Opacity Enforcing Supervisory Control Using Nondeterministic Supervisors

Yifan Xie, *Student Member, IEEE*, Xiang Yin, *Member, IEEE*,
and Shaoyuan Li, *Senior Member, IEEE*

***Abstract*—In this article, we investigate the enforcement of opacity via supervisory control in the context of discrete-event systems. A system is said to be opaque if the intruder, which is modeled as a passive observer, can never infer confidently that the system is at a secret state. The design objective is to synthesize a supervisor such that the closed-loop system is opaque even when the control policy is publicly known. In this article, we propose a new approach for enforcing opacity using nondeterministic supervisors. A nondeterministic supervisor is a decision mechanism that provides *a set of control decisions* at each instant, and randomly picks a specific control decision from the decision set to actually control the plant. Compared with the standard deterministic control mechanism, such a nondeterministic control mechanism can enhance the plausible deniability of the controlled system as the online control decision is a random realization and cannot be implicitly inferred from the control policy. We provide a sound and complete algorithm for synthesizing a nondeterministic opacity-enforcing supervisor. Furthermore, we show that nondeterministic supervisors are strictly more powerful than deterministic supervisors in the sense that there may exist a nondeterministic opacity-enforcing supervisor even when deterministic supervisors cannot enforce opacity.**

***Index Terms*—Discrete-event systems (DES), opacity, supervisory control.**

I. Introduction

INFORMATION security and privacy have become increasingly important issues in the analysis and design of modern engineering systems due to potential malicious attacks and information leakages in networks. In this article, we investigate an important information-flow security property called *opacity* in the context of discrete-event systems (DES). In this framework, a dynamic system is modeled as a DES and an intruder is modeled as a passive observer that monitors the behavior of the dynamic system via observable events. Essentially, opacity is a confidential property capturing whether or not the system can always deny of the possibility of executing of a secret behavior even when it may be true, i.e., it holds the plausible deniability for secret behaviors. Therefore, a system is said to be opaque with respect to a set of secret states if the intruder can never know for sure that the system is visiting a secret state.

Due to the increasing demands for security certification in safety-critical systems, the notion of opacity has drawn considerable attention in the past years in the literature, see, e.g., [2], [5], [31]. In particular, in the context of DES, different notions of opacity have been studied, including, e.g., current-state opacity [26], initial-state opacity [39], K-step and infinite-step opacity [51]. The verification of opacity has also been studied for different DES models including Petri nets [25], [37], [43], stochastic DES [8], [21], [47], [52], real-time systems [45], and networked DES [27], [49]. More recently, the notion of opacity has been extended to linear/nonlinear systems with infinite-states and continuous dynamics [1], [36], [53]. The reader is referred to the comprehensive surveys [19], [24] and the textbook [15] for recent advances on this active research area.

Given an open-loop system that is verified to be nonopaque, one important problem is to *enforce* opacity via some enforcement mechanisms. This is also referred to as the synthesis problem, which is a very active research topic in the literature and many different enforcement mechanisms have been proposed. For example, the authors in [4], [7], and [54] consider the enforcement of opacity via dynamic masks that change the output information dynamically. The idea of changing the output information has also been leveraged by using insertion functions [20], [22], [28], [46] and event shuffles [3]. In addition, event delays is also used to enforce opacity in [12].

One of the most widely investigated opacity enforcement mechanisms is the supervisory control theory [16], [38], [50]. In this framework, a supervisor is used to restrict the behavior of the system such that the closed-loop system is opaque [9], [38], [57]. For example, in [42], a formula for controllable and opaque sublanguage is provided. In [10], the authors solve the opacity control problem by assuming that all controllable events are observable and the observation of the intruder is included in the observation of the supervisor. In [50], a uniform approach is provided to solve the opacity-enforcing control problem without the assumption that controllable events are observable; however, it assumes that the observations of the supervisor and the intruder are equivalent. Recently in [44], the authors provide an algorithm

Manuscript received 19 October 2020; revised 18 May 2021 and 6 November 2021; accepted 19 November 2021. Date of publication 26 November 2021; date of current version 5 December 2022. This work was supported in part by the National Natural Science Foundation of China under Grant 62061136004, Grant 62173226, and Grant 61803259, and in part by the Shanghai Jiao Tong University Scientific and Technological Innovation Funds. Recommended by Associate Editor Jan Komenda. *(Corresponding author: Xiang Yin.)*

The authors are with the Department of Automation and Key Laboratory of System Control and Information Processing, Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: xyfan1234@sjtu.edu.cn; yinxiang@sjtu.edu.cn; syli@sjtu.edu.cn).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TAC.2021.3131125.

Digital Object Identifier 10.1109/TAC.2021.3131125

for synthesizing an opacity enforcing supervisor without any assumption on events set. However, it needs to assume that the control policy is not publicly known, which reduces the problem to the computation for a maximal controllable and observable sublanguage of the supremal opaque sublanguage.

Note that all existing works on opacity-enforcing supervisory control consider deterministic supervisors, which issue a specific control decision at each instant. However, such a deterministic decision mechanism may decrease the plausible deniability of the system. This is because, by knowing the control policy and by observing the occurrences of observable events, the intruder can recover the control decision made by the supervisor and, therefore, obtain a better state-estimate of the system.

In this article, we propose to use nondeterministic supervisors, for the first time, to enforce opacity. Unlike a deterministic supervisor that issues a specific control decision at each instant, a nondeterministic supervisor provides *a set of control decisions* at each instant and the specific control decision applied is chosen randomly via a "coin toss" manner. In other words, even if the intruder knows the control policy, it still does not know the specific control decision applied as it is decided randomly on-the-fly. Compared with the deterministic control mechanism, the nondeterministic control mechanism can significantly enhance the plausible deniability of the system, and, therefore, is more likely to enforce opacity.

The main contribution of this article is that we provide an algorithmic correct-by-construction procedure for synthesizing a nondeterministic supervisor that enforces opacity. This problem is fundamentally more challenging than the deterministic case as the observations of the supervisor and the intruder are *incomparable*. Specifically, although the specific control decision applied is unknown *a priori*, the supervisor will know this online choice after it is chosen. This information, however, is not available to the intruder. Hence, the supervisor's knowledge is strictly more than that of the intruder. In the standard opacity-enforcing control problem, it is sufficient to know the state-estimate of the system, which is not sufficient in our setting due to the issue of incomparable information. To address this issue, we propose a new information state (IS) that not only contains the state-estimate from the supervisor's point of view, but also contains the estimate of the supervisor's estimate from the intruder's point of view. In other words, the control decision should be made not only based on what the supervisor thinks about the plant, but also based on what the intruder thinks about the supervisor. Based on the proposed new IS, we provide a sound and complete approach that synthesizes a nondeterministic opacity-enforcing supervisor. In particular, we show that using nondeterministic supervisors is strictly more powerful than using deterministic supervisors, in the sense that, there may exist a nondeterministic opacity-enforcing supervisor even when deterministic supervisors cannot enforce opacity.

We note that the notion of nondeterministic supervisors was originally proposed in [18] to solve the standard supervisory control problem for safety and nonblockingness under partial observation. This approach was extended by [23]. Nondeterministic control mechanism has also been used for (bi)similarity enforcing supervisory control problems with nondeterministic models and specifications [11], [13], [40], [41], [56]. However, to the best of authors' knowledge, nondeterministic supervisors have never been applied to the opacity-enforcement problem. More importantly, the essence of why we use nondeterministic supervisors here is to enhance the plausible deniability of the system, which is fundamentally different from the essence of the existing works.

The rest of this article is organized as follows. In Section II, we introduce some necessary preliminaries. In Section III, we first provide a motivating example to illustrate the advantage of nondeterministic supervisors. Then, we formally present the nondeterministic control mechanism and formulate the corresponding opacity enforcement control problem. In Section IV, we propose a new type of IS that captures both the knowledge of the supervisor and the knowledge of the intruder and analyze the underlying information-flow. Then, we restrict our attention to the class of IS-based supervisors and discuss how an IS-based supervisor can be encoded as or be decoded from an IS-mapping. In Section V, we propose an algorithm to synthesize an IS-based nondeterministic opacity-enforcing supervisor based on the structure of the generalized bipartite transition system (G-BTS). In Section VI, we prove the correctness of the synthesis procedure proposed in Section V by showing that restricting our attention to IS-based supervisors is without loss of generality. Finally, we conclude this article in Section VII. Preliminary and partial versions of some of the results in this article are presented in [48]. First, all definitions, notations, and theorems in [48] have been reformulated in a more uniform manner. More importantly, the result in [48] is only sound as it restricts the solution space to a finite space *a priori*. In this work, we show that restricting to IS-based supervisor is without loss of generality using new techniques developed based on IS-mappings. This new result establishes both the soundness and the completeness of the synthesis algorithm, i.e., the nondeterministic synthesis problem is completely solved.

II. Preliminaries

A. System Model

Let Σ be a finite set of events. A string over Σ is a finite sequence $s = \sigma_1 \cdots \sigma_n, \sigma_i \in \Sigma$. We denote by Σ^* the set of all strings over Σ including the empty string ϵ. A language $L \subseteq \Sigma^*$ is a set of strings. For two languages L_1 and L_2, their concatenation is $L_1 L_2 = \{s_1 s_2 \in \Sigma^* : s_1 \in L_1, s_2 \in L_2\}$. The prefix-closure of language L is defined by $\overline{L} = \{v \in \Sigma^* : \exists u \in \Sigma^* \text{ s.t. } vu \in L\}$.

We assume basic knowledge of DES and use common notations; see, e.g., [6]. 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 [6]. 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

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

The natural projection is also extended to $P : 2^{\Sigma^*} \to 2^{\Sigma_o^*}$ by $P(L) = \{P(s) : s \in L\}$.

B. Deterministic Supervisory Control

In the framework of supervisory control, a supervisor dynamically enables/disables controllable events based on its observation. Formally, 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. A control decision $\gamma \in 2^\Sigma$ is a set of events such that $\Sigma_{uc} \subseteq \gamma$, i.e., uncontrollable events can never be disabled. We define $\Gamma = \{\gamma \in 2^\Sigma : \Sigma_{uc} \subseteq \gamma\}$ as the set of control decisions or control patterns. A *deterministic supervisor* is a function $S : P(\mathcal{L}(G)) \to \Gamma$. The language generated by the controlled system, denoted by $\mathcal{L}(S/G)$, is defined recursively by

1) $\epsilon \in \mathcal{L}(S/G)$;
2) For any $s \in \Sigma^*, \sigma \in \Sigma$, we have $s\sigma \in \mathcal{L}(S/G)$ iff $s\sigma \in \mathcal{L}(G), s \in \mathcal{L}(S/G)$ and $\sigma \in S(P(s))$.

C. Opacity

We assume that system G has a "secret", which is modeled as a set of secret states $X_S \subseteq X$. Furthermore, we consider a passive *intruder* having the following capabilities.

A1 The intruder knows the system model;
A2 The intruder can observe the occurrences of observable events.

Such an intruder is essentially an outside observer or an "eavesdropper." We say that system G is *opaque* w.r.t. X_S and Σ_o if

$$(\forall s \in \mathcal{L}(G) : \delta(s) \in X_S)(\exists t \in \mathcal{L}(G) : \delta(t) \notin X_S)[P(s) = P(t)].$$

That is, the intruder cannot infer for sure that the system is in a secret state based on the information flow.

When the original system is not opaque, one approach is to design a supervisor S such that the closed-loop system S/G is opaque; this is referred to as the *opacity-enforcing control problem*. In this setting, however, the implementation of such a supervisor may become a public information. To capture this severe scenario, we make the following assumption.

A3 The intruder knows the functionality of the supervisor, i.e., the control policy.

Note that, under the setting of deterministic supervisors, this knowledge together with the assumption that the intruder and the observer both observe Σ_o imply that the intruder knows precisely the control decision applied at each instant. Therefore, to define opacity of the controlled system, we should only consider strings in $\mathcal{L}(S/G)$ rather than all strings in $\mathcal{L}(G)$. Formally, we say that a deterministic supervisor $S : P(\mathcal{L}(G)) \to \Gamma$ enforces opacity on G, or the closed-loop system S/G is opaque, if for any string $s \in \mathcal{L}(S/G)$ such that $\delta(s) \in X_S$, there exists a string $t \in \mathcal{L}(S/G)$ such that $\delta(t) \notin X_S$ and $P(s) = P(t)$.

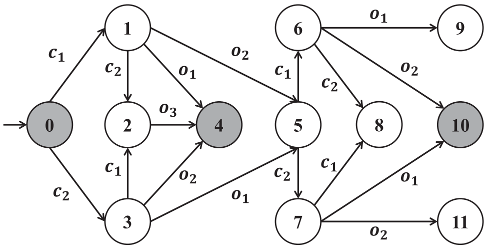

Fig. 1. System G with $\Sigma_c = \{c_1, c_2\}$, $\Sigma_o = \{o_1, o_2, o_3\}$, and $X_S = \{0, 4, 10\}$.

Finally, we introduce some operators that will be used in this article. Given a set of states $m \in 2^X$, we denote by $UR_\gamma(m)$, the *unobservable reach* of m under control decision $\gamma \in \Gamma$, i.e.,

$$UR_\gamma(m) = \{\delta(x, w) \in X : x \in m, w \in (\Sigma_{uo} \cap \gamma)^*\}. \quad (1)$$

We also denote by $NX_\sigma(m)$ the *observable reach* of m upon the occurrence of an observable event $\sigma \in \Sigma_o$, i.e.,

$$NX_\sigma(m) = \{\delta(x, \sigma) \in X : x \in m\}. \quad (2)$$

III. Enforcing Opacity Using Nondeterministic Supervisors

In this section, we propose to use nondeterministic supervisors to enforce opacity. First, we illustrate the advantage of using nondeterministic supervisors by a motivating example. Then, we formally define the functionality of the nondeterministic supervisor and opacity of nondeterministic control systems. We formulate the corresponding opacity-enforcing supervisory control problem that we want to solve in this article.

A. Motivating Example

Example 1: Let us consider system G shown in Fig. 1 with $\Sigma_o = \Sigma_{uc} = \{o_1, o_2, o_3\}$ and $X_S = \{0, 4, 10\}$. String $c_2o_1c_2o_1$ leads to secret state 10 and its observation is $P(c_2o_1c_2o_1) = o_1o_1$. By observing o_1o_1, the intruder cannot infer for sure that the system is at state 10 since $P(c_2o_1c_1o_1) = o_1o_1$ and $\delta(c_2o_1c_1o_1) = 9 \notin X_S$. However, by observing o_3, the intruder knows for sure that the system is at secret state 4 since for any string s such that $P(s) = o_3$, we have $\delta(s) = 4 \in X_S$. Therefore, the system is not opaque and we need to synthesize a supervisor to protect the system from revealing secret 4.

For this system, however, we cannot even synthesize a deterministic supervisor to enforce opacity. To see this clearly, let us evaluate what the supervisor can do initially. We have $\Gamma = \{\emptyset, \{c_1\}, \{c_2\}, \{c_1, c_2\}\}$.[1] Clearly, the supervisor cannot

[1] For the sake of simplicity, uncontrollable events are omitted in each control decision, i.e., $\emptyset$ standards for $\{o_1, o_2, o_3\}$ in this example.

choose $\emptyset$ as the initial control decision; otherwise secret state 0 will be the only reachable state. Also, the supervisor cannot make $\{c_1\}$ initially. This is because, under this control decision and by observing event o_1, the intruder knows for sure that the system is at state 4, which is reached via $0 \xrightarrow{c_1} 1 \xrightarrow{o_1} 4$. Note that transitions $0 \xrightarrow{c_2} 3 \xrightarrow{o_1} 5$ cannot provide the plausible deniability since c_2 is disabled initially. For the same reason, making $\{c_2\}$ initially will also reveal the secret. Finally, decision $\{c_1, c_2\}$ is also problematic initially as it makes state 2 reachable from which transition $2 \xrightarrow{o_3} 4$ will also reveal the secret. Therefore, we cannot enforce opacity for this system using a deterministic supervisor.

However, one can enforce opacity using the following control mechanism. Initially, the supervisor randomly chooses to either enable c_1 or enable c_2, but not enable both simultaneously. In other words, the control policy initially is a set $\{\{c_1\}, \{c_2\}\}$ and the specific choice is made randomly on-the-fly. Therefore, upon the occurrence of o_1 or o_2, the intruder does not know whether this event is from state 1 or from state 3 since it does not know whether or not the initial online control decision is $\{c_1\}$ or $\{c_2\}$ by just knowing the control policy $\{\{c_1\}, \{c_2\}\}$. On the other hand, since c_1 and c_2 will not be enabled simultaneously, state 2 is not reachable; hence, event o_3, which reveals the secret, will also not occur. Then, after observing o_1 or o_2, the supervisor can make decision $\{c_1, c_2\}$ deterministically, which prevents the system from revealing secret state 10.

The abovementioned example shows that using a nondeterministic control mechanism is more powerful than the deterministic one for the purpose of enforcing opacity. This result is intuitive as opacity is essentially a confidential property. Using a nondeterministic decision framework will, on the one hand, enhance the plausible deniability of the secret behavior of the system, and, on the other hand, decrease the confidentiality of the intruder's knowledge about the system. Hence, the system is more likely to be opaque under the nondeterministic mechanism.

B. Nondeterministic Supervisor

Now, we formally define the nondeterministic supervisor and the corresponding opacity enforcement problem.

Compared with a deterministic supervisor that issues a specific control decision at each instant, a nondeterministic supervisor works as follows. At each instant, the nondeterministic supervisor provides *a set of possible control decisions*. Then, it nondeterministically picks a specific control decision from this set in a "coin-toss" manner and keeps this specific control decision until a new observable event occurs. In other words, the control policy only determines a set of allowed decisions, but the specific control decision chosen is unknown *a priori*, which is a *random realization* under the control policy. Therefore, the supervisor makes decision not only based on observable events, but also depends on the specific control decisions chosen along the trajectory.

To define the "history" of the supervisor, we introduce the notion of the *extended string*, which is an alternating sequence of control decisions and events either ending up with a control decision in the form of

$$\rho = \gamma_0\sigma_1\gamma_1 \cdots \sigma_n\gamma_n \in \Gamma(\Sigma\Gamma)^* \tag{3}$$

or ending up with an event in the form of

$$\rho = \gamma_0\sigma_1\gamma_1 \cdots \sigma_n \in (\Gamma\Sigma)^*. \tag{4}$$

Then, the set of all extended strings is $\Gamma(\Sigma\Gamma)^* \cup (\Gamma\Sigma)^* = (\Gamma\Sigma)^*(\Gamma \cup \{\epsilon\})$. For any extended string $\rho \in (\Gamma\Sigma)^*(\Gamma \cup \{\epsilon\})$, we denote by $\rho|_\Sigma$ the projection to Σ^*, i.e., $\rho|_\Sigma = \sigma_1 \ldots \sigma_n$.

Since some events are unobservable for the supervisor and the supervisor cannot update its decision upon the occurrence of an unobservable event, similar to the natural projection, we define a new projection mapping

$$\mathcal{O} : (\Gamma\Sigma)^*(\Gamma \cup \{\epsilon\}) \rightarrow (\Gamma\Sigma_o)^*(\Gamma \cup \{\epsilon\}) \tag{5}$$

such that, for any extended string, projection $\mathcal{O}$ erases each unobservable event together with its successor control decision (if there exists one). Formally, for extended string ρ in the form of (3), let $1 \le i_1 < i_2 < \cdots < i_k \le n$ be all indices such that $\sigma_{i_j} \in \Sigma_o$. Then, we have

$$\mathcal{O}(\rho) = \gamma_0(\sigma_{i_1}\gamma_{i_1})(\sigma_{i_2}\gamma_{i_2}) \cdots (\sigma_{i_k}\gamma_{i_k}) \tag{6}$$

and for extended string ρ in the form of (4), we have

$$\mathcal{O}(\rho) = \gamma_0(\sigma_{i_1}\gamma_{i_1})(\sigma_{i_2}\gamma_{i_2}) \cdots (\sigma_{i_{k-1}}\gamma_{i_{k-1}})\sigma_{i_k}. \tag{7}$$

From the supervisor's point of view, each decision is made immediately (by first providing a set of decisions and then picking one from the set) after observing an observable event. Therefore, the supervisor should make decision based on alternating sequences that end up with observable events. Hence, the nondeterministic supervisor is defined as a function

$$S_N : (\Gamma\Sigma_o)^* \rightarrow 2^\Gamma \tag{8}$$

that maps an observable extended string $\mathcal{O}(\rho) = \gamma_0\sigma_1\gamma_1 \cdots \sigma_n \in (\Gamma\Sigma_o)^*$, which is referred to a *decision history*, to a set of possible control decisions. This definition essentially says that, although the control policy is nondeterministic, the supervisor knows the *realization*, i.e., which specific decision was picked at each previous instant. This is a reasonable setting as the supervisor always knows what it actually picks. Now, we define the language generated by a nondeterministic supervisor.

Definition 1: Let S_N be a nondeterministic supervisor. The set of extended strings generated by the closed-loop system, denoted by $\mathcal{L}_e(S_N/G)$, is defined recursively by

1) $\epsilon \in \mathcal{L}_e(S_N/G)$;
2) $\gamma_0 \in \mathcal{L}_e(S_N/G)$ if $\gamma_0 \in S_N(\epsilon)$;
3) for any $\rho = \gamma_0\sigma_1\gamma_1 \cdots \sigma_n\gamma_n\sigma_{n+1} \in (\Gamma\Sigma)^*$, we have $\rho \in \mathcal{L}_e(S_N/G)$, if and only if
   a) $\gamma_0\sigma_1\gamma_1 \ldots \sigma_n\gamma_n \in \mathcal{L}_e(S_N/G)$;
   b) $\sigma_1 \cdots \sigma_n\sigma_{n+1} \in \mathcal{L}(G)$;
   c) $\sigma_{n+1} \in \gamma_n$.
4) For any $\rho = \gamma_0\sigma_1\gamma_1 \cdots \sigma_n\gamma_n\sigma_{n+1}\gamma_{n+1} \in \Gamma(\Sigma\Gamma)^*$, we have $\rho \in \mathcal{L}_e(S_N/G)$, if and only if
   a) $\gamma_0\sigma_1\gamma_1 \cdots \sigma_n\gamma_n\sigma_{n+1} \in \mathcal{L}_e(S_N/G)$;
   b) $\gamma_{n+1} \in \begin{cases} \{\gamma_n\} & \text{if } \sigma_{n+1} \in \Sigma_{uo} \\ S_N(\mathcal{O}(\gamma_0\sigma_1\gamma_1 \ldots \sigma_n\gamma_n\sigma_{n+1})) & \text{if } \sigma_{n+1} \in \Sigma_o \end{cases}$

Then, a string $s \in \Sigma^*$ is said to be *generated* by S_N/G if there exists an extended string $\rho \in \mathcal{L}_e(S_N/G)$ such that $\rho|_\Sigma = s$. We define $\mathcal{L}(S_N/G) = \{\rho|_\Sigma \in \Sigma^* : \rho \in \mathcal{L}_e(S_N/G)\}$ as the language generated by the closed-loop system.

The intuition of the abovementioned definition is as follows. Initially, the first control decision should be included in the initial set of control decisions provided by S_N. When extended string $\gamma_0\sigma_1\gamma_1 \ldots \sigma_n\gamma_n$ is executed, the next event σ_{n+1} should be both feasible in the plant and enabled by the control decision applied currently, i.e., γ_n. Furthermore, if σ_{n+1} is unobservable, then the supervisor should not change the control decision, i.e., $\gamma_{n+1} = \gamma_n$. On the other hand, if σ_{n+1} is observable, then the supervisor may choose a specific control decision from the new set of all possible control decisions provided by S_N, i.e., $\gamma_{n+1} \in S_N(\mathcal{O}(\gamma_0\sigma_1\gamma_1 \ldots \sigma_n\gamma_n\sigma_{n+1}))$. We denote by $\mathcal{L}_e^o(S_N/G)$ the set of extended strings that end up with observable events including the empty string, i.e.,

$$\mathcal{L}_e^o(S_N/G) = \mathcal{L}_e(S_N/G) \cap (\{\epsilon\} \cup (\Gamma\Sigma)^*(\Gamma\Sigma_o)).$$

We also denote by $\mathcal{L}_e^d(S_N/G)$, the set of extended strings that end up with control decisions, i.e.,

$$\mathcal{L}_e^d(S_N/G) = \mathcal{L}_e(S_N/G) \cap \Gamma(\Sigma\Gamma)^*.$$

The supervisor always issues a decision (first generates a set of control decisions and then randomly picks one) when an extended string ρ in $\mathcal{L}_e^o(S_N/G)$ is generated. Then, for any observable extended string $\rho \in \mathcal{O}(\mathcal{L}_e^o(S_N/G))$, we define

$$\hat{\mathcal{E}}_S(\rho) = \{\delta(\rho'|_\Sigma) \in X : \exists \rho' \in \mathcal{L}_e^o(S_N/G) \text{ s.t. } \mathcal{O}(\rho') = \rho\} \quad (9)$$

as the set of all possible states that can be reached immediately after observing the last event from the supervisor's point of view, i.e., the state estimate of the supervisor without the unobservable tail.

Once the supervisor issues the last control decision, the set of states that can be reached unobservably can be determined. Formally, for any extended string $\rho \in \mathcal{O}(\mathcal{L}_e^d(S_N/G))$, we define

$$\mathcal{E}_S(\rho) = \{\delta(\rho'|_\Sigma) \in X : \exists \rho' \in \mathcal{L}_e(S_N/G) \text{ s.t. } \mathcal{O}(\rho') = \rho\} \quad (10)$$

as the state-estimate of the supervisor with the unobservable tail included. These two state estimates can be computed recursively as follows [50]:

1) $\hat{\mathcal{E}}_S(\epsilon) = \{x_0\}$;
2) $\mathcal{E}_S(\rho') = UR_\gamma(\hat{\mathcal{E}}_S(\rho))$ for $\rho' = \rho\gamma \in \mathcal{O}(\mathcal{L}_e^d(S_N/G))$;
3) $\hat{\mathcal{E}}_S(\rho'') = NX_\sigma(\mathcal{E}_S(\rho'))$ for $\rho'' = \rho'\sigma \in \mathcal{O}(\mathcal{L}_e^o(S_N/G))$.

Here, we use subscript "S" to emphasize that these state-estimates are from the supervisor's point of view.

Example 2: Still consider system G in Fig. 1 with $\Sigma_c = \{c_1, c_2\}$ and $\Sigma_o = \{o_1, o_2, o_3\}$. Suppose that the initial nondeterministic decision set is $S_N(\epsilon) = \{\gamma_1, \gamma_2, \gamma_3, \gamma_4\}$, where $\gamma_1 = \emptyset, \gamma_2 = \{c_1\}, \gamma_3 = \{c_2\}$ and $\gamma_4 = \{c_1, c_2\}$. Then, we have $\gamma_1, \gamma_2, \gamma_3, \gamma_4 \in \mathcal{L}_e(S_N/G)$. Suppose that the supervisor chooses γ_2 initially. Then, we have $\gamma_2 c_1 \in \mathcal{L}_e(S_N/G)$ and since c_1 is unobservable, we have $\gamma_2 c_1 \gamma_2 \in \mathcal{L}_e(S_N/G)$. When o_2 occurs, $\rho = \gamma_2 c_1 \gamma_2 o_2 \in \mathcal{L}_e^o(S_N/G)$ becomes the first extended string that ends up with an observable event. Then, the information available to the supervisor is $\mathcal{O}(\rho) = \gamma_2 o_2$. The state estimate of the supervisor is $\hat{\mathcal{E}}_S(\gamma_2 o_2) = NX_{o_2}(UR_{\gamma_2}(\hat{\mathcal{E}}(\epsilon))) = NX_{o_2}(\{0, 1\}) = \{5\}$, i.e., the supervisor knows for sure that system is at state 5 by first choosing γ_2 and then observing o_2.

Suppose that the supervisor then issues γ_4 deterministically, i.e., $S_N(\gamma_2 o_2) = \{\gamma_4\}$ and the supervisor can only choose γ_4; this yields extended string $\rho' = \gamma_2 c_1 \gamma_2 o_2 \gamma_4 \in \mathcal{L}_e^d(S_N/G)$ with the last control decision information attached, the information available to the supervisor is $\mathcal{O}(\rho') = \gamma_2 o_2 \gamma_4$. Then, the state estimate of the supervisor is $\mathcal{E}_S(\gamma_2 o_2 \gamma_4) = UR_{\gamma_4}(\hat{\mathcal{E}}_S(\gamma_2 o_2)) = \{5, 6, 7, 8\}$.

Again, extended string $\rho'' = \gamma_2 c_1 \gamma_2 o_2 \gamma_4 c_1 \gamma_4 o_1 \in \mathcal{L}_e^o(S_N/G)$ can be generated with $\mathcal{O}(\rho'') = \gamma_2 o_2 \gamma_4 o_1$. Then, the state estimate of the supervisor becomes $\hat{\mathcal{E}}_S(\gamma_2 o_2 \gamma_4 o_1) = NX_{o_1}(\mathcal{E}_S(\gamma_2 o_2 \gamma_4)) = \{9, 10\}$.

Remark 1: Finally, we note that some nondeterministic control decision sets may contain redundancy, i.e., for $\{\gamma_1, \ldots, \gamma_n\} \in 2^\Gamma$, $\gamma_i \subset \gamma_j$ for some $i, j = 1, \ldots, n$. In this case, removing γ_i from the nondeterministic control decision set does not change the behavior of the closed-loop system. Formally, we say that a nondeterministic control decision set $\{\gamma_1, \ldots, \gamma_n\} \in 2^\Gamma$ is irredundant if its elements are incomparable, i.e., $\forall i, j = 1, \ldots, n : \gamma_i \not\subset \gamma_j$. For the sake of simplicity and without loss of generality, hereafter, we only consider irredundant nondeterministic control decision sets.

Remark 2: Note that our definition of nondeterministic supervisor in (8) is language-based, which may require infinite memory to realize. However, we will show later in this article that finite-memory supervisors are always sufficient for our purpose. For this case, one may also realize a nondeterministic supervisor by a nondeterministic finite-state automaton and the closed-loop behavior can be then computed by taking the synchronous composition between the plant and the supervisor automaton.

C. Opacity of Nondeterministic Control Systems

In the definition of opacity for the standard deterministic setting, the intruder model has been specified by A1–A3. Here, we still consider the same intruder model, but we explain A3 more clearly in the nondeterministic setting:

A3′ The intruder knows the functionality of the supervisor. That is, the intruder knows the set of all possible control decisions the supervisor may pick according to the control policy. However, it does not know which specific control decision the supervisor picks online.

This assumption is reasonable in many applications as long as the communication channel between supervisor and the actuator is reliable. Then, under this setting, when the supervisor observes $\rho \in \mathcal{O}(\mathcal{L}_e(S_N/G))$, the intruder can only observes $\rho|_\Sigma \in P(\mathcal{L}(S_N/G))$. Therefore, the state estimate of the intruder essentially is more uncertain, which needs to estimate all possible realizations consistent with the control policy and the observation. Formally, for any observable string $s \in P(\mathcal{L}(S_N/G))$, we define $X_I(s)$ as the state estimate of the intruder, i.e.,

$$X_I(s) = \{\delta(s') \in X : \exists s' \in \mathcal{L}(S_N/G) \text{ s.t. } P(s') = s\}. \quad (11)$$

Then, opacity of control systems under nondeterministic supervisors is defined as follows.

Definition 2: Let $S_N : (\Gamma\Sigma_o)^* \rightarrow 2^\Gamma$ be a nondeterministic supervisor. We say the closed-loop system S_N/G is opaque (w.r.t. Σ_o and X_S) if $\forall s \in P(\mathcal{L}(S_N/G)){:}X_I(s) \not\subseteq X_S$.

The state estimate of the supervisor and the state estimate of the intruder can be related as follows. Since the intruder observes strictly less than the supervisor, its estimate of the system is essentially the union of its estimate of all possible supervisor's knowledge about the system. To see this more clearly, for any observable string $s \in P(\mathcal{L}(S_N/G))$, we also define

$$\hat{\mathcal{E}}_I(s)=\{\hat{\mathcal{E}}_S(\rho) \in 2^X : \rho \in \mathcal{O}(\mathcal{L}_e^o(S_N/G))\,\text{s.t.}\,\rho|_\Sigma = s\} \quad (12)$$

$$\mathcal{E}_I(s)=\{\mathcal{E}_S(\rho) \in 2^X : \rho \in \mathcal{O}(\mathcal{L}_e^d(S_N/G))\,\text{s.t.}\,\rho|_\Sigma = s\} \quad (13)$$

as the intruder's estimates of the state-estimations of the supervisor. Note that $\hat{\mathcal{E}}_I(s)$ and $\mathcal{E}_I(s)$ are, respectively, the state estimate immediately after observing an observable event and the state-estimate with the unobservable tail included. Note that we use subscript "I" to emphasize that these estimates are from the intruder's point of view. Then, we have the following result that connects $\mathcal{E}_I$ and X_I.

Proposition 1: For any $s \in P(\mathcal{L}(S_N/G))$, we have

$$X_I(s) = \bigcup \mathcal{E}_I(s).$$

Proof: By the definitions of $\mathcal{E}_I(s), \mathcal{E}_S(\rho)$, $\mathcal{L}(S_N/G)$ and mapping $\mathcal{O}$, we have

$$\begin{aligned}
&\bigcup \mathcal{E}_I(s)\\
&= \bigcup \{\mathcal{E}_S(\rho) \in 2^X : \rho \in \mathcal{O}(\mathcal{L}_e^d(S_N/G)) \text{ s.t. } \rho|_\Sigma = s\}\\
&= \{x \in \mathcal{E}_S(\rho) : \rho \in \mathcal{O}(\mathcal{L}_e^d(S_N/G)) \text{ s.t. } \rho|_\Sigma = s\}\\
&= \{\delta(\rho'|_\Sigma) : \rho' \in \mathcal{L}_e(S_N/G) \text{ s.t. } \mathcal{O}(\rho')|_\Sigma = s\}\\
&= \{\delta(s') : s' \in \mathcal{L}(S_N/G) \text{ s.t. } P(s') = s\}\\
&= X_I(s).
\end{aligned}$$

■

Given a nonopaque system, our goal is to synthesize a nondeterministic supervisor that restricts the system behavior such that opacity is satisfied for the closed-loop system. The opacity enforcement synthesis problem is formulated as follows.

Problem 1 (Opacity Enforcement Problem): Given system G and secret states $X_S \subseteq X$, synthesize a partial observation nondeterministic supervisor $S_N : (\Gamma\Sigma_o)^* \rightarrow 2^\Gamma$, such that S_N/G is opaque w.r.t. X_S and Σ_o.

Remark 3: Compared with deterministic supervisors, the additional power of nondeterministic supervisor, in terms of opacity enforcement, relies on assumption A3$'$. That is, the intruder is aware of the functionality of the nondeterministic supervisor but cannot eavesdrop the specific control decisions issued by the supervisor. Note that, if the intruder is completely not aware of the functionality of the supervisor (no matter deterministic or nondeterministic), then it has to make state estimation based on the *open-loop* system G. For this case, using nondeterministic supervisors does not provide any additional power compared with deterministic supervisors, and it suffices to solve the *supervisor-unaware* deterministic opacity-enforcement problem; see, e.g., [42], [44]. If the intruder is aware of the functionality of the nondeterministic supervisor, but at the same time, is also capable of eavesdropping the control decisions issued by the nondeterministic supervisor, then this essentially means that the nondeterministic control information can be resolved by the intruder. For this case, using nondeterministic supervisors is still the same as using deterministic supervisors in terms of the capability of enforcing opacity. Then, it suffices to solve a *supervisor-aware* deterministic opacity-enforcement problem; see, e.g., [10], [38], [50].

IV. IS and Its Flow

In the formulation of the opacity enforcement problem, the domain of the supervisor is defined over languages. Therefore, the solution space is infinite in general and there is no prior knowledge to bound the memory of the supervisor. To effectively solve the synthesis problem, in this section, we restrict our attention to a class of *IS-based supervisors*, where the space of ISs is finite. We first define the IS in the nondeterministic control problem and then discuss how the selected IS evolves. Also, we define an IS-mapping that can encode an IS-based supervisor. Our method for synthesizing a nondeterministic supervisor is to first synthesize an IS-mapping and then encode a supervisor from it. To this end, we finally put forward an algorithm that decodes a nondeterministic supervisor from IS-mapping. We will show later in Section VI that restricting our attention to IS-based supervisors is without loss of generality for the solvability of the general nondeterministic supervisor opacity enforcement problem.

A. Proposed Information Structure

In the deterministic control problem, it is known that 2^X is sufficient to realize an opacity-enforcing supervisor [50]. That is, a deterministic supervisor can be encoded as a state-based mapping $S : 2^X \rightarrow \Gamma$, which can be decoded by recursively estimating the state of the system and making decision based on the state-estimate (IS).

In the nondeterministic control problem, the supervisor and the intruder observe different information. Hence, the supervisor needs to make decision based on both the state estimates of itself and that of the intruder. To separate the observations of the supervisor and the intruder, we propose the following IS space

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

Each IS $\imath \in I$ is in the form of $\imath = (m, \mathbf{m})$. Intuitively, the first component aims to represent the state estimate of supervisor, while the second component aims to represent intruder's knowledge of the supervisor.

Formally, given a nondeterministic supervisor S_N and let $\rho \in \mathcal{O}(\mathcal{L}_e^o(S_N/G))$ be a decision history observed by the supervisor. We define

$$\mathcal{I}_{S_N}(\rho) = (\hat{\mathcal{E}}_S(\rho), \hat{\mathcal{E}}_I(\rho|_\Sigma)) \in 2^X \times 2^{2^X}$$

as the IS reached by ρ under S_N. Clearly, we have $\hat{\mathcal{E}}_S(\rho) \in \hat{\mathcal{E}}_I(\rho|_\Sigma)$ for any ρ by definition. We also define

$$\mathcal{I}_{S_N} := \{\mathcal{I}_{S_N}(\rho) : \rho \in \mathcal{O}(\mathcal{L}_e^o(S_N/G))\}$$

the set of all ISs reached by S_N.

Definition 3: A nondeterministic supervisor $S_N : (\Gamma\Sigma_o)^* \to 2^\Gamma$ is said to be *information-state-based* (IS-based) if

$$\begin{aligned}&\forall \rho, \rho' \in \mathcal{O}(\mathcal{L}_e^o(S_N/G)) :\\ &\mathcal{I}_{S_N}(\rho) = \mathcal{I}_{S_N}(\rho') \Rightarrow S_N(\rho) = S_N(\rho'). \qquad (14)\end{aligned}$$

An IS-based supervisor only makes decisions based on its current IS rather than the entire history. Therefore, we can encode an IS-based supervisor as a partial IS-mapping.

Definition 4: We say a partial IS-mapping $\Theta : I \to 2^\Gamma$ *encodes* supervisor $S_N : (\Gamma\Sigma_o)^* \to 2^\Gamma$ if

$$\forall \rho \in \mathcal{O}(\mathcal{L}_e^o(S_N/G)) : \Theta(\mathcal{I}_{S_N}(\rho)) = S_N(\rho).$$

Our general approach for synthesizing a nondeterministic supervisor is to synthesize its IS-mapping encoding. Clearly, given an IS-based supervisor S_N, we can easily encode it as an IS-mapping $\Theta : I \to 2^\Gamma$, which is defined at each state in $\mathcal{I}_{S_N}$. On the other hand, however, given a partial IS-mapping $\Theta : I \to 2^\Gamma$, it is not straightforward how to *decode* an IS-based supervisor from it. In fact, not every partial IS-mapping $\Theta : I \to 2^\Gamma$ actually encodes an IS-based supervisor. As a necessary requirement, the partial IS-mapping should be defined at state $\imath_0 = (\{x_0\}, \{\{x_0\}\})$, which is the initial IS of any IS-based supervisor. Then, one can argue inductively that, for any reachable IS, the partial IS-mapping should be defined, which suggests that the domain of the partial IS-mapping should contain the "reachability closure" from the initial-state $\imath_0$; otherwise, the decoded supervisor will "get stuck" at those states where the IS-mapping is undefined.

To compute such an "reachability closure," we need to investigate how the IS evolves. As we discussed earlier, the first component of the IS can be computed recursively based on ρ. However, the question is how to compute the second component. To this end, we should not only know the control decision for history ρ, but should also know the control decisions for those ρ' such that $\rho|_\Sigma = \rho'|_\Sigma$. In the remaining part of this section, we will elaborate on how $\hat{\mathcal{E}}_I(\rho|_\Sigma)$ can be computed recursively and by what information.

B. Micro/Macro States and Decisions

Before we proceed further, we define some necessary concepts. First, we introduce the notion of microstate, which is used to represent the knowledge of supervisor.

Definition 5 (Microstate): A *microstate* $m \in 2^X$ is a set of states and we define $M = 2^X$ as the set of microstates. An *augmented microstate* $m^+ = (m, \gamma) \in 2^X \times \Gamma$ is a microstate augmented with a control decision and we define $M^+ = 2^X \times \Gamma$ as the set of augmented microstates.

Then, we define the notion of macrostate, which is used to represent the knowledge of intruder about the supervisor.

Definition 6 (Macrostate): A *macrostate* $\mathbf{m} = \{m_1, m_2, \ldots, m_n\} \subseteq 2^X$ is a set of microstates and we define $\mathbb{M} = 2^{2^X}$ as the set of macrostates. An *augmented macrostate* $\mathbf{m}^+ = \{(m_1, \gamma_1), (m_2, \gamma_2), \ldots, (m_n, \gamma_n)\} \subseteq 2^X \times \Gamma$ is a set of augmented microstates and we define $\mathbb{M}^+ = 2^{2^X \times \Gamma}$ as the set of augmented macrostates.

In order to estimate the knowledge of the intruder, we should not only know the decision of the supervisor at a specific microstate, but also should know the decisions at other microstates in the same macrostate, which means that these microstates are indistinguishable from the intruder's point of view. This leads to the notion of macrocontrol-decision.

Definition 7 (Macrocontrol-Decision): A *macrocontrol-decision* is a set in the form of

$$d = \{(m_1, \Gamma_1), (m_2, \Gamma_2), \ldots, (m_n, \Gamma_n)\} \subseteq 2^X \times 2^\Gamma$$

where each (m_i, Γ_i) is a pair of microstate and a nondeterministic control decision (a set of control decisions). We denote by $D = 2^{2^X \times 2^\Gamma}$ the set of macrocontrol-decisions.

Let $\mathbf{m} = \{m_1, m_2, \ldots, m_n\} \in \mathbb{M}$ be a macrostate and $d \in D$ be a macrocontrol-decision. We say that d is *compatible* with $\mathbf{m}$ if it is in the form of

$$d = \{(m_1, \Gamma_1), (m_2, \Gamma_2), \ldots, (m_n, \Gamma_n)\} \subseteq 2^X \times 2^\Gamma$$

i.e., d essentially assigns each microstate $m_i \in \mathbf{m}$ a nondeterministic control decision $\Gamma_i \in 2^\Gamma$.

The unobservable reach of a macrocontrol-decision $d \in D$ is defined by

$$\odot(d) = \{(m,'\gamma) : \exists (m, \Gamma) \in d, \gamma \in \Gamma \text{ s.t. } m' = UR_\gamma(m)\}.$$

Let $\mathbf{m}^+$ be an augmented macrostate and $\sigma \in \Sigma_o$ be an observable event. Then, the observable reach of $\mathbf{m}^+$ upon the occurrence of σ is defined as

$$\begin{aligned}&\widehat{NX}_\sigma(\mathbf{m}^+)\\ &\quad = \{m' : \exists (m, \gamma) \in \mathbf{m}^+ \text{ s.t. } m' = NX_\sigma(m) \wedge \sigma \in \gamma\}.\end{aligned}$$

C. Information-Flow Analysis

Now, suppose that an IS-mapping $\Theta : I \to 2^\Gamma$ that encodes an IS-based supervisor S_N is given. Let $\mathbf{m} = \{m_1, \ldots, m_k\}$ be a macrostate representing the intruder's estimate of the supervisor's knowledge. We define

$$d_\Theta(\mathbf{m}) = \{(m_1, \Theta(m_1, \mathbf{m})), \ldots, (m_k, \Theta(m_k, \mathbf{m}))\}$$

as the macrocontrol-decision made by IS-based supervisor at macrostate $\mathbf{m}$.

Initially, the state-estimate of the supervisor is $m_0 = \{x_0\}$ and the intruder believes that this is the unique estimate of the system with estimate $\mathbf{m}_0 = \{m_0\}$, which forms the initial IS $\imath_0 = (m_0, \mathbf{m}_0)$.

Then, the supervisor issues a nondeterministic decision set $\Gamma_0 = S_N(\epsilon) = \Theta(m_0, \mathbf{m}_0)$. Note that, we have prespecified that the supervisor is IS-based. Therefore, we denote the control decision information at this instant by a macrocontrol-decision $d_\Theta(\mathbf{m}_0) = \{(m_0, \Theta(m_0, \mathbf{m}_0))\}$, which means that "*the supervisor will make control decision if its state-estimate is* m_0". Note that, at this instant, $d_\Theta(\mathbf{m}_0)$ is a singleton as the intruder does not yet have ambiguity about the supervisor, i.e., $\mathbf{m}_0 = \{m_0\}$.

Once the allowed decision set Γ_0 is specified, the supervisor will pick a concrete control decision in it. The intruder does not know which decision is chosen while the supervisor knows. Suppose that $\Gamma_0 = \{\gamma_0^1, \ldots, \gamma_0^k\}$ contains k control decisions. Then, the intruder's knowledge about the supervisor becomes

$$\begin{aligned}\mathbf{m}_0^+ =& \odot (d_\Theta(\mathbf{m}_0))\\ &= \{(UR_{\gamma_0^1}(m_0), \gamma_0^1), \ldots, (UR_{\gamma_0^k}(m_0), \gamma_0^k)\}\\ &= \{(m_0^1, \gamma_0^1), \ldots, (m_0^k, \gamma_0^k)\} \qquad (15)\end{aligned}$$

which means that the supervisor's estimate (with the unobservable tail) is possibly $UR_{\gamma_0^i}(m_0)$ and the control decision applied is γ_0^i. Note that, the supervisor knows precisely which augmented microstate (m_0^i, γ_0^i) it is at.

Then, when a new observable event $\sigma \in \Sigma_o$ occurs, and the intruder updates its knowledge to

$$\mathbf{m}_1 = \widehat{NX}_\sigma(\mathbf{m}_0^+) = \{m_1^1, \ldots, m_1^p\}. \qquad (16)$$

which is a macrostate containing at most k microstates, i.e., $p \leqslant k$.

Now, let us assume that, after some steps, the intruder's knowledge about the supervisor (immediately after the occurrence of an observable event) is

$$\mathbf{m}_n = \{m_n^1, \ldots, m_n^k\}.$$

Note that the supervisor knows the exact state estimate, i.e., $m_n^i \in \mathbf{m}_n$, and for each m_n^i, it allows nondeterministic decision set $\Gamma_i = \Theta(m_n^i, \mathbf{m}_n)$ as we assume the supervisor is IS-based and is encoded by Θ. Therefore, the corresponding macrocontrol-decision is

$$d_\Theta(\mathbf{m}_n) = \{(m_n^1, \Theta(m_n^1, \mathbf{m}_n)), \ldots, (m_n^k, \Theta(m_n^k, \mathbf{m}_n))\}. \quad (17)$$

Then, the intruder's knowledge about the supervisor is updated by adding this control information

$$\mathbf{m}_n^+ = \odot(d_\Theta(\mathbf{m}_n)),$$

which is an augmented marcostate containing at most $\sum_{i=1}^k |\Theta(m_n^i, \mathbf{m}_n)|$ augmented microstates.

Based on the abovementioned discussion, suppose that the intruder observes $\sigma_1 \cdots \sigma_n \in P(\mathcal{L}(S_N/G))$ and by assuming the fact that S_N is an IS-based supervisor encoded by Θ, it induces the following sequence:

$$\mathbf{m}_0 \xrightarrow{d_0} \mathbf{m}_0^+ \xrightarrow{\sigma_1} \mathbf{m}_1 \xrightarrow{d_1} \ldots \xrightarrow{\sigma_n} \mathbf{m}_n \xrightarrow{d_n} \mathbf{m}_n^+ \qquad (18)$$

where $\mathbf{m}_0 = \{\{x_0\}\}$, $d_i = d_\Theta(\mathbf{m}_i)$, $\mathbf{m}_i^+ = \odot(d_i)$, and $\mathbf{m}_{i+1} = \widehat{NX}_{\sigma_{i+1}}(\mathbf{m}_i^+)$. We note that σ_{i+1} is defined at $\mathbf{m}_i^+$ iff there exist $(m, \gamma) \in \mathbf{m}_i^+$ and $x \in m$ such that $\delta(x, \sigma_{i+1})!$ and $\sigma_{i+1} \in \gamma$. Therefore, the sequence in (18) is uniquely defined when $\sigma_1 \cdots \sigma_n$ and Θ are fixed; it is independent from the actual online choice of the supervisor at each instant.

Now we are ready to specify the reachability closure of an IS-mapping. Formally, let $\Theta : I \to 2^\Gamma$ be a partial IS-mapping and $\imath = (m, \mathbf{m}) \in I$ be an IS. Then, the *reachability closure* of $\imath$ under Θ, denoted by $\text{Reach}_\Theta(\imath) \subseteq I$, is defined recursively as follows:

1) $\imath \in \text{Reach}_\Theta(\imath)$;

2) $\imath' = (m,' \mathbf{m}') \in \text{Reach}_\Theta(\imath)$ if
   a) $m' \in \mathbf{m}'$;
   b) there exists $\imath'' = (m,'' \mathbf{m}'') \in \text{Reach}_\Theta(\imath)$ such that $\mathbf{m}'' \xrightarrow{d_\Theta(\mathbf{m}'')} \mathbf{m}''^+ \xrightarrow{\sigma} \mathbf{m}'$ for some $\sigma \in \Sigma_o$.

D. Property of the IS

The abovementioned analysis of information-flow is heuristic. In this section, we formally show that the proposed information updating rule indeed yields the state estimate of the intruder in the controlled system.

Theorem 1: Let Θ be an IS-mapping that encodes an IS-based supervisor S_N and $s = \sigma_1 \ldots \sigma_k \in P(\mathcal{L}(S_N/G))$ be an observable string available to the intruder. Let $\mathbf{m}_k$ and $\mathbf{m}_k^+$ be states induced by s and Θ according to (18). Then, we have

i) $\mathbf{m}_k = \hat{\mathcal{E}}_I(s)$;

ii)

$$\mathbf{m}_k^+ = \left\{(\mathcal{E}_S(\rho\gamma), \gamma) : \begin{array}{l}\rho \in \mathcal{O}(\mathcal{L}_e^o(S_N/G)) \textit{ s.t.}\\ \rho|_\Sigma = s \textit{ and } \gamma \in S_N(\rho)\end{array}\right\}.$$

Proof: We prove by induction on the length of s.

Induction Basis: For $|s| = 0$, i.e., $s = \epsilon$, from the definition of $\hat{\mathcal{E}}_I(s)$, we know that

$$\begin{aligned}\hat{\mathcal{E}}_I(\epsilon) &= \{\hat{\mathcal{E}}_S(\rho) \in 2^X : \rho \in \mathcal{O}(\mathcal{L}_e^o(S_N/G)) \text{ s.t. } \rho|_\Sigma = \epsilon\}\\ &= \{\hat{\mathcal{E}}_S(\epsilon)\}\\ &= \{\{\delta(\epsilon)\}\}\\ &= \{\{x_0\}\}\\ &= \mathbf{m}_0.\end{aligned}$$

Since $\mathbf{m}_0^+ = \odot(d_\Theta(\mathbf{m}_0))$ and $d_\Theta(\mathbf{m}_0) = \{(m_0, S_N(\epsilon))\}$, we have

$$\begin{aligned}\mathbf{m}_0^+ =& \odot (d_\Theta(\mathbf{m}_0))\\ &= \{(UR_\gamma(m_0), \gamma) : \gamma \in S_N(\epsilon)\}\\ &= \left\{(UR_\gamma(\hat{\mathcal{E}}_S(\epsilon)), \gamma) : \gamma \in S_N(\epsilon)\right\}\\ &= \{(\mathcal{E}_S(\gamma), \gamma) : \gamma \in S_N(\epsilon)\}.\end{aligned}$$

Note that $\rho = \epsilon$ is the only extended string in $\mathcal{O}(\mathcal{L}_e^o(S_N/G))$ such that $\rho|_\Sigma = s$. This completes the induction basis.

Induction Step: Let us assume that Theorem 1 holds for $|s| = k$. Then, we want to prove the case of $s\sigma_{k+1} \in P(\mathcal{L}(S_N/G))$. By the induction hypothesis, we know that

$$\mathbf{m}_k^+ = \left\{(\mathcal{E}_S(\rho\gamma), \gamma) : \begin{array}{l}\rho \in \mathcal{O}(\mathcal{L}_e^o(S_N/G)) \text{ s.t.}\\ \rho|_\Sigma = s \text{ and } \gamma \in S_N(\rho)\end{array}\right\}.$$

Then, we have

$$\begin{aligned}
&\mathbf{m}_{k+1}\\
&= \widehat{NX}_{\sigma_{k+1}}(\mathbf{m}_k^+)\\
&= \{NX_{\sigma_{k+1}}(m) : (m,\gamma) \in \mathbf{m}_k^+, \sigma_{k+1} \in \gamma\}\\
&= \left\{NX_{\sigma_{k+1}}(\mathcal{E}_S(\rho\gamma)) : \begin{array}{r} \rho \in \mathcal{O}(\mathcal{L}_e^o(S_N/G)) \text{ s.t.}\\ \rho|_\Sigma = s, \gamma \in S_N(\rho) \text{ and } \sigma_{k+1} \in \gamma \end{array}\right\}\\
&= \{\hat{\mathcal{E}}_S(\rho\gamma\sigma_{k+1}) : \rho \in \mathcal{O}(\mathcal{L}_e^o(S_N/G)) \text{ s.t. } \rho|_\Sigma = s\}\\
&= \{\hat{\mathcal{E}}_S(\rho') : \rho' \in \mathcal{O}(\mathcal{L}_e^o(S_N/G)) \text{ s.t. } \rho'|_\Sigma = s\sigma_{k+1}\}\\
&= \hat{\mathcal{E}}_I(s\sigma_{k+1}).
\end{aligned}$$

For $\mathbf{m}_{k+1}^+ = \odot(d_\Theta(\mathbf{m}_{k+1}))$. Suppose that $d_\Theta(\mathbf{m}_{k+1}) = \{(m_{k+1}^1, \Theta(m_{k+1}^1, \mathbf{m}_{k+1})), \ldots, (m_{k+1}^n, \Theta(m_{k+1}^n, \mathbf{m}_{k+1}))\}$. Note that we have

$$\begin{aligned}
\mathbf{m}_{k+1} &= \{\hat{\mathcal{E}}_S(\rho) : \rho \in \mathcal{O}(\mathcal{L}_e^o(S_N/G)) \text{ s.t. } \rho|_\Sigma = s\sigma_{k+1}\}\\
&= \{m_{k+1}^1, \ldots, m_{k+1}^n\}.
\end{aligned}$$

For each $\rho \in \mathcal{O}(\mathcal{L}_e^o(S_N/G))$ such that $\rho|_\Sigma = s\sigma_{k+1}$, since S_N is IS-based, we have $S_N(\rho) = \Theta(\hat{\mathcal{E}}_S(\rho), \mathbf{m}_{k+1})$. Then, we have the following:

$$\begin{aligned}
&\mathbf{m}_{k+1}^+\\
&= \odot(d_\Theta(\mathbf{m}_{k+1}))\\
&= \{(UR_\gamma(m), \gamma) : \exists (m, \Gamma) \in d_\Theta(\mathbf{m}_{k+1}) \text{ s.t. } \gamma \in \Gamma\}\\
&= \left\{(UR_\gamma(\hat{\mathcal{E}}_S(\rho)), \gamma) : \begin{array}{r} \rho \in \mathcal{O}(\mathcal{L}_e^o(S_N/G)) \text{ s.t.}\\ \gamma \in S_N(\rho) \text{ and } \rho|_\Sigma = s\sigma_{k+1} \end{array}\right\}\\
&= \left\{(\mathcal{E}_S(\rho\gamma), \gamma) : \begin{array}{r} \rho \in \mathcal{O}(\mathcal{L}_e^o(S_N/G)), \text{ s.t.}\\ \gamma \in S_N(\rho) \text{ and } \rho|_\Sigma = s\sigma_{k+1} \end{array}\right\}.
\end{aligned}$$

This completes the induction step, i.e., (ii) holds. ■

For any augmented macrostate $\mathbf{m}^+$, we define

$$M(\mathbf{m}^+) = \{m \in M : (m, \gamma) \in \mathbf{m}^+\}$$

as the macrostate obtained by removing the control decision components from $\mathbf{m}^+$. Then, the following result reveals that the abovementioned defined states set $M(\mathbf{m}_k^+)$ is indeed the state estimate of the intruder $\mathcal{E}_I(s)$.

Corollary 1: Let Θ be an IS-mapping that encodes an IS-based supervisor S_N and $s = \sigma_1 \ldots \sigma_k \in P(\mathcal{L}(S_N/G))$ be an observable string available to the intruder. Let $\mathbf{m}_k^+$ be the state reached according to the information-flow. Then, we have

$$M(\mathbf{m}_k^+) = \mathcal{E}_I(s).$$

Proof: By Theorem 1, we have

$$\mathbf{m}_k^+ = \left\{(\mathcal{E}_S(\rho\gamma), \gamma) : \begin{array}{r} \rho \in \mathcal{O}(\mathcal{L}_e^o(S_N/G)) \text{ s.t.}\\ \rho|_\Sigma = s \text{ and } \gamma \in S_N(\rho) \end{array}\right\}.$$

Therefore

$$\begin{aligned}
M(\mathbf{m}_k^+) &= \left\{\mathcal{E}_S(\rho\gamma) : \begin{array}{r} \rho \in \mathcal{O}(\mathcal{L}_e^o(S_N/G)) \text{ s.t.}\\ \rho|_\Sigma = s \text{ and } \gamma \in S_N(\rho) \end{array}\right\}\\
&= \{\mathcal{E}_S(\rho') : \rho' \in \mathcal{O}(\mathcal{L}_e^d(S_N/G)) \text{ s.t. } \rho'|_\Sigma = s\}\\
&= \mathcal{E}_I(s).
\end{aligned}$$

■

We explain the abovementioned concepts by the following example.

Example 3: Let us still consider system G in Fig. 1. We consider a nondeterministic supervisor S_N defined by

$$\forall \rho \in (\Gamma\Sigma_o)^* : S_N(\rho) = \{\{c_1\}, \{c_2\}\}.$$

Clearly, this supervisor is IS-based and it can be encoded by IS-mapping $\Theta : I \to 2^\Gamma$ such that $\forall \imath \in I : \Theta(\imath) = \{\{c_1\}, \{c_2\}\}$.

Initially, the supervisor's estimate is $m_0 = \{0\}$ and the intruder's estimate of supervisor's estimation is $\mathbf{m}_0 = \{\{0\}\}$, where the macrocontrol-decision induced by Θ is

$$d_\Theta(\mathbf{m}_0) = \{(\{0\}, \Theta(\{0\}, \{\{0\}\}))\} = \{(\{0\}, \{\{c_1\}, \{c_2\}\})\}.$$

Then, the intruder's knowledge is updated to

$$\begin{aligned}
\mathbf{m}_0^+ &= \odot(d_\Theta(\mathbf{m}_0))\\
&= \{(UR_{\{c_1\}}(\{0\}), \{c_1\}), (UR_{\{c_2\}}(\{0\}), \{c_2\})\}\\
&= \{(\{0,1\}, \{c_1\}), (\{0,3\}, \{c_2\})\}.
\end{aligned}$$

When event o_1 is observed, the intruder updates its knowledge to

$$\begin{aligned}
\mathbf{m}_1 &= \widehat{NX}_{o_1}(\mathbf{m}_0^+) = \{NX_{o_1}(\{0,1\}), NX_{o_1}(\{0,3\})\}\\
&= \{\{4\}, \{5\}\},
\end{aligned}$$

which means that the intruder guesses that the supervisor's state-estimate is either $\{4\}$ or $\{5\}$ based on the information available. Again, the macrocontrol-decision at $\mathbf{m}_1$ is

$$\begin{aligned}
d_\Theta(\mathbf{m}_1) &= \{(\{4\}, \Theta(\{4\}, \mathbf{m}_1)), (\{5\}, \Theta(\{4\}, \mathbf{m}_1))\}\\
&= \{(\{4\}, \{\{c_1\}, \{c_2\}\}), (\{5\}, \{\{c_1\}, \{c_2\}\})\},
\end{aligned}$$

which leads to

$$\begin{aligned}
\mathbf{m}_1^+ &= \odot(d_\Theta(\mathbf{m}_1))\\
&= \left\{\begin{array}{l} (UR_{\{c_1\}}(\{4\}), \{c_1\}), (UR_{\{c_2\}}(\{4\}), \{c_2\}),\\ (UR_{\{c_1\}}(\{5\}), \{c_1\}), (UR_{\{c_2\}}(\{5\}), \{c_2\}) \end{array}\right\}\\
&= \{(\{4\},\{c_1\}), (\{4\},\{c_2\}), (\{5,6\},\{c_1\}), (\{5,7\},\{c_2\})\}.
\end{aligned}$$

Similarly, from $\mathbf{m}_1^+$, observations can be observed and so forth.

E. Decode Supervisor From IS-Mapping

Finally, we are ready to discuss how to *decode* an IS-based nondeterministic supervisor from an IS-mapping $\Theta : I \to 2^\Gamma$. The decoded nondeterministic supervisor is denoted by S_Θ. Let $\textsc{Dom}(\Theta) = \{\imath \in I : \Theta(\imath)!\}$ be the domain of Θ. We say that IS-mapping Θ is reachability-closed if

$$\textsc{Reach}_\Theta(\imath_0) \subseteq \textsc{Dom}(\Theta)$$

Algorithm 1: Online Decoding of IS-Mapping Θ

```
$m \leftarrow \{x_0\}$ and $\mathbf{m} \leftarrow \{\{x_0\}\}$ and $\rho \leftarrow \epsilon$;
while $\rho = \epsilon$ or a new event $\sigma \in \Sigma_o$ is observed do
    if a new event $\sigma \in \Sigma_o \cap \gamma$ is observed then
        $m \leftarrow NX_\sigma(m^+)$ and $\mathbf{m} \leftarrow \widehat{NX}_\sigma(\mathbf{m}^+)$;
        $\rho \leftarrow \rho\sigma$;
    Define $S_\Theta(\rho) \leftarrow \Theta(m, \mathbf{m})$ as the current
    non-deterministic control decision;
    Randomly pick $\gamma \in S_\Theta(\rho)$ and apply this control
    decision online;
    $m^+ \leftarrow UR_\gamma(m)$ and $\mathbf{m}^+ \leftarrow \odot(d_\Theta(\mathbf{m}))$;
    $\rho \leftarrow \rho\gamma$;
```

where $\imath_0 = (\{x_0\}, \{\{x_0\}\})$ is the initial IS. Clearly, Θ is necessary to be reachability-closed; otherwise, the supervisor cannot make decision after some executions. Without loss of generality, we can further assume that $\text{Reach}_\Theta(\imath_0) = \text{Dom}(\Theta)$ as the mapping information of those unreachable states are not used.

When IS-mapping Θ is reachability-closed, we can decode a supervisor S_Θ as follows. For any decision history $\rho = \gamma_0\sigma_1\gamma_1 \ldots \gamma_{n-1}\sigma_n$, we have

$$S_\Theta(\rho) = \Theta(\hat{\mathcal{E}}_S(\rho), \hat{\mathcal{E}}_I(\rho|_\Sigma)). \tag{19}$$

Note that, based on the previous discussion, both $\hat{\mathcal{E}}_S(\rho)$ and $\hat{\mathcal{E}}_I(\rho|_\Sigma)$ can be computed recursively based on Θ. Therefore, in practice, $S_\Theta(\rho)$ can be executed online according to Algorithm 1. Specifically, we use parameters $m, m^+, \mathbf{m}$ and $\mathbf{m}^+$ to represent $\hat{\mathcal{E}}_S(\rho), \mathcal{E}_S(\rho), \hat{\mathcal{E}}_I(\rho|_\Sigma)$, and $\mathcal{E}_I(\rho|_\Sigma)$, respectively. Note that the updates of $\mathbf{m}$ and $\mathbf{m}^+$ use the online observation σ and the IS-mapping Θ to generate a nondeterministic control decision set Γ, in which an actual control decision applied $\gamma \in \Gamma$ is chosen randomly. However, the updates of m and m^+ only use the online observation σ and the actual decision γ applied.

By understanding how an IS-mapping Θ can be decoded as an IS-based supervisor, hereafter, we will also refer to a reachability-closed IS-mapping Θ as an IS-based supervisor directly. In order to solve the general opacity enforcement problem as formulated in Problem 1, our approach is to restrict our solution space to IS-based supervisors and solve the following IS-mapping synthesis problem.

Problem 2 (IS-Based Opacity Enforcement Problem): Given system G and secret states $X_S \subseteq X$, synthesize an IS-based supervisor $S_\Theta : (\Gamma\Sigma_o)^* \rightarrow 2^\Gamma$ decoded from IS-mapping $\Theta : I \rightarrow 2^\Gamma$, such that S_Θ/G is opaque w.r.t. X_S and Σ_o.

Remark 4: Problem 2 essentially restricts the solution space of Problem 1 to a finite domain. Clearly, if there exists an IS-based supervisor that enforces opacity, then there exists a nondeterministic opacity-enforcing supervisor. However, the following question arises immediately: *whether or not the nonexistence of an IS-based supervisor also implies the nonexistence of a general supervisor?* We will show later in Section VI that there exists a nondeterministic opacity enforcing supervisor *if and only if* there exists an IS-based one. In other words, restricting our attention to Problem 2 is without loss of generality for the solvability of Problem 1.

V. Synthesis of IS-Based Supervisors

In this section, we discuss how to synthesize an IS-based supervisor that enforces opacity. We first introduce the structure of the G-BTS. Then, we present a synthesis algorithm that returns a solution to Problem 2.

A. Bipartite Transition System (BTS)

By the analysis in the previous section, we see that the update of the intruder's knowledge consists of the following two steps: one is when the supervisor picks a macrocontrol-decision and the other is when a new observable event occurs. To separate these two steps, we adopt the idea of the BTS proposed in [50]. Here, we call the proposed structure G-BTS as it captures, in a more general manner, both the supervisor's estimate and the intruder's knowledge about the supervisor, while the original BTS in [50] only captures the supervisor's estimate.

Definition 8: A G-BTS T w.r.t. G is a 7-tuple

$$T = (Q_Y, Q_Z, h_{YZ}, h_{ZY}, \Sigma_o, D, y_0)$$

where

1) $Q_Y \subseteq \mathbb{M}$ is a set of macrostates;
2) $Q_Z \subseteq \mathbb{M}^+$ is the set of augmented macrostates;
3) $h_{YZ} : Q_Y \times D \rightarrow Q_Z$ is the transition function from Y-states to Z-states satisfying: for any $h_{YZ}(\mathbf{m}, d) = \mathbf{m}^+$, we have
    i) d is *compatible* with $\mathbf{m}$;
    ii) $\mathbf{m}^+ = \odot(d)$.
4) $h_{ZY} : Q_Z \times \Sigma_o \rightarrow Q_Y$ is the transition function from Z-states to Y-states satisfying: for any $h_{ZY}(\mathbf{m}^+, \sigma) = \mathbf{m}$, $\sigma \in \Sigma_o$, we have
    i) $\mathbf{m} = \widehat{NX}_\sigma(\mathbf{m}^+)$.
5) D is the set of macrocontrol-decisions;
6) Σ_o is the set of observable events of system G;
7) $y_0 = \{\{x_0\}\} \in Q_Y$ is the initial Y-state.

The G-BTS essentially captures the information-flow analyzed in Section IV. Specifically, at each Y-state, the IS-based supervisor makes a macrocontrol-decision d and then moves to a Z-state by updating the intruder's knowledge via unobservable reaches under the issued macrocontrol-decision d. When a new observable event $\sigma \in \Sigma_o$ occurs at a Z-state, we move to a Y-state by computing the observable reach, and so forth.

Example 4: Again, we consider system G in Fig. 1. An example of the G-BTS is shown in Fig. 2(a), in which rectangular states represent Y-states and oval states represent Z-states. Some states are omitted in Fig. 2(a) for simplicity. States are named by $s_1, \ldots, s_{32}$. The initial Y-state is $s_1 = \{\{0\}\}$, from which macrocontrol-decisions $d_1, \ldots, d_5$ that are compatible with s_1 can be made. For example, if the macrocontrol-decision made is $d_5 = \{(\{0\}, \{\{c_1\}, \{c_2\}\})\}$, then we move to Z-state $s_{10} = \odot(d_5)$. From this state, observable events o_1 and o_2 can occur, and both lead to the same Y-state s_{18}. From Y-state s_{18}, macrocontrol-decisions $d_9, \ldots, d_{13}$ that are compatible with s_{18}

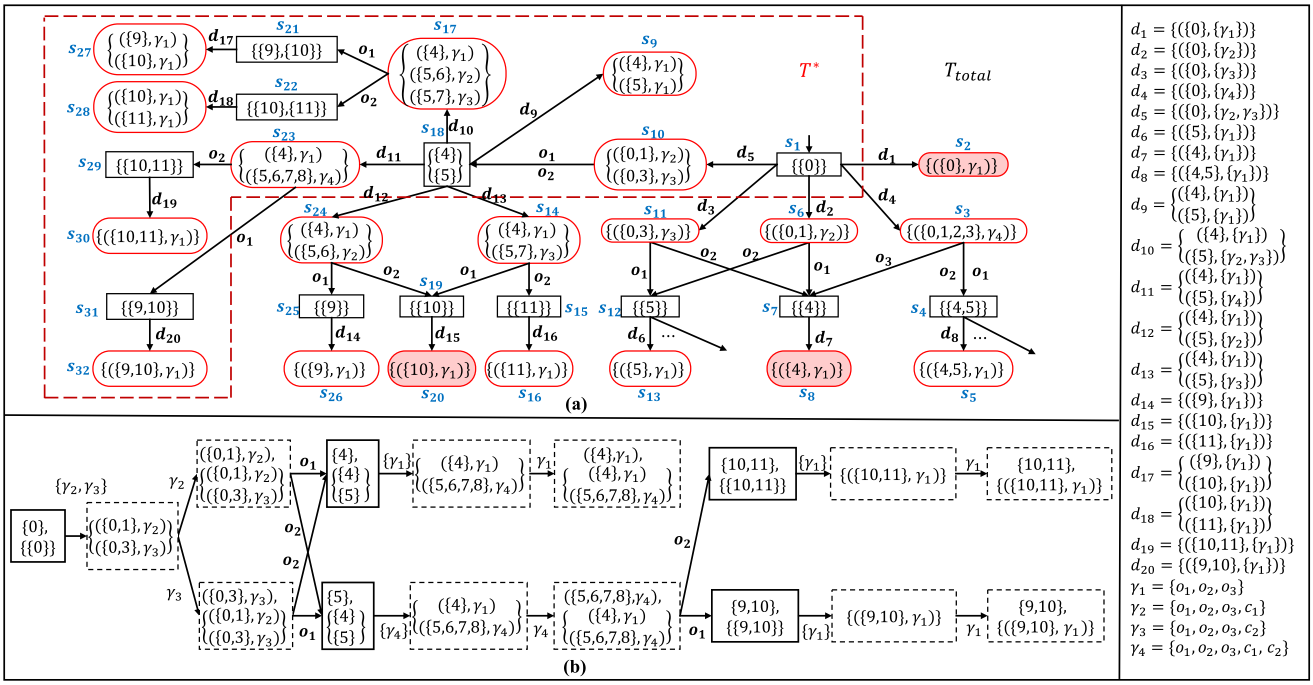
T^*
T_{total}
s_1 {{0}}
s_2 {({0}, γ_1)}
s_3 {({0,1,2,3}, γ_4)}
(a)
(b)
$d_1 = \{(\{0\},\{\gamma_1\})\}$
$d_2 = \{(\{0\},\{\gamma_2\})\}$
$d_3 = \{(\{0\},\{\gamma_3\})\}$
$d_4 = \{(\{0\},\{\gamma_4\})\}$
$d_5 = \{(\{0\},\{\gamma_2,\gamma_3\})\}$
$d_6 = \{(\{5\},\{\gamma_1\})\}$
$d_7 = \{(\{4\},\{\gamma_1\})\}$
$d_8 = \{(\{4,5\},\{\gamma_1\})\}$
$d_9 = \{(\{4\},\{\gamma_1\}), (\{5\},\{\gamma_1\})\}$
$d_{10} = \{(\{4\},\{\gamma_1\}), (\{5\},\{\gamma_2,\gamma_3\})\}$
$d_{11} = \{(\{4\},\{\gamma_1\}), (\{5\},\{\gamma_4\})\}$
$d_{12} = \{(\{4\},\{\gamma_1\}), (\{5\},\{\gamma_2\})\}$
$d_{13} = \{(\{4\},\{\gamma_1\}), (\{5\},\{\gamma_3\})\}$
$d_{14} = \{(\{9\},\{\gamma_1\})\}$
$d_{15} = \{(\{10\},\{\gamma_1\})\}$
$d_{16} = \{(\{11\},\{\gamma_1\})\}$
$d_{17} = \{(\{9\},\{\gamma_1\}), (\{10\},\{\gamma_1\})\}$
$d_{18} = \{(\{10\},\{\gamma_1\}), (\{11\},\{\gamma_1\})\}$
$d_{19} = \{(\{10,11\},\{\gamma_1\})\}$
$d_{20} = \{(\{9,10\},\{\gamma_1\})\}$
$\gamma_1 = \{o_1,o_2,o_3\}$
$\gamma_2 = \{o_1,o_2,o_3,c_1\}$
$\gamma_3 = \{o_1,o_2,o_3,c_2\}$
$\gamma_4 = \{o_1,o_2,o_3,c_1,c_2\}$

Fig. 2. (a) Example of the G-BTS, where rectangular states represent Y-states and oval states represent Z-states. (b) Decision diagram of the synthesized nondeterministic supervisor.

can be made. If the macrocontrol-decision made is d_{11}, then we move to $s_{23} = \odot(d_{11})$ and so forth.

B. Synthesis of IS-Based Supervisors

Now, we present how to synthesize IS-based nondeterministic opacity-enforcing supervisors represented by IS-mappings. Given a G-BTS T, for any Y-state $y \in Q_Y$, we define

$$C_T(y) := \{d \in D : h_{YZ}(y,d)!\}$$

as the set of macrocontrol-decisions defined at y in T. Also, we say that a Y-state y is *consistent* if $C_T(y) \neq \emptyset$; and a Z-state z is *consistent* if, for any $\sigma \in \Sigma_o$, we have

$$h_{ZY}(z,\sigma)! \Leftrightarrow (\exists (m,\gamma) \in z)[NX_\sigma(m) \neq \emptyset \wedge \sigma \in \gamma].$$

Intuitively, a Y-state is consistent if at least one macrocontrol-decision is defined and a Z-state is consistent if all feasible events are defined. Consistency is required for the purpose of control as the supervisor should be able to provide a control decision for any observation. We denote by Q^T_{const} the set of consistent states in T and we say that T is consistent if all reachable states are consistent.

As discussed earlier, we restrict our attention to IS-based supervisors. Our approach for synthesizing nondeterministic opacity-enforcing supervisors consists of the following two steps:

1) construct the largest consistent G-BTS in which all states are not secret-revealing;
2) extract one IS-based supervisor in the form of an IS-mapping from this largest G-BTS.

Since such an IS-based supervisor is extracted from T, by Theorem 1 and Corollary 1, we know that, upon the occurrence of any decision history, the Z-state $z \in \mathbb{M}^+$ reached is essentially the set of all possible state-estimates of the supervisor. Moreover, by Proposition 1, we know that $\bigcup M(z) = X_I(s)$, where s is the observation leading to the Z-state. Therefore, to make sure that the closed-loop system S_Θ/G is opaque, it suffices to guarantee that, for any Z-state $z \in \mathbb{M}^+$ reached, we have

$$\bigcup M(z) \not\subseteq X_S.$$

To this end, we define

$$Q_{\text{reveal}} = \{z \in \mathbb{M}^+ : \bigcup M(z) \subseteq X_S\}$$

as the set of *secret-revealing* Z-states.

In order to synthesize an IS-based supervisor, first, we construct the largest G-BTS w.r.t. G that enumerates all the feasible transitions satisfying the constraints of h_{YZ} and h_{ZY}. We denote such an all-feasible G-BTS by T_{total}. Then, we need to delete all secret-revealing Z-states in T_{total} and obtain a new G-BTS

$$T_0 = T_{\text{total}}\restriction_{(Q_Y \cup Q_Z)\setminus Q_{\text{reveal}}}$$

where $T\restriction_Q$ denotes the G-BTS obtained by restricting the state-space of T to $Q \subseteq Q_Y \cup Q_Z$.

However, by deleting secret-revealing states, the resulting G-BTS may become inconsistent. Hence, we also need to delete inconsistent states recursively. Specifically, we define an operator F that maps a G-BTS to a new G-BTS by

$$F : T \mapsto T\restriction_{Q^T_{\text{const}}}$$

and we define

$$T^* = \lim_{k\to\infty} F^k(T_0)$$

as the largest consistent G-BTS in which there is no secret-revealing state. The existence of the supremal fixed-point as well as the finite-convergence of iteration follow directly from the computation of winning region in two-player games [14] or the well-known supremal controllable sublanguage [6].

The construction of T^* follows directly from its definition and one can proceed in two steps. First, we construct T_0 by a depth-first search or a breadth-first search from the initial Y-state y_0. Specifically, at each state encountered, one needs to consider all possible successor states, until reaching a secret-revealing state or a state that has been visited. Second, we prune inconsistent states from T_0 by iterations. Specifically, we need to remove Y-states having no successor and Z-states at which some feasible transitions are undefined, until the structure converges. Similar searching and pruning procedure can be found in the literature; see, e.g., [50, Algorithm 1]. We illustrate this procedure by the following example.

Example 5: Consider again system G in Fig. 1. First, we construct the largest G-BTS T_{total} by enumerating all possible transitions, which is in fact the structure shown in Fig. 2(a). For the sake of simplicity, as we discussed earlier, redundant macrocontrol-decisions are omitted in T_{total}. For example, $d = \{(\{0\}, \{\gamma_1, \gamma_2\})\}$ is not listed at state s_1, since $\gamma_1 \subset \gamma_2$ and macrocontrol-decision d_2 is sufficient enough to carry this information.

Note that Z-states s_2, s_8 and s_{20} are secret-revealing states since $\bigcup M(s_2) = \{0\} \subseteq X_S$, $\bigcup M(s_8) = \{4\} \subseteq X_S$ and $\bigcup M(s_{20}) = \{10\} \subseteq X_S$. Therefore, we need to delete states s_2, s_8 and s_{20} to obtain T_0. However, this creates inconsistent states s_7 and s_{19} as no macrocontrol-decision is defined. Therefore, these two states are removed when applying operator F for the first time. Again, this further creates inconsistent states s_3, s_6, s_{11}, s_{14} and s_{24} since some feasible observations are not defined. Therefore, these states and the associated transitions are again deleted when applying operator F for the second time. This yields the final structure T^* including states s_1, s_9, s_{10}, s_{17}, s_{18}, s_{21}, s_{22}, s_{23}, $s_{27}, \ldots, s_{32}$, which is the largest consistent G-BTS having no secret-revealing state.

Based on T^*, Algorithm 2 is provided to synthesize an IS-based nondeterministic supervisor in the form of an IS-mapping via a depth-first search. Specifically, we start from the initial Y-state and pick a macrocontrol-decision d from the set of all macrocontrol-decisions defined at y. Then, for each pair (m, y), where $m \in y$, we use d to define the mapping value for IS (m, y), which is the unique nondeterministic decision set associated with m in d. Then, we move to the unique Z-state reached under macrocontrol-decision d and consider all successor Y-states by considering all possible observable events. If the new Y-state has not yet been visited, then we repeat the selection procedure by making a recursive call of procedure do deep-first search (DoDFS) until all reachable ISs are traversed. The computed IS-mapping Θ^* is reachability-closed by construction; hence can be used to decode a corresponding IS-based supervisor S_{Θ^*} according to Algorithm 1.

Algorithm 2: Synthesis of IS-Based Supervisor Θ^*

```
input : $T^*$
output: $\Theta^*$
1   $y_0 \leftarrow \{\{x_0\}\}$, VISITED$\leftarrow \{y_0\}$, DOM$(\Theta^*) \leftarrow \emptyset$ ;
2   if $y_0$ is not in $T^*$ then
3       return "no solution" ;
4   DoDFS$(\Theta^*, y_0)$;
5   return $\Theta^*$;
procedure DoDFS$(\Theta^*, y)$;
6   choose a macrocontrol-decision $d \in C_{T^*}(y)$ ;
7   for $m \in y$ do
8       DOM$(\Theta^*) \leftarrow$ DOM$(\Theta^*) \cup \{(m, y)\}$;
9       $\Theta^*(m, y) \leftarrow \Gamma_{m,d}$, where $\Gamma_{m,d}$ is the unique
        decision such that $(m, \Gamma_{m,d}) \in d$;
10  for $\sigma \in \Sigma_o$ such that $h_{ZY}(h_{YZ}(y, d), \sigma) = y'$ do
        if $y' \notin$ VISITED then
11          VISITED$\leftarrow$ VISITED$\cup\{y'\}$ ;
12          DoDFS$(\Theta^*, y')$;
```

We show the computation procedure in Algorithm 2 by the following example.

Example 6: We still consider our running example with T^* shown in Fig. 2 and we use Algorithm 2 to synthesize an IS-mapping from T^*. The algorithm starts from the initial Y-state $s_1 = \{\{0\}\}$, at which the macrocontrol-decision in T^* is unique. Therefore, the supervisor will pick d_5 which induces partial mapping value $\Theta^*(\{0\}, \{\{0\}\}) = \{\gamma_2, \gamma_3\}$. By choosing d_5, we move to Z-state s_{10} and we need to consider all possible successor Y-states of s_{10}. Here, both o_1 and o_2 from s_{10} lead to Y-state $s_{18} = \{\{4\}, \{5\}\}$, where three macrocontrol-decisions d_9, d_{10}, d_{11} are defined. Suppose that the supervisor chooses $d_{11} = \{(\{4\}, \{\gamma_1\}), (\{5\}, \{\gamma_4\})\}$. This again induces partial mapping values $\Theta^*(\{4\}, \{\{4\}, \{5\}\}) = \{\gamma_1\}$ and $\Theta^*(\{5\}, \{\{4\}, \{5\}\}) = \{\gamma_4\}$. If observable event o_2 occurs, Y-state s_{29} is reachable and the macrocontrol-decision defined is unique. Therefore, by choosing d_{19} at $s_{29} = \{\{10, 11\}\}$, partial mapping value $\Theta^*(\{10, 11\}, \{\{10, 11\}\}) = \{\gamma_1\}$ is induced. If observable event o_1 occurs, Y-state s_{31} is reachable and the macrocontrol-decision defined is d_{20}. Partial mapping value $\Theta^*(\{9, 10\}, \{\{9, 10\}\}) = \{\gamma_1\}$ is induced. This completes the construction of reachability-closed IS-mapping Θ^*, which can also be represented as the decision diagram shown in Fig. 2(b).

Remark 5: The main purpose of this article is to synthesize a nondeterministic supervisor that guarantees opacity. Our focus is the solvability of this problem and whether or not the synthesized solution is maximally permissive is out of the main scope of this work. Here, we provide a heuristic approach to improve the permissiveness of this solution. In line 6 of Algorithm 2, we do not put specific criterion for which macrocontrol-decision to choose from $C_{T^*}(y)$. To enhance the permissiveness of the supervisor, we can pick a *locally maximal* macrocontrol-decision at each

Y-state. Formally, given two nondeterministic decision sets Γ_1 and Γ_2, we denote

1) by $\Gamma_1 \leq \Gamma_2$ if $\forall \gamma \in \Gamma_1, \exists \gamma' \in \Gamma_2 : \gamma \subseteq \gamma'$;
2) by $\Gamma_1 < \Gamma_2$ if $\Gamma_1 \leq \Gamma_2$ and $\exists \gamma \in \Gamma_1, \exists \gamma' \in \Gamma_2 : \gamma \subset \gamma'$.

Then, for each Y-state $y = \{m_1, \ldots, m_k\}$ in T^* and two macrocontrol-decisions d_1 and d_2 defined at y, where $d_1 = \{(m_1, \Gamma_1), \ldots, (m_k, \Gamma_k)\}$, $d_2 = \{(m_1, \Gamma'_1), \ldots, (m_k, \Gamma'_k)\}$, we say d_2 is more permissive than d_1, denoted by $d_1 < d_2$ if

1) $\forall i \in \{1, \ldots, k\}, \Gamma_i \leq \Gamma'_i$; and
2) $\exists i \in \{1, \ldots, k\}, \Gamma_i < \Gamma'_i$.

Therefore, in line 6 of Algorithm 2, one can choose a locally maximal macrocontrol-decision $d \in C_{T^*}(y)$ in the sense of

$$\forall d' \in C_{T^*}(y) : d \not< d'.$$

For example, for T^* in Fig. 2(a), there are three macrocontrol-decisions $d_9 = \{(\{4\}, \{\gamma_1\}), (\{5\}, \{\gamma_1\})\}$, $d_{10} = \{(\{4\}, \{\gamma_1\}), (\{5\}, \{\gamma_2, \gamma_3\})\}$ and $d_{11} = \{(\{4\}, \{\gamma_1\}), (\{5\}, \{\gamma_4\})\}$ defined at s_{18}. Then, d_{11} is a locally maximally macrocontrol-decision among these three. For example, we have $d_{10} < d_{11}$ since $\gamma_2 \subset \gamma_4$ and $\gamma_3 \subset \gamma_4$. Therefore, for the sake of permissiveness, the IS-mapping synthesis procedure can pick d_{11} instead of d_9 or d_{10}.

We conclude this section by discussing the complexity of the proposed supervisor synthesis algorithm. To construct the largest consistent G-BTS T^*, first, we need to build T_{total}, which contains at most $2^{2^{|X|}}$ Y-states and $2^{2^{|X|+|\Sigma_c|}}$ Z-states. For each Y-state, there are at most $2^{|X|+|\Sigma_c|}$ transitions defined and for each Z-state, there are at most $|\Sigma_o|$ transitions defined. Overall, T_{total} contains, in the worst-case, $2^{2^{|X|+|\Sigma_c|}} + 2^{2^{|X|}}$ states and $2^{2^{|X|}} \cdot 2^{|X|+|\Sigma_c|} + |\Sigma_o| \cdot 2^{2^{|X|+|\Sigma_c|}}$ transitions. The complexity of removing all secret-revealing states to obtain G-BTS T_0 is linear in the size of T_{total}. The complexity of removing all inconsistent states iteratively to obtain T^* is quadratic in the size of T_{total}. Once T^* is constructed, we run Algorithm 2 to synthesize an IS-mapping Θ^*, which is simply a depth-first search over the space of T^* and the complexity is still linear in the size of T^*. The resulting IS-mapping contains at most $2^{2^{|X|}}$ elements in its domain. In order to execute the supervisor online, we use Algorithm 1 to decode IS-mapping Θ^*. To this end, the supervisor needs to store the IS-mapping Θ^* computed offline, and during the online execution, record both the current state estimate m and the current macrostate $\mathbf{m}$. Note that m contains at most $|X|$ states, while $\mathbf{m}$ contains at most $2^{|X|}$. By making a new control decision upon the occurrence of a new observable event, m and $\mathbf{m}$ can be updated, respectively, in polynomial-time and exponential-time in the size of G. Note that this online update transition can also be precomputed offline and be stored as transition rules together with the IS-mapping Θ^*. Overall, the entire complexity of the proposed synthesis approach is doubly exponential in the size of the original plant, where the major complexity is spent for the offline computation.

VI. Properties of the Synthesis Procedure

In this section, we formally prove the correctness of the synthesis procedure proposed in Section V. Note that in the formulation of Problem 1, supervisors make control decision based on the decision histories and can be non-IS-based in general. However, our algorithm in Section V solves a restricted version of Problem 1 by only considering IS-based supervisors as formulated in Problem 2. Therefore, to show the correctness of the proposed synthesis procedure in the context of Problem 1, our arguments consist of the following two steps:

1) first, we show that our solution to the IS-based synthesis problem, i.e., Problem 2, is sound and complete;
2) then we show that restricting Problem 1 to Problem 2 is without loss of generality, i.e., Problem 1 is solvable if and only if Problem 2 is solvable.

Throughout this section, we denote by S_{Θ^*} the IS-based supervisor synthesized by Algorithm 2.

A. Correctness of the IS-Based Synthesis Algorithm

In this section, we show that Algorithm 2 indeed solves Problem 2. First, we show that Algorithm 2 is sound in the sense that the synthesized supervisor S_{Θ^*} is opacity-enforcing.

Theorem 2: IS-based nondeterministic supervisor $S_{\Theta^*} : (\Gamma\Sigma_o)^* \to 2^\Gamma$ encoded from Θ^* enforces opacity.

Proof: Let $s = \sigma_1 \cdots \sigma_n \in P(\mathcal{L}(S_{\Theta^*}/G))$ be any observable string in closed-loop system S_{Θ^*}/G. Let $\mathbf{m}_n^+$ be state induced by s and IS-mapping Θ^* according to (18). By Corollary 1, we know that $\bigcup M(\mathbf{m}_n^+) = X_I(s)$. According to Algorithm 2, $\mathbf{m}_n^+$ is a reachable Z-state in T^* by construction. Furthermore, since T^* is obtained from T_0 where all Z-states in Q_{reveal} are removed. Therefore, we conclude that

$$X_I(s) = \bigcup M(\mathbf{m}^+) \not\subseteq X_S$$

which means that Θ^* enforces opacity. ■

Note that Algorithim 2 returns "no solution" when y_0 is not included in T^*. Next, we show that Algorithm 2 is also complete in the sense that there is indeed no solution to Problem 2 when y_0 is removed by operator F during the construction of T^*.

Theorem 3: If there exists a nondeterministic IS-based supervisor that enforces opacity, then y_0 must be included in T^*, i.e., Algorithm 2 will not return "no solution" when a solution to Problem 2 exists.

Proof: Suppose that there exists a reachability-closed IS-mapping $\Theta : I \to 2^\Gamma$ such that the encoded nondeterministic supervisor S_Θ enforces opacity. We construct a consistent G-BTS $T = (Q_Y, Q_Z, h_{YZ}, h_{ZY}, \Sigma_o, D, y_0)$ as follows: $Q_Y = \{\mathbf{m} : (m, \mathbf{m}) \in \mathcal{I}_{S_\Theta}\}$ and for any $y \in Q_Y$, $d = \{(m, \Theta(m, y)) : m \in y\}$ is the unique macrocontrol-decision defined at y and $Q_Z = \{\odot(d) : \exists y \in Q_Y \text{ s.t. } h_{YZ}(y, d)!\}$. Since S_Θ enforces opacity, we have $\forall s \in P(\mathcal{L}(S_\Theta/G)) : X_I(s) \not\subseteq X_S$. Let $\mathbf{m}_n^+$ be state induced by s and IS-mapping Θ according to (18). By the construction of T, we have $\mathbf{m}_n^+ \in Q_Z$. By Corollary 1, we know that $\bigcup M(\mathbf{m}_n^+) = X_I(s)$. Therefore, we have $Q_Z \cap Q_{\text{reveal}} = \emptyset$. Since T is included in T_0 and T itself is consistent, no states in T can be removed when iteratively applying operator F, which means that all states in T are also included in T^*. Therefore, the initial Y-state y_0 is included in T^* and Algorithm 2 will not return "no solution". ■

B. From Non-IS-Based Supervisors to IS-Based Supervisors

So far, we have shown that Algorithm 2 correctly solves Problem 2, which is a restrictive version of Problem 1. Clearly, Algorithm 2 is also sound for Problem 1 because an IS-based solution is also a solution to Problem 1. Then, it remains to show the completeness of Algorithm 2 in terms of Problem 1. To this end, it suffices to show that Problem 2 always has a solution when Problem 1 has one. Here, we provide a constructive procedure that always construct an IS-based opacity-enforcing supervisor that solves Problem 2 when a non-IS-based one that solves Problem 1 exists.

Suppose that there exists a (possibly non-IS-based) nondeterministic supervisor $S_N : (\Gamma\Sigma_o)^* \rightarrow 2^\Gamma$ that enforces opacity. We construct an IS-mapping Θ according to Algorithm 3. The idea is similar to the information-flow analysis for IS-mapping, which expands the IS space from the initial IS. We still use y to denote state-estimates immediately after an observable event and use z to denote state-estimates with the unobservable tail included. However, since the supervisor needs not to be IS-based, simply remembering the current IS is not sufficient and we also need to remember the history leading to each state estimate. Therefore, for each microstate m_i in a Y-state, we add an additional information ρ_i to track how this microstate is visited. Note that for each microstate m in a Y-state, the decision history may not be unique since there may have multiple ρ associated with the same m. Similarly, each augmented microstate in a Z-state is also attached with a decision history information. Then, procedure DoDSF implements a depth-first search to generate the domain of the IS-mapping. Note that, since S_N is not IS-based in general, it may take different actions for different histories visiting the same IS. Our approach is to fix the control decision for each IS as the union of the decisions for all its visits; the constructed mapping is, therefore, forced to be IS-based. Algorithm 3 clearly terminates in a finite number of states since it will stop when all possible macrostates are visited.

Algorithm 3: Construction of IS-Mapping Θ from S_N

```
   input : $S_N$
   output: $\Theta$
1  $\rho \leftarrow \epsilon, m \leftarrow \{x_0\}, \textsc{Visited} \leftarrow \{\{m\}\}$ ;
2  $y \leftarrow \{(m, \epsilon)\}$ ;
3  DoDFS($y$, Visited);
4  return $\Theta$;
   procedure DoDFS($y$, Visited);
5  $z \leftarrow \emptyset$;
6  Suppose $y = \{(m_1, \rho_1), \ldots, (m_k, \rho_k)\}$ ;
7  $\mathbf{m} \leftarrow \{m_1, \ldots, m_k\}$ ;
8  for $i = 1, \ldots, k$ do
9      $\textsc{Dom}(\Theta) \leftarrow \textsc{Dom}(\Theta) \cup \{(m_i, \mathbf{m})\}$;
10     $\Theta(m_i, \mathbf{m}) \leftarrow \bigcup\{S_N(\rho) : (m_i, \rho) \in Y\}$ ;
11     for $\gamma \in S_N(\rho_i)$ do
12         $z \leftarrow z \cup \{(UR_\gamma(m_i), \rho_i\gamma, \gamma)\}$;
13 Suppose $z = \{(m_1, \rho_1, \gamma_1), \ldots, (m_p, \rho_p, \gamma_p)\}$ ;
14 for $\sigma \in \Sigma_o$ do
15     $y' \leftarrow \emptyset$;
16     for $i = 1, \ldots, p$ do
17         if $\sigma \in \gamma_i$ then
18             $y' \leftarrow y' \cup \{(NX_\sigma(m_i), \rho_i\sigma)\}$;
19     if $\{m : (m, \rho) \in y'\} \notin \textsc{Visited}$ then
20         $\textsc{Visited} \leftarrow \textsc{Visited} \cup \{\{m : (m, \rho) \in y'\}\}$;
21         DoDFS($y'$);
```

The following result shows that Algorithm 3 indeed converts a non-IS-based opacity-enforcing supervisor into an IS-based opacity-enforcing supervisor.

Theorem 4: Let $S_N : (\Gamma\Sigma_o)^* \rightarrow 2^\Gamma$ be a nondeterministic supervisor enforcing opacity and $\Theta : I \rightarrow 2^\Gamma$ be the partial IS-mapping constructed by Algorithm 3. Then, IS-based supervisor S_Θ also enforces opacity.

Proof: First, by construction, for each macrostate (Y-state without the extended strings components) visited by IS-mapping Θ, i.e., $\mathbf{m} = \{m_1, \ldots, m_k\} \in \textsc{Visited}$, IS-mapping Θ defines a nondeterministic control decision for each microstate $m_i \in \mathbf{m}$ (lines 9–10). Also, for each Z-state reached by S_Θ, every possible observable events are defined (line 14). Therefore, for every IS $(m, \mathbf{m}) \in \textsc{Reach}_\Theta((\{x_0\}, \{\{x_0\}\}))$, $\Theta(m, \mathbf{m})$ is always well-defined, i.e., IS-mapping Θ is reachability-closed. Therefore, its decoded IS-based supervisor S_Θ is well-defined and we have

$$\mathcal{I}_{S_\Theta} = \textsc{Reach}_\Theta(\imath_0) = \textsc{Dom}(\Theta)$$

where $\imath_0 = (\{x_0\}, \{\{x_0\}\})$.

By Corollary 1 and Proposition 1, to show that S_Θ enforces opacity, it suffices to show that

$$\forall (m, \mathbf{m}) \in \textsc{Dom}(\Theta) : \bigcup M(\odot(d_\Theta(\mathbf{m}))) \not\subseteq X_S.$$

To this end, we consider how $\mathbf{m}$ is added. Suppose $y_0 y_1 \ldots y_n$ is the sequence of Y-states in the depth-first search such that y_n contributes $\mathbf{m}$ to Visited, and let $s = \sigma_1 \ldots \sigma_n$ be the observable events along this sequence. More clearly, suppose that $y_n = \{(m_1, \rho_1), \ldots, (m_k, \rho_k)\}$ and we have $\{m_1, \ldots, m_k\} = \mathbf{m}$. We claim that for y_n, we have

$$y_n = \{(\hat{\mathcal{E}}_S(\rho), \rho) : \rho \in \mathcal{O}(\mathcal{L}_e^o(S_N/G)), \rho|_\Sigma = s\}.$$

This claim can be seen inductively by the length of s. For $|s| = 0$, we have $y_0 = \{(\{x_0\}, \epsilon)\}$, where ϵ is the unique string in $\mathcal{O}(\mathcal{L}_e^o(S_N/G))$ whose projection is also ϵ and $\hat{\mathcal{E}}_S(\rho) = \{x_0\}$. Assume that this claim holds for $|s| = k$, then for the case of $s\sigma_{k+1}$, according to lines 11–12 and lines 16–17, we have

$$y_{k+1} = \left\{ \begin{array}{c} (NX_{\sigma_{k+1}}(UR_\gamma(m)), \rho\gamma\sigma_{k+1}) : \\ (m, \rho) \in y_k, \gamma \in S_N(\rho), \sigma_{k+1} \in \gamma \end{array} \right\}$$

$$= \left\{ \begin{array}{c} \left(\hat{\mathcal{E}}_S(\rho'), \rho'\right) : \\ \rho' \in \mathcal{O}\left(\mathcal{L}_e^o\left(S_N/G\right)\right), \rho'|_\Sigma = \sigma_1 \ldots \sigma_k \sigma_{k+1} \end{array} \right\}.$$

Now, still for the same $\mathbf{m}$ and string s leading to it. We have

$$\begin{aligned}
&M(\odot(d_\Theta(\mathbf{m})))\\
&= \{UR_\gamma(m) \in 2^X : (m,\Gamma) \in d_\Theta(\mathbf{m}), \gamma \in \Gamma\}\\
&= \{UR_\gamma(\hat{\mathcal{E}}_S(\rho)) \in 2^X : \rho \in \mathcal{O}(\mathcal{L}_e^o(S_N/G)),\\
&\quad \rho|_\Sigma = s, \gamma \in S_N(\rho)\}\\
&= \{\mathcal{E}_S(\rho) \in 2^X : \rho \in \mathcal{O}(\mathcal{L}_e^d(S_N/G)) \text{ s.t. } \rho|_\Sigma = s\}\\
&= \mathcal{E}_I(s).
\end{aligned}$$

Since S_N enforces opacity, we have $\bigcup \mathcal{E}_I(s) = X_I(s) \not\subseteq X_S$, which means that $\bigcup M(\odot(d_\Theta(\mathbf{m}))) \not\subseteq X_S$. ■

By combining Theorems 3–5, we have the following result immediately that finally establishes the correctness of the synthesis procedure.

Corollary 2: Algorithm 2 also correctly solves Problem 1, i.e., it is both sound and complete.

VII. Conclusion

In this article, we proposed to use nondeterministic control mechanism to enforce opacity. The essence is to leverage the nondeterministic mechanism to enhance the plausible deniability of the system. To this end, we formally defined the nondeterministic supervisor and formulated the corresponding opacity enforcement problem. Effective approach was provided to synthesize a nondeterministic opacity-enforcing supervisor based on both the information of the supervisor and the information of the intruder. We showed that the proposed algorithm is both sound and complete in the sense that it will correctly return a nondeterministic opacity-enforcing supervisor when one exists.

Although we show that nondeterministic supervisors are strictly more powerful than deterministic ones, the synthesis complexity is doubly exponential in the size of the plant, which is higher than the single-exponential complexity for the deterministic case. Intuitively, this complexity is paid because we should not only estimate all possible states of the system from the supervisor's point of view, but also need to estimate the supervisor's estimates from the intruder's point of view. Recently, some new efficient approaches, such as abstraction-based approach [17], [29], [30], [34], [53], [55] and compositional approach [32], [33], [35], have been proposed to reduce the computational complexity in the verification and synthesis of opacity. In the future work, we also would like to leverage these techniques to further mitigate the complexity of the proposed synthesis algorithm.

References

[1] L. An and G.-H. Yang, "Opacity enforcement for confidential robust control in linear cyber-physical systems," *IEEE Trans. Autom. Control*, vol. 65, no. 3, pp. 1234–1241, Mar. 2020.

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

[3] R. J. Barcelos and J. C. Basilio, "Enforcing current-state opacity through shuffle in event observations," *IFAC-PapersOnLine*, vol. 51, no. 7, pp. 100–105, 2018.

[4] B. Behinaein, F. Lin, and K. Rudie, "Optimal information release for mixed opacity in discrete-event systems," *IEEE Trans. Autom. Sci. Eng.*, vol. 16 no. 4, pp. 1960–1970, Oct. 2019.

[5] J. W. Bryans, M. Koutny, L. Mazaré, and P. Ryan, "Opacity generalised to transition systems," *Internationa J. Inf. Secur.*, vol. 7, no. 6, pp. 421–435, 2008.

[6] C. G. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems*. 2nd ed. Berlin, Germany: Springer, 2008.

[7] F. Cassez, J. Dubreil, and H. Marchand, "Synthesis of opaque systems with static and dynamic masks," *Formal Methods Syst. Des.*, vol. 40, no. 1, pp. 88–115, 2012.

[8] J. Chen, M. Ibrahim, and R. Kumar, "Quantification of secrecy in partially observed stochastic discrete event systems," *IEEE Trans. Automat. Sci. Eng.*, vol. 14, no. 1, pp. 185–195, Jan. 2017.

[9] P. Darondeau, H. Marchand, and L. Ricker, "Enforcing opacity of regular predicates on modal transition systems," *Discrete Event Dyn. Syst.*, vol. 25, no. 1/2, pp. 251–270, 2014.

[10] J. Dubreil, P. Darondeau, and H. Marchand, "Supervisory control for opacity," *IEEE Trans. Autom. Control*, vol. 55, no. 5, pp. 1089–1100, May 2010.

[11] M. Fabian and B. Lennartson, "On non-deterministic supervisory control," in *Proc. 35th IEEE Conf. Decis. Control*, 1996, pp. 2213–2218.

[12] Y. Falcone and H. Marchand, "Enforcement and validation (at runtime) of various notions of opacity," *Discrete Event Dyn. Syst.*, vol. 25, no. 4, pp. 531–570, 2015.

[13] H. Farhat, "Control of nondeterministic systems for bisimulation equivalence under partial information," *IEEE Trans. Autom. Control*, vol. 65, no. 12, pp. 5437–5443, Dec. 2020, doi: 10.1109/TAC.2020.2970148.

[14] E. Gradel and W. Thomas, *Automata, Logics, and Infinite Games: A Guide to Current Research*, vol. 2500. Berlin, Germany: Springer, 2002.

[15] C. N. Hadjicostis, *Estimation and Inference in Discrete Event Systems*. Berlin, Germany: Springer, 2020.

[16] Z. He, Z. Ma, and W. Tang, "Performance safety enforcement in strongly connected timed event graphs," *Automatica*, vol. 128, 2021, Art. no. 109605.

[17] J. Hou, X. Yin, S. Li, and M. Zamani, "Abstraction-based synthesis of opacity-enforcing controllers using alternating simulation relations," in *Proc. 58th IEEE Conf. Decis. Control*, 2019, pp. 7653–7658.

[18] K. Inan, "Nondeterministic supervision under partial observations," in *Proc. 11th Int. Conf. Anal. Optim. Syst. Discrete Event Syst.*, 1994, pp. 39–48.

[19] R. Jacob, J.-J. Lesage, and J.-M. Faure, "Overview of discrete event systems opacity: Models, validation, and quantification," *Annu. Rev. Control*, vol. 41, pp. 135–146, 2016.

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

[21] C. Keroglou and C. N. Hadjicostis, "Probabilistic system opacity in discrete event systems," *Discrete Event Dyn. Syst.*, vol. 28, pp. 289–314, 2018.

[22] C. Keroglou, L. Ricker, and S. Lafortune, "Insertion functions with memory for opacity enforcement," *IFAC-PapersOnLine*, vol. 51, no. 7, pp. 394–399, 2018.

[23] R. Kumar, S. Jiang, C. Zhou, and W. Qiu, "Polynomial synthesis of supervisor for partially observed discrete-event systems by allowing nondeterminism in control," *IEEE Trans. Autom. Control*, vol. 50, no. 4, pp. 463–475, Apr. 2005.

[24] S. Lafortune, F. Lin, and C. N. Hadjicostis, "On the history of diagnosability and opacity in discrete event systems," *Annu. Rev. Control*, vol. 45, pp. 257–266, 2018.

[25] D. Lefebvre and C. N. Hadjicostis, "Exposure time as a measure of opacity in timed discrete event systems," in *Proc. 18th Eur. Control Conf.*, 2019, pp. 1740–1745.

[26] F. Lin, "Opacity of discrete event systems and its applications," *Automatica*, vol. 47, no. 3, pp. 496–503, 2011.

[27] F. Lin, W. Chen, W. Wang, and F. Wang, "Information control in networked discrete event systems and its application to battery management systems," *Discrete Event Dyn. Syst.*, vol. 30, pp. 243–268, 2020.

[28] R. Liu, L. Mei, and J. Lu, "K-memory-embedded insertion mechanism for opacity enforcement," *Syst. Control Lett.*, vol. 145, 2020, Art. no. 104785.

[29] S. Liu, X. Yin, and M. Zamani, "On a notion of approximate opacity for discrete-time stochastic control systems," in *Proc. Amer. Control Conf.*, 2020, pp. 5413–5418.

[30] S. Liu and M. Zamani, "Verification of approximate opacity via barrier certificates," *IEEE Control Syst. Lett.*, vol. 5, no. 4, pp. 1369–1374, Oct. 2020.

[31] L. Mazaré, "Using unification for opacity properties," in *Proc. Workshop Issues Theory Secur.*, 2004, pp. 165–176.

[32] S. Mohajerani, Y. Ji, and S. Lafortune, "Compositional and abstraction-based approach for synthesis of edit functions for opacity enforcement," *IEEE Trans. Autom. Control*, vol. 65, no. 8, pp. 3349–3364, Aug. 2020.

[33] S. Mohajerani and S. Lafortune, "Transforming opacity verification to nonblocking verification in modular systems," *IEEE Trans. Autom. Control*, vol. 65, no. 4, pp. 1739–1746, Apr. 2020.

[34] M. Noori-Hosseini, B. Lennartson, and C. Hadjicostis, "Incremental observer reduction applied to opacity verification and synthesis," 2018, *arXiv:1812.08083*.

[35] M. Noori-Hosseini, B. Lennartson, and C. N. Hadjicostis, "Compositional visible bisimulation abstraction applied to opacity verification," *IFAC-PapersOnLine*, vol. 51, no. 7, pp. 434–441, 2018.

[36] B. Ramasubramanian, W. R. Cleaveland, and S. Marcus, "Notions of centralized and decentralized opacity in linear systems," *IEEE Trans. Autom. Control*, vol. 265, no. 4, pp. 1442–1455, Apr. 2020.

[37] I. Saadaoui, Z. Li, and N. Wu, "Current-state opacity modelling and verification in partially observed petri nets," *Automatica*, vol. 116, 2020, Art. no. 108907.

[38] A. Saboori and C. N. Hadjicostis, "Opacity-enforcing supervisory strategies via state estimator constructions," *IEEE Trans. Autom. Control*, vol. 57, no. 5, pp. 1155–1165, May 2012.

[39] A. Saboori and C. N. Hadjicostis, "Verification of initial-state opacity in security applications of discrete event systems," *Inf. Sci.*, vol. 246, pp. 115–132, 2013.

[40] S. Takai, "Bisimilarity enforcing supervisory control of nondeterministic discrete event systems with nondeterministic specifications," *Automatica*, vol. 108, 2019, Art. no. 108470.

[41] S. Takai, "Synthesis of maximally permissive supervisors for nondeterministic discrete event systems with nondeterministic specifications," *IEEE Trans. Autom. Control*, vol. 66, no. 7, pp. 3197–3204, Jul. 2021, doi: 10.1109/TAC.2020.3015453.

[42] S. Takai and Y. Oka, "A formula for the supremal controllable and opaque sublanguage arising in supervisory control," *SICE J. Control, Meas. Syst. Integration*, vol. 1, no. 4, pp. 307–311, 2008.

[43] Y. Tong, Z. Li, C. Seatzu, and A. Giua, "Verification of state-based opacity using Petri nets," *IEEE Trans. Autom. Control*, vol. 62, no. 6, pp. 2823–2837, Jun. 2017.

[44] Y. Tong, Z. Li, C. Seatzu, and A. Giua, "Current-state opacity enforcement in discrete event systems under incomparable observations," *Discrete Event Dyn. Syst.*, vol. 28, no. 2, pp. 161–182, 2018.

[45] L. Wang, N. Zhan, and J. An, "The opacity of real-time automata," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 37, no. 11, pp. 2845–2856, Nov. 2018.

[46] B. Wu, J. Dai, and H. Lin, "Synthesis of insertion functions to enforce decentralized and joint opacity properties of discrete-event systems," in *Proc. Amer. Control Conf.*, 2018, pp. 3026–3031.

[47] B. Wu and H. Lin, "Privacy verification and enforcement via belief abstraction," *IEEE Control Syst. Lett.*, vol. 2, no. 4, pp. 815–820, Oct. 2018.

[48] Y. Xie, X. Yin, and S. Li, "Opacity enforcing supervisory control using non-deterministic supervisors," in *Proc. 21st IFAC World Congr.*, vol. 53, no. 2, 2020, pp. 1763–1769.

[49] Y. Yao, Y. Tong, and H. Lan, "Initial-state estimation of multi-channel networked discrete event systems," *IEEE Control Syst. Lett.*, vol. 4, no. 4, pp. 1024–1029, Oct. 2020, doi: 10.1109/LCSYS.2020.2998610.

[50] X. Yin and S. Lafortune, "A uniform approach for synthesizing property-enforcing supervisors for partially-observed discrete-event systems," *IEEE Trans. Autom. Control*, vol. 61, no. 8, pp. 2140–2154, Aug. 2016.

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

[52] X. Yin, Z. Li, W. Wang, and S. Li, "Infinite-step opacity and K-step opacity of stochastic discrete-event systems," *Automatica*, vol. 99, pp. 266–274, 2019.

[53] X. Yin, M. Zamani, and S. Liu, "On approximate opacity of cyber-physical system," *IEEE Trans. Autom. Control*, vol. 66, no. 4, pp. 1630–1645, Apr. 2021.

[54] B. Zhang, S. Shu, and F. Lin, "Maximum information release while ensuring opacity in discrete event systems," *IEEE Trans. Automat. Sci. Eng.*, vol. 12, no. 4, pp. 1067–1079, Jul. 2015.

[55] K. Zhang, X. Yin, and M. Zamani, "Opacity of nondeterministic transition systems: A. (bi) simulation relation approach," *IEEE Trans. Autom. Control*, vol. 64, no. 12, pp. 5116–5123, Dec. 2019.

[56] C. Zhou, R. Kumar, and S. Jiang, "Control of nondeterministic discrete-event systems for bisimulation equivalence," *IEEE Trans. Autom. Control*, vol. 51, no. 5, pp. 754–765, May 2006.

[57] G. Zinck, L. Ricker, H. Marchand, and L. Hélouët, "Enforcing opacity in modular systems," in *Proc. IFAC World Congr.*, vol. 53, no. 2, 2020, pp. 2157–2164.

Yifan Xie (Student Member, IEEE) was born in Hubei, China, in 1999. She received the B.Eng. degree in automation from Beihang University, Beijing, China, in 2019. She is currently working toward the M.S. degree in control engineering with the Department of Automation, Shanghai Jiao Tong University, Shanghai, China.

Her current research interests include systems and control theory, formal methods, and discrete-event systems.

Xiang Yin (Member, IEEE) was born in Anhui, China, in 1991. He received the B.Eng. degree in electrical engineering from Zhejiang University, Hangzhou, China, in 2012, the M.S. and Ph.D. degrees in electrical engineering from the University of Michigan, Ann Arbor, MI, USA, in 2013 and 2017, respectively.

Since 2017, he has been with the Department of Automation, Shanghai Jiao Tong University, Shanghai, China, where he is currently an Associate Professor. His research interests include formal methods, discrete-event systems, and cyber-physical systems.

Dr. Yin was the recipient of the IEEE Conference on Decision and Control Best Student Paper Award Finalist in 2016. He is the Co-Chair of the IEEE CSS Technical Committee on Discrete Event Systems. He is also a member of the IEEE CSS Conference Editorial Board.

Shaoyuan Li (Senior Member, IEEE) was born in Hebei, China, in 1965. He received the B.S. and M.S. degrees in automation from the Hebei University of Technology, Tianjin, China, in 1987 and 1992, respectively, and the Ph.D. degree in control science from Nankai University, Tianjin, China, in 1997.

Since 1997, he has been with the Department of Automation, Shanghai Jiao Tong University, Shanghai, China, where he is currently a Professor. His current research interests include model predictive control, dynamic system optimization, and cyber-physical systems.