X_0 - S, \exists i'' \in X_0 \cap S, \exists s, s' \in \Sigma^*\{P(s) = P(s') = \omega, i \in \delta(i', s), i \in \delta(i'', s')\}\}$.

(ii) $(\mathbb{NR}, i) \in q \Leftrightarrow \{\forall i' \in X_0\{\nexists s \in \Sigma^*\{P(s) = \omega, i \in \delta(i', s)\}\}\}$. ■

Proof. (i) We prove part (i) by induction on the length of the string ω. Assume $\omega = \alpha_0 \ldots \alpha_n$ and denote the sequence of verifier states visited via ω by $q_0 \equiv X_{verifier,0}, q_1, \ldots, q_{n+1}$. First, we need to prove the induction hypothesis for $\omega = \alpha_0$. From Definition 5.1.1, we have $q_1 = q_0 \otimes M(\alpha_0)$. By

the definition of operator $\otimes$, we have

$$\begin{aligned}
&\{(\mathbb{Y}(i), i) \in q_1, \mathbb{Y}(i) \neq \mathbb{NR}\} \\
&\Leftrightarrow \{\exists i' \in X\{(i', i) \in M(\alpha_0), (\mathbb{Y}(i'), i') \in q_0, \mathbb{Y}(i') \neq \mathbb{NR}\}\} \\
&\Leftrightarrow \{\exists i' \in X, \exists s \in \Sigma^*\{i \in \delta(i', s), P(s) = \alpha_0, (\mathbb{Y}(i'), i') \in q_0, \\
&\quad \mathbb{Y}(i') \neq \mathbb{NR}\}\} \text{ (By definition of } M(\alpha_0)) \\
&\Leftrightarrow \{\exists i' \in X_0, \exists s \in \Sigma^*\{P(s) = \alpha_0, i \in \delta(i', s)\}\} \text{ (By definition of } q_0).
\end{aligned}$$

This completes the proof for $\omega = \alpha_0$. Next, we show that if the hypothesis of the induction is true for $\omega = \alpha_0\alpha_1 \ldots \alpha_{n-1}$, then it is also true for $\omega' = \alpha_0\alpha_1 \ldots \alpha_n$. From the induction hypothesis we have

$$\{(\mathbb{Y}(i'), i') \in q_n, \mathbb{Y}(i') \neq \mathbb{NR}\} \Leftrightarrow \{\exists j \in X_0, \exists r \in \Sigma^*\{P(r) = \omega, i' \in \delta(j, r)\}\}. \tag{5.1}$$

As mentioned before, the verifier state that is reachable via ω (ω') is q_n (q_{n+1}). From Definition 5.1.1, we have $q_{n+1} = q_n \otimes M(\alpha_n)$. By the definition of operator $\otimes$, we have

$$\begin{aligned}
&\{(\mathbb{Y}(i), i) \in q_{n+1}, \mathbb{Y}(i) \neq \mathbb{NR}\} \\
&\Leftrightarrow \{\exists i' \in X\{(i', i) \in M(\alpha_n), (\mathbb{Y}(i'), i') \in q_n, \mathbb{Y}(i') \neq \mathbb{NR}\}\} \\
&\Leftrightarrow \{\exists i' \in X, \exists t \in \Sigma^*\{i \in \delta(i', t), P(t) = \alpha_n, (\mathbb{Y}(i'), i') \in q_n, d\mathbb{Y}(i') \neq \mathbb{NR}\}\} \qquad (5.2) \\
&\Leftrightarrow \{\exists j \in X_0, \exists s \in \Sigma^*\{P(s) = \omega', i \in \delta(j, s)\}\} \qquad (5.3)
\end{aligned}$$

where (5.2) follows from the definition of $M(\alpha_0)$ and (5.3) follows from (5.1) with $s = rt$. This completes the proof for part (i).

(a), (b): We describe the proof for parts (a) and (b) focusing on part (a) (the proof for part (b) is similar). We use induction on the length of the string ω. Assume $\omega = \alpha_0 \ldots \alpha_n$ and denote the sequence of verifier states that are visited via ω by $q_0, q_1, \ldots, q_{n+1}$. First, we need to prove the induction hypothesis for $\omega = \alpha_0$. From Definition 5.1.1 we have $q_1 = q_0 \otimes M(\alpha)$. Note that the state-status mapping associated with the verifier initial state q_0 assigns a label $\mathbb{Y}$ from the set $\{\mathbb{S}, \mathbb{NS}\}$ to the pairs $(\mathbb{Y}, i)$ in the state-status mapping based on whether the system state element i of the pair is secret or not (and assigns $\mathbb{NR}$ if state i is not in X_0). From definition of the operator

$\otimes$, it follows that

$$
\begin{aligned}
&(\mathbb{S}, i) \in q_1 \\
&\Leftrightarrow \{\forall i' \in X_0\{(i', i) \in M(\alpha_0) \Rightarrow (\mathbb{S}, i') \in q_0\}\} \\
&\Leftrightarrow \{\forall i' \in X_0\{\{\exists s \in \Sigma^*\{P(s) = \alpha_0, i \in \delta(i', s)\}\} \Rightarrow \{(\mathbb{S}, i') \in q_0\}\}\} \quad (5.4) \\
&\Leftrightarrow \{\forall i' \in X_0\{\{\exists s \in \Sigma^*\{P(s) = \alpha_0, i \in \delta(i', s)\}\} \Rightarrow \{i' \in S\}\}\}, \quad (5.5)
\end{aligned}
$$

where (5.4) follows from definition of $M(\alpha_0)$ and (5.5) follows from definition of q_0. This proves the induction hypothesis for $\omega = \alpha_0$ for part (a).

Next, assuming that the hypothesis holds for $\omega = \alpha_0\alpha_1 \ldots \alpha_{n-1}$, we prove it for $\omega' = \alpha_0 \ldots \alpha_n$. Recall that q_{n+1} (q_n) is a verifier state reachable from the verifier initial state via ω' (ω) (in the theorem, verifier state q_{n+1} is denoted by q). By the induction hypothesis, we have

$$
(\mathbb{S}, i') \in q_n \Leftrightarrow \{\forall j \in X_0\{\{\exists r \in \Sigma^*\{P(r) = \omega, i' \in \delta(j, r)\}\} \Rightarrow \{j \in S\}\}\}. \quad (5.6)
$$

By construction of the verifier, we have $q_{n+1} = q_n \otimes M(\alpha_n)$. From definition of the operator $\otimes$ we have

$$
\begin{aligned}
&(\mathbb{S}, i) \in q_{n+1} \\
&\Leftrightarrow \{\forall i' \in X\{(i', i) \in M(\alpha_n) \Rightarrow (\mathbb{S}, i') \in q_n\}\} \\
&\Leftrightarrow \{\forall i' \in X\{\exists t \in \Sigma^*\{P(t) = \alpha_n, i \in \delta(i', t)\} \Rightarrow \{(\mathbb{S}, i') \in q_n\}\}\} \quad (5.7) \\
&\Leftrightarrow \{\forall j \in X_0\{\exists s \in \Sigma^*\{P(s) = \omega', i \in \delta(j, s)\} \Rightarrow \{j \in S\}\}\}, \quad (5.8)
\end{aligned}
$$

where (5.7) follows from definition of $M(\alpha_n)$ and (5.8) follows from (5.6) with $s = rt$. This completes the proof for part (a).

(c) Similar to the previous parts, we use induction on the length of the string ω. Assume $\omega = \alpha_0 \ldots \alpha_n$ and denote the sequence of verifier states visited via ω by $q_0, q_1, \ldots, q_{n+1}$. First, we need to prove the induction hypothesis

for $\omega = \alpha_0$. From the definition of operator $\otimes$, we have

$$
\begin{aligned}
&(\mathbb{M}, i) \in q_1 \Leftrightarrow \\
&\{\exists (i', i), (i'', i) \in M(\alpha_0)\{(\mathbb{NS}, i'), (\mathbb{S}, i'') \in q_0\}\} \Leftrightarrow \\
&\{\exists i', i'' \in X_0, \exists s, s' \in \Sigma^*\{P(s) = P(s') = \alpha_0, i \in \delta(i', s), i \in \delta(i'', s'), \\
&(\mathbb{NS}, i') \in q_0, (\mathbb{S}, i'') \in q_0\}\} \Leftrightarrow \\
&\{\exists i' \in X_0 - S, \exists i'' \in X_0 \cap S, \exists s, s' \in \Sigma^*\{P(s) = P(s') = \alpha_0, i \in \delta(i', s), \\
&i \in \delta(i'', s')\}\},
\end{aligned}
$$

where the last equation follows from the definition of q_0. The above argument establishes the hypothesis of the induction for $\omega = \alpha_0$.

Next, assuming that the hypothesis holds for $\omega = \alpha_0\alpha_1 \ldots \alpha_{n-1}$, we prove it for $\omega' = \alpha_0 \ldots \alpha_n$. Recall that q_{n+1} (q_n) is a verifier state reachable from the verifier initial state via ω' (ω) (in the theorem, verifier state q_{n+1} is denoted by q). By the induction hypothesis, we have

$$
\begin{aligned}
&(\mathbb{M}, i') \in q_n \Leftrightarrow \\
&\{\exists j \in X_0 - S, \exists j' \in X_0 \cap S, \exists r, r' \in \Sigma^*\{P(r) = P(r) = \omega, i' \in \delta(j, r), \\
&i' \in \delta(j', r')\}\}. \qquad (5.9)
\end{aligned}
$$

By construction of the verifier, we have $q_{n+1} = q_n \otimes M(\alpha_n)$. By definition of the operator $\otimes$, it follows that

$$
\begin{aligned}
&(\mathbb{M}, i) \in q_{n+1} \Leftrightarrow \\
&\{\{\exists (i', i), (i'', i) \in M(\alpha_n)\{(\mathbb{NS}, i'), (\mathbb{S}, i'') \in q_n\}\} \text{ or} \\
&\{\exists (i', i) \in M(\alpha_n)\{(\mathbb{M}, i') \in q_n\}\}\} \Leftrightarrow \\
&\{\{\exists (\mathbb{NS}, i'), (\mathbb{S}, i'') \in q_n, \exists t, t' \in \Sigma^*\{P(t') = P(t) = \alpha_n, \\
&i \in \delta(i', t), i \in \delta(i'', t')\}\} \text{ or} \qquad (5.10) \\
&\{\exists (\mathbb{M}, i') \in q_n, \exists t \in \Sigma^*\{P(t) = \alpha_n, i \in \delta(i', t)\}\}\} \Leftrightarrow \qquad (5.11) \\
&\{\exists j \in X_0 - S, \exists j' \in X_0 \cap S, \exists s, s' \in \Sigma^*\{P(s) = P(s') = \omega', \\
&i' \in \delta(j, s), i' \in \delta(j', s')\}\}. \qquad (5.12)
\end{aligned}
$$

Note that (5.10) is equivalent to (5.12) by parts (a) and (b) and (5.11) is equivalent to (5.12) by (5.9) with $s = rt$ and $s' = r't'$. This completes the

proof for part (c).

(ii) In part (i) we showed that pair $(\mathbb{Y}(i), i)$ in the state-status mapping q (which is reachable in the verifier via ω) has label $\mathbb{Y}(i) \neq \mathbb{NR}$ if and only if there exists an initial state i' and a string s such that $i \in \delta(i', s)$ is nonempty and $P(s) = \omega$. This is equivalent to $\mathbb{Y}(i) = \mathbb{NR}$ if and only if for all initial states i', there does not exist any string s such that $i \in \delta(i', s)$ and $P(s) = \omega$. Therefore, part (ii) follows from part (i). □

Since the verifier has all the information concerning the origin of any sequence of observations in the system, it can be used to verify initial-state opacity. We need the following definition before formalizing this with a theorem.

Definition 5.1.2. *Given a set of states X and a set of labels $\Delta = \{\mathbb{S}, \mathbb{NS}, \mathbb{M}, \mathbb{NR}\}$, a state-status mapping $q \in \Delta^X$ is $\mathbb{NR}$-certain if*

$$\forall i \in X\{\mathbb{Y}(i) = \mathbb{NR}\}.$$

A state-status mapping that is not $\mathbb{NR}$-certain is said to be $\mathbb{L}$-certain ($\mathbb{L} \in \{\mathbb{N}, \mathbb{NS}, \mathbb{M}\}$) if

$$\forall i \in X\{\mathbb{Y}(i) = \mathbb{L} \text{ or } \mathbb{Y}(i) = \mathbb{NR}\}.$$

■

Theorem 5.1.4. *Consider a non-deterministic finite automaton $G = (X, \Sigma, \delta, X_0)$ with set of states $X = \{0, 1, \ldots, N-1)$, a natural projection map P with respect to the set of observable events Σ_{obs} ($\Sigma_{obs} \subseteq \Sigma$), a set of secret states $S \subseteq X$, and the verifier $G_{verifier} = (X_{verifier}, \Sigma_{obs}, \delta_{verifier}, X_{verifier,0})$ constructed as in Definition 5.1.1. Automaton G is (S, P, ∞) initial-state opaque if and only if*

$$\forall q \in X_{verifier}\{q \text{ is not } \mathbb{S}\text{-certain}\}.$$

■

Proof. First observe that in an initial-state opaque system, for any string $t \in \Sigma^*$ that originates from a secret initial state i ($t \in L(G, i), i \in X_0 \cap S$), there exists another string $s \in \Sigma^*$ that originates from a non-secret initial state j ($s \in L(G, j), j \in X_0 - S$) that has the same projection as t, i.e., $P(s) = P(t)$.

More formally, $\forall i \in X_0 \cap S, \forall t \in \Sigma^*$

$$\{\delta(i,t) \neq \emptyset \Rightarrow \{\exists j \in X_0 - S, \exists s \in \Sigma^* \{P(s) = P(t), \delta(j,s) \neq \emptyset\}\}\}.$$

In Theorem 5.1.3, we proved that for each state of the verifier $G_{verifier}$, the associated state-status mapping captures the origin of any sequence in the system that leads to state $i \in X$, i.e., whether state i is reachable from secret initial states, non-secret initial states, both secret and non-secrets initial states, or not reachable from any initial state at all. If the origin of the given sequence is not reachable from any initial state, q is $\mathbb{NR}$-certain. In such case, initial-state opacity is not violated since the definition of initial-state opacity only restricts attention to sequences of observations that can be generated in the system; therefore, for $\omega \in \Sigma^*_{obs}$, $\omega \neq \epsilon$,

$$\{\exists q \in X_{verifier} \{q = \delta_{verifier}(X_{verifier,0}, \omega), q \text{ is } \mathbb{S}\text{-certain}\}\} \Leftrightarrow$$
$$\{\forall j \in X_0, \forall t \in \Sigma^* \{\{P(t) = \omega, \delta(j,t) \neq \emptyset\} \Rightarrow j \in S\}\}.$$

This violates initial-state opacity and (together with the initial case of $\omega = \epsilon$ which clearly holds) completes the proof. □

Remark 5.1.1. *By construction of the verifier, once the state-status mapping associated with a verifier state becomes $\mathbb{L}$-certain, $\mathbb{L} \in \{\mathbb{S}, \mathbb{NS}, \mathbb{M}\}$, it remains $\mathbb{L}$-certain or it becomes $\mathbb{NR}$-certain. Hence, for verification purposes, we need not generate subsequent states from a verifier state whose associated state-status mapping is $\mathbb{L}$-certain. Moreover, the state-status mapping associated with any verifier state reachable from a state for which the associated state-status mapping only contains labels $\mathbb{M}$, $\mathbb{NS}$, and/or $\mathbb{NR}$, cannot become $\mathbb{S}$-certain. Therefore, we need not generate subsequent states from verifier states whose associated state-status mappings only contain labels $\mathbb{M}$, $\mathbb{NS}$, and/or $\mathbb{NR}$.* ■

Example 5.1.3. *In this example, we first show that the DES G in Figure 5.1 with $X_0 = X$ is initial-state opaque with respect to $S = \{2\}$. For this, we construct the verifier $G_{verifier}$ for G as depicted in Figure 5.2. The verifier initial state is $q_0 = \{(\mathbb{NS}, 0), (\mathbb{NS}, 1), (\mathbb{S}, 2), (\mathbb{NS}, 3)\}$ denoting the fact that system state 2 is secret. Upon observing α (and following the details described*

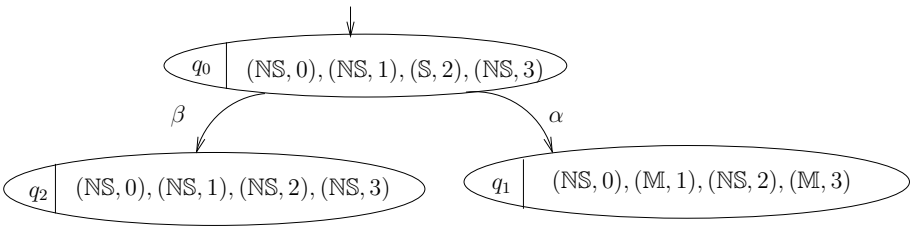

Figure 5.2: Verifier $G_{verifier}$ corresponding to G for $S = \{2\}$.

in Example 5.1.2), the verifier next state becomes

$$q_1 = \{(\mathbb{NS}, 0), (\mathbb{M}, 1), (\mathbb{NS}, 2), (\mathbb{M}, 3)\}.$$

This implies that upon observing α, the current system state is (i) either 0 or 2, which are only reachable from non-secret initial states, or (ii) either 1 or 3, each of which is reachable from both secret and non-secret initial states. The generation of the states in the $G_{verifier}$ is stopped at state q_1 with associated state-status mapping $\{(\mathbb{NS}, 0), (\mathbb{M}, 1), (\mathbb{NS}, 2), (\mathbb{M}, 3)\}$ since q_1 only contains labels $\mathbb{M}$ and $\mathbb{NS}$ (Remark 5.1.1). One can also verify that upon observing β from initial state q_0, the verifier state q_2 that is reached is associated with state-status mapping $\{(\mathbb{NS}, 0), (\mathbb{NS}, 1), (\mathbb{NS}, 2), (\mathbb{NS}, 3)\}$. Transitions from verifier state q_2 also need not be considered further since the state-status mapping of this state is $\mathbb{NS}$-certain (Remark 5.1.1). Since the verifier does not (and cannot) contain any state such that the associated state-status mapping is $\mathbb{S}$-certain, we conclude that DES G is initial-state opaque with respect to $S = \{2\}$.

Next, we assume that $S = \{1\}$. The verifier initial state in this case is

$$q_0 = \{(\mathbb{NS}, 0), (\mathbb{S}, 1), (\mathbb{NS}, 2), (\mathbb{NS}, 3)\}$$

denoting the fact that system state 1 is secret. The verifier construction can be completed for all possible observations (from each state) as shown in Figure 5.3. Note that for verifier states that are $\mathbb{S}$-, $\mathbb{NS}$-, or $\mathbb{M}$-certain, we do not need to consider any outgoing transitions (we ignore the $\mathbb{NR}$-certain state since it is only reached via sequences of observations that cannot be generated by the system). As can be observed, verifier state q_3, which is associated with state-status mapping $\{(\mathbb{S}, 0), (\mathbb{S}, 1), (\mathbb{S}, 2), (\mathbb{S}, 3)\}$ and is reachable from the verifier initial state via $\beta\alpha$, is $\mathbb{S}$-certain. Therefore, DES G is not initial-

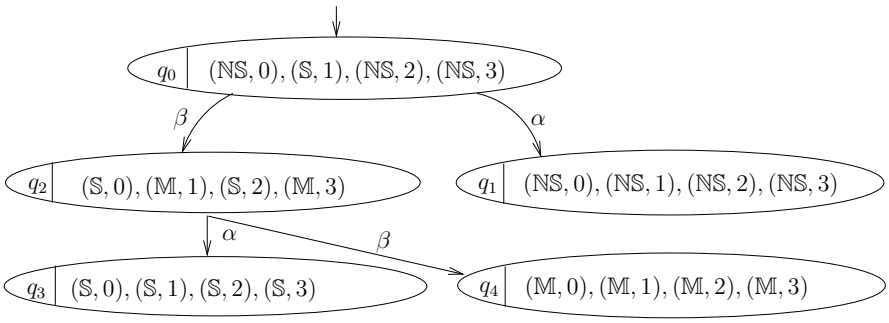

Figure 5.3: Verifier $G_{verifier}$ corresponding to G for $S = \{1\}$.

state opaque with respect to $S = \{1\}$. As mentioned before, this means that observing string $\beta\alpha$ positively determines the system initial state as state 1, which is within the set of secret states and hence violates initial-state opacity. ■

Remark 5.1.2. *The verifier introduced in Definition 5.1.1 has state complexity $O(4^N)$ where $N = |X|$ denotes the number of states of the underlying automaton G. As a result, checking for initial-state opacity has space complexity $O(4^N)$ and similar time complexity. When compared to the $O(2^{N^2})$ space and time complexity for verifying initial-state opacity using the ISE, the employment of a verifier results in a large reduction in complexity. However, there are situations where the use of the ISE might be preferable:*

(i) If the set of secret states is not known or is not invariant, then the reduced complexity method requires a new verifier to be constructed for each possible set of secret states, whereas the ISE can be used to verify initial-state opacity for any set. For $|X| = N$, there are at most 2^N possible sets of secret states and thus verifying initial-state opacity for all possible sets of secret states using a verifier has state space complexity $O(2^N \times 4^N)$ or equivalently, $O(8^N)$. This is still better than constructing an ISE which has state space complexity $O(2^{N^2})$.

(ii) The ISE can be used to provide a "measure" of opacity: specifically, the cardinality of the set of non-secret starting states (or the percentage of non-secret possible starting states over all possible starting states) of the state mappings associated with the states of the ISE can provide a measure of opacity (note that such simple measures of opacity have been frequently employed in cases where anonymity is desirable [38]).

(iii) If the set of secret states S changes along the observation, the ISE can provide useful information regarding initial-state opacity; this case is discussed in more detail in the next section. ■

5.1.3 Verifying Initial-State Opacity for a Time-Varying Set of Secret States

In this section, we consider cases where the set of secret states varies along the observation (in contrast to the invariant set of secret states that was assumed in the previous section). There are many security applications where this might be the case. For example, in a multi-level clearance system, upon an upgrade in the clearance level (modeled via an observable event), the user can have a more refined picture of the initial state of the system; similarly, upon a downgrade in the clearance level, the information that the user can infer about the initial state of the system becomes limited [39].

In the modeling of discrete event systems, the evolution of states is based upon the occurrence of events that may happen at *any* point in time. Hence, the notion of time is not captured within this framework. In order to overcome this restriction in modeling the time-varying behavior of the set of secret states, we follow an approach inspired by Ramadge and Wonham's supervisory control framework [15] (where control requirements are described as sub-languages of the system language $L(G)$). Specifically, we assume that the changes in the set of secret states are synchronized with observations, i.e., occurrences of observable events. In other words, we assume that the set of secret states is given by a mapping $S : P(L(G)) \rightarrow 2^X$ which maps each projection $\omega \equiv P(t)$ of a string t in $L(G)$ to a subset of the set X. Note that this constrains the nature of time-variations in the set of secret states (to be the same for system behaviors that generate the same sequence of observations, i.e., for $t_1, t_2 \in L(G)$ such that $P(t_1) = P(t_2)$).

In general, the mapping function $S(\cdot)$ may require infinite space to be described. For this reason, we assume that the given automaton G (for which initial-state opacity needs to be verified) is associated with a deterministic finite automaton $G_s = (X_s, \Sigma_{obs}, \delta_s, x_{s,0})$ that models the changes in the set of secret states. More specifically, each state of the automaton G_s is associated with a set of secret states (with the initial state $x_{s,0}$ of the automaton G_s

associated with the initial set of secret states); a state transition in G_s can occur after an observable event and models a change in the state of G_s and possibly in the associated set of secret states (note that multiple automaton states in X_s could be associated with the same set of secret states). It is assumed that the DES G_s contains all possible sequences of observations that can be generated by DES G (i.e., $P(L(G)) \subseteq L(G_s)$) so that, by following a sequence of observations, we can capture the corresponding set of secret states (in other words, $S(\omega) = \mathcal{F}(\delta_s(x_{s,0}, \omega))$ where $\mathcal{F} : X_s \to 2^X$ is the mapping that associates to each state of G_s a set of secret states). The definition of initial-state opacity for G can then be trivially extended to include this time-varying set of secret states as follows.

Definition 5.1.3 (Initial-State Opacity for Time-Varying Set of Secret States)**.** *Consider a non-deterministic finite automaton $G = (X, \Sigma, \delta, X_0)$, a natural projection map P with respect to the set of observable events Σ_{obs} ($\Sigma_{obs} \subseteq \Sigma$), and a (time-varying) set of secret states $S : P(L(G)) \to 2^X$ that is specified via a deterministic finite automaton $G_s = (X_s, \Sigma_{obs}, \delta_s, x_{s,0})$ that satisfies $P(L(G)) \subseteq L(G_s)$ and is associated with a function $\mathcal{F} : X_s \to 2^X$, such that*

$$S(\omega) = \mathcal{F}(\delta_s(x_{s,0}, \omega))$$

for any string $\omega \in P(L(G))$. Then, automaton G is initial-state opaque with respect to $S(\cdot)$ and P (or $(S(\cdot), P, \infty)$ initial-state opaque) if for all $t \in L(G, i)$ with $P(t) = \omega$ and for all $i \in X_0 \cap S(\omega)$, we have

$$\exists j \in X_0 - S(\omega), \exists s \in L(G, j)\{P(s) = \omega\}.$$

■

In order to verify initial-state opacity when the set of secret states S varies as in Definition 5.1.3, one needs to verify that for each string t in $L(G)$ with $P(t) = \omega$ such that t can originate from a state in $X_0 \cap S(\omega)$, there exists a string s with $P(s) = P(t) = \omega$ such that string s can originate from a state in $X_0 - S(\omega)$. In the following theorem, we show how one can modify the ISE construction to verify initial-state opacity for the case of a time-varying set of secret states.

Theorem 5.1.5. *Consider a non-deterministic finite automaton $G = (X, \Sigma, \delta, X_0)$, a natural projection map P with respect to the set of observable events*

Σ_{obs} ($\Sigma_{obs} \subseteq \Sigma$), and a time-varying set of secret states $S : P(L(G)) \rightarrow 2^X$ that is specified by the state of the deterministic automaton $G_s = (X_s, \Sigma_{obs}, \delta_s, x_{s,0})$ in Definition 5.1.3. Construct the ISE $G_{\infty,obs} = (X_{\infty,obs}, \Sigma_{obs}, \delta_{\infty,obs}, X_{\infty,0})$ and the automaton $G^*_{\infty,obs} = G_{\infty,obs} \times G_s = (X^*_{\infty,obs}, \Sigma_{obs}, \delta^*_{\infty,obs}, X^*_{\infty,0})$, where $X^*_{\infty,obs}$ is the set of states in $G^*_{\infty,obs}$ that are reachable from its initial state $X^*_{\infty,0} = X_{\infty,0} \times x_{s,0}$. Then, automaton G is $(S(\cdot), P, \infty)$ initial-state opaque if and only if for all $(m, x_s) \in X^*_{\infty,obs}$,

$$m(1) \nsubseteq \mathcal{F}(x_s) \text{ or } m(1) = \emptyset. \tag{5.13}$$

■

Proof. To prove the only if part, we assume that system is $(S(\cdot), P, \infty)$ initial-state opaque. Then for all $s \in \Sigma^*$ with $P(s) = \omega$

$$\{\exists i \in S(\omega) \cap X_0\{\delta(i, s) \neq \emptyset\}\} \Rightarrow$$

$$\{\exists t \in \Sigma^*, \exists i' \in X_0 - S(\omega)\{P(t) = \omega, \delta(i', t) \neq \emptyset\}\}.$$

Denote the state mapping associated with the state reached in $G_{\infty,obs}$ via ω, by m. By Theorem 4.2.1, we know that $m(1)$ contains both system states i and i'. Since $i \in X_0 \cap S(\omega)$ and $i' \in X_0 - S(\omega)$, this implies that $m(1)$ contains system states in both $X_0 \cap S(\omega)$ and $X_0 - S(\omega)$; therefore, $m(1) \nsubseteq S(\omega)$ which completes the proof for the only if part. Obviously, for all $s \in \Sigma^*$ such that $s \notin L(G)$, $G^*_{\infty,obs}$ will be driven by $\omega = P(s)$ to a state (m, x_s) such that $m = \emptyset$.

To prove the (if) part, assume that (5.13) holds and use contradiction. If the system is not $(S(\cdot), P, \infty)$ initial-state opaque, then there exists $s \in \Sigma^*$ with $P(s) = \omega$ such that $\exists i \in X_0 \cap S(\omega)\{\delta(i, s) \neq \emptyset$ and

$$\{\forall t \in \Sigma^*, \forall i' \in X_0\{P(t) = \omega, \delta(i', t) \neq \emptyset\} \Rightarrow \{i' \in X_0 \cap S(\omega)\}\}.$$

Denote the state mapping associated with the state reached in $G_{\infty,obs}$ via ω, by m. By Theorem 4.2.1, we know that $m(1)$ contains system state i and all system states i' from which strings t with $P(t) = \omega$ can originate. Also, since $i, i' \in S(\omega)$, this implies that $m(1) \neq \emptyset$ and $m(1)$ contains system states in $S(\omega)$; this contradicts (5.13) and hence completes the proof for the (if) part. □

5.2 Verifying K-Step Opacity

In this section, we show that for a DES G to be K-step opaque, it is necessary and sufficient that each k-delayed state estimate $\hat{X}_{|\omega|-k}(\omega)$, $0 \leq k \leq \min(K, |\omega|)$, associated with a sequence of observations ω contain at least one state outside the set of secret states S (unless the sequence of observations ω cannot be generated by G in which case $\hat{X}_{|\omega|-k}(\omega) = \emptyset$).

Theorem 5.2.1. *Given a non-deterministic finite automaton $G = (X, \Sigma, \delta, X_0)$, a natural projection map P with respect to the set of observable events Σ_{obs} ($\Sigma_{obs} \subseteq \Sigma$), and a set of secret states $S \subseteq X$, automaton G is (S, P, K)-opaque if and only if for all $\omega \in \Sigma^*_{obs}$, $0 \leq k \leq \min(K, |\omega|)$*

$$\hat{X}_{|\omega|-k}(\omega) \not\subseteq S \text{ or } \hat{X}_{|\omega|-k}(\omega) = \emptyset, \tag{5.14}$$

where $\hat{X}_{|\omega|-k}(\omega)$ is the k-delayed state estimate associated with the sequence of observations ω. ■

Proof. (Only if) Proof by contradiction. Assume that there exists a k-delayed state estimate $\hat{X}_{|\omega|-k}(\omega)$, $0 \leq k \leq K$, associated with a sequence of observations $\omega = \alpha_0 \ldots \alpha_n$ with $|\omega| \geq K$ (for now), such that $\hat{X}_{|\omega|-k}(\omega) \subseteq S$ and $\hat{X}_{|\omega|-k}(\omega) \neq \emptyset$. From Definition 4.5.1, this implies that there exists $t \in \Sigma^*$ with $P(t) = \alpha_0\alpha_1 \ldots \alpha_n$ such that for some k in $0 \leq k \leq K$ we have

$$\forall i \in X_0\{\delta(i, t) \neq \emptyset\} \Rightarrow$$

$$\{\forall t' \in \bar{t}\{|P(t)/P(t')| = k\}\forall j \in \delta(i, t')\{\delta(j, t/t') \neq \emptyset \Rightarrow j \in S\}\}. \tag{5.15}$$

Unless $\hat{X}_{|\omega|-k}(\omega) = \emptyset$ (in which case $\delta(i, t)$ would be empty as well), we conclude that there exists at least one string t that passes through the set of secret states S at some point k within the past K observations and all sequences in the system that generate the same sequence of observations pass through secret states when t passes through secret states; therefore, the system is not K-step opaque.

To handle the case when $|\omega| < K$, we assume that there exists a k-delayed state estimate $\hat{X}_{|\omega|-k}(\omega)$, $0 \leq k \leq |\omega|$, associated with a sequence of observations $\omega = \alpha_0 \ldots \alpha_n$ with $|\omega| < K$, such that $\hat{X}_{|\omega|-k}(\omega) \subseteq S$ and $\hat{X}_{|\omega|-k}(\omega) \neq \emptyset$. From Definition 4.5.1, this implies that (5.15) holds and that the system is not K-step opaque (following the same reasoning as in the case for $|\omega| \geq K$).

(If) Assume that the system is not K-step opaque. Hence, from Definition 3.2.1, there exists a string t in $L(G)$ that visits state(s) j in S within the past K observations (i.e., there exists $t' \in \bar{t}$ such that for some $i \in X_0$ and $j \in S$, $j \in \delta(i, t')$, $\delta(j, t/t') \neq \emptyset$) such that for all strings s in $L(G)$ with $P(s) = P(t)$, when string t passes through the state(s) j in S, string s passes through state(s) $j' \in S$. This means that (i) state(s) j reachable from initial state(s) $i \in X_0$ via string $t' \in \bar{t}$ and with continuation(s) via string t/t' belongs (belong) to S; (ii) state(s) $j' \in S$ reachable from initial state $i' \in X_0$ via string $s' \in \bar{s}$ with continuation(s) via string s/s' belongs (belong) to S; and (iii) $P(t') = P(s')$ and $|P(t)/P(t')| = |P(s)/P(s')| = k$ with $0 \leq k \leq K$. Now consider the k-delayed state estimate $\hat{X}_{|\omega|-k}(\omega)$ with $\omega \equiv P(t)$. From Definition 4.5.1 we know that $\hat{X}_{|\omega|-k}(\omega)$ includes all states j and j' characterized above. Also, since by assumption, the sequence of observations ω violates K-step opacity, we have for all $j \in \hat{X}_{|\omega|-k}(\omega)$: $j \in S$. Hence, there exists $\omega \in \Sigma_{obs}^*$ and $0 \leq k \leq \min(K, |\omega|)$ such that $\hat{X}_{|\omega|-k}(\omega) \subseteq S$ and $\hat{X}_{|\omega|-k}(\omega) \neq \emptyset$, which completes the proof. □

Corollary 4.6.1 proves that SM-KDE captures the set of all k-delayed state estimates, $0 \leq k \leq K$, via its $(K+1)$-dimensional state mappings and, hence, by Theorem 5.2.1, it can be used for verifying K-step opacity.

Theorem 5.2.2. *Consider a non-deterministic finite automaton* $G = (X, \Sigma, \delta, X_0)$ *with a natural projection map* P *with respect to the set of observable events* Σ_{obs} *(*$\Sigma_{obs} \subseteq \Sigma$*), and its SM-KDE* $G_{K,obs} = (X_{K,obs}, \Sigma_{obs}, \delta_{K,obs}, X_{K,0})$ *constructed as in Definition 4.6.1. Automaton* G *is* (S, P, K)*-opaque if and only if for all* $m \in X_{K,obs}$, $k \in \{0, \ldots, K\}$

$$m(k) \not\subseteq S \text{ or } m(k) = \emptyset. \tag{5.16}$$

■

Proof. Follows from Theorem 5.2.1 and Corollary 4.6.1. Note that $m(k) = \emptyset$ implies that this state is reached only via sequences of observations that cannot be generated by the original system. Also, we do not need to treat the case when $|\omega| < K$ separately because for $|\omega| \leq k \leq K$, $\hat{X}_{|\omega|-K}(\omega) = X_0$ or $\hat{X}_{|\omega|-K}(\omega) = \emptyset$. □

Remark 5.2.1. *If we assume that the system is "alive with respect to observable events" (i.e., if we require that it is always possible to observe a sequence*

of events of arbitrary length), then it is not hard to argue that (5.16) can be replaced with the following simpler form:

$$\forall m \in X_{K,obs} : m(K) \not\subseteq S \text{ or } m(K) = \emptyset. \tag{5.17}$$

In other words, assuming that the system is alive with respect to observable events, in order to verify K-step opacity, one need only check the K-delayed state estimates and verify that they do not fall entirely within the set of secret states (as opposed to (5.16) where all k-delayed state estimates, $0 \leq k \leq K$, need to be verified). This assumption is less stringent than the commonly accepted assumptions for the diagnosis problem [40] where it is required that no sequence of arbitrary length of unobservable events can occur in the system. In other words, the diagnosis problem is formulated under the assumptions that (i) G is live, and (ii) there exist no cycles of unobservable events. For Condition (5.16) to be relaxed into Condition (5.17), we require (i) to hold for observable events but (ii) is not required. For a detailed comparison between diagnosability and K-step opacity please refer to Section 3.5.4. ■

Example 5.2.1. *DES G in Figure 4.1-a with $X_0 = \{0, 1, 2, 3, 4\}$ is not $(\{0\}, P, 2)$-opaque due to the existence of state m_4 (or m_6) in the state mapping-based 2-delay state estimator depicted in Figure 4.3. If the system generates the sequence of observations $\alpha\beta$ (or $\beta\beta$), then (since the only state from which $\alpha\beta$ or $\beta\beta$ can be observed is state 0) we can conclude with certainty that the system was in state 0 two steps ago. This violates the 2-step opacity requirement since state 0 is a secret state. The unit-delay state estimator for this system is shown in Figure 5.4 (again we have not included the state that corresponds to the empty state mapping); it can be verified that for each of the 2-dimensional state mappings m associated with its states, every set of intermediate states $m(k), 0 \leq k \leq 1$, contains at least one element outside S. Hence, DES G is $(\{0\}, P, 1)$-opaque. This demonstrates that K-step opacity does not in general imply K'-step opacity for $K' > K$. The converse, however, is trivially true, i.e., K'-step opacity implies K-step opacity for $K' > K$.* ■

Corollary 4.6.2 proves that the OS-KDE captures the set of all k-delayed state estimates, $0 \leq k \leq K$, via its $(K + 2)$-tuples and hence, by Theorem 5.2.1, it can be used for verifying K-step opacity.

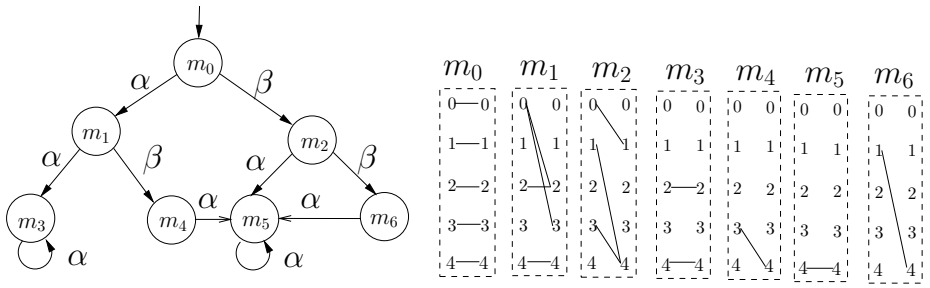

Figure 5.4: State mapping-based 1-delay state estimator corresponding to DES G.

Theorem 5.2.3. *Consider a non-deterministic finite automaton* $G = (X, \Sigma, \delta, X_0)$ *and a natural projection map* P *with respect to the set of observable events* Σ_{obs} *(*$\Sigma_{obs} \subseteq \Sigma$*), and its OS-KDE* $G_{K,obs}^{observation} = (X_{K,obs}^{observation}, \Sigma_{obs}, \delta_{K,obs}^{observation}, X_{K,0}^{observation})$ *constructed as in Definition 4.6.2. Automaton* G *is* (S, P, K)*-opaque if and only if for all* $Q = (\Omega, Z_K, \ldots, Z_0) \in X_{K,obs}^{observation}$, $k \in \{0, \ldots, K\}$,

$$Z_k \not\subseteq S \text{ or } Z_k = \emptyset. \tag{5.18}$$

■

Proof. Follows from Theorem 5.2.1 and Corollary 4.6.2. Note that $Z_k = \emptyset$ implies that this state is reached only via sequences of observation that cannot be generated by the original system. □

Example 5.2.2. *As discussed in Example 5.2.1, DES* G *in Figure 4.1-a with* $X_0 = \{0, 1, 2, 3, 4\}$ *is not* $(\{0\}, P, 2)$*-opaque since observing the sequence of observations* $\alpha\beta$ *(or* $\beta\beta$*) reveals that the system was in state 0 two steps ago and state 0 is a secret state. Note that the reachable states in the observation sequence-based 2-delay state estimator (depicted in Figure 4.4) via* $\alpha\beta$ *(or* $\beta\beta$*) are* $Q_4 = (\alpha\beta, \{0\}, \{3\}, \{4\})$ *(or* $Q_6 = (\beta\beta, \{0\}, \{1\}, \{4\})$*). The 2-delayed state estimate associated with either of these states is* $\{0\}$ *and falls entirely within the set of secret states, which means that the system is not* $(\{0\}, P, 2)$*-opaque.* ■

The exponential complexity of the algorithms proposed in this chapter for verifying K-step opacity is not desirable for implementation purposes. However, in Chapter 6, we show that deciding whether the non-deterministic finite automaton G is K-step opaque is NP-hard for $|\Sigma_{obs}| > 1$. This implies

that it is unlikely that any algorithm can verify K-step opacity in polynomial time.

Remark 5.2.2. *In Section 3.5.1, we introduce the notion of trajectory-based K-step opacity. It is not hard to see that DES G is trajectory-based K-step opaque if and only if for any given sequence of observations ω, there always exists at least one compatible sequence of states such that G visits exclusively non-secret states while generating the last K events in ω. Since the SM-KDE captures the sequence of states via the K-dimensional state mappings, we easily conclude that trajectory-based K-step opacity can be verified using K-delay state estimators: automaton G is trajectory-based K-step opaque if and only if, in the SM-KDE associated with the system, for all nonempty $m \in X_{K,obs}$, there exists $(i_0, i_1, \ldots, i_K) \in m$ such that $i_0, i_1, \ldots, i_K \in X - S$.*■

5.3 Verifying Infinite-Step Opacity

5.3.1 Verifying Infinite-Step Opacity Using K-Delay State Estimator

As mentioned before, infinite-step opacity can be considered as the limiting case of K-step opacity as K approaches infinity. Note that in this case, the K-delay state estimator is not a finite structure anymore (because K approaches infinity) and hence it is not useful in modeling the intruder or verifying infinite-step opacity. However, in this section, we show that for $K \geq 2^{N^2} - 1$, K-step opacity implies infinite-step opacity and hence the verification method for K-step opacity based on K-delay state estimators can be used to verify infinite-step opacity. Specifically, we show that for $K' > K \geq 2^{N^2} - 1$, K-step opacity and K'-step opacity are equivalent; therefore, since infinite-step opacity is the limiting case of K-step opacity as $K \to \infty$, we can state that K-step opacity for $K \geq 2^{N^2} - 1$ is equivalent to infinite-step opacity (the other direction is obviously true: infinite-step opacity always implies K-step opacity). Note that K-step opacity does not in general imply K'-step opacity for $K' > K$ (in fact, Example 5.2.1 is a demonstration of this).

The idea behind the proof is the following. Fix a point in the system's

state trajectory; in the K-step opacity problem we are interested in finding how much we can say regarding the membership of the state, at that fixed point in time, to the set of secret states, even after we make K additional observations. We can gain insight to this question by considering the estimate of the state at this fixed point as the *initial uncertainty* for an initial-state estimation problem. Note that here we are concerned with the information conveyed by the *set* of all sequences of observations of length at most K, and not a specific sequence of observations.

We carry the formal proof of the fact that K-step opacity and K'-step opacity are equivalent for $K' > K \geq 2^{N^2} - 1$ in two theorems. First, in Theorem 5.3.1, we prove that for the aforementioned fixed point in the state trajectory, for $K \geq 2^{N^2} - 1$, the information required for opacity verification (i.e., membership of the state at that point in time in the set of secret states S) is equivalent over all possible observation sequences of length K. Then, in Theorem 5.3.2, we show that this equivalence results in the equivalence of K-step opacity and K'-step opacity for $K' > K \geq 2^{N^2} - 1$.

Theorem 5.3.1. *Consider a non-deterministic finite automaton* $G = (X, \Sigma, \delta, X_0)$ *with* $|X| = N$*, and construct* $G_{K,obs}$ *and* $G_{K^*,obs}$ *for* $K > K^* = 2^{N^2} - 1$ *as delayed state estimators as described in Definition 4.6.1. Then, for any* $(K+1)$*-dimensional state mapping* m *associated with the SM-KDE state reached in* $G_{K,obs}$ *via* $\omega = \alpha_0\alpha_1 \ldots \alpha_n$ *with* $|\omega| > 2^{N^2} - 1$ *and for each* $m(k)$*,* $2^{N^2} \leq k \leq K$*, there exists a* (K^*+1)*-dimensional state mapping* m' *associated with a state reached in* $G_{K^*,obs}$ *via some* $\omega' = \alpha_0\alpha_1 \ldots \alpha_{n-k}\alpha'_{n-k+1} \ldots \alpha'_{n'}$ *for some* $n' \leq n + 2^{N^2} - 1 - k$ *and with* $\alpha'_{n-p} \in \Sigma_{obs}, k-1 \leq p \leq n-n'$*, such that* $m(k) = m'(k + n' - n)$. ■

Proof. Recall that in any K-delay state estimator $G_{K,obs}$, the k-delayed state estimate due to observation ω is captured via the set $m(k)$, where m is the $(K+1)$-dimensional state mapping associated with the state reached in $G_{K,obs}$ via $\omega = \alpha_0\alpha_1 \ldots \alpha_n$ (k satisfies $0 \leq k \leq K$). Now consider the fixed point in time after the sequence of observations $\alpha_0\alpha_1 \ldots \alpha_{n-k}$ has been observed. Once k more observations are made (i.e., once $\alpha_{n-k+1}\alpha_{n-k+2} \ldots \alpha_n$ are observed), the set $m(k)$ denotes the k-delayed state estimate at that fixed point due to the sequence of observations $\omega = \alpha_0\alpha_1 \ldots \alpha_{n-k}\alpha_{n-k+1} \ldots \alpha_n$. Similarly, $m'(l)$ denotes the l-delayed state estimates at that fixed point due to the sequence of observations $\omega' = \alpha_0\alpha_1 \ldots \alpha_{n-k}\alpha'_{n-k+1} \ldots \alpha'_{n'}$ for $n' - l = n - k$. In other

words, $m(k)$ represents the k-delayed state estimate, if after the passage of the system through the state at that fixed point $\alpha_{n-k+1}\alpha_{n-k+2}\ldots\alpha_n$ is observed, whereas $m'(l)$ denotes the l-delayed state estimate at this same point if $\alpha'_{n-k+1}\ldots\alpha'_{n'}$ is observed. To prove Theorem 5.3.1, we need to show that assuming $k \geq 2^{N^2}$, there exists an $l \leq 2^{N^2}-1$ such that the l-delayed state estimate at that same fixed time due to a shorter sequence of observations $\omega' = \alpha_0\alpha_1\ldots\alpha_{n-k}\alpha'_{n-k+1}\ldots\alpha'_{n'}$ with $n' = l+n-k$ is the same as the k-delayed state estimate of that fixed point due to the sequence of observations $\omega = \alpha_0\alpha_1\ldots\alpha_{n-k}\alpha_{n-k+1}\ldots\alpha_n$.

Denote the estimate of the system's (current) state at that point (i.e., the estimate after observing $\alpha_0\alpha_1\ldots\alpha_{n-k}$) by $Z \subseteq X$. The problem of k-delayed estimation of the state of the system at the fixed point in time after observing $\omega = \alpha_0\alpha_1\ldots\alpha_n$ can be viewed as an initial-state estimation problem where (due to the observations $\alpha_0\alpha_1\ldots\alpha_{n-k}$ that have been made before reaching that fixed point) the initial uncertainty about the "initial state" is the set Z. Hence, the set $m(k)$ after observing $\omega = \alpha_0\alpha_1\ldots\alpha_n$ is the same as the set of starting states of the state mapping that is associated with the state that is reached via $\alpha_{n-k+1}\alpha_{n-k+2}\ldots\alpha_n$ in the ISE whose initial state is associated with the state mapping $\odot_2(Z)$. Note that the string $\alpha_{n-k+1}\alpha_{n-k+2}\ldots\alpha_n$ has length $k > 2^{N^2}-1$. Since the ISE has at most 2^{N^2} states, strings of length at most $2^{N^2}-1$ can be chosen to visit any (reachable) ISE state. This implies that the state reached in this ISE via the string $\alpha_{n-k+1}\alpha_{n-k+2}\ldots\alpha_n$ of length k can also be reached via a string of length less than or equal to $2^{N^2}-1$, which we denote by $\alpha'_{n-k+1}\alpha'_{n-k+2}\ldots\alpha'_{n'}$ for some $n' \leq n+2^{N^2}-1-k$. Since the states reached in the ISE via either of these strings are identical, the k-delayed state estimate due to ω is the same as the $(k-(n-n'))$-delayed state estimate due to ω'. This completes the proof. □

The above result can be used to show that K'-step opacity is equivalent to K-step opacity for $K' > K \geq 2^{N^2}-1$. We prove this by showing that for $K \geq 2^{N^2}$, K-step opacity is equivalent to K^*-step opacity with $K^* = 2^{N^2}-1$.

Theorem 5.3.2. *For a non-deterministic finite automaton $G = (X, \Sigma, \delta, X_0)$, K-step opacity is equivalent to K^*-step opacity for $K > K^* = 2^{N^2}-1$ where $N = |X|$.* ■

Proof. (K-step opacity $\Rightarrow$ K^*-step opacity) Recall that DES is K-step opaque if and only if (5.16) in Theorem 5.2.2 holds. Consider the $(K+1)$-dimensional

state mapping m and (K^*+1)-dimensional state mapping m' associated with the states reached in $G_{K,obs}$ and $G_{K^*,obs}$ respectively via ω. Observe that $m(k) = m'(k), 0 \leq k \leq 2^{N^2} - 1$; since both $m(k)$ and $m'(k)$ denote the k-delayed state estimate due to observation ω, they are identical sets of states. Therefore, (5.16) implies that $\forall m \in X_{K^*,obs}, \forall k \in \{0, \ldots, K^*\} : m(k) \nsubseteq S$ or $m(k) = \emptyset$, which implies that K^*-step opacity holds.

(K^*-step opacity $\Rightarrow$ K-step opacity) We need to show (5.16). From Theorem 5.3.1 we have: for any $(K+1)$-dimensional state mapping m associated with states of $G_{K,obs}$ reached via a string ω with $|\omega| \geq 2^{N^2} - 1$ and $2^{N^2} \leq k \leq K$, there exists a (K^*+1)-dimensional state mapping m' associated with states of $G_{K^*,obs}$ and some l satisfying $0 \leq l \leq 2^{N^2} - 1$ such that $m(k) = m'(l)$. Now if DES G is K^*-step opaque, then all sets of intermediate states $m'(l)$ of all (K^*+1)-dimensional state mappings m' associated with states in $G_{K^*,obs}$ contain states outside the set of secret states or are empty; following the previous discussion, for $2^{N^2} \leq k \leq K$, all sets of intermediate states $m'(k)$ of all (K^*+1)-dimensional state mappings m' associated with states of $G_{K,obs}$ contain states outside the set of secret states. This implies (5.16) for $2^{N^2} \leq k \leq K$. Moreover, the discussion in part (i) implies (5.16) for $0 \leq k \leq 2^{N^2} - 1$. Therefore, (5.16) holds if m is reached in $G_{K,obs}$ via a string ω with $|\omega| \geq 2^{N^2} - 1$. If m is reached via a shorter string t with $|t| < 2^{N^2} - 1$, then the discussion in part (i) still implies (5.16) for $0 \leq k < 2^{N^2} - 1$; moreover, for $2^{N^2} - 1 \leq k \leq K$, we have $m(k) \equiv X_0$ (since we have yet to make enough observations) which trivially satisfies (5.16). □

5.3.2 Verifying Infinite-Step Opacity Using a Bank of Initial-State Estimators

The verification method for infinite-step opacity introduced in the previous section requires the construction of a $(2^{N^2} - 1)$-delayed state estimator which has state complexity $O(|\Sigma_{obs}|^{2^{N^2}-1} \times (2^N)^2)$ (by Section 4.6.3) or, equivalently for $|\Sigma_{obs}| \geq 2$, $O(|\Sigma_{obs}|^{2^{N^2}-1})$. In this section, we introduce a different method that uses both the ISE and the current-state estimator to verify initial-state opacity with complexity $O(2^{N^2})$.

In order to verify that a system is infinite-step opaque, we need to verify that at any point during the observation process, knowing the sequence of

observations before reaching that point, *in addition* to a future observation sequence (that is possible from that point onward), does not (and will not) allow us to determine whether the set of possible states at that point is a subset of the set of secret states. We perform this verification using a two-phase approach: (i) finding all possible estimates of the system's current state along any possible sequence of observations, and (ii) for each point in this trajectory (set of possible system states), calculating the information that can be gained about the state at that point by observation sequences that are possible from that point onward. The first phase can be achieved via a standard current-state estimator [4] (see Example 4.6.1). The second phase requires the construction of an ISE-like state estimator for each possible uncertainty about the current-state estimate (which is now used as the initial state estimate for the ISE-like state estimator). In other words, for each set of state estimates $Z \subseteq X$ provided in the first phase, we construct an ISE whose initial state is associated with the state mapping $\odot_2(Z)$. Clearly, if any of these ISEs contains a state with associated (non-empty) state mapping m such that its set of starting states contains elements only in S (i.e., if $m(1) \subseteq S$), then DES G is not infinite-step opaque. The following theorem formalizes the above discussion and proves that the two-phase approach is correct.

Theorem 5.3.3. *Consider a non-deterministic finite automaton* $G = (X, \Sigma, \delta, X_0)$*, a natural projection map* P *with respect to the set of observable events* Σ_{obs} *(*$\Sigma_{obs} \subseteq \Sigma$*), and a set of secret states* $S \subseteq X$*. For each set of current-state estimates* Z_n *associated with a state of its current-state estimator* $G_{0,obs}$*, construct the initial-state estimator* $G^{(n)}_{\infty,obs} = (X^{(n)}_{\infty,obs}, \Sigma_{obs}, \delta^{(n)}_{\infty,obs}, X^{(n)}_{\infty,0})$ *by setting its initial state* $X^{(n)}_{\infty,0}$ *to be* $\odot_2(Z_n)$*. Then, automaton* G *is* (S, P, ∞)*-opaque if and only if for all* n*, and for all* $m \in X^{(n)}_{\infty,obs}$*,*

$$m(1) \not\subseteq S \text{ or } m(1) = \emptyset. \tag{5.19}$$

■

Proof. If for some n and $m \in X^{(n)}_{\infty,obs}$, we have $m(1) = \emptyset$, then m is reachable via an infeasible sequence of observations which is not important in the definition of infinite-step opacity.

(If) We prove this by contradiction. Assume that (5.19) holds but DES G

is not (S, P, ∞)-opaque. Hence, there exists a string t that passes through a secret state j such that every other string s with $P(s) = P(t)$ also passes through a secret state j' when string t passes through the secret state j. More specifically, suppose that string t (string s) originates from state $i \in X_0$ (state $i' \in X_0$) and that string t (string s) can be written as $t = t't''$ ($s = s's''$) such that $P(t') = P(s')$, state i (state i') reaches state j (state j') via string t' (string s'), and state j (state j') reaches some state k (state k') via string t'' (string s''). To keep notation simple (and without loss of generality), we assume that there is only one string s, $s \neq t$ with $P(s) = P(t)$; the properties of the current-state estimator [34] imply that $Z_n \equiv \{j, j'\}$ can be associated with some state of $G_{0,obs}$. The corresponding $G^{(n)}_{\infty,obs}$ has initial state $X^{(n)}_{\infty,0} = \{(j, j), (j', j')\}$. Consider the state m that is reached in $G^{(n)}_{\infty,obs}$ from $X^{(n)}_{\infty,0}$ via $P(t'')$ (which equals $P(s'')$ because $s = s's''$, $t = t't''$, $P(s) = P(t)$ and $P(s') = P(t')$): since state j (state j') reaches state k (state k') via string t'' (string s''), the sequence of observations $P(t'')(=P(s''))$ could have originated from both states j and j' (and only from these two since string s was assumed without loss of generality to be unique). By Corollary 4.2.1, we know that the starting and ending states in the state mapping associated with the ISE state reached via string ω are, respectively, the set of states from which the observation ω could have originated and the set of states that are reached from such initial states. This implies that $m = \{(j, k), (j', k')\}$ which in turn implies that $m(1) = \{j, j'\}$ and $m(1) \subseteq S$ since we assumed $\{j, j'\} \in S$. This is a contradiction to our initial assumption that (5.19) holds and completes the (If) part of the proof.

(Only if) Assume that DES G is (S, P, ∞)-opaque. We need to prove that for all n and $m \in X^{(n)}_{\infty,obs}$, $(j, k) \in m$ and $j \in S$ imply that there exists $(j', k') \in m$ such that $j' \in X - S$. We prove this by contradiction. Assume that there exist n and $m \in X^{(n)}_{\infty,obs}$ such that for all $(j, k) \in m$ we have $j \in S$, and also that m is reached via ω in $G^{(n)}_{\infty,obs}$. By properties of the ISE states [5], this implies that for any state j in the initial state $X^{(n)}_{\infty,0}$ of $G^{(n)}_{\infty,obs}$ and for all $r \in \Sigma^*$ such that $P(r) = \omega$ and $\delta(j, r)$ is defined, we have $j \in S$. Without loss of generality, assume that there are only two such initial states in $X^{(n)}_{\infty,0}$ and denote them by j and j' (if there is only one such initial state, define $j = j'$); also, without loss of generality, we assume that from each of the states j and j', only one string can originate with projection ω; denote the string that originates from j (j') by t'' (s'') (note that it is

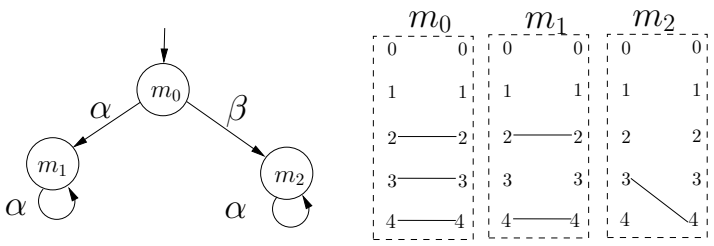

Figure 5.5: ISE $G^{(4)}_{\infty,obs}$ corresponding to states $\{2,3,4\}$.

possible that $t'' = s''$). As mentioned before, the initial state $X^{(n)}_{\infty,0}$ of $G^{(n)}_{\infty,obs}$ is constructed using an estimate of the current state Z that is a reachable state in $G_{0,obs}$, e.g., via a string Ω. By construction of $G_{0,obs}$, we know that there exist $i, i' \in X_0$ and $t', s' \in \Sigma^*$ such that $j \in \delta(i, t')$, $j' \in \delta(i', s')$, and $P(s') = P(t') = \Omega$. Define $t = t't''$ and $s = s's''$ and assume that we observe $\Omega\omega$. For this sequence of observations, there is a string t such that every other string s with $P(s) = P(t)$, passes through the set of secret states when t does. This violates infinite-step opacity which is a contradiction and hence the proof is complete. □

Remark 5.3.1. *In practice, since the set of initial state estimates can only decrease with additional observations [5], we only need to construct $G^{(n)}_{\infty,obs}$ for Z_n's which have at least one secret state.* ■

Example 5.3.1. *In this example, we show that DES G in Figure 4.1-a is not $(\{3\}, P, \infty)$-opaque. To verify infinite-step opacity we need to first construct the current-state estimator $G_{0,obs}$ as in Figure 4.2. This state estimator has five states $Z_1 = \{4\}$, $Z_2 = \{1,4\}$, $Z_3 = \{2,4\}$, $Z_4 = \{2,3,4\}$, $Z_5 = \{0,1,2,3,4\}$; hence, we need to construct five ISEs. Since only state mappings $Z_5 = \{(0,0),(1,1),(2,2),(3,3),(4,4)\}$ and $Z_4 = \{(2,2),(3,3),(4,4)\}$ contain the secret state 3, by Remark 5.3.1, we only need to construct two ISEs: (i) ISE $G^{(5)}_{\infty,obs}$ with initial state mapping corresponding to Z_5 is indeed the initial-state estimator in Figure 4.1-b which we constructed previously in Example 4.2.1. It can be easily verified that the set of starting states of all (non-empty) state mappings associated with this ISE has states outside the set of secret states. (ii) The ISE $G^{(4)}_{\infty,obs}$ with initial state corresponding to Z_4 is depicted in Figure 5.5 (again ignoring the empty state mapping reached via sequences of observations that cannot be generated by G). State $m_2 = \{(3,4)\}$ in $G^{(4)}_{\infty,obs}$ violates $(\{3\}, P, \infty)$-opacity since its set of starting*

states only contains state 3 *which is a secret state. State* m_2 *is reachable in* $G^{(4)}_{\infty,obs}$ *via* β *from* m_0*. Moreover,* m_0 *in this ISE corresponds to the state in* $G_{0,obs}$ *(in Figure 4.2) that is reached via observation* α*. Putting these two pieces of information together, we can conclude that observing* $\alpha\beta$ *reveals that the system has gone through state* 3*, which is a secret state.* ■

The verification of infinite-step opacity using Theorem 5.3.3 requires that for each state of the current-state estimator, an ISE-like state estimator be constructed. Since there are at most 2^N states for the current-state estimator, this implies that the complexity of this method is $O(2^N \times 2^{N^2})$ or equivalently $O(2^{N^2})$. This exponential complexity[1] is not desirable for implementation purposes; as we show in Chapter 6, however, verifying infinite-step opacity is PSPACE-hard and hence it is unlikely that any algorithm can verify this property in polynomial-time [37].

Remark 5.3.2. *In Section 3.5.3, we extend the definition trajectory-based* K*-step opacity to trajectory-based infinite-step opacity. It is not hard to argue that if we define the language* $E \subseteq L(G)$ *to be the set of strings in* G *that visit at least one secret state, then* $L(G) - E$ *is the set of strings in* G *that only visit non-secret states. Thus,* G *is trajectory-based infinite-step opaque if and only if* $P(E) \subseteq P(L(G) - E)$*.*

If we construct a (possibly non-deterministic) automaton $\tilde{G} = (X, \Sigma, \tilde{\delta}, \tilde{X}_0)$ *from* G *by removing all transitions from non-secret states in* G *that end in a secret state, and if we set* $\tilde{X}_0 = X_0 \cap (X - S)$*, we easily see that* $L(\tilde{G}) = L(G) - E$*. Therefore, DES* G *is trajectory-based infinite-step opaque if and only if* $P(L(\tilde{G})) = P(L(G))$*.* ■

[1] A more careful argument (that takes into account the fact that the state mapping in the ISE construction can only include at most $|X_0|$ states in the set of starting states) can be used to argue that the complexity of this method is $O(2^{|X_0|N})$.

CHAPTER 6

COMPUTATIONAL COMPLEXITY OF VERIFYING OPACITY

In this chapter we consider the computational complexity of verifying various notions of opacity. We start with initial-state opacity, which as shown in Chapter 5 can be verified (for a fixed set of secret states S) with space and time complexity $O(4^N)$, where $N = |X|$ is the number of states of the given automaton. In fact, in this chapter, we establish that the verification of initial-state opacity for $|\Sigma_{obs}| > 1$ is a PSPACE-complete problem [37].

Verifying K-step opacity using KDE was shown in Chapter 5 to have space and time complexity $O((|\Sigma_{obs}| +1)^K \times 2^N)$; in this chapter, we establish that deciding whether the non-deterministic finite automaton G is K-step opaque is NP-hard for $|\Sigma_{obs}| > 1$ [37].

Finally, verifying infinite-step opacity using the current-state estimator and a bank of initial-state estimators has space and time complexity $O(2^{N^2})$. In this chapter, we show that verifying infinite-step opacity is PSPACE-hard for $|\Sigma_{obs}| > 1$ [37].

6.1 Review of Complexity Theory

We now briefly review some necessary results and definitions from complexity theory (see [37] for further details). A *problem* is a parameterized question to be answered. An *instance* of a problem is obtained by specifying particular values for all problem parameters. A *decision* problem is one whose answer, depending on the instance, is either "yes" or "no". An algorithm *solves* a problem if it produces a correct answer when applied to any instance of the problem. In the sequel, we only consider decision problems.

The class of problems that can be solved by an algorithm that is polynomial in the size (encoding) of the problem is called P. NP stands for the class of decision problems that are "verifiable" by a polynomial-time algorithm. A

decision problem is NP-hard if any other decision problem in NP can be reduced to this problem using a polynomial-time algorithm. If a decision problem is NP-hard and is in NP, then it is called NP-complete. It is widely conjectured that P is a proper subset of NP. If this is true, then there is no polynomial-time algorithm for any NP-complete or NP-hard problem.

The class of decision problems that can be solved using space that is polynomial in the size (encoding) of the problem is called PSPACE. A PSPACE-hard problem is a decision problem such that any other decision problem in PSPACE can be reduced to this problem using a polynomial-time algorithm. If a PSPACE-hard problem is in PSPACE, then it is called PSPACE-complete. Although all problems solvable in polynomial-time can be solved in polynomial-space, it is widely believed that there exist problems solvable in polynomial-space that cannot be solved in polynomial-time. Also, it is known that NP$\subseteq$PSPACE and the inclusion is widely believed to be proper. A PSPACE-complete problem can be solved in polynomial-time if P=NP and NP=PSPACE. In other words, showing that a problem is PSPACE-complete is strong evidence that the problem is computationally expensive.

One of the first problems proved to be NP-complete is the *non-tautology (NT)* problem introduced in [41]. In order to describe this problem formally, we introduce some notation. Let $U = \{u_1, u_2, \ldots, u_M\}$ be a set of *Boolean* variables (i.e., variables that take value in $\{0, 1\}$). A truth assignment for U is a function $T : U \rightarrow \{0, 1\}$. If $T(u) = 1$ we say that u is true under T. If u is a variable in U, u and $\neg u$ (negated u) are *literals* over U. A *Boolean expression* is built from a set of Boolean variables U, conjunction (logical AND) $\wedge$, disjunction (logical OR) $\vee$, logical negation $\neg$, and parentheses for grouping. A *phrase* p is a conjunction of literals. A Boolean expression E is considered to be in (L, P)-disjunctive normal form ((L, P)-DNF) if it is a disjunction of P phrases $\{p_1, \ldots, p_P\}$ each with at most L literals. E can also be represented as $\{p_1, \ldots, p_P\}$. We assume that for each variable u, at most one of the literals u or $\neg u$ appears in every phrase.

Definition 6.1.1 (Non-Tautology (NT) Problem)**.** *Given a set of variables $U = \{u_1, \ldots, u_M\}$ and a Boolean expression $E = \{p_1, \ldots, p_P\}$ in (L, P)-DNF, for $L, P > 0$, does there exist a truth assignment T for U that makes E false?* ■

The authors of [41] prove that the NT problem is NP-complete for $M > 1$

and $P > 2$.

6.2 Verification of Initial-State Opacity is PSPACE-Complete for $|\Sigma_{obs}| > 1$

In this section we study the complexity class of the verification of the initial-state opacity (INI) problem. We establish that the INI problem, for $|\Sigma_{obs}| > 1$, is PSPACE-complete. We achieve this using a reduction from the *language containment for non-deterministic finite automata (LC)* problem, which is known to be PSPACE-complete[1] for $|\Sigma| > 1$ [42, 43]. We define the INI and the LC problems formally below.

Definition 6.2.1 (Initial-State Opacity Verification (INI) problem)**.** *Given a non-deterministic finite automaton $G = (X, \Sigma, \delta, X_0)$, a natural projection map P with respect to the set of observable events Σ_{obs} ($\Sigma_{obs} \subseteq \Sigma$), and a set of secret states $S \subseteq X$, is automaton G (S, P, ∞) initial-state opaque?* ■

Definition 6.2.2 (Language Containment (LC) problem)**.** *Given two non-deterministic automata $G_1 = (X_1, \Sigma, \delta_1, X_{1,0})$ and $G_2 = (X_2, \Sigma, \delta_2, X_{2,0})$ with sets of initial states $X_{1,0}$ and $X_{2,0}$, is $L(G_1) \subseteq L(G_2)$?* ■

Theorem 6.2.1. *The INI problem is PSPACE-complete for $|\Sigma_{obs}| > 1$.* ■

Proof. We first prove that the INI problem is in PSPACE for $|\Sigma_{obs}| > 1$ by introducing a polynomial-time algorithm which reduces every instance of the INI problem with $|\Sigma_{obs}| > 1$ to an instance of the LC problem with $|\Sigma| > 1$; since the LC problem is in PSPACE for $|\Sigma| > 1$, this proves that the INI problem is also in PSPACE for $|\Sigma_{obs}| > 1$. Given a non-deterministic automaton $G = (X, \Sigma, \delta, X_0)$ and a natural projection map P with respect to the set of observable events Σ_{obs} ($\Sigma_{obs} \subseteq \Sigma$), the *unobservable reach* $UR(x, \alpha)$ of state x of G under event $\alpha \in \Sigma_{obs}$ is the set of states reachable from x with a sequence of events s in which the only observable event is event α that appears exactly once (no other observable event appears, i.e., $P(s) = \alpha$). Then, the non-deterministic automaton $G_o = (X, \Sigma_{obs}, \delta_o, X_0)$ is constructed from G by removing all unobservable events and by introducing for each event $\alpha \in \Sigma_{obs}$ transitions associated with label α from each state x to each state in the

[1] The problem can be solved in polynomial time if $|\Sigma| = 1$.

unobservable reach $UR(x, \alpha)$. Note that computing the unobservable reach takes $O(N^3)$ time [19], where N denotes the number of states of DES G. Next, we construct two non-deterministic automata $G_1 = (X, \Sigma_{obs}, \delta_o, X_{1,0})$ and $G_2 = (X, \Sigma_{obs}, \delta_o, X_{2,0})$ which have the same set of states, event set, and state transition function as G_o, but differ in their sets of initial states, which are taken respectively to be $X_{1,0} = X_0 \cap S$ and $X_{2,0} = X_0 - S$. Since in constructing these two automata, the structure of G_o is preserved and only the set of initial-states is modified through set intersection, this construction requires $O(N)$ time.

Next, we show that $L(G_1) \subseteq L(G_2)$ if and only if G is (S, P, ∞) initial-state opaque. Define

$$L(G, X_0 \cap S) := \bigcup_{i \in X_0 \cap S} L(G, i),$$

and

$$L(G, X_0 - S) := \bigcup_{i \in X_0 - S} L(G, i).$$

By (3.1), DES G is (S, P, ∞) initial-state opaque if and only if

$$P(L(G, X_0 \cap S)) \subseteq P(L(G, X_0 - S)). \tag{6.1}$$

By construction of G_o, we have that $L(G_o) = P(L(G))$ and, since $X_{1,0} = X_0 \cap S$ and $X_{2,0} = X_0 - S$, (6.1) is equivalent to $L(G_1) \subseteq L(G_2)$. This proves that the INI problem is in PSPACE for $|\Sigma_{obs}| > 1$.

In order to show that the INI problem is PSPACE-hard for $|\Sigma_{obs}| > 1$, we reduce the LC problem with $|\Sigma| > 1$ to an instance of the INI problem with $|\Sigma_{obs}| > 1$ via a polynomial-time algorithm. Given two non-deterministic automata $G_1 = (X_1, \Sigma, \delta_1, X_{1,0})$ and $G_2 = (X_2, \Sigma, \delta_2, X_{2,0})$, define the non-deterministic automaton $G = (X, \Sigma, \delta, X_0)$ with the set of states $X = X_1 \cup X_2$, set of initial states $X_0 = X_{1,0} \cup X_{2,0}$, and state transition function $\delta : X \times \Sigma \to 2^X$ given by[2]

$$\delta(x, \alpha) = \begin{cases} \delta_1(x, \alpha), & \text{if } x \in X_1 \\ \delta_2(x, \alpha), & \text{if } x \in X_2. \end{cases}$$

[2]Without loss of generality, we assume that $X_1 \cap X_2 = \emptyset$ (one can always rename the states to ensure this).

Note that the time- and state-complexity of constructing G is $O(m^2 + n^2)$ where $m = |X_1|$ and $n = |X_2|$, i.e., polynomial in the number of states of G_1 and G_2. We show that $L(G_1) \subseteq L(G_2)$ if and only if G is (S, P, ∞) initial-state opaque where $S \equiv X_{1,0}$ and projection mapping P is with respect to the set of observable events $\Sigma_{obs} = \Sigma$. Using (6.1) and assuming that $\Sigma_{obs} = \Sigma$, we have

$$
\begin{aligned}
&G \text{ is } (S, P, \infty) \text{ initial-state opaque } \Leftrightarrow \\
&L(G, X_0 \cap S) \subseteq L(G, X_0 - S) \Leftrightarrow \\
&L(G, X_{1,0}) \subseteq L(G, X_0 - X_{1,0}) \quad (S = X_{1,0} \subseteq X_0) \Leftrightarrow \\
&L(G, X_{1,0}) \subseteq L(G, X_{2,0}) \quad (X_0 = X_{1,0} \dot{\cup} X_{2,0}) \Leftrightarrow \\
&L(G_1) \subseteq L(G_2).
\end{aligned}
$$

Since $\Sigma = \Sigma_{obs}$, this proves that each instance of the LC problem with $|\Sigma| > 1$ can be reduced to an instance of the INI problem with $|\Sigma_{obs}| > 1$ via a polynomial-time algorithm; therefore, the INI problem is PSPACE-hard for $|\Sigma_{obs}| > 1$. Since the INI problem is PSPACE-hard and is in PSPACE, we conclude that the INI problem is PSPACE-complete. □

Remark 6.2.1. *The authors of [32] study the existence of a unique input/output (UIO) sequence for a state i of a given deterministic finite state machine which can be modeled as a Mealy machine (see Section 3.5.5). Recall that a Mealy machine is a finite state machine that generates an output based on its current state and input [19]. A UIO sequence for state i is an input sequence x such that the output sequence generated by the machine in response to x from initial state i is different from the response to x from any other initial state. Assuming that the uncertainty about the initial state of the system is X (i.e., $X_0 = X$) and that the state transition function is deterministic (i.e., from each state and with each input, the next state is unique), the authors of [32] show that determining whether a given state i has a UIO sequence is a PSPACE-complete problem.*

One can always transform the Mealy machine modeling of [32] (where each state transition is triggered by an input/output pair) to the automaton modeling of this thesis (where each state transition is triggered by an event) in polynomial time [19]; therefore, it is possible to extend the definition of UIO in [32] to our framework as follows: a UIO sequence for state i is a string

s such that the sequence of observations $P(s)$ generated by DES G from any state other than i is different than that from i. Using this definition, and setting $S = \{i\}$ and $X_0 = X$, it is not hard to see that state i has no UIO sequence if and only if DES G is initial-state opaque. This demonstrates that with $X_0 = X$, the verification of initial-state opacity is a PSPACE-hard problem even if the given DES is deterministic (in the sense that the cardinality of the state transition function for each state/event pair is at most 1). Note that in this section, we also showed that the verification of initial-state opacity is a PSPACE-complete problem but under different assumptions (we made no assumption about the set of initial states X_0 but we assumed that the given DES is non-deterministic). ■

6.3 Verification of K-Step Opacity is NP-Hard for $|\Sigma_{obs}| > 1$

In order to establish the complexity class of the verification of K-step opacity problem, we need the language-containment for non-deterministic automata with finite languages (LC-FIN) and therefore describe it formally. First, we need to define the language *marked* by an automaton.

Definition 6.3.1 (Marked Language)**.** *Given a non-deterministic finite automaton $G = (X, \Sigma, \delta, X_0, X_m)$ with set of marked states $X_m \subseteq X$, define the marked language $L_m(G)$ of G as*

$$L_m(G) = \{s \in L(G) | \exists i \in X_0 \{\delta(i, s) \cap X_m \neq \emptyset\}\}.$$

■

Note that in general, $L_m(G)$ is not prefix-closed. Also, we have $L_m(G) \subseteq L(G)$; if $X_m = X$, then $L_m(G) = L(G)$.

Definition 6.3.2 (LC-FIN Problem)**.** *Given two non-deterministic automata $G_1 = (X_1, \Sigma, \delta_1, X_{1,0}, X_{m,1})$ and $G_2 = (X_2, \Sigma, \delta_2, X_{2,0}, X_{m,2})$ with finite languages $L_m(G_1)$ and $L_m(G_2)$, is $L_m(G_1) \subseteq L_m(G_2)$?* ■

The authors of [44] prove that the LC-FIN problem is NP-complete for $|\Sigma| > 1$.

Next, we define the LC-FIN-CLOSED problem as a special case of the LC-FIN problem when $X_m = X$. When $X_m = X$, X_m is dropped from the 5-tuple description of G. In order to characterize the complexity class of the K-step opacity verification (KSTEP) problem, we first prove that the LC-FIN-CLOSED problem for $|\Sigma| > 1$ is NP-complete, and then show that the LC-FIN-CLOSED problem for $|\Sigma| > 1$ can be reduced in polynomial time to an instance of the KSTEP problem for $|\Sigma_{obs}| > 1$. This proves that the KSTEP problem is NP-hard for $|\Sigma_{obs}| > 1$. Below, we describe the KSTEP and the LC-FIN-CLOSED problems formally.

Definition 6.3.3 (K-Step Opacity Verification (KSTEP) problem)**.** *Given a non-deterministic finite automaton $G = (X, \Sigma, \delta, X_0)$, a natural projection map P with respect to the set of observable events Σ_{obs} ($\Sigma_{obs} \subseteq \Sigma$), and a set of secret states $S \subseteq X$, is automaton G (S, P, K)-opaque?* ■

Definition 6.3.4 (LC-FIN-CLOSED Problem)**.** *Given two non-deterministic automata $G_1 = (X_1, \Sigma, \delta_1, X_{1,0})$ and $G_2 = (X_2, \Sigma, \delta_2, X_{2,0})$ with finite languages $L(G_1)$ and $L(G_2)$, is $L(G_1) \subseteq L(G_2)$?* ■

Theorem 6.3.1. *The LC-FIN-CLOSED problem is NP-complete for $|\Sigma| > 1$.* ■

Proof. In order to show that the LC-FIN-CLOSED problem is NP-hard, we reduce the NT problem (defined in Section 6.1) with $M > 1$ and $P > 2$ (where M is the number of Boolean variables and P is the number of phrases) to an instance of the LC-FIN-CLOSED problem with $|\Sigma| > 1$ via a polynomial-time algorithm.

Consider the Boolean expression $E = \{p_1, \ldots, p_P\}$ in (L, P)-DNF (i.e., an expression which is the disjunction of P phrases, each with maximum number of L literals) and assume that E is defined over the set of variables $U = \{u_1, \ldots, u_M\}$. We construct the non-deterministic[3] finite automaton $G = (X, \Sigma, \delta, X_0)$ with state set $X = (U \cup \{u_f\}) \times \{1, \ldots, P\}$, event set $\Sigma = \{0, 1\}$, set of initial states $X_0 = \{u_1\} \times \{1, \ldots, P\}$, and state transition function $\delta : X \times \Sigma \to X$ defined for $(u_i, j) \in X$, $1 \leq i \leq M - 1$, $1 \leq j \leq P$ as follows:

[3]The non-determinism is due to the fact that the initial state is a set.

i) for $0 \in \Sigma$ we have

$$\delta((u_i, j), 0) = \begin{cases} (u_{i+1}, j), & \text{if } (\neg u_i \in p_j) \vee (u_i \notin p_j \wedge \neg u_i \notin p_j), \\ \text{undefined}, & \text{otherwise.} \end{cases}$$

ii) for $1 \in \Sigma$ we have

$$\delta((u_i, j), 1) = \begin{cases} (u_{i+1}, j), & \text{if } (u_i \in p_j) \vee (u_i \notin p_j \wedge \neg u_i \notin p_j), \\ \text{undefined}, & \text{otherwise.} \end{cases}$$

The state transition function is defined for $(u_M, j) \in X$, $1 \leq j \leq P$ as follows:
i) for $0 \in \Sigma$ we have

$$\delta((u_M, j), 0) = \begin{cases} (u_f, j), & \text{if } (\neg u_M \in p_j) \vee (u_M \notin p_j \wedge \neg u_M \notin p_j), \\ \text{undefined}, & \text{otherwise.} \end{cases}$$

ii) and for $1 \in \Sigma$ we have

$$\delta((u_M, j), 1) = \begin{cases} (u_f, j), & \text{if } (u_M \in p_j) \vee (u_M \notin p_j \wedge \neg u_M \notin p_j), \\ \text{undefined}, & \text{otherwise.} \end{cases}$$

Since there is no transition defined out of the states (u_f, j) $(1 \leq j \leq P)$ and there are no cycles in the automaton (one can never return to state (u_i, j) from a state $(u_{i'}, j)$ with $i' > i$), we can easily see that $L(G)$ is a finite language of length M. Each string s in $L(G)$ has length M and visits the sequence of states $(u_1, j) \rightarrow (u_2, j) \rightarrow \ldots \rightarrow (u_M, j) \rightarrow (u_f, j)$. We can define a truth assignment as follows: u_i is set to true if the i^{th} event in the string s is 1. Moreover, by construction of G, the set of strings in G that can originate from the initial state (u_1, j) and reach the state in (u_f, j) $(1 \leq j \leq P)$ defines the set of all truth assignments which makes phrase p_j true. Expression E is true if and only if at least one of its phrases is true. Therefore, $L(G)$ captures all truth assignments that make E true. For E to be a tautology (i.e., for E to be true under all truth assignments), it is necessary and sufficient that $L(G)$ contains all possible strings of length M and, since $L(G)$ is assumed to be prefix-closed, it is necessary and sufficient that $L(G)$ contains all possible

strings of length at most M; in other words, E is a tautology if and only if

$$L(G) = \{\epsilon\} \bigcup_{k=1}^{M} \Sigma^k,$$

where M is the number of Boolean variables. Therefore, checking whether E is a tautology with $M = |U| > 1$ reduces to an instance of LC-FIN-CLOSED with $\Sigma = \{0, 1\}$ ($|\Sigma| > 1$). Also, note that this reduction can be done in time polynomial in the size of E (i.e., in the total number of literals summed over all phrases). This proves that the LC-FIN-CLOSED for $|\Sigma| > 1$ is NP-hard. Note that the LC-FIN-CLOSED problem is a special case of the LC-FIN problem, and since the LC-FIN problem is in NP, the LC-FIN-CLOSED problem is also in NP. This completes the proof. □

Remark 6.3.1. *The reduction technique used in the proof of Theorem 6.3.1 is similar to the one used in [44] in proving NP-completeness of the LC-FIN problem. However, in the construction introduced in [44], the languages of the obtained automata are not necessarily prefix-closed; more specifically, the state transition function of the constructed automaton in [44] is total and only one state in the automaton is marked. In the proof of Theorem 6.3.1, we extend this construction to guarantee that the constructed languages are all prefix-closed by marking all states and allowing the state transition function to be partial.* ■

In the following example, we illustrate the reduction introduced in the proof of Theorem 6.3.1.

Example 6.3.1. *Consider the Boolean expression* $E = \{(u_1 \wedge u_2), (\neg u_2 \wedge u_3 \wedge u_4)\}$ *which is in (3,2)-DNF (i.e., the number of phrases is* $P = 2$*, and the maximum number of literals in each phrase is* $L = 3$*) and is defined over the set of Boolean variables* $U = \{u_1, u_2, u_3, u_4\}$*. Following the algorithm in the proof of Theorem 6.3.1, we construct the non-deterministic automaton* $G = (X, \Sigma, \delta, X_0)$ *with* $X = (U \cup \{u_f\}) \times \{1, 2\}$*,* $\Sigma = \{0, 1\}$*, and* $X_0 = \{u_1\} \times \{1, 2\}$ *as depicted in Figure 6.1. Since variable* u_1 *is present in the phrase* p_1*, we have* $\delta((u_1, 1), 1) = (u_2, 1)$ *and* $\delta((u_1, 1), 0)$ *is undefined. Also, since variable* u_2 *is not present in the phrase* p_2*, we have* $\delta((u_1, 2), 0) = \delta((u_1, 2), 1) = (u_2, 2)$*. Following this approach, one can complete the construction of* G*. As can be easily verified, the string* 0100 *is not in* $L(G)$ *which implies that*

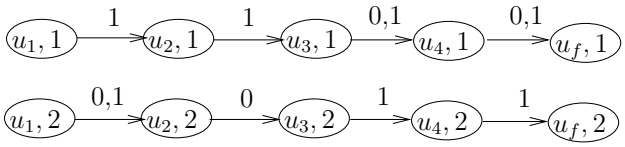

Figure 6.1: DES G constructed from the (3,2)-DNF Boolean expression $E = \{(u_1 \wedge u_2), (\neg u_2 \wedge u_3 \wedge u_4)\}$.

$L(G) \neq \{\epsilon\} \cup \Sigma \cup \Sigma^2 \cup \Sigma^3 \cup \Sigma^4$; hence E is not a tautology. For example, the truth assignment $u_1 = 0$, $u_2 = 1$, $u_3 = 0$, and $u_4 = 0$ makes E false. ■

Next, we reduce the LC problem to an instance of K-step opacity.

Theorem 6.3.2. *Given two non-deterministic automata*[4] *$G_i = (X_i, \Sigma, \delta_i, X_{i,0})$, $i = 1, 2$, with finite languages of length K and $|\Sigma| > 1$, define the non-deterministic automaton $G = (X, \Sigma, \delta, X_0)$ with the state set $X = X_1 \cup X_2$, set of initial states $X_0 = X_{1,0} \cup X_{2,0}$, and state transition function $\delta : X \times \Sigma \to 2^X$ given by $\delta(x, \alpha) = \delta_1(x, \alpha)$ if $x \in X_1$, and $\delta(x, \alpha) = \delta_2(x, \alpha)$ if $x \in X_2$. Then $L(G_1) \subseteq L(G_2)$ if and only if G is (S, P, K)-opaque where $S \equiv X_{1,0}$ and projection map P is with respect to the set of observable events $\Sigma_{obs} = \Sigma$.* ■

Proof. Note that in an automaton with a finite language, there cannot exist any cycle. In other words, none of the strings that originate from state i can visit this state again in the future. Also, since G_1 and G_2 are both non-deterministic automata with finite languages of length K, the construction of G ensures that G is also a non-deterministic automaton with finite language of length K. This implies that the length of all strings in G that start from a state i in the set of initial states $X_0 = X_{1,0} \cup X_{2,0}$ is at most K and that these strings cannot visit state i again in the future; since $S \equiv X_{1,0}$, the set of strings that pass through S within the past K observations is $L(G_1)$ and, in fact, these strings only pass through S at the system startup. Also note that none of the strings in $L(G_2)$ ever passes through S.

According to Definition 3.2.1, DES G is (S, P, K)-opaque if for every string t in $L(G)$ that passes through S within the past K observations, there exists a string s in $L(G)$ with $P(s) = P(t)$ such that when string t passes through S, string s does not. Since $\Sigma_{obs} = \Sigma$, $P(s) = P(t)$ implies $s = t$. Also, since all of the strings in $L(G)$ have length at most K, (S, P, K)-opacity of G implies

[4]Without loss of generality, we assume that $X_1 \cap X_2 = \emptyset$. One can always rename the states to achieve this.

that the set of strings t in $L(G)$ that pass through S is a subset of the set of strings s in $L(G)$ that do not pass through S. From the previous discussion, we know that the set of strings in $L(G)$ that pass through $S = X_{1,0}$ is $L(G_1)$ and the set of strings that do not pass through S is $L(G_2)$ (i.e., strings that start from $X_{2,0}$). This implies that (S, P, K)-opacity of G is equivalent to $L(G_1) \subseteq L(G_2)$. Observing that $|\Sigma_{obs}| = |\Sigma| > 1$ completes the proof. □

The time complexity of constructing G is $N_1^2 + N_2^2$ with $N_1 = |X_1|$ and $N_2 = |X_2|$, which is polynomial in the number of states of G_1 and G_2. Therefore, Theorem 6.3.2 proves that deciding whether the non-deterministic finite automaton G is K-step opaque is NP-hard for $|\Sigma_{obs}| > 1$.

6.4 Verification of Infinite-Step Opacity is PSPACE-Hard for $|\Sigma_{obs}| > 1$

In order to characterize the complexity class of the infinite-step opacity verification (INF) problem, we show that each instance of the INI problem (studied in Section 6.2) can be reduced (via an algorithm which has complexity polynomial in the number of states of automaton G) to an instance of the INF problem. This proves that the INF problem is an NP-hard problem for $|\Sigma_{obs}| > 1$ since the INI problem is a PSPACE-complete problem for $|\Sigma_{obs}| > 1$.

If none of the secret states of system G is reachable after startup (i.e., if none of the strings in the system can pass through the set of secret states except at startup), then it is not hard to see that infinite-step opacity and initial-state opacity become equivalent. In the following lemma, we use this insight to reduce each instance of the INI problem to an instance of the INF problem. First, we define the INF problem.

Definition 6.4.1 (Infinite-Step Opacity Verification (INF) problem)**.** *Given a non-deterministic finite automaton $G = (X, \Sigma, \delta, X_0)$, a natural projection map P with respect to the set of observable events Σ_{obs} ($\Sigma_{obs} \subseteq \Sigma$), and a set of secret states $S \subseteq X$, is automaton G (S, P, ∞)-opaque?* ■

Definition 6.4.2. *Given a non-deterministic finite automaton $G = (X, \Sigma, \delta, X_0)$, define the non-deterministic finite automaton $\hat{G} = (\hat{X}, \Sigma, \hat{\delta}, X_0)$ with*

state set $\hat{X}$ *constructed from* X *by adding duplicates* x_0' *for each* $x_0 \in X_0$ *(we denote this by* $x_0 \stackrel{d}{\equiv} x_0'$*), and state transition function* $\hat{\delta} : \hat{X} \times \Sigma \to 2^{\hat{X}}$ *defined for* $\alpha \in \Sigma$ *and* $x \in X$ *as*

$$\hat{\delta}(x, \alpha) = \{y | y \in \delta(x, \alpha) - X_0\} \cup \{y' | y \in \delta(x, \alpha) \cap X_0, y \stackrel{d}{\equiv} y'\},$$

and for $\alpha \in \Sigma$ *and* $x' \in \hat{X} - X$ *as*

$$\hat{\delta}(x', \alpha) = \hat{\delta}(x, \alpha),$$

where $x' \stackrel{d}{\equiv} x$. ■

Lemma 6.4.1. *Given a non-deterministic finite automaton* $G = (X, \Sigma, \delta, X_0)$*, a natural projection map* P *is taken with respect to the set of observable events* Σ_{obs} *(*$\Sigma_{obs} \subseteq \Sigma$*), and a set of secret states* $S \subseteq X$*, construct the non-deterministic finite automaton* $\hat{G}$ *as in Definition 6.4.2. Then,*

$$(S, P, \infty) \textit{ initial-state opacity for } G \Leftrightarrow (S \cap X_0, P, \infty)\textit{-opacity for } \hat{G}.$$

■

Proof. The state transition function $\hat{\delta}$ of $\hat{G}$ is the same as the state transition function δ of G except for transitions $x \to y$ from a state $x \in X$ to a state $y \in X_0$. In $\hat{G}$, these transitions are replaced with $x \to y'$ where state y' is the duplicate of state y. The state set of $\hat{G}$ is extended to include such duplicates. Note that the transitions defined from (duplicate) state y' are the same as those of the (original) state y, which guarantees that none of the strings in $\hat{G}$ that originate from a state $y \in X_0$ will ever visit this state (or any other state in X_0) in the future. In other words, if a string $s \in L(G)$ starts from a state $x \in X_0$ in G and visits some state $y \in X_0$ in its path, then the same string in $\hat{G}$ will visit the duplicated state y' when state y is visited in G. Note that $\hat{G}$ contains the same strings as G, i.e., $L(\hat{G}) = L(G)$. If system $\hat{G}$ is infinite-step opaque with respect to $\hat{S}$ and P, this implies that the membership of system state to the set $\hat{S} = S \cap X_0$ cannot be determined at any time during the observation. However, system $\hat{G}$ can only visit states in X_0 (and hence in $S \cap X_0$) at startup. This implies that infinite-step opacity and initial-state opacity with respect to $\hat{S}$ and P are equivalent for system $\hat{G}$. Moreover, systems $\hat{G}$ and G have the same set of initial states, closed

behavior, and set of observable events. Therefore, initial-state opacity of $\hat{G}$ with respect to $\hat{S}$ and P is equivalent to initial-state opacity of G with respect to $\hat{S}$ and P. Also, for G, initial-state opacity with respect to $\hat{S}$ and P is equivalent to initial-state opacity with respect to S and P since in the definition of initial-state opacity, states in $S - X_0$ are not important (recall that $\hat{S} = S \cap X_0$). Hence, initial-state opacity of $\hat{G}$ with respect to $\hat{S}$ and P is equivalent to initial-state opacity of G with respect to S and P. This completes the proof. □

Note that the reduction technique introduced in Lemma 6.4.1 clearly has (state or time) complexity that is polynomial in the number of states of G.

Corollary 6.4.1. *The INF problem is PSPACE-hard for* $|\Sigma_{obs}| > 1$. ■

CHAPTER 7

ENFORCING OPACITY

In this chapter, using the state estimator constructions of Chapter 4, we consider the problem of designing a (minimally restrictive) supervisor which: (i) limits the system's behavior within some pre-specified legal behavior, and (ii) enforces (either initial-state or infinite-step) opacity requirements by disabling, at any given time, the least possible number of events.

For enforcing initial-state opacity, we establish that the set of solutions can be characterized as the intersection of controllable, normal, and *opaque* languages. Using this characterization, we then show that the solution to the problem of enforcing our requirements is a supervisor that enforces the supremal element of such languages. We argue that, under some mild assumptions, the supremal element exists and derive a formulation for it. Moreover, assuming that the given legal behavior is regular, we show that the supremal element is also regular. Finally, we propose a procedure that uses initial-state estimators to implement this supremal element, effectively integrating the verification and control problems.

For enforcing K-step opacity, we briefly discuss that we can follow a similar approach for the previous case (i.e., for the case of initial-state opacity), and use a K-delay state estimator (instead of an initial-state estimator) to construct the supervisor that enforces K-step opacity (instead of initial-state opacity).

For enforcing infinite-step opacity, we leverage the results on initial-state opacity and build a finite bank of supervisors that implement minimally restrictive supervisory strategies.

7.1 Review of Supervisory Control Theory

7.1.1 Language-Based Approach

In the Ramadge and Wonham framework introduced in [15], it is assumed that the event set Σ can be partitioned into the sets of controllable events (Σ_c) and uncontrollable events (Σ_{uc}) so that $\Sigma_c \cap \Sigma_{uc} = \emptyset$ and $\Sigma_c \cup \Sigma_{uc} = \Sigma$. Control is achieved by means of a *supervisor* which at any given time can disable one or more controllable events. Formally, given a non-deterministic finite automaton $G = (X, \Sigma, \delta, X_0)$ and a natural projection map P with respect to the set of observable events Σ_{obs} ($\Sigma_{obs} \subseteq \Sigma$), a *feasible* supervisor ν_o (subscript o denotes the partial observation) for G is a map $\nu_o : P(L(G)) \rightarrow \{\Sigma' \subseteq \Sigma | \Sigma_{uc} \subseteq \Sigma'\}$ which defines the set of events Σ' that remain enabled after observing a particular string from the system (note that Σ' necessarily includes all uncontrollable events). If we denote the closed-loop system by ν_o/G, the *minimally restrictive feasible supervisor problem* (MS) is defined as the design of a feasible supervisor ν_o such that: (i) $L(\nu_o/G) \subseteq E$ for a given (prefix-closed) language E that describes desirable behavior (control objective), and (ii) $L(\nu_o/G)$ is as least restrictive as possible (i.e., for any other feasible supervisor ν'_o such that $L(\nu'_o/G) \subseteq E$, we have $L(\nu'_o/G) \subseteq L(\nu_o/G)$). If we exclude requirement (ii) (that the supervisor is minimally restrictive), the following theorem from [19, 28] characterizes all solutions to the supervisory control problem for certain prefix-closed languages. In the sequel we assume that $\Sigma_c \subseteq \Sigma_{obs}$.

Theorem 7.1.1 ([28])**.** *Given a non-deterministic finite automaton $G = (X, \Sigma, \delta, X_0)$, a natural projection map P with respect to the set of observable events Σ_{obs} ($\Sigma_{obs} \subseteq \Sigma$), a set of controllable events Σ_c ($\Sigma_c \subseteq \Sigma_{obs}$), a set of uncontrollable events $\Sigma_{uc} = \Sigma - \Sigma_c$, and a prefix-closed language $K \subseteq L(G)$ with $K \neq \emptyset$, there exists a feasible supervisor ν_o for G such that $L(\nu_o/G) = K$ if and only if: (i) K is $(L(G), \Sigma_{uc})$–controllable [19, 28] (i.e., $\overline{K}\Sigma_{uc} \cap L(G) \subseteq \overline{K}$), and (ii) K is $(L(G), P)$–normal [19, 28] (i.e., $K = L(G) \cap P^{-1}(P(K))$).* ■

The following lemma from [45] characterizes a class of normal languages which is used in Chapter 7.

Lemma 7.1.1. *([45]) Given a non-deterministic finite automaton $G = (X, \Sigma,$*

δ, X_0), a natural projection map P with respect to the set of observable events Σ_{obs} ($\Sigma_{obs} \subseteq \Sigma$), and a set of controllable events Σ_c ($\Sigma_c \subseteq \Sigma$), a set of uncontrollable events $\Sigma_{uc} = \Sigma - \Sigma_c$, for any prefix-closed language $M \subseteq P(L(G))$, the language $K = P^{-1}(M) \cap L(G)$ is normal with respect to $(L(G), P)$. ■

For any $E \subseteq L(G)$, we define $\mathcal{N}(E)$ ($\mathcal{C}(E)$) to be the set of all prefix-closed sublanguages of E that are normal (controllable). The set $\mathcal{CN}(E) \equiv \mathcal{C}(E) \cap \mathcal{N}(E)$ is closed under union and, hence, there exists a unique supremal element $sup\mathcal{CN}(E)$ under the partial order of set inclusion for this set [19, 28]. We denote $sup\mathcal{CN}(E)$ by $E^{\uparrow CN}$. Using this, we can formulate the solution $\nu_o^{\uparrow CN}$ to MS, when limited to normal sublanguages of E, as $E^{\uparrow CN}$. The following lemma (taken from [45]) characterizes this solution.[1] In the sequel, the superscript $E^{\uparrow C_o}$ ($E^{\uparrow N}$) denotes the supremal prefix-closed and $(P(L(G)), \Sigma_{uc})$–controllable ($(L(G), P)$–normal) sublanguage of E.

Lemma 7.1.2 ([45])**.** *Given a non-deterministic finite automaton $G = (X, \Sigma, \delta, X_0)$, a natural projection map P with respect to the set of observable events Σ_{obs} ($\Sigma_{obs} \subseteq \Sigma$), a set of controllable events Σ_c, a set of uncontrollable events $\Sigma_{uc} = \Sigma - \Sigma_c$, and a prefix-closed language $E \subseteq L(G)$, we have $E^{\uparrow CN} = L(G) \cap P^{-1}((P(E^{\uparrow N}))^{\uparrow C_o})$.* ■

7.1.2 State-Based Approach

Another approach for defining supervisory control problems is the state-based approach where, instead of specifying the legal behavior as a prefix-closed language E, a set of forbidden states is provided via some predicate $R : X \to \{0, 1\}$ with $R(x) = 0$ capturing the fact that $x \in X$ is a forbidden state. This set of forbidden states needs to be avoided via a *state-feedback* supervisor $\nu_s : X \to \{\Sigma' \subseteq \Sigma | \Sigma_{uc} \subseteq \Sigma'\}$ ([19,28]). Given the state the system is in, the supervisor determines which controllable events to disable. It can be shown that there exists a state-feedback supervisor such that all states x for which $R(x) = 1$ can be visited under supervision, if and only if R is *controllable* [19, 28], i.e., if and only if it satisfies the following: (i) if state m satisfies R then m is reachable from the initial state of G via a string of states satisfying R, and (ii) at any of the visited states, uncontrollable events

[1] Note that if $\mathcal{CN}(E) = \emptyset$ then there is no solution to MS.

take the system to states which again satisfy R. If R is not controllable, we can seek a controllable predicate that best approximates R from below. Specifically, we say that predicate R_1 refines R_2 if for all $x \in X$, $R_1(x) = 1$ implies $R_2(x) = 1$. Now define $\mathcal{CR}(R)$ to be the set of all predicates that are controllable and refine R. Then, $\mathcal{CR}(R)$ is closed under union and, hence, has a supremal element $sup\mathcal{CR}(R)$ (denoted by $R^{\uparrow CR}$). Let $\nu_s^{\uparrow CR}$ be the state-feedback supervisor that synthesizes[2] the predicate $sup\mathcal{CR}(R)$; also denote by R/G the accessible part of automaton G when all the states that do not satisfy R are removed. Then, for any predicate R, we have $L(\nu_s^{\uparrow CR}/G) = L^{\uparrow C}(R/G)$ [19, 28]. In other words, to find the state-feedback supervisor to synthesize the predicate $sup\mathcal{CR}(R)$, one can first remove all states that do not satisfy R and then find the supremal controllable sublanguage of the closed-behavior of the remaining state transition diagram.

7.2 Minimally Restrictive Opacity-Enforcing Feasible Supervisor Problem (MOES)

Opacity is defined to be a property of the states of the given finite automaton; however, the application of supervisory control to the system modifies the original structure of the automaton and hence its states. The remedy to this problem is to find a way to map states of the supervised system to states of the original system and, hence, re-define the set of secret states for the system under supervision to include all those states that are mapped to secret states in the original system.

Definition 7.2.1 (Opacity for Supervised System)**.** *Given a non-deterministic finite automaton $G = (X, \Sigma, \delta, X_0)$, a natural projection map P with respect to the set of observable events Σ_{obs} ($\Sigma_{obs} \subseteq \Sigma$), and a set of secret states $S \subseteq X$, we say that the non-deterministic automaton $G' = (X', \Sigma', \delta', X_0')$ is (S, P, ∞) initial-state opaque ((S, P, ∞)-opaque) with respect to G if $G_p = G' \times G \equiv (X_p, \Sigma_p, \delta_p, X_{0p})$ is (S_p, P, ∞) initial-state opaque ((S_p, P, ∞)-opaque), where $X_{0p} = \{(x_o', x_o) | x_o' \in X_o', x_o \in X_o\}$ and $S_p = \{(x', x) \in X_p | x \in S\}$.* ■

Using Definition 7.2.1, initial-state (infinite-step) opacity is enforced under

[2]Note that if $\mathcal{CR}(R) = \emptyset$ then the problem has no solution.

supervision if ν_o/G (the supervised system) is (S, P, ∞) initial-state opaque ((S, P, ∞)-opaque) with respect to G. Note that this definition requires ν_o/G to be regular. A feasible supervisor that achieves this property is called an *opacity-enforcing* feasible supervisor for the system and is denoted by ν_{op}.

Definition 7.2.2 (MOES)**.** *Given a non-deterministic finite automaton $G = (X, \Sigma, \delta, X_0)$, a natural projection map P with respect to the set of observable events Σ_{obs} ($\Sigma_{obs} \subseteq \Sigma$), a set of secret states $S \subseteq X$, a prefix-closed and regular language $E \subseteq L(G)$, and a set of controllable events Σ_c ($\Sigma_c \subseteq \Sigma_{obs}$), find an opacity-enforcing feasible supervisor ν_{op} for G such that (i) $L(\nu_{op}/G) \subseteq E$, and (ii) $L(\nu_{op}/G)$ is as large as possible.* ■

The MOES problem that requires enforcing initial-state opacity is denoted with MOES^0 and the MOES problem that requires enforcing infinite-step opacity is denoted with MOES$^\infty$. Following this convention, we denote the initial-state (infinite-step) opacity-enforcing supervisor by ν_{op}^0 (ν_{op}^∞).

7.3 Solution to MOES^0

7.3.1 Characterizing the Solution to MOES^0

The solutions to MOES^0 can be characterized using machinery that already exists in the literature on supervisory control (e.g., [15,19]). We first recall a language-based formulation of the state-based notion of initial-state opacity.

Definition 7.3.1 (Language-Based Definition of Initial-State Opacity)**.** *Given a non-deterministic finite automaton $G = (X, \Sigma, \delta, X_0)$, a natural projection map P with respect to the set of observable events Σ_{obs} ($\Sigma_{obs} \subseteq \Sigma$), and a set of secret states $S \subseteq X$, language $K \subseteq L(G)$ is (S, P, ∞) initial-state opaque with respect to G if*

$$\forall i \in X_0 \cap S, \forall t \in L(G, i) \cap \bar{K}, \exists j \in X_0 - S, \exists s \in L(G, j) \cap \bar{K}\{P(s) = P(t)\}.$$

■

The following lemma relates Definition 7.3.1 to Definition 7.2.1 for a regular language K.

Lemma 7.3.1. *Given a non-deterministic finite automaton* $G = (X, \Sigma, \delta, X_0)$*, a natural projection map* P *with respect to the set of observable events* Σ_{obs} $(\Sigma_{obs} \subseteq \Sigma)$*, a set of secret states* $S \subseteq X$*, a prefix-closed and regular language* $K \subseteq L(G)$ *and the non-deterministic finite automaton* $G_K = (X_K, \Sigma_K, \delta_K, X_{0K})$ *such that* $L(G_K) = K$*,* K *is* (S, P, ∞) *initial-state opaque with respect to* G *if and only if* G_K *is* (S, P, ∞) *initial-state opaque with respect to* G*.* ■

Proof. Observe that $t \in K$ is equivalent to the fact that there exists a state $i_K \in X_{0K}$ such that $t \in L(G_K, i_K)$. Hence, the statement that

$$\forall i \in X_0 \cap S, \forall t \in L(G, i) \cap K, \exists j \in X_0 - S, \exists s \in L(G, j) \cap \bar{K}\{P(s) = P(t)\}, \tag{7.1}$$

is equivalent to saying that for all $i_K \in X_{0K}$ and $i \in S \cap X_0$, if $L(G_K, i_K) \cap L(G, i) \neq \emptyset$ then

$$\forall t \in L(G_K, i_K) \cap L(G, i), \exists j \in X_0 - S, \exists j_K \in X_{0K}, \exists s \in L(G, j) \cap L(G_K, j_K) \{P(s) = P(t)\}. \tag{7.2}$$

Define $G_p = (X_p, \Sigma_p, \delta_p, X_{0p})$ as $G_p = G_K \times G$. Then, $L(G_K, i_K) \cap L(G, i) \neq \emptyset$ is equivalent to the existence of some t such that $\delta_p((i_K, i), t)$ exists for $(i_K, i) \in X_{0p}$. In the sequel, we use i_p to denote (i_K, i). Note that since $i \in S \cap X_0$, we have $i_p \in S_p \cap X_{0p}$. Hence, $L(G_K, i_K) \cap L(G, i) \neq \emptyset$ is equivalent to $\exists i_p \in S_p \cap X_{0p}$ such that $\delta_p(i_p, t)$ exists. Similarly, the statement $\exists s \in L(G, j) \cap L(G_K, j_K)$ is equivalent to $\exists j_p \equiv (j_K, j) \in X_{0p} - S_p$ such that $\delta_p(j_p, s)$ exists. Hence, (7.2) is equivalent to saying that for all $i_p \in S_p \cap X_{0p}$ and for all $t \in L(G_p, i_p)$ there exist $j_p \in X_{0p} - S_p$ and $s \in L(G_p, j_p)$ such that $P(s) = P(t)$. We conclude that (7.1) is equivalent to G_p being (S_p, P, ∞) initial-state opaque. Based on Definition 7.2.1, this is equivalent to G_K being (S, P, ∞) initial-state opaque with respect to G, which concludes the proof. □

Using the notion of initial-state opacity for languages, we now characterize all opacity-enforcing feasible supervisors and derive the solution to MOES^0.

Theorem 7.3.1. *Given a non-deterministic finite automaton* $G = (X, \Sigma, \delta, X_0)$*, a natural projection map* P *with respect to the set of observable events*

Σ_{obs} ($\Sigma_{obs} \subseteq \Sigma$), *a set of secret states* $S \subseteq X$, *a set of controllable events* Σ_c ($\Sigma_c \subseteq \Sigma_{obs}$), *and a set of uncontrollable events* $\Sigma_{uc} = \Sigma - \Sigma_c$, *there exists an opacity-enforcing feasible supervisor* ν_{op}^0 *for* G *such that* $L(\nu_{op}^0/G) = K$, *if and only if: (i)* K *is* $(L(G), \Sigma_{uc})$*–controllable; (ii)* K *is* $(L(G), P)$*–normal; (iii)* K *is* (S, P, ∞) *initial-state opaque with respect to* G. ■

Proof. Follows from Theorem 7.1.1 and Lemma 7.3.1. □

For any $E \subseteq L(G)$, define $\mathcal{P}^0(E)$ to be the set of prefix-closed sublanguages of E that are initial-state opaque. Then, using Theorem 7.3.1, for any supervisor ν_{op}^0 that enforces initial-state opacity we have $L(\nu_{op}^0/G) \in \mathcal{CNP}^0(E) := \mathcal{C}(E) \cap \mathcal{N}(E) \cap \mathcal{P}^0(E)$. MOES^0 requires the minimally restrictive opacity-enforcing feasible supervisor, and since MOES^0 assumes that $\Sigma_c \subseteq \Sigma_{obs}$, Theorem 7.3.1 can be used to characterize the solution to MOES^0 as the supervisor $\nu_{op}^{\uparrow CNP^0}$ such that $L(\nu_{op}^{\uparrow CNP^0}/G) = sup\mathcal{CNP}^0(E) \equiv E^{\uparrow CNP^0}$. In the next section, we prove that such supremal element exists and provide a formulation for it.

7.3.2 Properties of Initial-State Opaque Languages

In this section, through various lemmas, we characterize the language $E^{\uparrow CNP^0}$ and, hence, obtain the solution to MOES^0.

Lemma 7.3.2. *Given a non-deterministic finite automaton* $G = (X, \Sigma, \delta, X_0)$, *a natural projection map* P *with respect to the set of observable events* Σ_{obs} ($\Sigma_{obs} \subseteq \Sigma$), *and a set of secret states* $S \subseteq X$, *for any language* $E \subseteq L(G)$, *we have*

$$\mathcal{P}^0(E) = \{K \subseteq E | K = \overline{K}, K \subseteq L(G) \cap P^{-1}(P(K \cap L(G, X_0 - S)))\}.$$

■

Proof. Define $V \equiv K \cap L(G, X_0 - S)$ and $W \equiv K \cap L(G, X_0 \cap S)$. Recall that

$$\mathcal{P}^0(E) = \{K \subseteq E | K = \overline{K}, \forall t \in K, \forall i \in X_0 \cap S \{t \in L(G, i) \Rightarrow$$

$$\{\exists s \in K, \exists j \in X_0 - S \{s \in L(G, j), P(s) = P(t)\}\}\}$$

which implies that

$$\mathcal{P}^0(E) = \{K \subseteq E | K = \overline{K}, \forall t \in \Sigma^* \{t \in W \Rightarrow$$

$$\{\exists s \in \Sigma^* \{s \in V, P(s) = P(t)\}\}\}\}.$$

Therefore $K \in \mathcal{P}^0(E)$ implies that $P(W) \subseteq P(V)$. Moreover, we have $K = V \cup W$, which implies that $P(K) = P(V) \cup P(W)$ and, since $P(W) \subseteq P(V)$, that $P(K) = P(V)$, or equivalently, $K \subseteq P^{-1}(P(V))$. Moreover, $K \subseteq E \subseteq L(G)$, and hence $K \subseteq (L(G) \cap P^{-1}(P(V)))$. Putting all of these together, we have

$$\mathcal{P}^0(E) = \{K \subseteq E | K = \overline{K}, K \subseteq (L(G) \cap P^{-1}(P(V)))\}$$

which concludes the proof. □

The next lemma states that the set of initial-state opaque and prefix-closed sublanguages of E has a supremal element.

Lemma 7.3.3. *$\mathcal{P}^0(E)$ is nonempty and closed under arbitrary unions; in particular, the supremal element sup$\mathcal{P}^0(E)$ exists in $\mathcal{P}^0(E)$.*

Proof. We have $\emptyset \in \mathcal{P}^0(E)$ and hence $\mathcal{P}^0(E)$ is non-empty. Now let $K_\lambda \in \mathcal{P}^0(E)$ for $\lambda \in \Lambda$, some index set. We show that $K := \bigcup K_\lambda \in \mathcal{P}^0(E)$. This is straightforward since for all $\lambda \in \Lambda$:

$$K_\lambda \in \mathcal{P}^0(E) \Rightarrow K_\lambda \subseteq (L(G) \cap P^{-1}(P(K_\lambda \cap L(G, X_0 - S)))));$$

hence,

$$\begin{aligned} K &\subseteq \bigcup (L(G) \cap P^{-1}(P(K_\lambda \cap L(G, X_0 - S)))) \\ &= (L(G) \cap \bigcup P^{-1}(P(K_\lambda \cap L(G, X_0 - S)))) \\ &= (L(G) \cap P^{-1}(P(\bigcup K_\lambda \cap L(G, X_0 - S)))) \\ &= (L(G) \cap P^{-1}(P(K \cap L(G, X_0 - S)))), \end{aligned}$$

which implies that $K \in \mathcal{P}^0(E)$. Therefore, $\mathcal{P}^0(E)$ is closed under arbitrary unions and hence the supremal element exists. □

Assuming that E is prefix-closed, the following theorem derives a formulation for the supremal element $E^{\uparrow P^0}$ of $\mathcal{P}^0(E)$.

Theorem 7.3.2. *Given a non-deterministic finite automaton $G = (X, \Sigma, \delta, X_0)$, a natural projection map P with respect to the set of observable events Σ_{obs} ($\Sigma_{obs} \subseteq \Sigma$), and a set of secret states $S \subseteq X$, for any prefix-closed language $E \subseteq L(G)$, we have $E^{\uparrow P^0} = E \cap P^{-1}(P(E \cap L(G, X_0 - S)))$.* ■

Proof. To prove the theorem, we show that: (a) every $K \in \mathcal{P}^0(E)$ is a subset of $H \equiv E \cap P^{-1}(P(E \cap L(G, X_0 - S)))$, and (b) $H \in \mathcal{P}^0(E)$. From Lemma 7.3.2, $K \in \mathcal{P}^0(E)$ implies that $K \subseteq L(G) \cap P^{-1}(P(K \cap L(G, X_0 - S)))$. Since $K \subseteq E$, we have $K \subseteq L(G) \cap P^{-1}(P(E \cap L(G, X_0 - S)))$. Finally, $K \subseteq E \subseteq L(G)$ implies that $K \subseteq H$. To show $H \in \mathcal{P}^0(E)$ we need to show that: (i) H is initial-state opaque, and (ii) H is prefix-closed. For (i), note that H contains all strings s in E for which there exists a string t in E that has the same projection (as s) and starts from a non-secret state. Hence, based on the definition of initial-state opacity, H is initial-state opaque. Moreover, $L(G, X_0 - S)$ is prefix-closed by construction; this implies that $P^{-1}(P(L(G, X_0 - S)))$ is also prefix-closed since projection and inverse projection preserve prefix closure [19, 28]. Furthermore, the intersection of two prefix-closed languages is prefix-closed [19, 28], thus H is prefix-closed. This completes the proof that $H = E^{\uparrow P^0}$. □

Corollary 7.3.1. *Given a non-deterministic finite automaton $G = (X, \Sigma, \delta, X_0)$, a natural projection map P with respect to the set of observable events Σ_{obs} ($\Sigma_{obs} \subseteq \Sigma$), and a set of secret states $S \subseteq X$, normality is preserved under $^{\uparrow P^0}$ operator for a prefix-closed and normal language $E \subseteq L(G)$.* ■

Proof. Define $M \equiv P(E \cap L(G, X_0 - S))$. Note that M is prefix-closed since prefix-closure is preserved under projection and inverse projection [19, 28]. Using Theorem 7.3.2, $E^{\uparrow P^0}$ can also be written (since $E \subseteq L(G)$) as $E^{\uparrow P^0} = E \cap P^{-1}(M) \cap L(G)$. It follows from Lemma 7.1.1 that $P^{-1}(M) \cap L(G)$ is normal. If E is normal, then $E^{\uparrow P^0}$ is normal since normality is preserved under intersection for prefix-closed languages [19, 28]. □

In the next lemma we prove that for any prefix-closed and normal language E and for any $M = \overline{M} \subseteq P(E \cap L(G, X_0 - S))$, $K \equiv L(G) \cap P^{-1}(M)$ is a sublanguage of E that is both normal and initial-state opaque.

Lemma 7.3.4. *Given a non-deterministic finite automaton* $G = (X, \Sigma, \delta, X_0)$*, a natural projection map* P *with respect to the set of observable events* Σ_{obs} $(\Sigma_{obs} \subseteq \Sigma)$*, and a set of secret states* $S \subseteq X$*, for any prefix-closed and normal language* $E \subseteq L(G)$ *and any prefix-closed language* $M \subseteq P(E \cap L(G, X_0 - S))$*, we have* $K \equiv P^{-1}(M) \cap L(G) \in \mathcal{NP}^0(E)$*.* ■

Proof. Define $U \equiv E \cap L(G, X_0 - S)$ and $V \equiv K \cap L(G, X_0 - S)$. To prove that $K \in \mathcal{NP}^0(E)$ we need to show that: (i) $K \subseteq E$, (ii) K is normal, and (iii) K is initial-state opaque. For (i), it follows from the definition of M that $K \equiv P^{-1}(M) \cap L(G) \subseteq P^{-1}(P(E)) \cap L(G)$. Since E is normal, we have $P^{-1}(P(E)) \cap L(G) = E$ and hence $K \subseteq E$. Condition (ii) is true by Lemma 7.1.1. For (iii), we again use the definition of M to infer that $K \subseteq P^{-1}(P(U)) \cap L(G)$; hence $s \in K$ implies that $s \in P^{-1}(P(U))$ which in turn implies that there exists $t \in U$ such that $P(s) = P(t)$. Now $t \in U$ implies that $t \in L(G)$ and, hence, by normality of K (and since $P(s) = P(t)$) we have $t \in K$; moreover, $t \in L(G, X_0 - S)$ and, hence, $t \in V$. Thus, for all $s \in K$ there exists $t \in V$ such that $P(s) = P(t)$, i.e., $P(K) \subseteq P(V)$ which implies that $K \subseteq P^{-1}(P(V))$ and since $K \subseteq L(G)$, that $K \subseteq P^{-1}(P(V)) \cap L(G)$. By Lemma 7.3.2, $K \in \mathcal{P}^0$ which concludes the proof. □

We complete the analysis on the properties of initial-state opaque languages by considering the controllability condition.

Lemma 7.3.5. *Given a non-deterministic finite automaton* $G = (X, \Sigma, \delta, X_0)$*, a natural projection map* P *with respect to the set of observable events* Σ_{obs} $(\Sigma_{obs} \subseteq \Sigma)$*, a set of secret states* $S \subseteq X$*, a set of controllable events* Σ_c $(\Sigma_c \subseteq \Sigma_{obs})$*, and a set of uncontrollable events* $\Sigma_{uc} = \Sigma - \Sigma_c$*, initial-state opacity is preserved under* $^{\uparrow CN}$ *operator for any prefix-closed and normal language* $E \subseteq L(G)$*.* ■

Proof. We show that $K \in \mathcal{NP}^0(E)$ implies $K^{\uparrow CN} \in \mathcal{P}^0$. Since K is prefix-closed, from Lemma 7.1.2 we have $K^{\uparrow CN} = L(G) \cap P^{-1}((P\big(K^{\uparrow N}\big))^{\uparrow C_o})$. Moreover, $K \in \mathcal{N}(E)$ implies that $K^{\uparrow N} = K$ and, hence, $K^{\uparrow CN} = L(G) \cap P^{-1}((P(K))^{\uparrow C_o})$. Define $M \equiv (P(K))^{\uparrow C_o}$ such that $K^{\uparrow CN} = L(G) \cap P^{-1}(M)$. Note that $K \in \mathcal{P}^0$ which by Lemma 7.3.2 implies that $K \subseteq L(G) \cap P^{-1}(P(K \cap L(G, X_0 - S)))$ or, equivalently, $P(K) = P(K \cap L(G, X_0 - S))$ (refer to the proof of Lemma 7.3.2). Therefore, $M = (P(K))^{\uparrow C_o} \subseteq P(K) = P(K \cap L(G, X_0 - S))$; by Lemma 7.3.4, $K^{\uparrow CN} = L(G) \cap P^{-1}(M) \in \mathcal{NP}^0(K)$ which

implies that $K^{\uparrow CN} \in \mathcal{NP}^0(E)$ since $K \subseteq E$. This in turn implies that $K^{\uparrow CN} \in \mathcal{P}^0$ which completes the proof. □

The following theorem derives a formulation for $E^{\uparrow CNP^0}$.

Theorem 7.3.3. *Given a non-deterministic finite automaton $G = (X, \Sigma, \delta, X_0)$, a natural projection map P with respect to the set of observable events Σ_{obs} ($\Sigma_{obs} \subseteq \Sigma$), a set of secret states $S \subseteq X$, a set of controllable events Σ_c ($\Sigma_c \subseteq \Sigma_{obs}$), and a set of uncontrollable events $\Sigma_{uc} = \Sigma - \Sigma_c$, a prefix-closed language $E \subseteq L(G)$, we have*

$$E^{\uparrow CNP^0} = L(G) \cap P^{-1}((P((E^{\uparrow N})^{\uparrow P^0}))^{\uparrow C_o}).$$

■

Proof. By Lemma 7.3.5, initial-state opacity is preserved under $^{\uparrow CN}$ operator for normal languages; hence $E^{\uparrow CNP^0} = (E^{\uparrow NP^0})^{\uparrow CN}$. Also, by Corollary 7.3.1, normality is preserved under the $^{\uparrow P^0}$ operator for normal languages; therefore, $E^{\uparrow NP^0} = (E^{\uparrow N})^{\uparrow P^0}$. Putting these two facts together, we have $E^{\uparrow CNP^0} = ((E^{\uparrow N})^{\uparrow P^0})^{\uparrow CN}$. Using Lemma 7.1.2, the proof is complete. □

7.3.3 Implementing the Solution to MOES^0 using the Initial-State Estimator

MOES^0 requires the minimally restrictive opacity-enforcing feasible supervisor ν_{op} that can enforce the legal behavior described via the prefix-closed language E. Theorem 7.3.1 states that the solution to MOES^0 boils down to $E^{\uparrow CNP^0}$ and Theorem 7.3.3 (assuming that E is normal) characterizes this solution as $E^{\uparrow CNP^0} = L(G) \cap P^{-1}((P(E^{\uparrow P^0}))^{\uparrow C_o})$. Observe that Theorem 7.3.2 characterizes $E^{\uparrow P^0} = E \cap P^{-1}(P(E \cap L(G, X_0 - S)))$ which implies that $E^{\uparrow P^0}$ can be implemented using projection and intersection operations on languages. Also [45] provides a formulation for the supremal controllable sublanguage $E^{\uparrow C}$ (assuming that E is prefix-closed) using *concatenation*, *intersection* and *complementation* operations on languages. For regular languages, all these operations can be implemented using operations on finite automata [27]. Since in MOES^0 E is assumed to be regular, $E^{\uparrow CNP^0}$ can be obtained (and, hence, the solution to MOES^0 can be implemented) via

operations on the automata describing E and G, which implies that $E^{\uparrow CNP^0}$ is regular. We now point out that the ISE construction can be used for verifying initial-state opacity and propose an algorithm that efficiently integrates the verification and control problems. In the sequel we assume, without loss of generality, that E is normal (if this assumption is not satisfied, we can always first compute $E^{\uparrow N}$ using the results in [45]).

Algorithm A: Consider a non-deterministic finite automaton $G = (X, \Sigma, \delta, X_0)$, with a set of observable events Σ_{obs} ($\Sigma_{obs} \subseteq \Sigma$), a set of controllable events Σ_c ($\Sigma_c \subseteq \Sigma_{obs}$), a set of secret states $S \subseteq X$, and a prefix-closed, normal and regular language $E \subseteq L(G)$, $E \neq \emptyset$, describing the legal behavior. We can obtain a minimally restrictive supervisor $\nu_s^{\uparrow CR}$ for G that enforces initial-state opacity with respect to the natural projection map $P : \Sigma \to \Sigma_{obs}$ via the following steps: (i) Construct automaton $G_E = (X_E, \Sigma, \delta_E, X_{0E})$ such that $L(G_E) = E$. (ii) Construct $G_p = G_E \times G = (X_p, \Sigma, \delta_p, X_{0p})$; define $S_p \equiv \{(x_e, x) \in X_p | x \in S\}$. (iii) Construct the ISE $G^{(p)}_{\infty,obs} = (X^{(p)}_{\infty,obs}, \Sigma_{obs}, \delta^{(p)}_{\infty,obs}, X^{(p)}_{\infty,0})$ corresponding to G_p. (iv) Construct the state-feedback supervisor $\nu_s^{\uparrow CR}$ for $G^{(p)}_{\infty,obs}$ that avoids all states in $G^{(p)}_{\infty,obs}$ for which the set of starting states of the associated state mapping (is nonempty and) contains no state outside S_p; for this, first define the predicate $R : X^{(p)}_{\infty,obs} \to \{0, 1\}$ as $R(m) = 0$, if $m(1) \subseteq S_p$ and $m \neq \emptyset$, and $R(m) = 1$, otherwise. Then construct the accessible part $R/G^{(p)}_{\infty,obs}$ of the ISE $G^{(p)}_{\infty,obs}$ when all the states that do not satisfy R are removed, and find the supremal controllable sublanguage of the closed-behavior of the remaining state transition diagram. ■

Theorem 7.3.4. *Given a prefix-closed, normal and regular language $E \subseteq L(G)$, $E \neq \emptyset$, the control action of the solution $\nu_{op}^{\uparrow CNP^0}$ to MOES^0 after observing ω is the same as the control action of the state-feedback supervisor $\nu_s^{\uparrow CR}$ (as synthesized by Algorithm A) at the state reached in $G^{(p)}_{\infty,obs}$ via ω.*■

Proof. We first show that steps (i)-(iv) implement $P(E^{\uparrow P^0})$. In [5], we showed that in the ISE $G_{\infty,obs}$ for G, $P(L(G, X_0 - S))$ is characterized via the set of strings in $G_{\infty,obs}$ that start from its initial state and reach a state in $G_{\infty,obs}$ for which the associated state mapping contains in its set of starting states at least one state outside the set of secret states S. Using this result, we can argue that $P(E \cap L(G, X_0 - S))$ is characterized via the set of strings in $G^{(p)}_{\infty,obs}$ (as constructed in Algorithm A) that start from initial state $X^{(p)}_{0,obs}$

and reach a state in $G^{(p)}_{\infty,obs}$ for which the associated state mapping contains at least one state outside S_p in its set of starting states. Theorem 7.3.2 states that $E^{\uparrow P^0} = E \cap P^{-1}(P(E \cap L(G, X_0 - S)))$ which implies that $P(E^{\uparrow P^0}) = P(E \cap P^{-1}(P(E \cap L(G, X_0 - S))))$ and thus $P(E^{\uparrow P^0}) = P(E) \cap P(P^{-1}(P(E \cap L(G, X_0 - S))))$, since both E and $P^{-1}(P(E \cap L(G, X_0 - S)))$ are prefix closed; in turn, this implies that $P(E^{\uparrow P^0}) = P(E \cap L(G, X_0 - S))$, which means that one can implement $P(E^{\uparrow P^0})$ by removing in the ISE the states that do not satisfy R, i.e., $L(R/G^{(p)}_{\infty,obs}) = P(E^{\uparrow P^0})$. Removing states and the associated labels from ISE $G_{\infty,obs}$ (i.e., evaluating $R/G^{(p)}_{\infty,obs}$) might violate the controllability condition, hence step (v) of the algorithm implements $(P(E^{\uparrow P^0}))^{\uparrow C_o}$; therefore, the state-feedback supervisor $\nu_s^{\uparrow CR}$ synthesizes the predicate $sup\mathcal{CR}(R)$. We have $L(\nu_s^{\uparrow CR}/G) = L^{\uparrow C}(R/G)$ [19, 28] and, hence, $L(\nu_s^{\uparrow CR}/G^{(p)}_{\infty,obs}) = (L(R/G^{(p)}_{\infty,obs}))^{\uparrow C_o} = (P(E^{\uparrow P^0}))^{\uparrow C_o}$.

Finally, to prove the theorem we need to show that

$$\nu_{op}(\omega) \equiv \nu_s^{\uparrow CR}(\delta^{(p)}_{\infty,obs}(X^{(p)}_{\infty,0}, \omega))$$

is the solution to MOES^0 where $\delta^{(p)}_{\infty,obs}(X^{(p)}_{\infty,0}, \omega)$ denotes the state in $G^{(p)}_{\infty,obs}$ reached via ω from initial state $X^{(p)}_{\infty,0}$. For this, we can equivalently prove that the transition structure of the automaton $\tilde{G} \equiv \nu_s^{\uparrow CR}/G^{(p)}_{\infty,obs}$ can be used to implement the solution to MOES^0 as follows: a string s can be executed in the closed loop system ν_{op}/G if its projection $P(s) = \omega$ belongs to $L(\tilde{G})$. For this, we show that: (i) $\tilde{G}$ is an opacity-enforcing feasible supervisor, and (ii) $L(\tilde{G}/G) = E^{\uparrow CNP^0}$. Note that automaton $\tilde{G}$ is a feasible supervisor for G since $L(\tilde{G})$ is $(P(L(G)), \Sigma_{uc})$–controllable [19, 28]. Moreover, $L(\tilde{G})$ is initial-state opaque by construction. Hence, $\tilde{G}$ is an opacity-enforcing feasible supervisor. To show (ii), note that the supervisor $\tilde{G}$ observes the behavior through the projection map P; therefore, the behavior of $\tilde{G}$ can be described via $L(\tilde{G}/G) = L(G) \cap P^{-1}(L(\tilde{G}))$. Moreover, $L(G) \cap P^{-1}(L(\tilde{G})) = L(G) \cap P^{-1}((P(E^{\uparrow P^0}))^{\uparrow C_o}) = E^{\uparrow CNP^0}$ which results in $L(\tilde{G}/G) = E^{\uparrow CNP^0}$. □

Example 7.3.1. *Consider the DES G in Figure 4.1-a. Assume that $\Sigma_c = \{\beta\}$. Also suppose that $E = L(G)$ meaning that the only requirement for the supervisor is to enforce initial-state opacity. This implies that steps (i)-(iii) of Algorithm A can be replaced by the construction of the ISE for G. Figure 4.1-b depicts this ISE for G and denotes it by $G_{\infty,obs}$. This system is not $(\{1, 3\}, P, \infty)$ initial-state opaque due to the existence of ISE state*

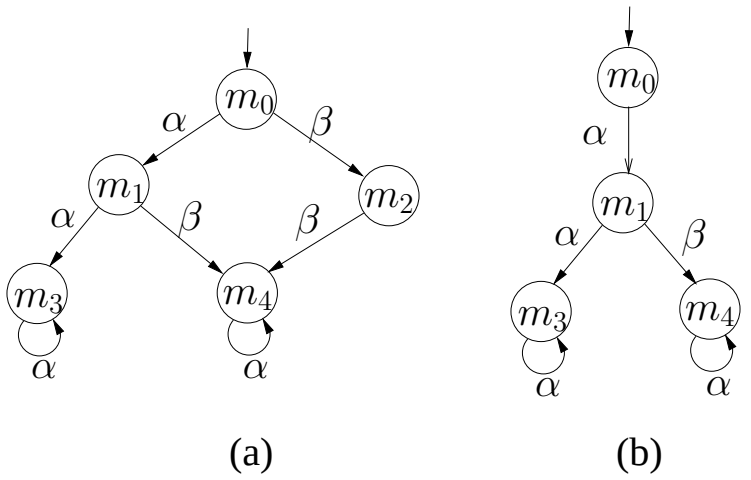

Figure 7.1: (a) $R/G_{\infty,obs}$; (b) $\tilde{G} \equiv \nu_s^{\uparrow CR}/G_{\infty,obs}$ as well as the minimally restrictive opacity-enforcing supervisor $\nu_{op}^{\uparrow CNP^0}$ for G in Figure 4.1-a.

$m_5 = \{(1,4),(3,4)\}$ *whose set of starting states* ($m_5(1) = \{1,3\}$) *is strictly within* S. *In other words, observing* $\beta\alpha(\alpha)^*$ *determines the system initial state to be within the set* $\{1,3\}$.

To obtain the minimally restrictive supervisor that enforces $(\{1,3\},P,\infty)$ *initial-state opacity, following step (iv) of Algorithm A, we first remove from* $G_{\infty,obs}$ *the states that violate initial-state opacity, in this case,* m_5*; Figure 7.1-a depicts the accessible part of the remaining automaton* $R/G_{\infty,obs}$. *Next, following step (v) of Algorithm A, we check for the controllability condition. At state* m_2, α *is disabled which is an uncontrollable event. Hence, access to state* m_2 *should be rejected earlier, which is accomplished by disabling* β *at state* m_0. *Figure 7.1-b depicts the automaton associated with the supremal controllable sublanguage of the automaton in Figure 7.1-a. Based on this, the supervisor does not allow observing* β *as the first observation. Indeed, observing* $\beta\alpha$ *determines the system initial state to be within the set* $\{1,3\}$, *which consists exclusively of secret states (and, hence, violates initial-state opacity).* ■

7.4 Solution to MOES$^\infty$

Verifying infinite-step opacity using Theorem 5.3.3 consists of two phases:

(i) The first phase constructs a standard current-state estimator and, as part of verifying infinite-step opacity, we need to ensure that none of these

current-state estimates lies entirely within the set of secret states, i.e., the system is current-state opaque (or 0-step opaque). If current-state opacity is violated, we can follow an approach similar to the one in Section 7.3, to obtain a current-state estimator (instead of an initial-state estimator) and then construct a state-feedback supervisor that enforces current-state opacity (instead of initial-state opacity). Without loss of generality, we assume that the system under consideration is current-state opaque (otherwise, one can always design an optimal supervisor that enforces current-state opacity and then consider the controlled system under this supervisor as the new system to be controlled).

(ii) The second phase constructs, for each current-state estimate $Z \subseteq X$ provided in the first phase, an ISE whose initial state is associated with the state mapping $\odot(Z)$. We argued that if any of these ISEs contains a state with associated (non-empty) state mapping m such that the set of starting states $m(1)$ contains elements only in S (i.e., $m(1) \subseteq S$ and $m(1) \neq \emptyset$), then DES G is not infinite-step opaque. Therefore, one can enforce infinite-step opacity by prohibiting sequences of observations which reach such ISE states in *one* of the associated ISEs. This requirement can be equivalently described as a MOES^0 problem where the intruder knows that the initial state of the system is Z. We denote this special case of the MOES^0 problem as MOES^0_Z where subscript $Z \subseteq X$ represents the initial uncertainty about the initial state is Z (as opposed to MOES^0 where this uncertainty is taken to be X_0). Given any discrete event system G with N states, assume that the current-state estimator has $M := |X_{0,obs}| \leq 2^N$ states, which we denote by $X_{0,obs} = \{Z_0, \ldots, Z_{M-1}\}, Z_q \subseteq X, 0 \leq q \leq M-1$. We therefore obtain supervisors that solve MOES$^0_{Z_q}$, $0 \leq q \leq M-1$, as $\nu^{\uparrow CNP^0}_{op,q}$. Recall that for any solution $\nu^{\uparrow CNP^0}_{op}$ to MOES^0, we have an equivalent state-feedback supervisor $\nu^{\uparrow CR}_{s}$. Thus, for any solution $\nu^{\uparrow CNP^0}_{op,q}$ to MOES$^0_{Z_q}$, the equivalent state-feedback supervisor is denoted by $\nu^{\uparrow CR}_{s,q}$. The following algorithm formalizes the construction of $\nu^{\uparrow CR}_{s,q}$, $0 \leq q \leq M-1$.

Algorithm B Consider a non-deterministic finite automaton $G = (X, \Sigma, \delta, X_0)$, with a set of observable events Σ_{obs} ($\Sigma_{obs} \subseteq \Sigma$), a set of controllable events Σ_c ($\Sigma_c \subseteq \Sigma_{obs}$), a set of secret states $S \subseteq X$, and a prefix-closed, normal and regular language $E \subseteq L(G)$, $E \neq \emptyset$, describing the legal behavior. Assume (without loss of generality) that G is current-state opaque. We can obtain a bank of state-feedback supervisors $\nu^{\uparrow CR}_{s,q}$ for G

that enforce initial-state opacity with respect to the natural projection map $P : \Sigma \rightarrow \Sigma_{obs}$ via the following steps: (i) Construct automaton $G_E = (X_E, \Sigma, \delta_E, X_{0E})$ such that $L(G_E) = E$. (ii) Construct $G_p = G_E \times G = (X_p, \Sigma, \delta_p, X_{0p})$ and define $S_p \equiv \{(x_e, x) \in X_p | x \in S\}$. (iii) Construct the current-state estimator $G^{(p)}_{0,obs} = (X^{(p)}_{obs}, \Sigma_{obs}, \delta^{(p)}_{obs}, X_{0p})$ corresponding to G_p. (iv) For each set of current-state estimates Z_q associated with a state of the current-state estimator $G^{(p)}_{0,obs}$, construct the initial-state estimator $G^{(p,q)}_{\infty,obs} = (X^{(p,q)}_{\infty,obs}, \Sigma_{obs}, \delta^{(p,q)}_{\infty,obs}, X^{(p,q)}_{\infty,0})$ by setting its initial state $X^{(p,q)}_{\infty,0}$ to be $\odot(Z_q)$. (v) For each ISE $G^{(p,q)}_{\infty,obs}$, $0 \leq q \leq M-1$, where M denotes the number of states of the current-state estimator $G^{(p)}_{0,obs}$, construct the state-feedback supervisor $\nu^{\uparrow CR}_{s,q}$ for $G^{(p,q)}_{\infty,obs}$ that avoids all states in $G^{(p,q)}_{\infty,obs}$ for which the set of starting states of the associated state mapping contains no state outside S_p; for this, first define the predicate $R : X^{(p,q)}_{\infty,obs} \rightarrow \{0, 1\}$ as $R(m) = 0$, if $m(1) \subseteq S_p$ and $m \neq \emptyset$, and $R(m) = 1$, otherwise. Then construct the accessible part $R/G^{(p,q)}_{\infty,obs}$ of the ISE $G^{(p,q)}_{\infty,obs}$ when all the states that do not satisfy R are removed, and find the supremal controllable sublanguage of the closed behavior of the remaining state transition diagram. ■

Following the approach in Theorem 7.3.4, it can be shown that the control action of the solution $\nu^{\uparrow CNP^0}_{op,q}$ to $\text{MOES}^0_{Z_q}$ after observing ω *and assuming that the initial uncertainty is* Z_q is the same as the control action of the state-feedback supervisor $\nu^{\uparrow CR}_{s,q}$ (as synthesized by Algorithm B) at the state reached in $G^{(p,q)}_{\infty,obs}$ via ω. In the sequel, we define the state of the supervisor $\nu^{\uparrow CR}_{s,q}$ as the current state of the automaton $\nu^{\uparrow CR}_{s,q}/G^{(p,q)}_{\infty,obs}$.

We use the bank of state-feedback supervisors $\nu^{\uparrow CR}_{s,q}$, $0 \leq q \leq M-1$, associated with each state of the current-state estimator, to enforce infinite-step opacity as follows. Following a sequence of observations ω, we enable the state-feedback supervisor $\nu^{\uparrow CR}_{s,q}$ associated with the state Z_q of the current-state estimator (Z_q is reachable via ω). Once the state-feedback supervisor $\nu^{\uparrow CR}_{s,q}$ is enabled, it will stay enabled for the remainder of the operation of the system. Note that a state-feedback supervisor $\nu^{\uparrow CR}_{s,q}$ starts operating from its (unique) initial state and its current state is updated according to new observations. This guarantees that the membership of the system's state to the set of secret states (at the point where the sequence of observations ω is made) will be kept opaque for all possible subsequent observations. Since the control action of the state-feedback supervisor $\nu^{\uparrow CR}_{s,q}$ is merely determined by its state, we simply need to track its current state based on the observed

events. Upon observing a new label α, we need to first update the current state of all enabled state-feedback supervisors, and *also* enable a new supervisor to ensure that the membership of the system's state to the set of secret states (at the point where the sequence of observations $\omega\alpha$ is made) will be kept opaque for all possible future observations. As a result, after observing the sequence of observations ω, $|\omega|$ state-feedback supervisors are enabled and the overall control action is defined to be the intersection of the individual control actions of these $|\omega|$ supervisors.

Theorem 7.4.1. *Consider a non-deterministic finite automaton* $G = (X, \Sigma, \delta, X_0)$*, with a set of observable events* Σ_{obs} *(*$\Sigma_{obs} \subseteq \Sigma$*), a set of controllable events* Σ_c *(*$\Sigma_c \subseteq \Sigma_{obs}$*), a set of secret states* $S \subseteq X$*, and a prefix-closed, normal and regular language* $E \subseteq L(G)$*,* $E \neq \emptyset$*, describing the legal behavior. Assume that the system is current-state opaque and construct the current-state estimator* $G^{(p)}_{0,obs} = (X^{(p)}_{obs}, \Sigma_{obs}, \delta^{(p)}_{obs}, X_{0p})$ *and the bank of state-feedback supervisors* $\nu^{\uparrow CR}_{s,q}$ *associated with the bank of initial-state estimators* $G^{(p,q)}_{\infty,obs} = (X^{(p,q)}_{\infty,obs}, \Sigma_{obs}, \delta^{(p,q)}_{\infty,obs}, X^{(p,q)}_{\infty,0})$*,* $0 \leq q \leq M-1$*, where* M *denotes the number of states of the current-state estimator* $G^{(p)}_{0,obs}$ *(as in Algorithm B). Also assume that upon observing the sequence of observations* $\omega = \alpha_0\alpha_1 \ldots \alpha_n$*, the sequence of states visited by the current-state estimator* $G^{(p)}_{0,obs}$ *is given by* $Z_{q_0} = X_{0p}$*,* Z_{q_1}*, ...,* Z_{q_n}*,* $Z_{q_{n+1}}$*,* $0 \leq q_i \leq M-1$*,* $0 \leq i \leq n+1$*, which we denote by* $Z_{q_0} \xrightarrow{\alpha_0} Z_{q_1} \xrightarrow{\alpha_1} \ldots Z_{q_n} \xrightarrow{\alpha_n} Z_{q_{n+1}}$*. Then:*

(i) The control action of the solution $\nu^{\uparrow CNP^\infty}_{op}$ *to MOES*$^\infty$ *after observing the sequence of observations* ω *is*

$$\nu^{\uparrow CNP^\infty}_{op}(\omega) = \nu^{\uparrow CR}_{s,q_0}(z_0) \cap \nu^{\uparrow CR}_{s,q_1}(z_1) \cap \ldots \cap \nu^{\uparrow CR}_{s,q_n}(z_n) \cap \nu^{\uparrow CR}_{s,q_{n+1}}(z_{n+1}), \quad (7.3)$$

where z_i*,* $0 \leq i \leq n$*, is the state in* $G^{(p,q_i)}_{\infty,obs}$ *that is reached from its initial state* $X^{(p,q_i)}_{\infty,0}$ *via* $\alpha_i\alpha_{i+1} \ldots \alpha_n$*, i.e.,* $z_i = \delta^{(p,q_i)}_{\infty,obs}(X^{(p,q_i)}_{\infty,0}, \alpha_i\alpha_{i+1} \ldots \alpha_n)$ *and* $z_{n+1} = X^{(p,q_{n+1})}_{\infty,0}$*.*

(ii) The solution $\nu^{\uparrow CNP^\infty}_{op}$ *to MOES*$^\infty$ *always exists.* ■

Proof. (i) We prove (7.3) by induction on the length of the sequence of observations. The control action of $\nu^{\uparrow CNP^\infty}_{op}$ for $\omega = \epsilon$, i.e., before any observation is made, enforces the opacity of the membership of the initial state of the system to the set of secret states. Therefore, the control action of $\nu^{\uparrow CNP^\infty}_{op}$ for $\omega = \epsilon$ is equivalent to control action of $\nu^{\uparrow CR}_{s,q_0}(z_0)$ which enforces initial-state

opacity. Next, assuming that (7.3) holds for $\omega = \alpha_0\alpha_1 \ldots \alpha_n$, we prove that for any $\alpha_{n+1} \in \Sigma_{obs}$, (7.3) holds for $\omega = \alpha_0\alpha_1 \ldots \alpha_n\alpha_{n+1}$. Denote the (last) state-feedback supervisor that is enabled after observing $\omega = \alpha_0\alpha_1 \ldots \alpha_{n+1}$ by $\nu_{s,q}^{\uparrow CR}$.

To establish the theorem, we essentially show that the control actions of previously enabled state-feedback supervisors do not affect the control action of state-feedback supervisor $\nu_{s,q}^{\uparrow CR}$, and visa versa. This implies that the MOES$^\infty$ problem is equivalent to a set of *independent* MOES^0 problems, and the optimality of the solution to MOES$^\infty$ follows from the optimality of the solutions to each of the MOES^0 problems.

We first describe how the design of supervisor ν_j may be affected because certain behavior in the system has already been disabled prior to the enabling of supervisor ν_j due to the actions of supervisors that were enabled earlier, say supervisor ν_i. For notational simplicity we take $j = 2$ and $i = 1$, but the discussion can be easily extended to any $i, j \geq 0$. We start by describing what the potential problems might be and argue that these potential problems do not surface. Assume that string $s = s_1s_2s_3$ with $P(s_1) = \omega_1$, $P(s_2) = \omega_2$, and $P(s_3) = \omega_3$ is removed by supervisor ν_1 after observing ω_1 and that supervisor ν_2 becomes enabled when the sequence of observations $\omega_1\omega_2$ is observed. Also, denote the current-state estimate at the point when sequence of observations $\omega_1\omega_2$ was observed by X_{12}. Recall that supervisor ν_2 enforces opacity of the membership of the system state to the set of secret states at the point when the sequence of observations $\omega_1\omega_2$ was observed for all future observations. Now, if the only string with projection ω_3 that can originate from non-secret states in X_{12} is s_3 and there exist(s) other string(s) s_1s_2t with $P(t) = \omega_3$ such that string t can originate from secret states in X_{12}, then disabling string $s_1s_2s_3$ may reveal later on that the system actual state at the point when the sequence of observations $\omega_1\omega_2$ was observed belonged to the set of secret states. Note that ν_2 is designed without taking into account the effect of the supervisor ν_1; this implies that the supervisor ν_2 will not disable string t since ν_2 is under the impression that in the corresponding system model, there exists a string s_3 that originates from a non-secret state and has the same projection as string t, i.e., $P(s_3) = P(t) = \omega_3$.

The situation described above cannot occur because when a string s is removed from the closed behavior $L(G)$ of the DES G due to the action of supervisor ν_1 with *partial observation*, all strings t with the same projection,

i.e., all strings in $P^{-1}(P(s))$, will also be removed from $L(G)$. This implies that if string $s_1 s_2$ is removed by supervisor ν_1, all other strings $s_1 t$ with $P(s_1 t) = P(s_1 s_2)$ are also removed by supervisor ν_1. Hence, the situation previously described cannot occur. We can use a similar argument to establish that the control actions of ν_2 need not be considered in the design of ν_1. This proves that the designs of ν_1 and ν_2 can be considered independently, which completes the proof of part (i).

(ii) In part (i) we show that the solution $\nu_{op}^{\uparrow CNP^{\infty}}$ to MOES$^{\infty}$ exists if and only if $\nu_{s,q}^{\uparrow CR}$ exists for $0 \leq q \leq M-1$. Theorem 7.3.1 proves that $\nu_{s,q}^{\uparrow CR}$, $0 \leq q \leq M-1$, always exists and hence the proof for part (ii) is complete. □

In practice, we only need to enable state-feedback supervisors associated with a current-state estimator state Z such that $Z \cap S \neq \emptyset$ because, when $Z \cap S = \emptyset$, no sequence of observations can refine the initial state uncertainty (captured by the set Z) to a nonempty subset of S. Nevertheless, since we enable a new state-feedback supervisor each time we observe a new label, it seems that an infinite number of supervisors needs to be used. As it turns out, however, this scheme can be implemented with finite space complexity by taking advantage of the fact that there is a finite number of supervisors, each with a finite number of states. Next, we describe the details of this implementation.

During the operation of the system, if the current-state estimator state Z_q is visited twice (for example, after observing the sequence of observations ω and $\omega\Omega$) then the state-feedback supervisor $\nu_{s,q}^{\uparrow CR}$ associated with the current-state estimator state Z_q should be enabled twice. To model this, we store a single copy of $\nu_{s,q}^{\uparrow CR}$ but allow it to simultaneously lie in more than one state. Then, the control action for each $\nu_{s,q}^{\uparrow CR}$ can be defined as the intersection of the control actions at each of its current states. The implementation of the supervisor in this way results in the implementation of a non-deterministic finite automaton which is relatively straightforward and requires finite memory. From this point onwards, upon observing a new label, both of the states are updated and the control action is obtained by taking the intersection of the control actions in the resulting states. In this way, we can implement two or more state-feedback supervisors that share the same structure $\nu_{s,q}^{\uparrow CR}$ but may differ in their current states (and hence control actions). If the state-feedback supervisor $\nu_{s,q}^{\uparrow CR}$ is re-enabled, we can use the same approach:

at all times, the control action is defined as the intersection of the control actions of all possible current states of $\nu_{s,q}^{\uparrow CR}$.

The previous discussion implies that the control action of the solution to MOES$^{\infty}$ is the intersection of the control actions of all enabled supervisors as indicated by their possible multiple current states. To keep track of the enabled state-feedback supervisors along with their current states, we define a binary *state-indicator vector* for each of the state-feedback supervisors in the bank. The size of this vector equals the number of states of the corresponding state-feedback supervisor and each of its elements corresponds to a state of the state-feedback supervisor. Once a supervisor is enabled, the first element of the state-indicator vector (corresponding to the initial state of this supervisor) becomes "1" (initially all elements are set to "0"). If, following the observation of an event $\alpha \in \Sigma_{obs}$, the supervisor state evolves from state i to state j, the j^{th} element in the indicator vector for the next state (initially taken to be the all zero vector) becomes "1". If this observation also requires that the state-feedback supervisor is re-enabled, i.e., if the associated current-state estimator state is revisited, we update the indicator vector of the next state as described above and also insert a "1" in the first element of the state-indicator vector. Upon a new observation, the state evolution of the supervisor is updated so that the states that are reachable from the current state(s) via the new observation have a "1" at the corresponding location of the state-indicator vector. The following algorithm formalizes this construction.

Algorithm C Consider a non-deterministic finite automaton $G = (X, \Sigma, \delta, X_0)$, with a set of observable events Σ_{obs} ($\Sigma_{obs} \subseteq \Sigma$), a set of controllable events Σ_c ($\Sigma_c \subseteq \Sigma_{obs}$), a set of secret states $S \subseteq X$, and a prefix-closed, normal and regular language $E \subseteq L(G)$, $E \neq \emptyset$, describing the legal behavior. Construct the non-deterministic finite automaton $G_E = (X_E, \Sigma, \delta_E, X_{0E})$ with $L(G_E) = E$, the non-deterministic finite automaton $G_p = G_E \times G = (X_p, \Sigma, \delta_p, X_{0p})$, the current-state estimator $G_{0,obs}^{(p)} = (X_{obs}^{(p)}, \Sigma_{obs}, \delta_{obs}^{(p)}, X_{0p})$ associated with G_p, and the bank of state-feedback supervisors $\nu_{s,q}^{\uparrow CR}$ associated with the bank of initial-state estimators $G_{\infty,obs}^{(p,q)} = (X_{\infty,obs}^{(p,q)}, \Sigma_{obs}, \delta_{\infty,obs}^{(p,q)}, X_{\infty,0}^{(p,q)})$, $0 \leq q \leq M-1$, where M denotes the number of states of the current-state estimator $G_{0,obs}^{(p)}$, as indicated in Algorithm B. Construct the state-indicator vector $\mathbf{I}_{\mathbf{q}}^{(\omega)}$ associated with each state-feedback supervisor $\nu_{s,q}^{\uparrow CR}$, $0 \leq q \leq M-1$, after observing the sequence of observations $\omega = \alpha_0\alpha_1 \ldots \alpha_n$ by recursively

applying the following for $i = 0$ to $i = n$.
(i) Initialize the state-indicator vector $\mathbf{I_q^{(\alpha_0\alpha_1\ldots\alpha_i)}} = \underline{\mathbf{0}}$, $0 \leq q \leq M-1$, where the size of the zero-vector $\underline{\mathbf{0}}$ equals the number of states of the automaton $\nu_{s,q}^{\uparrow CR}/G_{\infty,obs}^{(p,q)}$.
(ii) Let Z_q be the state in the current-state estimator $G_{0,obs}^{(p)}$ that is reached from its initial state X_{0p} via the string $\alpha_0 \ldots \alpha_i$. If $Z_q \cap S \neq \emptyset$, then enable the q^{th} state-feedback supervisor $\nu_{s,q}^{\uparrow CR}$ in the bank of state-feedback supervisors by setting the first element in its associated state-indicator vector $\mathbf{I_q^{(\alpha_0\alpha_1\ldots\alpha_i)}}$ to "1", i.e., set $\mathbf{I_q^{(\alpha_0\alpha_1\ldots\alpha_i)}}(0) = 1$.
(iii) If the state-feedback supervisor $\nu_{s,q}^{\uparrow CR}$ is (already enabled and is) at state j when the sequence of observations $\alpha_0\alpha_1 \ldots \alpha_{i-1}$ is observed (i.e., if $\mathbf{I_q^{(\alpha_0\alpha_1\ldots\alpha_{i-1})}}(j) = 1$), and if state j transitions to state j' in automaton $\nu_{s,q}^{\uparrow CR}/G_{\infty,obs}^{p,q}$ upon observing the last label α_i, then set entry j' of the state-indicator vector $\mathbf{I_q^{(\alpha_0\alpha_1\ldots\alpha_i)}}$ to "1", i.e., set $\mathbf{I_q^{(\alpha_0\alpha_1\ldots\alpha_i)}}(j') = 1$. ■

Definition 7.4.1. *Consider the state-feedback supervisor* $\nu_s : X_H \rightarrow \{\Sigma' \subseteq \Sigma | \Sigma_{uc} \subseteq \Sigma'\}$ *associated with finite automaton* $H = (X_H, \Sigma_{obs}, \delta_H, X_{0,H})$, *with set of states* $X_H = \{0, 1, \ldots, Q-1\}$ *and state-indicator vector* $\mathbf{I}$. *Define the control action of the state-feedback supervisor* ν_s *on the state-indicator vector* $\mathbf{I} = (i_0, \ldots, i_{Q-1})^T$ *for* $0 \leq q \leq Q-1$ *as* $\nu_s(\mathbf{I}) := \bigcap_{q:i_q=1} \nu_s(q)$. ■

Using Theorem 7.4.1, Algorithm C, and Definition 7.4.1, it is easy to see that the control action of the solution $\nu_{op}^{\uparrow CNP^\infty}$ to MOES$^\infty$ after observing the sequence of observations $\omega = \alpha_0\alpha_1 \ldots \alpha_n$ is $\nu_{op}^{\uparrow CNP^\infty}(\omega) = \bigcap_{q=0}^{M-1} \nu_{s,q}^{\uparrow CR}(\mathbf{I_q^{(\omega)}})$.

Example 7.4.1. *Consider the DES* G *in Figure 4.1-a with* $\Sigma_c = \{\beta\}$ *and* $E = L(G)$. *As mentioned in Example 5.3.1, this system is not* $(\{3\}, P, \infty)$*-opaque since observing* $\alpha\beta\alpha^*$ *reveals that secret state* 3 *was visited in the past. We follow Algorithm B to design a minimally restrictive supervisor which enforces infinite-step opacity. First, we need to construct the current-state estimator and the associated bank of ISEs. In Example 5.3.1, we carried this step as part of the verification process and argued that we only need to construct the two ISEs* $G_{\infty,obs}^{p,3}$ *and* $G_{\infty,obs}^{p,4}$ *associated with current-state estimator states* $Z_3 = \{2, 3, 4\}$ *and* $Z_4 = \{0, 1, 2, 3, 4\}$. *Next, following Algorithm B, we construct the state-feedback supervisor* $\nu_{s,3}^{\uparrow CR}$ ($\nu_{s,4}^{\uparrow CR}$) *which avoids the states*

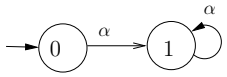

Figure 7.2: Supervisor $\nu_{s,3}^{\uparrow CR}$ defined in Algorithm B.

in ISE $G_{\infty,obs}^{p,3}$ ($G_{\infty,obs}^{p,4}$) for which the set of starting states of the associated state mapping contains no state outside the set of secret states $\{3\}$.

It can be easily verified that the sets of starting states of all state mappings associated with ISE $G_{\infty,obs}^{p,4}$ (depicted in Figure 4.1-b) have states outside the set of secret states $\{3\}$ and, hence, no supervision is required (i.e., we do not need to construct $\nu_{s,4}^{\uparrow CR}$). On the other hand, the set of starting states of state mapping $m_2 = \{(3,4)\}$ associated with ISE $G_{\infty,obs}^{p,3}$ (depicted in Figure 5.5) only contains state 3 and, hence, violates infinite-step opacity. The supervisor $\nu_{s,3}^{\uparrow CR}$ which avoids reaching that state is depicted in Figure 7.2. As a result of all of these discussions, the bank of state-feedback supervisors only contains one supervisor $\nu_{s,3}^{\uparrow CR}$. The state-indicator vector $\mathbf{I_3}$ associated with the state-feedback supervisor $\nu_{s,3}^{\uparrow CR}$ is initialized to $\mathbf{I_3^\epsilon} = (0,0)^T$. Upon observing α, state Z_3 is reached in the current-state estimator from its initial state Z_4 and, hence, the state-feedback supervisor $\nu_{s,3}^{\uparrow CR}$ associated with state Z_3 becomes enabled, i.e., $\mathbf{I_3^\alpha} = (1,0)^T$. Upon activation of $\nu_{s,3}^{\uparrow CR}$, the controllable event β is disabled. After that, supervisor $\nu_{s,3}^{\uparrow CR}$ transitions to state 1, i.e., $\mathbf{I_3^{\alpha\alpha}} = (0,1)^T$, and does not limit the behavior of the system anymore. Also, since state Z_3 is not reachable in the current-state estimator again, supervisor $\nu_{s,3}^{\uparrow CR}$ will not be re-enabled in future. Hence, the net effect of this supervision is the removal of the sequence of observations $\alpha\beta\alpha^*$ from the set of observations that the system could generate. (Note that if at the very beginning β is observed, then no supervisor will ever be enabled.) ■

Remark 7.4.1. *In order to implement the solution to MOES$^\infty$ using Algorithm B, one needs to store (i) the bank of state-feedback supervisors $\nu_{s,q}^{\uparrow CR}$, $0 \leq q \leq M-1$, and (ii) the bank of state-indicator vectors $\mathbf{I_q}$, $0 \leq q \leq M-1$, associated with each state-feedback supervisor. Here, M is the number of states of the current-state estimator associated with DES G. If N is the number of states of G, then the bank of state-feedback supervisors has $O(2^N \times 2^{N^2})$ or, equivalently, $O(2^{N^2})$ state-space complexity (each state-feedback supervisor is constructed from an ISE-like automaton (which has $O(2^{N^2})$ state-space complexity) and there are at most $O(2^N)$ such state-feedback supervisors (as-*

sociated with each state of the current-state estimator)). Each state-indicator vector is a vector of 0s and 1s and contains at most 2^{N^2} elements (which is the maximum number of states of a state-feedback supervisor). Therefore, storing the bank of state-indicator vectors requires $O(2^N \times 2^{N^2})$ or, equivalently, $O(2^{N^2})$ bits. The exponential complexity of the solutions to MOES$^\infty$ is not surprising since algorithms that implement minimally restrictive supervisors for non-deterministic automata with partial observations (e.g., to enforce diagnosability [46]) typically have similar complexity [46]. ■

Remark 7.4.2. *For enforcing K-step opacity, one can follow a similar approach for the case of initial-state opacity to show that the minimally restrictive supervisor that enforces K-step opacity can be constructed using the K-delay state estimator. For this, one needs to extend the state-based notion of K-step opacity to languages and then follow a similar procedure as in Section 7.3 to establish that the set of K-step opaque languages has supremal element. Due to the similarity of the materials, we will not include the details here.* ■

CHAPTER 8

APPLICATION EXAMPLE: TRACKING OF MOBILE AGENTS IN SENSOR NETWORKS

As mentioned in earlier chapters, there are many areas where infinite-step opacity (or more generally, state-based notions of opacity) can be used to characterize security requirements of interest. In Chapter 3, we studied conditions which may cause the seed (i.e., the initial state) or the current state of a pseudo-random generator in a cryptographic protocol to be compromised, and showed that this problem can be formulated and analyzed using a state-based opacity framework. We also demonstrated how coverage properties of mobile agents in sensor networks can be studied using state-based notions of opacity.

In this chapter, we show how the theoretical developments of Chapters 4 and 5 can be applied to analyze tracking problems in some representative sensor networks.

8.1 Introduction

Modern tactical intelligence, surveillance and reconnaissance (ISR) technologies include a vast assortment of air and ground based radars, unmanned autonomous vehicles, unattended ground-based sensors (UGS) and human intelligence observations. These sensing platforms produce streams of events based on what they detect within their physical range. The resulting event stream must then be combined with the physical/logical limitations of the environment in order to identify and track vehicles, aircraft, and/or people. As a result, the problem of tracking in sensor networks has received considerable attention [47–49].

In this section, we study the related problem of security and privacy in *planar* sensor networks. More specifically, we study the problem of tracking the trajectory of the location of a given vehicle (how it positions itself as time

goes on) with respect to a certain set of strategic (secret) locations, using information from a set of sensors that are deployed in a given planar region. These trajectories can be of interest for a variety of reasons. For example, feasible trajectories can be exploited in order to hide the origin of a trajectory from an observer who is employing the sensor network trying to identify whether the vehicle originated from a strategically important location or determine whether the vehicle passed from this particular set of strategic locations at some specific instant of time. It is assumed that the locations of the deployed sensors are known and remain constant, but the sensor coverages are allowed to differ (i.e., *heterogenous* sensor networks are considered).

In order to study this problem, we use the notion of infinite-step opacity introduced in Chapter 3.3. Recall that this notion is defined assuming that the underlying system is a discrete event system (DES) that can be modeled as a non-deterministic finite automaton with partial observation on its transitions. As shown later in this chapter, non-deterministic finite automata with partial observation on their transitions can be used to conveniently model the movement of mobile agents in various terrains; thus, state-based opacity notions (including infinite-step opacity) are ideally suited for coverage analysis and verification of security/privacy properties of interest. By defining the set of secret states to be the set of strategic locations, infinite-step opacity translates to the ability of the intruder to determine, given all available (past and current) sensor readings, whether the vehicle has gone through some strategic locations at a specific instant in time.

Due to the cost associated with sensor deployment, designers are typically faced with limitations on the number (or location) of deployed sensors. *Minimal sensor selection problems* aim at finding the set of sensors such that: (i) properties of interest about the sensor network hold, and (ii) if any of the sensors in the set is turned off, the property ceases to hold [50]. This chapter studies sensor selection problems in which the desired property is taken to be the ability of the sensor network to identify the passage of the vehicle through strategic locations; thus, the minimal sensor selection problem translates to the problem of finding a set of sensors (from a given set of available sensors) such that turning off any of the sensors in this set renders the system infinite-step opaque. In this chapter, we also propose a top-down algorithm to solve the minimal sensor selection problem and establish the correctness of the algorithm by showing that lack of infinite-step opacity is

a *mask-monotonic* property [51]. The effectiveness and applicability of this algorithm are subsequently evaluated via an extensive set of simulations.

8.1.1 Related Work

The framework considered here for tracking analysis is related to the *weak model* introduced in [47] for studying the *trackability* of sensor networks. A sensor network is trackable if the rate of growth of the number of state sequences that are consistent with a given sequence of observations is *sub-exponential* in the length of the observations. The authors of [47] show that the rate of growth of the number of consistent state sequences is either polynomial or exponential in the size of the observations; they also obtain necessary and sufficient conditions on the placement of the sensors such that this growth is polynomial, i.e., such that the sensor network is trackable. While the model in this chapter is similar to that of [47], this thesis studies a different problem: trackability studies the *number* of state sequences that are consistent with the given sequence of observations, while infinite-step opacity studies the state estimates that appear in these state sequences (and whether they fall exclusively in the set of secret states at specific points in time).

The authors of [49] study the problem of *multiple-target* tracking for sensor networks with deterministic deployment subject to communication delays, noisy observations, false alarms, and data packet losses. Assuming that the noisy observation of the state of the object is measured with a known detection probability (less than one) and that the number of false alarms follows a *Poisson* distribution, the authors of [49] use the Markov Chain Monte Carlo Association algorithm to estimate the number of moving objects and their state trajectories. Unlike the framework of [49], the formulation study in this chapter does not assume probabilities associated with the target movements. Furthermore, the focus is on characterizing the target trajectories *with respect to a set of secret locations*.

The sensor selection problem studied here is related to the *coverage* problem in sensor networks which has been studied with respect to different objectives and metrics (see, for example, [52–54]). The characteristic attribute used to classify different approaches to the coverage problem is whether sensor deployment is deterministic or stochastic. In deterministic sensor de-

ployment, the possible locations of the sensors are preselected, whereas in stochastic sensor deployment, sensors are deployed according to a *known* probability distribution. The coverage problem in the deterministic case boils down to the problem of finding the optimal placement for sensors such that a target coverage objective is met [53, 54]. In the stochastic case, the coverage problem reduces to finding the number of sensors that must be deployed, given the sensor deployment distribution, in order for every point in the field of interest to be covered by at least k sensors with some target probability p [52]. The sensor selection problem studied here is related to the coverage problem with deterministic sensor deployment [53].

The problem of connected coverage has been studied by [53], where the authors provide a geometric analysis that relates coverage to connectivity, and also define the necessary conditions for a sensor network to be connected. The authors of [54] studied the problem of deterministic coverage under the additional constraint that each sensor must have at least k neighbors. In its simplest form (full event observation), the minimum sensor selection problem considered here can be seen as the study of the *connectivity* of the sensor network graph with respect to the set of secret states (strategic locations).

8.2 Sensor Selection Related Questions

Due to the cost associated with deploying sensors, designers are typically faced with limitations on the number (or location) of deployed sensors. *Minimum sensor selection problems* aim to find the minimum number of sensors (or, more generally, a set of sensors of minimum cost) such that certain properties about the sensor network hold [50]. These properties vary depending on the underlying application and include (among others) *observability, normality, diagnosability,* and *co-observability* [51,55]. In this chapter, by letting the desired property be the ability of the sensor network to identify the passage of the vehicle through some set of strategic cells S, the minimum sensor selection problem is defined as the problem of finding the set of sensors (from a given set of available sensors) that has the minimum possible cardinality and guarantees that the system is *not* infinite-step opaque with respect to S. This problem is defined formally below.

Definition 8.2.1 (Minimum Sensor Selection Problem)**.** *Given an $n_1 \times n_2$ 2-dimensional grid, the kinematic model H of the vehicle that moves in this grid, sensor coverage areas expressed in terms of aggregations of cells (for each sensor in the given set), and a set of strategic (secret) cells S, find a subset of the given set of sensors that has minimum cardinality and ensures that the system is* not *infinite-step opaque.* ■

Clearly, the solution to the minimum sensor selection problem can be obtained by searching through all sensor configurations and obtaining the set of sensors (of minimum cardinality) such that the system is not infinite-step opaque. The problem has exponential complexity in the number of available sensors (because there is an exponential number of sensor configurations).

Another version of sensor selection problem is the minimal sensor selection problem which aims at finding a *minimal* solution, i.e., a set of sensors that have the following properties: (i) if all sensors in the set are selected, the system is not infinite-step opaque, (ii) by turning off any one sensor in this set, the system becomes infinite-step opaque. This problem is defined formally below.

Definition 8.2.2 (Minimal Sensor Selection Problem)**.** *Given an $n_1 \times n_2$ 2-dimensional grid, the kinematic model H of the vehicle that moves on the grid, sensor coverage areas expressed in terms of aggregations of cells (for each sensor in the given set), and a set of strategic (secret) cells S, find a subset of of the given set of sensors which is minimal and ensures that the system is infinite-step opaque, i.e., find a set of sensors such that (i) when all sensors in the set are selected the system is not infinite-step opaque, and (ii) if any one of the sensors in this set is turned off, the system becomes infinite-step opaque.* ■

The authors of [51] study minimal solutions to the sensor selection problem for a general property P, and propose a *top-down* algorithm for finding a minimal solution with complexity polynomial in the number of available sensors and in the time needed to verify property P. They also argue that as long as property P is *mask-monotonic*, the top-down algorithm obtains a minimal solution. For the property P to be mask-monotonic the following needs to be true: if the given setting G satisfies P with set of deployed sensors I; then the given setting satisfies P with any other set of deployed sensors I' that includes I (i.e., $I \subseteq I'$).

In the sequel, we show that lack of infinite-step opacity is mask-monotonic. First, for simplicity it is assumed that in Definitions 8.2.1 and 8.2.2 sensor coverage areas are not overlapping. In this case, turning on a sensor α can be modeled as adding event α to the set of observable events Σ_{obs} (and removing it from the set of unobservable events). Assume that the system is not infinite-step opaque for a given set of deployed sensors I_1 denoted by projection map $P_{\Sigma_{o1}}$; then, there exists a string s that passes through the set of secret states such that no other string t with the same projection, $P_{\Sigma_{o1}}(s) = P_{\Sigma_{o1}}(t)$, passes through a non-secret state when s passes through the set of secret states. Next, one turns on more sensors, and denotes the set of deployed sensors by $I_2 \supseteq I_1$ and the corresponding projection map by $P_{\Sigma_{o2}}$. Since $\Sigma_{o1} \subseteq \Sigma_{o2}$,

$$P_{\Sigma_{o2}}(s) = P_{\Sigma_{o2}}(t) \Rightarrow P_{\Sigma_{o1}}(s) = P_{\Sigma_{o1}}(t),$$

which implies that strings s and t with $P_{\Sigma_{o2}}(s) = P_{\Sigma_{o2}}(t)$ will also satisfy $P_{\Sigma_{o1}}(s) = P_{\Sigma_{o1}}(t)$; thus, string t has to pass through secret states when string s passes through secret states even under mapping $P_{\Sigma_{o2}}$. This implies that lack of infinite-step opacity is mask monotonic as long as sensor coverage is not overlapping.

Next, we consider the case when sensors are overlapping. Recall that the label of transitions ending in cells which are covered by more than one sensor is chosen to be a special label that indicates the set of all sensors covering that cell. Therefore, if the set of sensors that cover a cell changes — due to turning on more sensors — then the associated label with transitions ending in that cell will also change. However, if the transitions that end in a specific cell have identical labels before turning on a sensor, they will also have identical labels after turning on that sensor (although the associated label may now be different). Thus if s and t are such that $P_{\Sigma_{o2}}(s) = P_{\Sigma_{o2}}(t)$, then it also holds that $P_{\Sigma_{o1}}(s) = P_{\Sigma_{o1}}(t)$. The rest of the proof is similar to the previous case.

The above discussion implies that lack of infinite-step opacity is mask monotonic and one can use the top-down algorithm proposed in [51] to find the minimal solution to the sensor selection problem in Definition 8.2.1. The top-down algorithm works as follows: one starts by selecting all sensors and switches sensors off one at a time (in some arbitrary order), until the system

becomes infinite-step opaque. The last sensor is kept so that the resulting set of selected sensors ensures that the system is not infinite-step opaque and, from this set of sensors, one tries to find a sensor to turn off such that the system does not become infinite-step opaque (e.g., by going through the sensors in some predefined order). If this is possible, that sensor is turned off and one continues until no sensor can be turned off, obtaining in this way a minimal solution to the sensor selection problem in Definition 8.2.1.

It should be noted that there may exist more than one minimal solution and that the minimal solution does not necessarily correspond to the set of sensors that has minimum cardinality. This also explains why obtaining a minimal solution is generally easier than obtaining the minimum solution.

8.3 Simulation Studies

8.3.1 Generating Kinematic Models and Sensor Coverages

To study the verification process of infinite-step opacity in tracking problems in the context of sensor networks, one first needs to obtain a system model. We use the approach described in Section 3.4.2 and model the terrain that the vehicle is moving on as a grid of $n \times n$ cells, which are identified (for different values of n) as follows:

$$\begin{bmatrix} 1 & 2 & \dots & n \\ n+1 & n+2 & \dots & 2n \\ \vdots & \vdots & \ddots & \vdots \\ (n-1)n+1 & (n-1)n+2 & \dots & n^2 \end{bmatrix}.$$

This implies that the kinematic model H (which describes the limitations on the vehicle movements) has n^2 states. It is assumed that the vehicle in each cell can only move to neighboring cells so that from each state i ($1 \leq i \leq n^2$) of the kinematic model H there are at most[1] 9 possible transitions: $i \to i-n-1$, $i \to i-n$, $i \to i-n+1$, $i \to i-1$, $i \to i$, $i \to i+1$, $i \to i+n-1$, $i \to i+n$, and $i \to i+n+1$.

In order to show the applicability of the results to typical kinematic models

[1]Of course, states on the edges of the grid have fewer possible transitions.

on an $n \times n$ grid, the possible transitions (movements) between neighboring cells are chosen randomly. This is accomplished by creating for each state i, $1 \leq i \leq n^2$, a random binary vector of size 9, each element of which corresponds to one of the 9 possible state transitions for state i. The index of the element is the index of the state transition (in the ordering described above) and the randomly obtained binary value indicates whether the given transition is possible (value 1) or not (value 0). `Matlab` is used to create this random binary vector (0 and 1 are chosen with equal probability) and the resulting information is stored in a text file (in a format appropriate for the `UMDES` Library tool [56] which is described in more detail later in this chapter).

It is also assumed that there are m sensors in the sensor network and different values of m are tried at the beginning of each simulation. The coverage of sensors is determined randomly by creating a rectangular coverage region for each sensor. More specifically, the top left and bottom right vertices of the coverage region are chosen with probability that is uniform in the given grid. Once sensor coverage areas are chosen as described above, labels are assigned to the transitions in the kinematic model H to obtain automaton G (as described in Section 3.4.2). Again, `Matlab` is used to create the random sensor coverage areas and store the resulting automaton G in a format appropriate for the `UMDES` library tool.

8.3.2 Using `UMDES` to Verify Infinite-Step Opacity

In order to implement the verification method for infinite-step opacity described in Section 5.3.2, the `UMDES` Library is used. `UMDES` is a library of C routines developed at the University of Michigan [56] for studying discrete event systems modelled as finite automata. The routines currently available in the `UMDES` Library implement many of the basic operations needed for fault diagnosis [16] and supervisory control [28] for discrete event systems. The `UMDES` Library also includes basic functions pertaining to the manipulation and composition of finite automata.

The routines in the `UMDES` Library have a command-line interface. Inputs (i.e., finite automata) to these routines are either command-lines or text files formatted appropriately. In particular, an automaton with N states is stored

as a text file with extension `.fsm` and has a header (the first line of the file) and exactly N paragraphs. The header denotes the number of states (N) and the i^{th} paragraph, $1 \leq i \leq N$, describes the identifier of the i^{th} state followed by the transitions defined out of that state. Each transition is associated with a label and letter "o" if observable and "uo" if unobservable (the letter associated with each label is assumed to be consistent throughout the text file).

The first step in verifying infinite-step opacity (as described in Theorem 5.3.3) requires constructing the current-state estimator. The `UMDES` Library provides a routine for creating the current-state estimator for a given automaton. Note that this routine is developed assuming that the given automaton has one (unique) initial state but one can overcome this limitation by adding a dummy state x to G and by connecting it to all states in X_0 using a dummy unobservable transition. It is not hard to see that the current-state estimator $\hat{G}_{0,obs}$ constructed for this new automaton $\hat{G}$ (with state x as its unique initial state) is the same as the current-state estimator $G_{0,obs}$ constructed for automaton G with X_0 as its initial state (except that the initial state of $\hat{G}_{0,obs}$ has the extra element x in its initial state estimate compared to the initial state estimate of $G_{0,obs}$). Another issue that arises when using the `UMDES` Library routine for constructing the current-state estimator is the output of this routine. As part of the verification process, the set of states of the current-state estimator needs to be parsed from the file describing it. As will be explained later, `Matlab` is used to call all `UMDES` Library routines and to manipulate their output. The output of the `UMDES` Library routine for current-state estimation (is a text file that) has some extra information which makes parsing the current-state estimator states impossible using the routines provided in `Matlab` for reading text (e.g., `textscan`). In order to solve this problem, one can use Linux capabilities to manipulate text files and discard the information that is not necessary, so that the `textscan` routine can be used to parse the information regarding the current-state estimator states from this modified file.

The next step in verifying infinite-step opacity (as described in Theorem 5.3.3) requires constructing ISEs with their initial state set to $\odot_2(Z_0)$ for each state Z_0 of the current-state estimator which contains a state in the set of secret states S. As there is no support for constructing ISEs in the `UMDES` Library, we employ a transformation that enables us to use the routine for

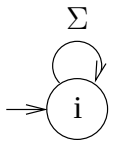

Figure 8.1: Automaton H_i.

current-state estimation in order to construct ISEs. Specifically, to construct the ISE associated with $G = (X, \Sigma, \delta, X_0)$, we perform the following steps:

(i) For each $i \in X_0$, one constructs $G_i = (\{i\} \times X, \Sigma, \delta_i, \{i\} \times X_0)$ where for $j \in X, \alpha \in \Sigma$: $\delta_i((i,j),\alpha) \equiv (i, \delta(j,\alpha))$. In words, G_i is the automaton that is obtained by annotating each state of G with label i and assuming that the automaton starts from state i.

(ii) The automaton $\hat{G} = (\bigcup_{i \in X_0} X_i, \Sigma, \hat{\delta}, \odot_2(X_0))$ is constructed, with X_i, $i \in X_0$, denoting the set of states of G_i and $\hat{\delta}((i,j),\alpha) = \delta_i((i,j),\alpha)$. In words, $\hat{G}$ is obtained by taking the *union* of all G_i, $i \in X_0$, and setting its set of initial states equal to $\odot_2(X_0)$.

(iii) By constructing the current-state estimator for $\hat{G}$, one can obtain the ISE for G as described in Chapter 4.

Remark 8.3.1. *In Step (i), in order to annotate automaton G, one uses the product routine provided in the* `UMDES` *Library and the auxiliary automaton H_i depicted in Figure 8.1.*

In Step (ii), one cannot use the union routine provided in the `UMDES` *Library to obtain $\hat{G}$. The problem is that the union routine provided by the* `UMDES` *Library acts on two non-deterministic automata G_1 and G_2, and returns the deterministic automaton G' that is the determinization of the non-deterministic automaton $G_1 \cup G_2$ (constructed using the subset construction [19]). This implies that the output G' of the union operator and $G_1 \cup G_2$ can have different sets of states. In Step (ii) of the transformation described above, however, the set of states (and non-determinism) of the automaton from the union operator needs to be intact; therefore the union routine provided in the* `UMDES` *Library cannot be used here. This issue is solved by using Linux capabilities to manipulate files. As mentioned before, each G_i is stored as a text file, e.g., as* `Gi.fsm`*. In order to construct the union $\cup G_i$ without modifying the set of states, the header from each file* `Gi.fsm` *(corresponding to each G_i, $i \in X_0$) is removed, and then all of these files are merged into one file with its header (i.e., number of states) equal to $|X_0| \times N$ (since the*

number of automata of the form G_i that are merged is $|X_0|$, and each G_i has N states). The automaton corresponding to this file is $\hat{G}$ in Step (ii) of the above transformation. ■

The above transformation obtains the ISE by constructing the current-state estimator for an annotated system model but, compared to the method described in the previous section (which constructs the ISE using state mappings), it is not as efficient in terms of memory and/or computational time. One obvious reason for this is that many states in $\hat{G}$ that are not reachable from its set of initial-states are nevertheless constructed and stored as part of the algorithm. This issue was ignored for the sake of using existing routines from the `UMDES` Library. The reader, however, should keep in mind that the effect of this inefficiency on the time required for simulation is large. Therefore, when looking at the reported times required for running the simulations, one needs to keep in mind that these time measurements are not indicative of the actual time required for verifying infinite-step opacity (they are longer than the actual time); instead, these timing measurements should be used as a relative measure of difference between simulations with different parameters.

The last step in verifying infinite-step opacity (as described in Theorem 5.3.3) requires checking whether the set of starting states of any of the state mappings in the constructed ISEs lies entirely within the set of secret states. This checking is accomplished and a wrapper for the program is obtained using `Matlab` code. The code first creates a random kinematic model with associated sensor coverage as described in Section 8.3.1. The result is stored in a text file readable by `UMDES` Library routines as described in this section. Then, the current-state estimator and the required initial-state estimators based on the transformation procedure described in this section are constructed. Note that `UMDES` Library routines can be called within `Matlab` using the `system` routine.

8.3.3 Simulations Results

The first part of the simulation (small example) studies the effect that the number of deployed sensors has on infinite-step opacity and hence indirectly solves the (minimal) sensor selection problem. It is assumed that the vehicle

is moving on a 6×6 grid and that $S = \{5, 16, 27\}$ and $X_0 = X$. The kinematic model used throughout this part of the simulation is depicted in Figure 8.2. It is assumed that up to 7 sensors are available for deployment, with coverage as depicted in Figure 8.2. One starts assuming that all sensors are selected. At each step, sensors are turned off one at a time, and one checks whether the system becomes infinite-step opaque. Sensors are turned off in the same order as their identifier with the largest one (i.e., 7) turned off first. The results are summarized in Table 8.1: after turning off sensors 7, 6, and 5, turning off any additional sensor violates infinite-step opacity; therefore, the solution to the minimal sensor selection problem is the set of sensors $\{1, 2, 3, 4\}$. As can be observed, the algorithm runs relatively fast.

The solution to the minimum sensor selection problem was also studied via an exhaustive search over all possible sensor configurations. For this part, one starts by turning off all sensors. The proposed algorithm first enumerates all 7 subsets of set $I = \{1, 2, 3, 4, 5, 6, 7\}$ of size 1 using the function `nchoosek` in `Matlab`. For each such subset,[2] the corresponding sensor is turned on and one verifies whether the system is infinite-step opaque. If the system is not infinite-step opaque, the algorithm stops and reports the chosen sensor as the solution to minimum sensor selection problem. Otherwise, that sensor is turned off and the algorithm proceeds with the next subset. Once each sensor corresponding to a sensor subset of size 1 is turned on and the system is verified to be infinite-step opaque, the algorithm proceeds by enumerating all subsets of set I of size 2, checking whether the selected subset makes the system infinite-step opaque. As soon as the algorithm finds a sensor configuration for which the system is not infinite-step opaque, the algorithm stops. The proposed algorithm stops after enumerating the set $\{1, 2, 3, 4\}$. This implies that the solution to the minimal sensor selection obtained previously is also a solution to the minimum sensor selection problem. In the second part of the simulation, one studies the minimal solution to the sensor selection problem in Definition 8.2.1 for randomly generated grids with randomly selected sensor coverage. For this, one randomly creates 100 grids of size 6×6 as described in Section 8.3.1. It is also assumed that up to 7 sensors can be deployed with sensor coverage chosen randomly as described in Section 8.3.1. Table 8.2 describes the cardinality of the solution to the minimal sensor se-

[2]The order follows the order of the resulting vector from function `nchoosek`.

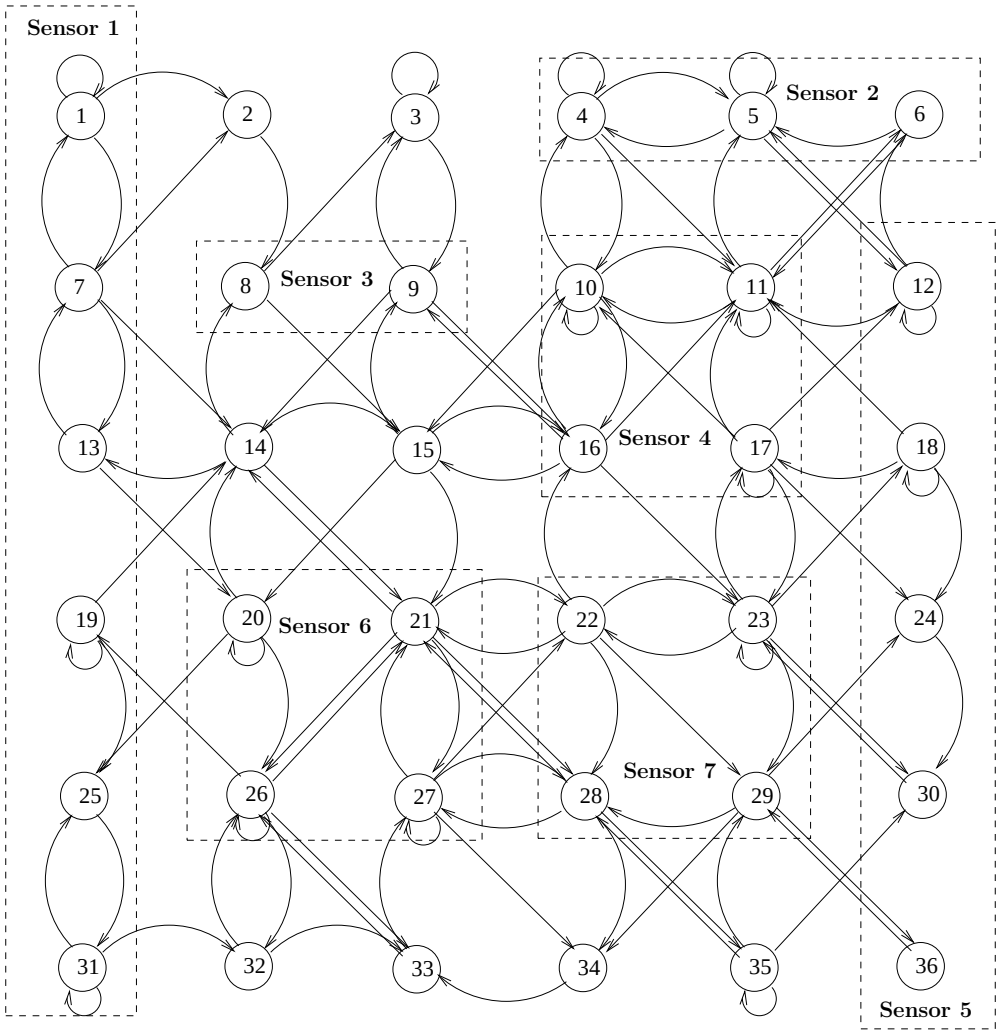

Figure 8.2: Example of a 6×6 sensor grid.

Table 8.1: Infinite-step opacity for the 6×6 grid in Figure 8.2. $|X_{obs}|$ denotes the number of current-state estimator states and $\max |X^n_{\infty,obs}|$ denotes the maximum number of ISE's states. Here $i : j$ means sensors $i, i+1, \ldots, j$ are deployed.

Deployed sensors	1:7	1:6	1:5	1:4	1,2,3	1,2,4	1,3,4	2,3,4
$\lvert X_{obs}\rvert$	53	37	17	11	6	10	7	8
$\max \lvert X^n_{\infty,obs}\rvert$	2451	915	322	193	64	64	64	64
Time (Sec)	3.55	1.9	2.4	3.8	3.4	3.8	3.4	3.17
Infinite-Step Opaque?	No	No	No	No	Yes	Yes	Yes	Yes

Table 8.2: Solution to the minimal sensor selection problem for 100 randomly generated 6×6 grid with up to 7 sensors.

Cardinality of the minimal solution	1	2	3	4	5	6	7
Number of grids	0	0	72	20	2	5	0

Table 8.3: Number of ISE's states as a function of grid size.

Grid size	6×6	8×8	10×10
Number of current-state estimator states	48	44	80
Number of ISE	11	15	24
Maximum Number of ISE's states	2243	1935	5070
Time (Sec)	13	30.5	1048

lection problem in Definition 8.2.1 for these randomly generated grids. This table should be interpreted as follows: out of 100 grids, 72 needed 3 sensors to violate infinite-step opacity, 20 needed 4 sensors, and so forth. As can be seen from this table, on average 3 sensors suffice to violate infinite-step opaque.

The third part of the simulation studies the effect of the grid size on the number of states of the generated ISE's. For this, one doubled and tripled the size of the grid. More specifically, n= 6, 8, 10 were used, each time choosing S randomly such that it includes 3% of system states and also spreads over the grid. It was also assumed that $X_0 = X$. For each value of n, 5 sensors were considered with location and coverage chosen randomly (as described earlier). The results are summarized in Table 8.3. As the number of cells in the grid triples (from 36 to 100), the size of constructed ISE's also triples. Also observe that (not surprisingly) the algorithm takes a relatively long time for large size grids.

Remark 8.3.2. *In this chapter, we study the application of the notion of infinite-step opacity to coverage analysis of mobile agent trajectory and existing tools were employed and appropriate transformations were devised to verify infinite-step opacity using a current-state estimator and a bank of initial-state estimators. While in this chapter we focus on the notion of infinite-step opacity, other state-based notions of opacity introduced in Chapter 3 can be used to characterize other interesting security properties of sensor networks. For example, the notion of initial-state opacity (K-step opacity) can be employed to hide the fact that the vehicle has started (passed) from some partic-*

ular set of strategically important locations initially (within the past K sensor readings) from an observer who is employing the sensor network. The verification technique that was introduced in this chapter can easily be extended for verifying initial-state opacity. For verifying K-step opacity, a Java toolbox is now available in [57]. ■

CHAPTER 9

CONCLUSION AND FUTURE WORK

9.1 Conclusion

In this thesis, motivated by a variety of security applications, we follow a state-based approach to define, analyze, and verify three notions of opacity:

(i) **Initial-state opacity**: For a system to be (S, P, ∞) initial-state opaque, the membership of the initial state of the system to the set of secret states S needs to remain opaque (for the whole length of the observation) to an outside observer who is observing the system behavior through a static natural projection map P.

(ii) **K-step opacity**: For a system to be (S, P, K)-opaque, for $K \geq 0$, the entrance of the system to the set of secret states S, at any time during the past K observations, should remain opaque to outsiders.

(iii) **Infinite-step opacity**: For a system to be (S, P, ∞)-opaque, the entrance of the system to the set of secret states S, at any time during the observations, should remain opaque to outsiders. Infinite-step opacity can be considered as the limiting case of K-step opacity as K approaches infinity.

While in these definitions, we assumed that the projection map is a static natural projection map of the form $P : \Sigma \to \Sigma_{obs}$; more general projections of the form $P : \Sigma \to \Delta \cup \{\epsilon\}$ that may map multiple events to a label in the set $\Delta \cup \{\epsilon\}$ can also be handled in a straightforward manner. To keep notation simple, in this thesis, we only discuss the natural projection.

To verify initial-state opacity, we construct an initial-state estimator which provides initial-state estimates. We show that for a system to be initial-state opaque, all initial-state estimates (associated with states of this estimator) need to contain at least one state outside the secret set S. We also investigate the complexity of the verification method and show that it can be largely reduced if the set of secret states S is fixed. Extensions to scenarios where

the set of secret states is time-varying are also discussed. We also establish that the verification of initial-step opacity is a PSPACE-complete problem.

To verify K-step opacity, we introduce the K-delay state estimator which provides K-delayed state estimates. These are the estimates of the state of the system k observations ago, $0 \leq k \leq K$, and are consistent with all observations so far (including the last K observations). We show that for a system to be K-step opaque, all k-delayed state estimates, $0 \leq k \leq K$ (associated with states of the K-delay state estimator), need to contain at least one state outside the secret set S. We also investigate the complexity of the verification method and establish that the verification of K-step opacity is an NP-hard problem.

To verify infinite-step opacity, we show that for any $K \geq 2^{N^2} - 1$ (where N is the number of states of the discrete event system), K-step opacity and infinite-step opacity become equivalent; hence, the verification method for K-step opacity can be applied for verifying infinite-step opacity with $K = 2^{N^2} - 1$. We also introduce a different method to verify infinite-step opacity using the current-state estimator and a bank of initial-state estimators; this verifier method has considerably lower space and time complexity compared to the method that constructs a K-delay state estimator for $K = 2^{N^2} - 1$. We also establish that the verification of infinite-step opacity is a PSPACE-hard problem.

To address the problem of designing feasible supervisors that enforce opacity, we formulate the problem as a supervisory control problem that enforces opacity while limiting the behavior of the system to a subset behavior, called legal behavior and described by a prefix-closed language E. We show that there always exists a solution to this problem and characterize the set of solutions as the set of sublanguages of E that are controllable, observable, and initial-state opaque. We show that there always exists a minimally restrictive solution to this problem and under the assumption that $\Sigma_c \subseteq \Sigma_{obs}$, we propose a method to find the supremal language among such languages (which is the solution of our minimally restrictive supervisory control problem).

9.2 Future Work

There are many interesting future directions. In the near future, we plan to investigate the following.

1. Extension to modular settings: In this setting, the given system is modeled as a *composition* (synchronous product) of M modules $\{G_1, G_2, \ldots, G_M\}$ where each module G_i is modelled as a non-deterministic finite automaton with N_i states where the set of secret states S is of the form $S = \{(x_1, x_2, \ldots, x_M) | x_i \in S_i\}$ for set of secret states S_i for each module G_i. Extensions of the definition of state-based notions of opacity to such systems is a topic of ongoing research.

2. Introducing probabilistic metrics to this framework: Initial-state opacity does not consider the likelihood of sequences of observations that violate the initial-state opacity requirement (instead it simply reports whether a given system is opaque or not). In addition, it does not attempt to characterize the confidence of the intruder when initial-state opacity is not violated (e.g., the probability that the system initial state belongs to the set S). This can limit the appropriateness of the notion of initial-state opacity in applications where the *confidence* of the intruder can serve as a measure of security. An example of an application where such confidence concerns have been considered is anonymity protocols [38, 58–60]. Such systems consist of a set of users whose known actions generate associated outputs that are observed by intruders who then try to infer the identity of the originator of the action. The goal of any anonymity protocol is to hide the origin (user) for certain actions in the system despite the observed outputs. Depending on the kind of protection offered by an anonymity protocol, a probabilistic notion of initial-state opacity can be more appropriate for describing the security requirements.

   Introducing probabilistic metrics to our framework can be achieved as follows: consider a scenario where we are given a stochastic discrete event system (SDES) that can be modeled as a probabilistic finite automaton (PFA) with partial observation on its transitions; assuming that the initial-state distribution vector is known, we can define two notions of opacity: *almost initial-state opacity* and *probabilistic initial-*

state opacity. The former notion requires that the probability of behavior that violates initial-state opacity lies below a threshold; probabilistic initial-state opacity, on the other hand, considers the probability that the system initial-state lies in the set of secret states (and hence, the confidence of the intruder) given any possible observation in the system, and requires that this probability lies below a threshold for all possible behavior in the system. Such probabilistic extensions are currently the topic of ongoing research.

3. Another extension is to connect the design of control policies for stochastic systems under suitable optimality criteria for probabilistic opacity.

4. Finally, we are interested in studying tracking problems involving two (or more) mobile agents with possibly different kinematic models or kinematic models with common patterns (in case where the mobile agents share group formations) [48,61,62]. In this case, sensor readings can be triggered by any of the vehicles, which adds uncertainty to the problem and can potentially be handled by using projection mappings more general than natural projection. The study of this problem and variations of it, along with potential applications of modular verification techniques [63] will be part of future work.

REFERENCES

[1] R. Focardi and R. Gorrieri, "A taxonomy of trace–based security properties for CCS," in *Proc. of the 7th Workshop on Computer Security Foundations*, June 1994, pp. 126–136.

[2] S. Schneider and A. Sidiropoulos, "CSP and anonymity," in *Proc. of the 4th European Symposium on Research in Computer Security*, September 1996, pp. 198–218.

[3] J. Bryans, M. Koutny, L. Mazare, and P. Ryan, "Opacity generalised to transition systems," *International Journal of Information Security*, vol. 7, no. 6, pp. 421–435, November 2008.

[4] A. Saboori and C. N. Hadjicostis, "Notions of security and opacity in discrete event systems," in *Proc. of the 46th IEEE Conference on Decision and Control*, December 2007, pp. 5056–5061.

[5] A. Saboori and C. N. Hadjicostis, "Verification of initial–state opacity in security applications of DES," in *Proc. of the 9th International Workshop on Discrete Event Systems*, May 2008, pp. 328–333.

[6] A. Saboori and C. N. Hadjicostis, "Verification of K-step opacity and analysis of its complexity," in *Proc. of the 48th IEEE Conference on Decision and Control*, December 2009, pp. 205–210.

[7] A. Saboori and C. N. Hadjicostis, "Verification of infinite-step opacity and analysis of its complexity," in *Proc. of the 2009 Workshop on Dependable Control of Discrete Systems*, June 2009, pp. 51–56.

[8] M. Bishop, *Computer Security: Art and Science*. Boston, MA: Addison Wesley Professional, 2003.

[9] S. Bose, A. Patra, and S. Mukhopadhyay, "On observability with delay: Antitheses and syntheses," *IEEE Transactions on Automatic Control*, vol. 39, no. 4, pp. 803–806, April 1994.

[10] P. E. Caines, R. Greiner, and S. Wang, "Dynamical logic observers for finite automata," in *Proc. of 27th IEEE Conference on Decision and Control*, vol. 1, December 1988, pp. 226–233.

[11] S. Hashtrudi Zad, R. H. Kwong, and W. M. Wonham, "Fault diagnosis in discrete event systems: Framework and model reduction," *IEEE Transactions on Automatic Control*, vol. 48, no. 7, pp. 1199–1212, July 2003.

[12] F. Lin and W. Wonham, "On observability of discrete event systems," *Information Sciences*, vol. 44, no. 3, pp. 173–198, April 1988.

[13] M. Oishi, I. Hwang, and C. Tomlin, "Immediate observability of discrete event systems with application to user–interface design," in *Proc. of 42nd IEEE Conference on Decision and Control*, vol. 3, December 2003, pp. 2665–2672.

[14] C. M. Özveren and A. S. Willsky, "Observability of discrete event dynamic systems," *IEEE Transactions on Automatic Control*, vol. 35, no. 7, pp. 797–806, July 1990.

[15] P. J. Ramadge and W. M. Wonham, "Supervisory control of a class of discrete event processes," *SIAM Journal on Control and Optimization*, vol. 25, pp. 206–230, January 1987.

[16] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. C. Teneketzis, "Failure diagnosis using discrete event models," *IEEE Transactions on Automatic Control*, vol. 4, no. 2, pp. 105–124, March 1996.

[17] C. M. Özveren, A. S. Willsky, and P. J. Antsaklis, "Stability and stabilizability of discrete event dynamic systems," *Journal of the Association for Computing Machinery*, vol. 38, no. 3, pp. 729–751, July 1991.

[18] C. M. Özveren and A. S. Willsky, "Invertibility of discrete event dynamic systems," *Mathematics of Control, Signals, and Systems*, vol. 5, no. 4, pp. 365–390, July 1992.

[19] C. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*. New York, NY: Kluwer Academic Publishers, 2008.

[20] J. W. Bryans, M. Koutny, and P. Y. A. Ryan, "Modelling opacity using Petri nets," *Electronic Notes in Theoretical Computer Science*, vol. 121, pp. 101–115, February 2005.

[21] J. Bryans, M. Koutny, and P. Ryan, "Modelling dynamic opacity using Petri nets with silent actions," in *Formal Aspects in Security and Trust*. New York, NY: Springer, 2005, vol. 173, pp. 159–172.

[22] E. Badouel, M. Bednarczyk, A. Borzyszkowski, B. Caillaud, and P. Darondeau, "Concurrent secrets," *Discrete Event Dynamic Systems*, vol. 17, no. 4, pp. 425–446, December 2007.

[23] J. Dubreil, P. Darondeau, and H. Marchand, "Opacity enforcing control synthesis," in *Proc. of the 9th International Workshop on Discrete Event Systems*, May 2008, pp. 28–35.

[24] N. Hadj-Alouane, S. Lafrance, L. Feng, J. Mullins, and M. Yeddes, "On the verification of intransitive noninterference in multilevel security," *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)*, vol. 35, no. 5, pp. 948–958, October 2005.

[25] F. Cassez, J. Mullins, and O. Roux, "Synthesis of non-interferent distributed systems," *Computer Network Security*, vol. 1, pp. 159–170, August 2007.

[26] D. Thorsley and D. Teneketzis, "Intrusion detection in controlled discrete event systems," in *Proc. of the 45th IEEE Conference on Decision and Control*, December 2006, pp. 6047–6054.

[27] J. E. Hopcroft, R. Motwani, and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation.* Boston, MA: Addison Wesley, 2006.

[28] W. M. Wonham, "Supervisory control of discrete event systems," Systems and Control Group, Department of Electrical and Computer Engineering, University of Toronto, 2009. [Online]. Available: www.utoronto.ca/DES

[29] D. K. Pradhan and M. Chatterjee, "GLFSR — A new test pattern generator for built–in–self–test," *IEEE Transactions on Computer–Aided Design of Integrated Circuits and Systems*, vol. 18, no. 2, pp. 238–247, February 1999.

[30] S. Jiang, Z. Huang, V. Chandra, and R. Kumar, "A polynomial algorithm for testing diagnosability of discrete event systems," *IEEE Transactions on Automatic Control*, vol. 46, no. 8, pp. 1318–1321, 2001.

[31] T.-S. Yoo and S. Lafortune, "Polynomial–time verification of diagnosability of partially observed discrete event systems," *IEEE Transactions on Automatic Control*, vol. 47, no. 9, pp. 1491–1495, 2002.

[32] D. Lee and M. Yannakakis, "Testing finite state machines: State identification and verification," *IEEE Transactions on Computers*, vol. 43, no. 3, pp. 306–320, March 1994.

[33] D. Lee and M. Yannakakis, "Principles and methods of testing finite state machines — A survey," *Proceedings of the IEEE*, vol. 84, no. 8, pp. 1090–1123, August 1996.

[34] P. E. Caines, R. Greiner, and S. Wang, "Classical and logic–based dynamic observers for finite automata," *IMA Journal of Mathematical Control and Information*, vol. 8, no. 1, pp. 45–80, March 1991.

[35] K. Naik, "Efficient computation of unique input/output sequences in finite–state machines," *IEEE/ACM Transactions on Networking*, vol. 5, no. 4, pp. 585–599, August 1997.

[36] J. S. Meditch, "A survey of data smoothing for linear and nonlinear dynamic systems," *Automatica*, vol. 9, no. 2, pp. 151–162, March 1973.

[37] M. R. Garey and D. S. Johnson, *Computers and Intractability – A Guide to the Theory of NP-Completeness*. New York, NY: Freeman, 1979.

[38] D. Chaum, "The dining cryptographers problem: Unconditional sender and recipient untraceability," *Journal of Cryptology*, vol. 1, no. 1, pp. 65–75, March 1988.

[39] D. Bell and L. LaPadula, "Secure computer systems: Mathematical foundations," MITRE Corporation, Bedford, MA, Tech. Rep. MTR-2547, March 1973.

[40] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis, "Diagnosibility of discrete event systems," *IEEE Transactions on Control Systems Technology*, vol. 40, no. 9, pp. 1555–1575, September 1995.

[41] S. A. Cook, "The complexity of theorem-proving procedures," in *Proc. of the 3rd Annual ACM Symposium on Theory of Computing*, April 1971, pp. 151–158.

[42] J. Y. Kao, A. J. Malton, N. Rampersad, and J. Shallit, "On NFAs where all states are final, initial, or both," *CoRR*, vol. abs/0808.2417, 2008. [Online]. Available: http://arxiv.org/abs/0808.2417

[43] S. C. Kleene, "Representation of events in nerve nets and finite automata," in *Automata Studies*, C. E. Shannon and M. McCarthy, Eds. Princeton, NJ: Princeton University Press, 1956, no. 34, pp. 3–41.

[44] H. B. Hunt III, "On the time and tape complexity of languages, I," in *Proc. of the 5th Annual ACM Symposium on Theory of Computing*, April 1973, pp. 10–19.

[45] R. D. Brandt, V. Garg, R. Kumar, F. Lin, S. I. Marcus, and W. M. Wonham, "Formulas for calculating supremal controllable and normal sublanguages," *System and Control Letters*, vol. 15, no. 2, pp. 111–117, 1990.

[46] M. Sampath, S. Lafortune, and D. Teneketzis, "Active diagnosis of discrete event systems," *IEEE Transactions on Automatic Control*, vol. 43, no. 7, pp. 908–929, July 1998.

[47] V. Crespi, G. Cybenko, and G. Jiang, "The theory of trackability with applications to sensor networks," *ACM Transactions on Sensor Networks*, vol. 4, no. 3, pp. 1–42, May 2008.

[48] D. Reid, "An algorithm for tracking multiple targets," *IEEE Transactions on Automatic Control*, vol. 24, no. 6, pp. 843–854, 1979.

[49] S. Oh, L. Schenato, P. Chen, and S. Sastry, "A scalable real-time multiple-target tracking algorithm for sensor networks," University of California, Berkeley, Tech. Rep. UCB//ERL M05/9, February 2005.

[50] R. Debouk, S. Lafortune, and D. Teneketzis, "On an optimization problem in sensor selection," *Discrete Event Dynamic Systems: Theory and Application*, vol. 12, no. 4, pp. 417–445, October 2002.

[51] S. Jiang, R. Kumar, and H. E. Garcia, "Optimal sensor selection for discrete event systems with partial observation," *IEEE Transactions on Automatic Control*, vol. 48, no. 3, pp. 369–381, March 2003.

[52] L. Lazos and R. Poovendran, "Stochastic coverage in heterogeneous sensor networks," *ACM Transactions on Sensor Networks*, vol. 2, no. 3, pp. 325–358, August 2006.

[53] G. Xing, X. Wang, Y. Zhang, C. Lu, R. Pless, and C. Gill, "Integrated coverage and connectivity configuration for energy conservation in sensor networks," *ACM Transactions on Sensor Networks*, vol. 1, no. 1, pp. 36–72, August 2005.

[54] S. Poduri and G. S. Sukhatme, "Constrained coverage for mobile sensor networks," in *Proc. of IEEE International Conference on Robotics and Automation*, April 2004, pp. 165–172.

[55] T.-S. Yoo and S. Lafortune, "NP–completeness of sensor selection problems arising in partially observed discrete event systems," *IEEE Transactions on Automatic Control*, vol. 47, no. 9, pp. 1495–1499, September 2002.

[56] UMDES-LIB software library, 2009. [Online]. Available: http://www.eecs.umich.edu/umdes/toolboxes.html

[57] TAKOS: A Java toolbox for analyzing the K-opacity of systems, 2010. [Online]. Available: http://toolboxopacity.gforge.inria.fr/

[58] D. L. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," *Communications of the ACM*, vol. 24, no. 2, pp. 84–90, February 1981.

[59] M. K. Reiter and A. D. Rubin., "Crowds: Anonymity for web transactions," *ACM Transactions on Information and System Security*, vol. 1, no. 1, pp. 66–92, November 1998.

[60] S. Zhong, Z. Yang, and T. Chen, "k-Anonymous data collection," *Information Sciences*, vol. 179, no. 17, pp. 2948–2963, August 2009.

[61] Y. Bar-Shalom, T. Kirubarajan, and X.-R. Li, *Multitarget-Multisensor Tracking: Principles and Techniques*. Boston, MA: Yaakov Bar-Shalom Publishing, 1995.

[62] H. Yang and B. Sikdar, "A protocol for tracking mobile targets using sensor networks," in *Proc. of 1st IEEE Workshop on Sensor Network Protocols and Applications*, May 2003, pp. 71–81.

[63] A. Saboori and C. N. Hadjicostis, "Reduced-complexity verification for initial-state opacity in modular discrete event systems," in *Proc. of the 10th International Workshop on Discrete Event Systems (to appear)*, September 2010.