2020 American Control Conference
Denver, CO, USA, July 1-3, 2020

Supervisory Control of Discrete-Event Systems for Infinite-Step Opacity

Yifan Xie and Xiang Yin

Abstract— **In this paper, we investigate the problem of synthesizing maximally-permissive opacity-enforcing supervisors for partially-observed discrete-event systems. We focus on a specific type of opacity called *infinite-step opacity* requiring that the intruder, which is modeled as a passive observer, can never determine for sure that the system is/was at a secret state for any specific instant. When the original system is verified to be non-opaque, one is interested in *synthesizing* a supervisor that controls the system in a *least-restrictive* manner such that the closed-loop system is opaque. We provide an effective approach for solving the infinite-step opacity control problem by re-formulating this problem as a two-player game over a generalized information structure that involves information delays. We show that the proposed synthesis algorithm is sound and complete, and the resulting supervisor is maximally-permissive. Most of the existing works on opacity-enforcing supervisory control only consider current-state opacity in which no delayed information is involved. Our work provides a complete solution to the infinite-step opacity control problem without any assumption on the relationship between controllable events and observable events.**

I. Introduction

In this paper, we investigate a formal information-flow security property called *opacity* in the context of Discrete-Event Systems (DES). Roughly specking, a system is *opaque* if its "secret" can never be revealed to a malicious intruder. Essentially, opacity is a confidentiality property guaranteeing the plausible deniability for the secret of the system.

In the context of DES, opacity has been studied very extensively in the past few years; see, e.g., [4], [8], [12], [13], [15], [21], [26]. Especially, to characterize different types of security requirements, different notions of opacity are proposed in the literature. For example, current-state opacity [14] requires that the intruder should never know for sure that the system is currently at a secret state by utilizing the information available up to the current instant. In many situations, the intruder may further use future information to better infer the security status of the system for some previous instant. To capture this scenario, the notions of K-step opacity [7], [19] and infinite-step opacity [19], [24] are proposed. Particularly, infinite-step opacity is the strongest one among all notions of opacity mentioned above, which requires that the intruder can never know that the system is/was at a secret state for any specific instant even by using future information.

When a system is verified to be non-opaque, one important problem is to enforce opacity via some mechanisms. In the literature, several opacity enforcement mechanisms have been investigated, e.g., by supervisory control [1], [5], [6], [20], [22], [23], by dynamic masks [3], [25], [27] and by insertion functions [9]–[11]. In particular, supervisory control of opacity aims to find a *supervisor* that restricts the behavior of the system dynamically such that the closed-loop system is opaque. For example, [6] studies the supervisory control problem for current-state opacity assuming that all controllable events are observable and the intruder cannot observe more events than the supervisor. The work of [22] relaxes above assumptions but assumes that the intruder does not know the implementation of the supervisor. Note that all the above mentioned works on opacity-enforcing supervisor synthesis consider current-state opacity.

In this paper, we study the problem of synthesizing maximally-permissive supervisors for *infinite-step opacity*. This problem is significantly more challenging than the current-state opacity enforcement problem. Specifically, in the infinite-step setting, whether or not a secret can be revealed to the intruder not only depends on the information available currently, but also depends on the information in the future. To handle this future information, in the verification problem, we can look ahead in the original open-loop system. However, in the synthesis problem, the future information of the closed-loop system is *unknown* and depends on the control policy in the future which is *to be determined*. Hence, how to handle the dependence between the delayed information and the control policy is the main difficulty here. In this paper, we propose an effective approach that solves the infinite-step opacity control synthesis problem. In particular, we show that the proposed synthesis algorithm is sound and complete, and the resulting supervisor is maximally-permissive.

The approach in this paper essentially generalizes the two-player-game-based synthesis framework proposed in our previous work [23] to the case of delayed information. Similarly information state has also been used in our recent work for dynamic mask synthesis [25]. However, dynamic masks can only change the observation of the system, while supervisor will change the *actually behavior* of the system. Hence, different state updating rules are proposed here to handle the control problem. Finally, as we mentioned earlier, most of the existing works on opacity-enforcing supervisory control only consider current-state opacity. One exception is [18], where the authors propose a method to enforce infinite-step opacity by synthesizing *a set of supervisors* that run synchronously. However, our approach synthesizes a single

This work was supported by the National Natural Science Foundation of China (61803259, 61833012) and by Shanghai Jiao Tong University Scientific and Technological Innovation Funds.

Y. Xie and X. Yin are with Department of Automation and Key Laboratory of System Control and Information Processing, Shanghai Jiao Tong University, Shanghai 200240, China. E-mail: {xyfan1234,yinxiang}@sjtu.edu.cn.

978-1-5386-8266-1/$31.00 ©2020 AACC

supervisor directly. Moreover, the approach in [18] is based on the assumption that all controllable events are observable; this assumption is also relaxed in our approach.

II. Preliminary

A. System Model

We assume basic knowledge of DES and use common notations; see, e.g., [2]. A DES is modeled as a deterministic finite-state automaton $G = (X, \Sigma, \delta, x_0)$, where X is the finite set of states, Σ is the finite set of events, $\delta : X \times \Sigma \to X$ is the partial transition function, where $\delta(x, \sigma) = y$ means that there is a transition labeled by event σ from state x to y, and $x_0 \in X$ is the initial state. The transition function can also be extended to $\delta : X \times \Sigma^* \to X$ in the usual manner [2]. For simplicity, we write $\delta(x, s)$ as $\delta(s)$ when $x = x_0$. The language generated by G is defined by $\mathcal{L}(G) := \{s \in \Sigma^* : \delta(x_0, s)!\}$, where ! means "is defined".

When the system is partially observed, Σ is partitioned into two disjoint sets: $\Sigma = \Sigma_o \dot{\cup} \Sigma_{uo}$, where Σ_o is the set of observable events and Σ_{uo} is the set of unobservable events. The natural projection $P : \Sigma^* \to \Sigma_o^*$ is defined by

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

For any observation $\alpha\beta \in P(\mathcal{L}(G))$, we denote by $\hat{X}_G(\alpha|\alpha\beta)$ the *delayed state estimate* that captures the set of all possible states the system could be in at the instant when α is observed given the entire observation $\alpha\beta$, i.e.,

$$\hat{X}_G(\alpha|\alpha\beta) := \left\{ \delta(s) \in X : \begin{array}{c} \exists st \in \mathcal{L}(G) \text{ s.t.} \\ P(s) = \alpha \wedge P(st) = \alpha\beta \end{array} \right\}.$$

For simplicity, we define $\hat{X}_G(\alpha) := \hat{X}_G(\alpha|\alpha)$ as the current state estimate upon the occurrence of α.

B. Secret and Intruder

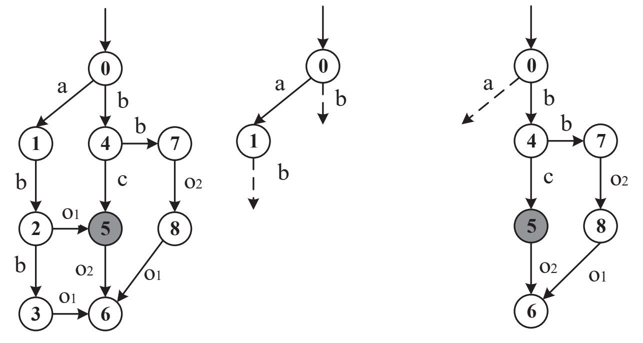
We assume that system G has a "secret" modeled as a set of secret states $X_S \subseteq X$. We consider an intruder modeled as a passive observer that may also observe the occurrences of observable events. Then the intruder can infer whether or not the system was/is at a secret state based on the information-flow available. If for any string that leads to a secret state, the intruder can never determine whether the system is/was at a secret state no matter what future information is generated, then we say that the system is *infinite-step opaque*, which is formally defined as follows.

Definition 1: Given system G, a set of observable events Σ_o and a set of secret states X_S, system G is said to be infinite-step opaque w.r.t. X_S and Σ_o if

$$\forall \alpha\beta \in P(\mathcal{L}(G)) : \hat{X}_G(\alpha|\alpha\beta) \not\subseteq X_S.$$

The following example illustrates infinite-step opacity.

Example 1: Let us consider system G in Fig.1(a) with $\Sigma_o = \{o_1, o_2\}$ and $X_S = \{5\}$. This system is actually current-state opacity. For example, for observation $o_1 \in P(\mathcal{L}(G))$, we have $\hat{X}_G(o_1) = \{5, 6\}$, i.e., when string o_1 is observed, the intruder cannot know for sure that the system is currently at secret state 5 or is at non-secret state 6. However, G is not infinite-step opaque w.r.t. X_S and Σ_o. To see this, we consider string abo_1o_2 with $P(abo_1o_2) = o_1o_2$ and the delay-state estimate is $\hat{X}_G(o_1|o_1o_2) = \{5\} \subseteq X_S$. This means that the system was at secret state 5 one step ago for sure when string o_1o_2 is observed. Upon the occurrence of o_1o_2, secret state 5 will be revealed to the intruder.



(a) System G (b) Solution G_1 (c) Solution G_2

Fig. 1: $\Sigma_o = \{o_1, o_2\}, \Sigma_c = \{a, b, c\}$ and $X_S = \{5\}$.

III. Problem Formulation

When a given system G is not infinite-step opaque, we need to enforce opacity on the system. One typical approach is to synthesize a *supervisor* that restricts the system's behavior to a sublanguage that satisfies the opacity requirement.

In the framework of supervisory control, a supervisor can control the transition function of G, which means it can dynamically enable/disable the occurrences of events. We assume that the events set is further partitioned as $\Sigma = \Sigma_c \dot{\cup} \Sigma_{uc}$, where Σ_c is the set of controllable events and Σ_{uc} is the set of uncontrollable events. Controllable events are events that can be prevented from happening, or disabled, by supervisor; uncontrollable events cannot be prevented from happening by supervisor. A control decision $\gamma \in 2^{\Sigma}$ is said to be admissable if $\Sigma_{uc} \subseteq \gamma$, namely uncontrollable events can never be disabled. We define $\Gamma = \{\gamma \in 2^{\Sigma} : \Sigma_{uc} \subseteq \gamma\}$ as the set of admissable control decisions. Since a supervisor can only make decisions based on its observation, a partial-observation supervisor is a function $S_P : P(\mathcal{L}(G)) \to \Gamma$. We use the notation S_P/G to represent the controlled system and the language generated by S_P/G, denoted by $\mathcal{L}(S_P/G)$, is defined recursively in the usual manner [2].

In this paper, we want to synthesize a supervisor that disables events dynamically such that the controlled system S_P/G is infinite-step opaque. Note that, when the implementation of the supervisor becomes a public information, the intruder may further know that some behavior in the original open-loop system is no longer feasible in the closed-loop system. Therefore, for any supervisor S_P and observable string $\alpha\beta \in P(\mathcal{L}(S_P/G))$, we define

$$\hat{X}_{S_P/G}(\alpha \mid \alpha\beta) := \left\{ \delta(s) \in X : \begin{array}{c} \exists st \in \mathcal{L}(S_P/G) \text{ s.t.} \\ P(s) = \alpha \wedge P(st) = \alpha\beta \end{array} \right\}$$

as the delayed state estimate in the closed-loop system. Similarly, we define $\hat{X}_{S_P/G}(\alpha) = \hat{X}_{S_P/G}(\alpha \mid \alpha)$. Then we say that the closed-loop system S_P/G is infinite-step opaque w.r.t. X_S and Σ_o if

$$\forall \alpha\beta \in P(\mathcal{L}(S_P/G)) : \hat{X}_{S_P/G}(\alpha \mid \alpha\beta) \not\subseteq X_S. \quad (1)$$

Clearly, the less events enabled, the system is more likely to be opaque. It is desired that we can enable as many events as possible. Therefore, our goal is to find a maximally permissive supervisor. Then the supervisory control problem for enforcing infinite-step opacity under partial observation is formulated as follows.

Problem 1: (*Infinite-Step Opacity Control Problem*) Given system G and secret states $X_S \subseteq X$, synthesize a partial-observation supervisor $S_P : P(\mathcal{L}(G)) \to \Gamma$, such that:

(1) S_P/G is infinite-step opaque w.r.t. X_S and Σ_o; and

(2) For any supervisor S'_P satisfying (1), we have $\mathcal{L}(S_P/G) \not\subset \mathcal{L}(S'_P/G)$.

The second condition implies that the synthesized supervisor is maximal in the sense that its permissiveness cannot be improved anymore. As we will see in the following example, such maximal solution is not unique in general.

Example 2: Again, we consider system G shown in Fig.1(a) and we assume that $\Sigma_c = \{a, b, c\}$ is the set of controllable events. Actually, we have two different ways to get an infinite-step opaque controlled system. If the supervisor disables event b initially, then we get an infinite-step opaque system G_1 shown in Fig.1(b). This system is clearly opaque as no secret state can be visited. Another approach is to disable event a initially and we can get a closed-loop system G_2 shown in Fig.1(c). This system is also infinite-step opaque since $\hat{X}_{G_2}(\epsilon \mid o_2) = \{0, 4, 5, 7\} \not\subseteq X_S$. One can verify that both of these two solutions cannot be further improved, i.e., they are maximal. Moreover, we have $\mathcal{L}(G_1) \not\subset \mathcal{L}(G_2)$ and $\mathcal{L}(G_2) \not\subset \mathcal{L}(G_1)$, i.e., they are two incomparable maximal solutions. Note that, we cannot achieve $\mathcal{L}(G_1) \cup \mathcal{L}(G_2)$ as there is a control conflict for event b in two solutions at the initial step.

Finally, we introduce some necessary operators that will be used for further developments. Let $q \in 2^X$ be a set of states, $\gamma \in \Gamma$ be a control decision and $\sigma \in \Sigma_o$ be an observable event. The *unobservable reach* of $q \subseteq X$ under control decision $\gamma \subseteq \Sigma$ is defined by

$$UR_\gamma(q) := \{\delta(x, w) \in X : x \in q, w \in (\Sigma_{uo} \cap \gamma)^*\}. \quad (2)$$

which is the set of states that can be reached from states in q via unobservable strings allowed in γ. The observable reach of $q \subseteq X$ upon the occurrence of $\sigma \in \Sigma_o$ is defined by

$$NX_\sigma(q) := \{\delta(x, \sigma) \in X : x \in q\}. \quad (3)$$

Similarly, let $\rho \in 2^{X \times X}$ be a set of state pairs, which represents that the system is currently at the second state and is from the first state at some instant. Let $\gamma \in \Gamma$ be a control decision and $\sigma \in \Sigma_o$ be an observable event. We define:

$$\widetilde{UR}_\gamma(\rho) = \{(x_1, \delta(x_2, w)) \in X^2 : (x_1, x_2) \in \rho, w \in (\Sigma_{uo} \cap \gamma)^*\}$$
$$\widetilde{NX}_\sigma(\rho) = \{(x_1, \delta(x_2, \sigma)) \in X^2 : (x_1, x_2) \in \rho\}$$
$$\odot_\gamma(q) = \{(x, \delta(x, w)) \in X^2 : x \in q, w \in (\Sigma_{uo} \cap \gamma)^*\}$$

In particular, $\odot_\gamma(q)$ maps q to a set of state pairs such that, in each pair of states, the first state can reach the second state via enabled but unobservable strings.

Example 3: Let us consider system G in Fig.1(a) again. Let $q = \{1, 5\} \in 2^X$ be a set of states and $\gamma = \{o_1, o_2, a, b\} \in \Gamma$ be the control decision. Then we have $UR_\gamma(\{1, 5\}) = \{1, 2, 3, 5\}$. Upon the occurrence observable event $o_1 \in \Sigma_o$, we have $NX_{o_1}(\{1, 2, 3, 5\}) = \{5, 6\}$. Let $\rho = \{(0, 1), (4, 5)\} \in 2^{X \times X}$ be a set of state pairs, which represents that the system is either (i) currently at state 1 from state 0; or (ii) currently at state 5 from state 4. Let $\gamma = \{o_1, o_2, a, b\} \in \Gamma$ be the control decision. Then we have $\widetilde{UR}_{\{o_1, o_2, a, b\}}(\{(0, 1), (4, 5)\}) = \{(0, 1), (0, 2), (0, 3), (4, 5)\}$. When event $o_1 \in \Sigma_o$ occurs, we have $\widetilde{NX}_{o_1}(\{(0, 1), (0, 2), (0, 3), (4, 5)\}) = \{(0, 5), (0, 6)\}$. Besides, let $q = \{1, 2, 3, 5\} \in 2^X$ and $\gamma = \{o_1, o_1, a, b\}$, we have $\odot_\gamma(q) = \{(1, 1), (1, 2), (1, 3), (2, 2), (2, 3), (3, 3), (5, 5)\}$.

IV. Information-State for Infinite-Step Opacity

In this section, we study how information evolves in the closed-loop system when delayed information is involved.

A. Notion of Information-State

In the synthesis of partially-observed systems, 2^X is usually chosen as the set of information states representing current information of the system. This information state has been shown to be suitable for current-state opacity enforcement. However, infinite-step opacity requires that the secret cannot be revealed at any instant currently and in the future; hence, 2^X is not sufficient enough.

Note that, the requirement of infinite-step opacity in Equation (1) can be equivalent written as

$$\forall \alpha \in P(\mathcal{L}(S_P/G)), \forall \alpha' \in \overline{\{\alpha\}} : \hat{X}_{S_P/G}(\alpha' | \alpha) \not\subseteq X_S. \quad (4)$$

where $\overline{\{\alpha\}}$ denotes the prefixes of all the strings in $\{\alpha\}$, i.e. $\overline{\{\alpha\}} := \{s \in \mathcal{L}(S_P/G) : \exists t \in \mathcal{L}(S_P/G) \text{ s.t. } st \in \overline{\{\alpha\}}\}$. This reformulation suggests that, to enforce infinite-step opacity, it is sufficient to capture the delayed estimates of all previous instants. Therefore, instead of using 2^X, we use another set of information states defined by

$$I := 2^X \times 2^{2^{X \times X}}.$$

Then each information state $\imath \in I$ is in the form of $\imath = (C(\imath), D(\imath))$, where

- The first component $C(\imath) \in 2^X$ is a set of states that capture the current state estimate of the system; and

- The second component $D(\imath) \in 2^{2^{X \times X}}$ is a set of state pairs sets that captures all possible delayed state estimates in history. Specifically, for $(x, x') \in 2^{X \times X}$, x represents the state of the system at some previous instant and x' represents the current state of the system, and $D(\imath)$ essentially contains all such pairs sets for all previous instants, where $\{(x_1, x'_1), (x_2, x'_2), \cdots, (x_n, x'_n)\} \in D(\imath)$.

B. Information-State Update Rules

Suppose that the system's information-state is $\imath = (C(\imath), D(\imath))$. Then, upon the occurrence of an observable event $\sigma \in \Sigma_o$ (should also be allowed by the previous control decision), the supervisor will issue a new control decision $\gamma \in \Gamma$. Therefore, (σ, γ) is our new information about the system and the information-state $\imath = (C(\imath), D(\imath))$ needs to updated to $\imath' = (C(\imath'), D(\imath'))$ as follows:

$$\begin{cases} C(\imath') = UR_\gamma(NX_\sigma(C(\imath))) \\ D(\imath') = \{\widetilde{UR}_\gamma(\widetilde{NX}_\sigma(\rho)) \in 2^{X \times X} : \rho \in D(\imath)\} \\ \qquad\qquad \cup \{\odot_\gamma(C(\imath'))\} \end{cases} \tag{5}$$

Intuitively, the first equation simply updates the current state estimate of the system, while the second equation updates all possible delayed state estimates (pairs) of previous instants and adds the current state estimate to the history. Then we consider a controlled system S_P/G. Let $\alpha = \sigma_1\sigma_2 \cdots \sigma_n \in P(\mathcal{L}(S_P/G))$ be an observable string. Then the following information states evolution will happen

$$\imath_0 \xrightarrow{(\sigma_1, S_P(\sigma_1))} \imath_1 \xrightarrow{(\sigma_2, S_P(\sigma_1\sigma_2))} \cdots \xrightarrow{(\sigma_n, S_P(\sigma_1 \cdots \sigma_n))} \imath_n$$

where $\imath_0 = (UR_{S_P(\epsilon)}(\{x_0\}), \{\odot_{S_P(\epsilon)}(UR_{S_P(\epsilon)}(\{x_0\}))\})$ represents the initial information state and each $\imath_{i-1} \xrightarrow{(\sigma_i, S_P(\sigma_1 \cdots \sigma_i))} \imath_i$ means that $\imath_i$ is obtained from $\imath_{i-1}$ with new information $(\sigma_i, S_P(\sigma_1 \cdots \sigma_i))$ according to Equation (5). We denote by $\mathcal{I}(\alpha)$ the information state reached by α, i.e., $\mathcal{I}(\alpha) = \imath_n$.

We illustrate the above information update procedure by the following example.

Example 4: Let S_P be the supervisor that results in closed-loop system G_2 in Fig.1(c). Specifically, supervisor S_P enables all the events except a initially, i.e., $S_P(\epsilon) = \{o_1, o_2, b, c\}$. Let us consider observable string $o_2 o_1 \in P(\mathcal{L}(S_P/G))$. Then we have

$$\mathcal{I}(\epsilon) = (UR_{S_P(\epsilon)}(\{x_0\}), \{\odot_{S_P(\epsilon)}(UR_{S_P(\epsilon)}(\{x_0\}))\}),$$

where

$$\begin{aligned} C(\mathcal{I}(\epsilon)) =& \{0, 4, 5, 7\} \\ D(\mathcal{I}(\epsilon)) =& \left\{\left\{ \begin{matrix} (0,0), (0,4), (0,5), (0,7), \\ (4,4), (4,5), (4,7), (5,5), (7,7) \end{matrix} \right\}\right\} \end{aligned}$$

Once event o_2 is observed, new control decision $S_P(o_2) = \{o_1, o_2, a, b, c\}$ is made, and the updated information state is $\mathcal{I}(o_2)$, where

$$\begin{aligned} C(\mathcal{I}(o_2)) =& UR_{\{o_1, o_2, a, b, c\}}(NX_{o_2}(C((\epsilon)))) \\ =& \{6, 8\} \\ D(\mathcal{I}(o_2)) =& \{\widetilde{UR}_{\{o_1, o_2\}}(\widetilde{NX}_{o_2}(D(\mathcal{I}(\epsilon))))\} \\ & \cup \{\odot_{\{o_1, o_2, a, b, c\}}(C(\mathcal{I}(o_2))\} \\ =& \left\{ \begin{matrix} \{(0,6), (4,6), (5,6), (0,8), (4,8), (7,8)\}, \\ \{(6,6), (8,8)\} \end{matrix} \right\} \end{aligned}$$

Then event o_1 is observed and new control decision $S_P(o_2 o_1) = \{o_1, o_2, a, b, c\}$ is made. The information state is then updated to $\mathcal{I}(o_2 o_1)$, where

$$\begin{aligned} C(\mathcal{I}(o_2 o_1)) =& UR_{S_P(o_2 o_1)}(NX_{o_1}(C(\mathcal{I}(o_2)))) \\ =& \{6\} \\ D(\mathcal{I}(o_2 o_1)) =& \{\widetilde{UR}_{S_P(o_2 o_1)}(\widetilde{NX}_{o_1}(D(\mathcal{I}(o_2))))\} \\ & \cup \{\odot_{S_P(o_2 o_1)}(C(\mathcal{I}(o_2 o_1))\} \\ =& \{\{(0,6), (4,6), (7,6)\}, \{(8,6)\}, \{(6,6)\}\} \end{aligned}$$

C. Property of the Information-State

In the following result, we formally show that the proposed information updating rule indeed yields the desired delayed state estimate in the controlled system.

Proposition 1: Let S_P be a supervisor, $\alpha \in P(\mathcal{L}(S_P/G))$ be an observable string and $\mathcal{I}(\alpha)$ be the information state reached. Then we have

(i) $C(\mathcal{I}(\alpha)) = \hat{X}_{S_P/G}(\alpha)$; and
(ii) $D(\mathcal{I}(\alpha)) = \{\rho_{\beta,\alpha} \in 2^{X \times X} : \beta \in \overline{\{\alpha\}}\}$, where

$$\rho_{\beta,\alpha} = \left\{ (\delta(s), \delta(st)) \in X \times X : \begin{matrix} \exists st \in \mathcal{L}(S_P/G) \text{ s.t.} \\ P(s) = \beta \wedge P(st) = \alpha \end{matrix} \right\}. \tag{6}$$

Note that each information-state is in the form of $\imath = (C(\imath), D(\imath)) = 2^X \times 2^{2^{X \times X}}$, where $D(\imath)$ is a set of possible state-pairs. We define

$$D_1(\imath) := \{\{x \in X : (x, x') \in \rho\} : \rho \in D(\imath)\} \tag{7}$$

as the set of its first components. For $\mathcal{I}(o_2) = (\{6, 8\}, \{\{(0,6), (4,6), (5,6), (0,8), (4,8), (7,8)\}, \{(6,6), (8, 8)\}\})$ in Example 4, we have $D_1(\mathcal{I}(o_2)) = \{\{0, 4, 5, 7\}, \{6, 8\}\}$. The following result shows that $D_1(\imath)$ indeed captures all possible delayed state estimates.

Corollary 1: Let S_P be a supervisory controller, $\alpha \in P(\mathcal{L}(S_P/G))$ be an observable string and $\mathcal{I}(\alpha)$ be the information state reached. Then we have

$$D_1(\mathcal{I}(\alpha)) = \{\hat{X}_{S_P/G}(\beta \mid \alpha) \in 2^X : \beta \in \overline{\{\alpha\}}\}.$$

V. Generalized Bipartite Transition System and Synthesis Algorithm

A. Bipartite Transition System

In Section IV, we have proposed a new type of information state that can capture all possible delayed state estimates. Note that, the information state updating rule as defined in Equation (5) essentially consists of two steps: the immediate observable reach when a new observable event occurs and the unobservable reach when a new control decision is issued.

However, this is based on the assumption that supervisor S_P is given. In the synthesis problem, the control decision at each instant is unknown and to be determined. Therefore, we need to separate these two update steps clearly. To this end, we define the *generalized bipartite transition system* (G-BTS) which is a generalized BTS structure in our previous work [23] by incorporating the new information state.

Definition 2: A generalized bipartite transition system (G-BTS) T w.r.t. G is a 7-tuple.

$$T = (Q_Y^T, Q_Z^T, h_{YZ}^T, h_{ZY}^T, \Sigma_o, \Gamma, y_0^T), \tag{8}$$

where

- $Q_Y^T \subseteq I$ is the set of Y-states. Therefore, a Y-state $y \in Q_Y^T$ is in the form of $y = (C(y), D(y))$;
- $Q_Z^T \subseteq I \times \Gamma$ is the set of Z-states. For each $z \in Q_Z^T$, $I(z)$ and $\Gamma(z)$ denote, respectively, the information state and the control decision, so that $z = (I(z), \Gamma(z)) \in 2^X \times 2^{2^{X\times X}} \times \Gamma$. For simplicity, we further write $z = (C(I(z)), D(I(z)), \Gamma(z))$ as $z = (C(z), D(z), \Gamma(z))$;
- $h_{YZ}^T : Q_Y^T \times \Gamma \to Q_Z^T$ is the partial transition function from Y-states to Z-states satisfying the following constraint: for any $h_{YZ}^T(y, \gamma) = z$, we have

$$\begin{cases} C(z) = UR_\gamma(C(y)) \\ D(z) = \{\widetilde{UR}_\gamma(\rho) \in 2^{X\times X} : \rho \in D(y)\} \cup \{\odot_\gamma(C(z))\} \\ \Gamma(z) = \gamma \end{cases} \tag{9}$$

- $h_{ZY}^T : Q_Z^T \times \Sigma \to Q_Y^T$ is the partial transition function from Z-states to Y-states satisfying the following constraint: for any $h_{ZY}^T(z, \sigma) = y$, we have $\sigma \in \Gamma(z) \cap \Sigma_o$ and

$$\begin{cases} C(y) = NX_\sigma(C(z)) \\ D(y) = \{\widetilde{NX}_\sigma(\rho) \in 2^{X\times X} : \rho \in D(z)\} \end{cases} \tag{10}$$

- Σ_o is the set of observable events of G;
- Γ is the set of admissible control decisions of G;
- $y_0^T := (\{x_0\}, \{\emptyset\}) \in Q_Y^T$ is the initial Y-state.

The G-BTS structure is essentially a game structure between the controller and the environment. When the controller picks a control decision $\gamma \in 2^\Sigma$ at a Y-state, the game moves to a Z-state. A transition from Y-state to Z-state is an unobservable reach under the issued control decision and remembers the control decision. When the environment picks an observation $\sigma \in \Sigma_o \cap \gamma$ at a Z-state, the game moves to a Y-state, and so forth. A transition from Z-state to Y-state is the observable reach. Transitions from Z-states to Y-states and transitions from Y-states to Z-states are indeed the information updating rule in Equation (5), but we separate the updating procedure into two parts. Specifically, for any $z \in Q_Z^Y, y \in Q_Y^T, \gamma \in \Gamma, \sigma \in \Sigma_o$, we have $h_{YZ}^T(h_{ZY}^T(z, \sigma), \gamma) = (\imath', \gamma)$, where $I(z) \xrightarrow{(\sigma,\gamma)} \imath'$. The basic structure of the BTS was originally proposed in [23]. Here we call the proposed structure *generalized* BTS as it is used information states capturing delays rather than the current-state-type information state used in [23].

Example 5: Again, consider system G in Fig.1(a). Then T^* in Fig.2 is a G-BTS. For the sake of simplicity, uncontrollable events o_1 and o_2 are omitted in each control decision in Fig.2. From the initial Y-state $y_0 = (\{0\}, \{\emptyset\})$, by making control decision $\gamma = \{b, c, o_1, o_2\}$, we will reach Z-state $z_1 = h_{YZ}^T(y_0, \gamma) = (\{0, 4, 5, 7\}, \{\{(0,0), (0,4), (0,5), (0,7), (4,4), (4,5), (4,7), (5,5), (7,7)\}\}, \{a, b, c\})$. From z_1, the occurrence of observable event o_2 leads to the next Y-state $y_1 = h_{ZY}^T(z_1, o_2) = (\{6, 8\}, \{\{(0,6), (0,8), (4,6), (4,8), (5,6), (7,8)\}\})$. From y_1, by making control decision $\gamma = \{o_1, o_2, a, b, c\}$, we will reach Z-state $z_2 = h_{YZ}^T(y_1, \gamma) = (\{6, 8\}, \{\{(0,6), (0,8), (4, 6), (4,8), (5,6), (7,8)\}, \{(6,6), (8,8)\}\}, \{a, b, c\})$. From z_2, the observable event o_1 can happen and we will reach the next Y-state $y_2 = h_{ZY}^T(z_2, o_1) = (\{6\}, \{\{(0,6), (4,6), (7, 6)\}, \{8, 6)\}\})$. Then we make the control decision $\gamma = \{o_1, o_2, a, b, c\}$ and reach to Z-state $z_3 = h_{YZ}^T(y_2, \gamma) = (\{6\}, \{\{(0,6), (4,6), (7,6)\}, \{(8,6)\}, \{(6,6)\}\}, \{a, b, c\})$.

B. Construction of Supervisor

By Equation (4) and Corollary 1, to make sure that the closed-loop system S_P/G is infinite-step opaque, it suffices to guarantee that, for any information state $\imath$ reached, we have $\forall q \in D_1(\imath), q \nsubseteq X_S$. Therefore, we define

$$Q_{unsafe} = \{z \in Q_Z : \exists q \in D_1(I(z)) \text{ s.t. } q \subseteq X_s\}$$

as the set of *unsafe* Z-states. To synthesize a supervisor that enforces infinite-step opacity, the controlled system S_P/G needs to guarantee that all the reachable information states are safe states. Furthermore, as the supervisor can only play at Y-states, we need to make sure that (i) there is at least one choice at each Y-state; and (ii) all choices at each Z-state should be considered. Therefore, for each G-BTS T, we say that a state is *consistent* if

- at least one transition is defined when it is a Y-state;
- all feasible observations are defined when it is a Z-state.

Let $\mathcal{T}$ be the set of all G-BTSs. For any G-BTS $T \in \mathcal{T}$, we define $Q_{consist}^T$ as the set of consistent states in T. Also, for a set of states $Q \subseteq Q_Y^T \cup Q_Z^T$, we define $T|_Q$ as the restriction of T to Q, i.e., $T|_Q$ is the G-BTS obtained by removing states and their transitions not in Q from T.

In order to synthesize an opacity-enforcing supervisor, first, we construct the largest G-BTS that enumerates all possible transitions for each state, i.e., a transition is defined whenever it satisfies the constraints in h_{YZ}^T or h_{ZY}^T. We define $T_{total} \in \mathcal{T}$ as the largest G-BTS that includes all possible transitions. Second, we remove all unsafe states from T_{total} and define

$$T_0 = T_{total}|_{Q\setminus Q_{unsafe}}.$$

Then, we need to iteratively remove inconsistent states from T until the G-BTS is consistent. Specifically, we define an operator

$$F : \mathcal{T} \to \mathcal{T}$$

by: for any T, we have $F(T) = T|_{Q_{consist}^T}$. Note that $T|_{Q_{consist}^T}$ need not be consistent in general since removing

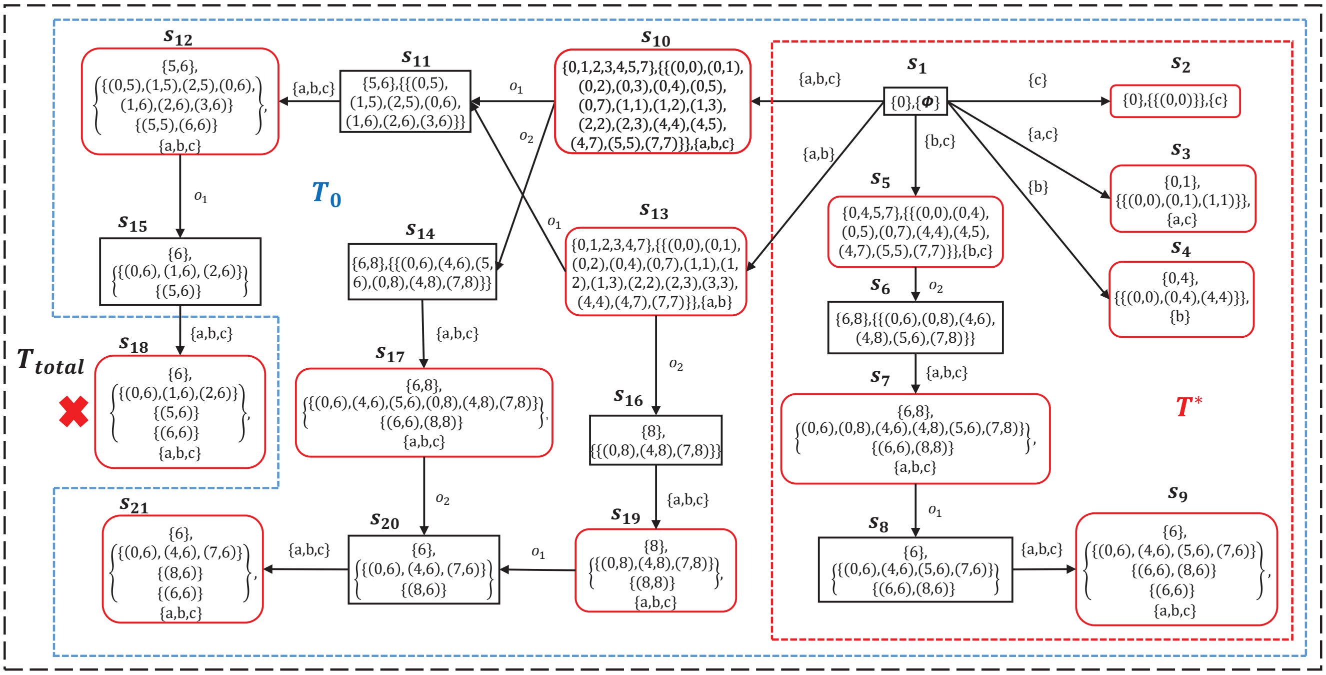

s_{12}
{5,6}, {{(0,5),(1,5),(2,5),(0,6),(1,6),(2,6),(3,6)}, {(5,5),(6,6)}}, {a,b,c}
{a,b,c}
s_{11}
{5,6},{{(0,5),(1,5),(2,5),(0,6),(1,6),(2,6),(3,6)}}
o_1
o_2
s_{10}
{0,1,2,3,4,5,7},{{(0,0),(0,1),(0,2),(0,3),(0,4),(0,5),(0,7),(1,1),(1,2),(1,3),(2,2),(2,3),(4,4),(4,5),(4,7),(5,5),(7,7)}},{a,b,c}
{a,b,c}
s_1
{0},{Φ}
{c}
s_2
{0},{{(0,0)}},{c}
{a,b}
{b,c}
{a,c}
s_3
{0,1}, {{(0,0),(0,1),(1,1)}}, {a,c}
{b}
s_4
{0,4}, {{(0,0),(0,4),(4,4)}}, {b}
o_1
T_0
s_{15}
{6}, {{(0,6), (1,6), (2,6)}, {(5,6)}}
s_{14}
{6,8},{{(0,6),(4,6),(5,6),(0,8),(4,8),(7,8)}}
o_1
s_{13}
{0,1,2,3,4,7},{{(0,0),(0,1),(0,2),(0,4),(0,7),(1,1),(1,2),(1,3),(2,2),(2,3),(3,3),(4,4),(4,7),(7,7)}},{a,b}
s_5
{0,4,5,7},{{(0,0),(0,4),(0,5),(0,7),(4,4),(4,5),(4,7),(5,5),(7,7)}},{b,c}
s_6
o_2
{6,8},{{(0,6),(0,8),(4,6),(4,8),(5,6),(7,8)}}
{a,b,c}
s_{18}
T_{total}
✖
{6}, {{(0,6),(1,6),(2,6)}, {(5,6)}, {(6,6)}}, {a,b,c}
{a,b,c}
s_{17}
{6,8}, {{(0,6),(4,6),(5,6),(0,8),(4,8),(7,8)}, {(6,6),(8,8)}}, {a,b,c}
o_2
s_{16}
{8}, {{(0,8),(4,8),(7,8)}}
s_7
{a,b,c}
{6,8}, {(0,6),(0,8),(4,6),(4,8),(5,6),(7,8)}, {(6,6),(8,8)}}, {a,b,c}
T^*
s_{21}
{6}, {{(0,6), (4,6), (7,6)}, {(8,6)}, {(6,6)}}, {a,b,c}
{a,b,c}
s_{20}
o_2
{6}, {{(0,6), (4,6), (7,6)}, {(8,6)}}
o_1
s_{19}
{a,b,c}
{8}, {{(0,8),(4,8),(7,8)}, {(8,8)}}, {a,b,c}
s_8
o_1
{6}, {{(0,6),(4,6),(5,6),(7,6)}, {(6,6),(8,6)}}
{a,b,c}
s_9
{6}, {{(0,6), (4,6), (5,6), (7,6)}, {(6,6), (8,6)}, {(6,6)}}, {a,b,c}

Fig. 2: Example of the construction of the G-BTS. Rectangular (black) states correspond to Y-states and rectangular with rounded corners (red) states correspond to Z-states. T_{total} is the largest G-BTS that enumerates all transitions.

inconsistent states may create new inconsistent states. Note that, iterating operator F always converges in a finite number of steps since we need to remove at least one state for each iteration and there are only finite number of states in T. We define

$$T^* = \lim_{k\to\infty} F^k(T_0), \text{ where } T_0 = T_{total}|_{Q\setminus Q_{unsafe}}$$

as the resulting G-BTS after the convergence of operator F.

Before we proceed further, we illustrate the above procedure by the following example.

Example 6: We still consider G in Fig.1(a). We want to synthesize a partial-observation supervisor S_P to enforce infinite-step opacity. First, we construct the largest G-BTS that enumerates all possible transitions for each state, which is T_{total} shown in the box marked with black-dashed lines in Fig.2. We rename each state in T_{total} as $S_1, \cdots, S_{21}$. Then, to obtain T_0, we need to restrict T_{total} to Q_{safe}. Since $D_1(S_{18}) = \{\{0,1,2\},\{5\},\{6\}\}$ and $\{5\} \subseteq X_S$, S_{18} is a unsafe state, which needs to be removed. This results in T_0 shown in the box marked with blue-dashed lines in Fig.2. Then, we need to remove all inconsistent states from T_0. Since S_{18} has been removed, S_{15} becomes an inconsistent Z-state, which is also removed by applying operator F for the first time. Then S_{12} becomes a new inconsistent Z-state as feasible observable event o_1 is not defined; hence S_{12} is deleted when applying operator F for the second time. We keep applying operator F and need to remove state S_{11}. This makes S_{10} and S_{13} inconsistent because feasible observable event o_1 is not defined. So operator F will further delete S_{10} and S_{13}. Then operator F converges and results in T^* shown in the box marked with red-dashed lines in Fig.2.

Next, we show that synthesizing a supervisor within T^* is without loss of generality.

Theorem 5.1: The opacity-enforcing synthesis problem has no solution if T^* is empty.

Given a G-BTS T, we define $C_T(y) := \{\gamma \in \Gamma : h^T_{YZ}(y,\gamma)!\}$ as the set of control decisions defined at $y \in Q^T_Y$ When T^* is not empty, we can synthesize a supervisor S^* as follows. At each instant, the supervisor S^* will remember the current information state (Y-state) and pick a *locally maximal* control decision γ defined at the current Y-state in T^*, i.e., a control decision $\gamma \in C_{T^*}(y)$ such that $\nexists \gamma' \in C_{T^*}(y) : \gamma \subset \gamma'$. [1] Note that locally maximal control decision is not unique in general and we denote by $LocMax(y, T^*)$ the set of locally maximal control decisions at y in T^*. Then we update the information state based on the control decision issued and wait for the next observable event and so forth. The execution of S^* is formally described as Algorithm 1.

Next, we show that the proposed supervisor S^* indeed solves Problem 1.

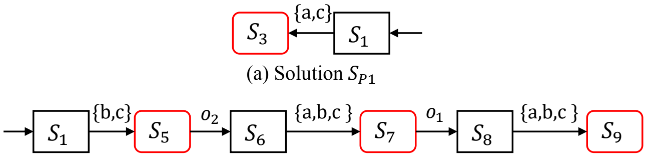
Theorem 5.2: Supervisor S^* defined by Algorithm 1 enforces infinite-step opacity and is maximally permissive.

Example 7: Let us consider system G in Fig.1(a) and we use Algorithm 1 to solve the infinite-step opacity control problem. At the initial Y-state $y_0 = (\{x_0\}, \{\emptyset\})$, we have

[1] Some decisions are "equivalent" at a Y-state in the sense that some events in the control decision may not be feasible within the unobservable reach. For example, decisions $\{a\}$ and $\{a,c\}$ are equivalent at S_1 in Fig. 2 as event c is not feasible. For those equivalent decisions, we only draw the one with all redundant event included in the figure. However, when comparing decisions to find local maximal decisions, those redundant events should not be counted.

Algorithm 1: Online Execution of Supervisor S^*

```
$y \leftarrow \{\{x_0\}, \{\emptyset\}\}$;
find a locally maximal control decision
    $\gamma \in \textsc{LocMax}(y, T^*)$;
make initial control decision $\gamma$;
$z \leftarrow h^T_{YZ}(y, \gamma)$;
while new event $\sigma \in \gamma \cap \Sigma_o$ is observed do
    $y \leftarrow h^T_{ZY}(z, \sigma)$;
    find a locally maximal control decision
        $\gamma \in \textsc{LocMax}(y, T^*)$;
    update the control decision to $\gamma$;
    $z \leftarrow h^T_{YZ}(y, \gamma)$;
```



(a) Solution S_{P1}

(b) Solution S_{P2}

Fig. 3: Solution S_P.

$\textsc{LocMax}(y_0, T^*) = \{\{a, c\}, \{a, b\}\}$. If we choose control decision $\{a, c\}$, then we reach Z-state S_3 and the system has no future observation. This gives supervisor S_{P1} shown in Fig.3(a), which is the one resulting in G_1 in Fig.1(b). On the other hand, if we choose control decision $\{b, c\}$ initially, then the system moves to S_5. From S_5, the occurrence of o_2 leads to S_6 and the supervisor picks $\{a, b, c\}$ as the maximal control decision and the system moves to S_7. Then observable events o_1 occurs and the system moves to S_8, where supervisor S_{P2} picks $\{o_1, o_2, a, b, c\}$ leading the system to state S_9. This gives supervisor S_{P2} shown in Fig.3(b), which is the one resulting in G_1 in Fig.1(c). As we have discussed in Example 2, both supervisors are maximal and they are incomparable.

VI. Conclusion

We solved the maximally-permissive infinite-step opacity control problem without assumption on controllable events or observable events. We defined a new class of generalized bipartite transition structures that captures the delayed information in the control synthesis problem over a game structure. Based on the G-BTS, we proposed an effective algorithm to construct a maximally-permissive supervisor that enforces infinite-step opacity. Note that the worst complexity of the proposed algorithm is doubly-exponential due to the difficulty of the delayed information. In the future, we plan to further mitigate the computational complexity using abstraction-based techniques [16], [17], [28].

References

[1] E. Badouel, M. Bednarczyk, A. Borzyszkowski, B. Caillaud, and P. Darondeau. Concurrent secrets. *Discrete Event Dynamic Systems: Theory & Appllications*, 17(4):425–446, 2007.

[2] C.G. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems*. Springer, 2nd edition, 2008.

[3] F. Cassez, J. Dubreil, and H. Marchand. Synthesis of opaque systems with static and dynamic masks. *Formal Methods in System Design*, 40(1):88–115, 2012.

[4] J. Chen, M. Ibrahim, and R. Kumar. Quantification of secrecy in partially observed stochastic discrete event systems. *IEEE Trans. Automation Science and Engineering*, 14(1):185–195, 2017.

[5] P. Darondeau, H. Marchand, and L. Ricker. Enforcing opacity of regular predicates on modal transition systems. *Discrete Event Dynamic Systems: Theory & Appllications*, 25(1-2):251–270, 2014.

[6] J. Dubreil, P. Darondeau, and H. Marchand. Supervisory control for opacity. *IEEE Trans. Automatic Control*, 55(5):1089–1100, 2010.

[7] Y. Falcone and H. Marchand. Enforcement and validation (at runtime) of various notions of opacity. *Discrete Event Dynamic Systems: Theory & Appllications*, 25(4):531–570, 2015.

[8] R. Jacob, J.-J. Lesage, and J.-M. Faure. Overview of discrete event systems opacity: Models, validation, and quantification. *Annual Reviews in Control*, 41:135–146, 2016.

[9] Y. Ji, Y.-C. Wu, and S. Lafortune. Enforcement of opacity by public and private insertion functions. *Automatica*, 93:369–378, 2018.

[10] Y. Ji, X. Yin, and S. Lafortune. Enforcing opacity by insertion functions under multiple energy constraints. *Automatica*, 108:108476, 2019.

[11] Y. Ji, X. Yin, and S. Lafortune. Opacity enforcement using nondeterministic publicly known edit functions. *IEEE Trans. Automatic Control*, 64(10):4369–4376, 2019.

[12] C. Keroglou and C.N. Hadjicostis. Probabilistic system opacity in discrete event systems. *Discrete Event Dynamic Systems: Theory & Appllications*, pages 1–26, 2017.

[13] S. Lafortune, F. Lin, and C.N. Hadjicostis. On the history of diagnosability and opacity in discrete event systems. *Annual Reviews in Control*, 45:257–266, 2018.

[14] F. Lin. Opacity of discrete event systems and its applications. *Automatica*, 47(3):496–503, 2011.

[15] T. Masopust and X. Yin. Complexity of detectability, opacity and A-diagnosability for modular discrete event systems. *Automatica*, 101:290–295, 2019.

[16] S. Mohajerani, Y. Ji, and S. Lafortune. Compositional and abstraction-based approach for synthesis of edit functions for opacity enforcement. *IEEE Trans. Automatic Control*, 2019.

[17] M. Noori-Hosseini, B. Lennartson, and C. Hadjicostis. Compositional visible bisimulation abstraction applied to opacity verification. In *14th Int. Workshop on Discrete Event Systems*, pages 434–441, 2018.

[18] A. Saboori and C.N. Hadjicostis. Opacity-enforcing supervisory strategies via state estimator constructions. *IEEE Trans. Automatic Control*, 57(5):1155–1165, 2011.

[19] A. Saboori and C.N. Hadjicostis. Verification of infinite-step opacity and complexity considerations. *IEEE Trans. Automatic Control*, 57(5):1265–1269, 2012.

[20] S. Takai and Y. Oka. A formula for the supremal controllable and opaque sublanguage arising in supervisory control. *SICE J. Control, Measu. & Syst. Integration*, 1(4):307–311, 2008.

[21] Y. Tong, Z. Li, C. Seatzu, and A. Giua. Verification of state-based opacity using Petri nets. *IEEE Trans. Automatic Control*, 62(6):2823–2837, 2017.

[22] Y. Tong, Z. Li, C. Seatzu, and A. Giua. Current-state opacity enforcement in discrete event systems under incomparable observations. *Discrete Event Dynamic Systems: Theory & Appllications*, 28(2):161–182, 2018.

[23] X. Yin and S. Lafortune. A uniform approach for synthesizing property-enforcing supervisors for partially-observed discrete-event systems. *IEEE Trans. Automatic Control*, 61(8):2140–2154, 2016.

[24] X. Yin and S. Lafortune. A new approach for the verification of infinite-step and k-step opacity using two-way observers. *Automatica*, 80:162–171, 2017.

[25] X. Yin and S. Li. Synthesis of dynamic masks for infinite-step opacity. *IEEE Trans. Automatic Control*, 2019.

[26] X. Yin, Z. Li, W. Wang, and S. Li. Infinite-step opacity and k-step opacity of stochastic discrete-event systems. *Automatica*, 99:266–274, 2019.

[27] B. Zhang, S. Shu, and F. Lin. Maximum information release while ensuring opacity in discrete event systems. *IEEE Trans. Automation Science and Engineering*, 12(4):1067–1079, 2015.

[28] K. Zhang, X. Yin, and M. Zamani. Opacity of nondeterministic transition systems: A (bi) simulation relation approach. *IEEE Trans. Automatic Control*, 64(12):5116–5123, 2019.