# Current-State Opacity Enforcement in Discrete Event Systems Under Incomparable Observations

Yin Tong, Zhiwu Li, Carla Seatzu, and Alessandro Giua,

**Abstract**

In this paper we tackle the opacity enforcement problem in discrete event systems using supervisory control theory. In particular, we consider the case where the intruder and the supervisor may observe different sets of events and neither of these sets needs to be contained in the other one. Moreover, there may be controllable events that cannot be observed by the supervisor. We propose a finite structure, called an *augmented I-observer*, to characterize the strings that will not leak the secret. Based on such a structure, a locally optimal supervisor enforcing current-state opacity is designed.

**Index Terms**

Opacity Discrete Event Systems Supervisory Control Finite State Automatapacity Discrete Event Systems Supervisory Control Finite State Automata

**Published as**:

Y. Tong, Z.W. Li, C. Seatzu, A. Giua, “Current-state opacity enforcement in discrete event systems under incomparable observations,” Discrete Event Dynamic Systems, Vol. 28, No. 2, 2018: 161-182.

This work was supported by the National Natural Science Foundation of China under Grants Nos. 61374068, 61472295, 61673309, the Science and Technology Development Fund, MSAR, under Grant No. 078/2015/A3, and the Fundamental Research Funds for the Central Universities with Grant No. 2682017ZDPY10.

Y. Tong is with the School of Information Science and Technology, Southwest Jiaotong University, Chengdu 611756, China (e-mail: yintong@swjtu.edu.cn).

Z. W. Li is with the Institute of Systems Engineering, Macau University of Science and Technology, Taipa, Macau, and also with the School of Electro-Mechanical Engineering, Xidian University, Xi’an 710071, China (e-mail: zhwli@xidian.edu.cn).

C. Seatzu is with the Department of Electrical and Electronic Engineering, University of Cagliari, 09123 Cagliari, Italy (e-mail: seatzu@diee.unica.it)

A. Giua is with the Department of Electrical and Electronic Engineering, University of Cagliari, 09123 Cagliari, Italy (email: giua@unica.it).

## I. Introduction

Motivated by the concern about security and privacy in computer systems, communication protocols, etc., various notions of secrecy have been formulated, such as *non-interference* [1], [2], *anonymity* [3], [4] and *opacity* [5], [6], [7], [8], [9]. Among them opacity is a useful notion for describing in a unitary framework some other security properties such as trace-based non-interference and anonymity [10]. In this paper we focus on an opacity property, called *current-state opacity*, in discrete event systems (DES). This opacity property was introduced in [6]. Given a system, a subset of its states is considered as "secret". There exists a malicious observer (called intruder) who attempts to detect when the system is in a secret state so that an attack can be launched. It is usually assumed that the intruder knows the structure of the system but has only partial observation of the system's evolution. The system is said to be *current-state opaque* with respect to the given secret if based on its observations the intruder cannot determine with certainty if the current state of the system belongs to the secret.

It is proven that opacity verification problems in bounded systems are decidable [40]. There are several ways to verify opacity using the so-called *observer automaton* [7], [9], [11], [12] or Petri net based techniques [13], [14], [15], [16]. Meanwhile, the *opacity enforcement* problem is another active topic that has received a lot of attention in the DES community. Given a system that is not opaque, the opacity enforcement problem consists in turning the system into an opaque one. Approaches to opacity enforcement may rely on supervisory control [17], [18], [19], [20], [21], dynamically restraining the observability of events [22], inserting additional events in the output behavior of the system [23], [24] and the runtime validation technique [25]. The aim of this work is to enforce *current-state opacity* [6], [9] using supervisory control.

Given a system that is not current-state opaque with respect to a given secret, our purpose is to design a maximally permissive supervisor that restricts the behavior of the system to ensure that the controlled system is current-state opaque. There has been some related work on the design of supervisors to enforce opacity properties. In [26], the authors consider the secret defined as a set of event sequences (such an opacity property is usually called *language-based opacity*) and a set of intruders having different observations. They assume that all events are observable and controllable to the supervisor, and show that the optimal supervisor always exists. Considering the same language-based opacity enforcement problem but with only one intruder, Dubreil et al. [18], [27] study a more general case where the supervisor may observe a set of events different from the one observed by the intruder in the presence of uncontrollable events. [20] propose methods for designing optimal supervisors to enforce two different opacity properties: initial-state opacity and infinite-step opacity, with the assumption that the supervisor can observe all events. More recently, the common assumption that all controllable events are also observable [18], [20], [26], [27] is relaxed in [21] to enforce current-state opacity.

We point out that all the aforementioned works are carried out in the framework of finite automata and rely on Ramadge and Wonham's basic theory of supervisory control for DES [28]. Note that the objective of opacity enforcement is not concerned with liveness since opacity properties focus on a set of indiscernible runs from the perspective of the intruder instead of individual runs. In this paper we tackle the current-state opacity enforcement problem in the framework of finite automata and what distinguishes our work from the existing works consists in three aspects.

- No containment relation is assumed between the sets $E_I$, $E_S$ of events observable by the intruder and by the supervisor, respectively. We call this general setting *incomparable observations*. In this sense, the problem considered here is more general than the one in [18], [20], [21], [26], [27].
- We also relax the assumption made in [17], [18], [19], [20], [29] that all controllable events $E_C$ should be observable.
- Finally, we consider the problem of enforcing opacity under the assumption that the intruder does not know that a supervisor

TABLE I
COMPARISON BETWEEN THE PROPOSED APPROACH AND PREVIOUS APPROACHES.

| Works | [18] | [21] | [29] | This Paper |
|---|---|---|---|---|
| Assumptions | $E_I \subseteq E_S$ (or $E_S \subseteq E_I$) $E_C \subseteq E_S$ | $E_I \subseteq E_S$ | $E_C \subseteq E_S$ | None |
| Does the intruder know the supervisor? | Yes | Yes | No | No |
| Complexity | $\mathcal{O}(\lvert X\rvert \times 2^{\lvert X\rvert})$ | $\mathcal{O}(2^{2(\lvert X\rvert+\lvert E_C\rvert)})$ | $O(2^{2^{2\lvert X\rvert}})$ | $\mathcal{O}(2^{2(\lvert X\rvert\times 2^{\lvert X\rvert}+\lvert E_C\rvert)})$ |

is acting on the system. To address this problem, we define *G-opacity* of a language. We show that if a controlled system $Sup/G$ is current-state opaque then its generated language $L(Sup/G)$ is $G$-opaque but the converse does not hold in the general case. However, if the intruder does not know the supervisor, $L(Sup/G)$ being $G$-opaque is sufficient to guarantee that the intruder cannot detect if the current state belongs to the secret.

To be more clear, comparison between the proposed approach and previous ones [18], [21], [29] is summarized in Table I. All the approaches are developed for deterministic finite automata but under different assumptions. The last row of Table I presents their computational complexity, where $X$ is the set of states of the system and $E_C$ is the set of controllable events. Note that the assumption that the intruder does not know the supervisor simplifies the problem of opacity enforcing due to the incomplete structural information available to the supervisor's adversary.

In this paper, first a structure called *augmented I-observer* is constructed. The augmented I-observer of a system is a deterministic finite automaton, where each state contains the current-state estimate of the intruder. Based on the augmented I-observer, evolutions of the system that satisfy current-state opacity can be characterized. Then we show that the current-state opacity enforcement problem can be reduced to the basic supervisory control problem under partial observation [30]. Note that the maximally permissive supervisor enforcing current-state opacity may not be unique. Thus we obtain a set of *locally optimal* supervisors where the adverb "locally" points out that the behavior of the controlled system under each of them is not strictly included in another. Finally, we show that based on the proposed approach it is possible to solve current-state opacity enforcement problem assuming that the intruder does not know the supervisor. To summarize, there are three main contributions of the paper:

- The definition of $G$-opacity of a language that enables us to formalize the opacity enforcement problem under the assumption that the intruder has no knowledge (or at most a partial knowledge) of the superviso,
- The definition of a novel finite structure, the augmented I-observer, that enables one to relax the assumptions $E_S \subseteq E_I$ (or $E_I \subseteq E_S$) and $E_C \subseteq E_S$, and
- The demonstration that based on the notion of $G$-opacity and the augmented I-observer, the current-state opacity enforcement problem can be reduced to the basic supervisory control problem under partial observation, which is a result that has not been previously discussed in the literature. Then, locally optimal supervisors are achieved using appropriate supervisory control techniques.

This paper improves the results presented in the previous work [29] to a more general setting, by removing the assumption that all events controllable by the supervisor should be observable. In addition, under the same assumptions, the proposed approach has lower complexity than the approach in [29].

The rest of this paper is organized as follows. Basic notions on automata and supervisory control theory are recalled in

Section II. Section III presents the definition of current-state opacity and the corresponding verification approach. In Section IV the current-state opacity enforcement problem is formalized and a method for the synthesis of an optimal supervisor is proposed. Finally, this paper is concluded in Section V, where our future work in this topic is also discussed.

## II. Background

In this section we recall some elementary notions on automata and supervisory control. For more details, we refer the reader to [28], [30].

### A. Automata

A system is modeled in this paper as a *deterministic finite automaton* (DFA) $G = (X, E, \delta, x_0)$, where $X$ is the finite set of *states*, $E$ is the set of *events*, $\delta : X \times E \to X$ is the (partial) *transition function*, $x_0 \in X$ is the *initial state*. The transition function can be extended to $\delta : X \times E^* \to X$ recursively: for all $x \in X$, $\delta(x, \varepsilon) = x$ and $\delta(x, \sigma e) = \delta(\delta(x, \sigma), e)$ for $\sigma \in E^*$ and $e \in E$. We denote by $\delta(x, \sigma)!$ the fact that $\sigma$ is defined at $x$. The *generated language* of a DFA $G = (X, E, \delta, x_0)$ is defined as

$$L(G) = \{\sigma \in E^* | \delta(x_0, \sigma)!\}.$$

A string $\sigma'$ is a prefix of a string $\sigma \in E^*$ if for some $\sigma'' \in E^*$, $\sigma = \sigma'\sigma''$. The *prefix closure* of a language $L \subseteq E^*$ is defined to be the language

$$\overline{L} = \{\sigma' \in E^* | \sigma'\sigma'' \in L \text{ for some } \sigma'' \in E^*\}.$$

If $\overline{L} = L$, we say that $L$ is *prefix-closed*. Clearly, the generated language of a DFA is always prefix-closed, i.e., $L(G) = \overline{L(G)}$.

To model the partial observation of event sequences by the intruder and the supervisor, we denote by $E_I \subseteq E$ and $E_S \subseteq E$ the sets of events observable by the intruder and the supervisor, respectively. The natural *projection* $P_I : E^* \to E_I^*$ on $E_I$ is defined as i) $P_I(\varepsilon) = \varepsilon$; ii) for all $\sigma \in E^*$ and $e \in E$, $P_I(\sigma e) = P_I(\sigma)e$ if $e \in E_I$, and $P_I(\sigma e) = P_I(\sigma)$, otherwise. Similarly, the natural projection $P_S : E^* \to E_S^*$ on $E_S$ can be defined. Given an event sequence $\sigma \in E^*$, its projection $w_i = P_I(\sigma)$ (resp., $w_s = P_S(\sigma)$) on $E_I$ (resp., $E_S$) is called an *observation* of the intruder (resp., supervisor). We denote by $E_{UI}$ (resp., $E_{US}$) the set of events that cannot be observed by the intruder (resp., supervisor). We denote by $E_C \subseteq E$ (resp. $E_{UC} = E \setminus E_C$) the set of events that can (resp. cannot) be controlled by the supervisor.

*Definition 2.1 (Unobservable Reach):* Given a system $G = (X, E, \delta, x_0)$ and a state $x$, the *unobservable reach* $R_I(x, \varepsilon)$ of $x$ with respect to the intruder is

$$R_I(x, \varepsilon) = \{x' \in X | \exists \sigma \in E_{UI}^* : \delta(x, \sigma) = x'\}.$$

◇

Obviously, $x \in R_I(x, \varepsilon)$. Given an event $e \in E_I$, the *e-reach* $R_I(x, e)$ of $x$ is defined as $R_I(x, e) = R_I(x', \varepsilon)$, where $x' = \delta(x, e)$.

### B. Supervisory Control

Given a system $G = (X, E, \delta, x_0)$, the goal of supervisory control is to design a control agent (called supervisor) that restricts the behavior of the system within a *specification language* $K \subseteq L(G)$. The supervisor observes a set of *observable events* $E_S \subseteq E$ and is able to control a set of *controllable events* $E_C \subseteq E$. The supervisor *enables* or *disables* controllable

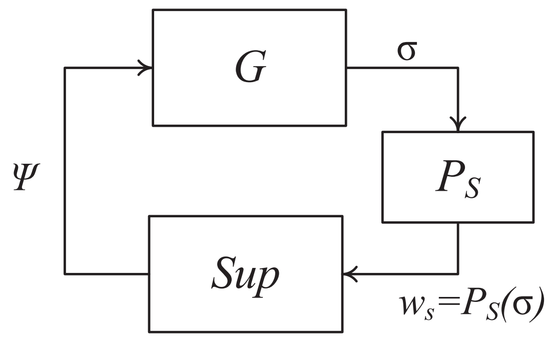

Fig. 1. Supervisory control under partial observation

events. When an event is enabled (resp., disabled) by the supervisor, all transitions labeled by the event are allowed to occur (resp., prevented from occurring). After the supervisor observes a string generated by the system it tells the system the set of events that are enabled next to ensure that the system will not violate the specification. A supervisor can be represented by $Sup = (Y, E_S, \delta_s, y_0, \Psi)$, where $(Y, E_S, \delta_s, y_0)$ is an automaton and

$$\Psi : Y \to \{E' \subseteq E | E_{UC} \subseteq E'\}$$

specifies the set of events enabled by the supervisor in each state. Fig. 1 illustrates the paradigm of supervisory control under partial observation. Let $\sigma \in L(G)$ be the string generated by the system and $w_s = P_S(\sigma)$ be the corresponding observation of the supervisor. Then the set of events enabled by the supervisor is $\Psi(y)$, where $y = \delta_s(y_0, w_s)$. System $G$ under the control of a suitable supervisor $Sup$ is denoted as $Sup/G$, and it satisfies $L(Sup/G) \subseteq K$.

*Definition 2.2 (Controllability):* [28] Given a DFA $G$, a set of controllable events $E_C$, and a language $K \subseteq L(G)$, $K$ is said to be *controllable* (wrt $L(G)$ and $E_C$) if

$$\overline{K} E_{UC} \cap L(G) \subseteq \overline{K},$$

where $E_{UC} = E \setminus E_C$. ◇

In other words, the controllability of $K$ requires that for any prefix $\sigma$ of a string in $K$, if $\sigma$ followed by an uncontrollable event $e \in E_{UC}$ is in $L(G)$, then it must also be a prefix of a string in $K$. It is known that controllability is preserved under arbitrary unions and consequently the supremal controllable sublanguage of a given language exists.

*Definition 2.3 (Observability):* [28] Given a DFA $G$, a set of controllable events $E_C$, a set of observable events $E_S$, and a language $K \subseteq L(G)$, $K$ is said to be *observable* (wrt $L(G)$, $E_S$ and $E_C$) if for all $\sigma, \sigma' \in \overline{K}$ and all $e \in E_C$ such that $\sigma e \in L(G)$, $\sigma' e \in \overline{K}$ and $P_S(\sigma) = P_S(\sigma')$, $\sigma e \in \overline{K}$ holds. ◇

Roughly speaking, observability requires that supervisor's observation of the system (i.e., the projection of $\sigma$ on $E_S$) provides sufficient information to decide after the occurrence of a controllable event whether the resultant string is still in $\overline{K}$. Unlike controllability, observability is however not preserved under union, therefore the supremal observable sublanguage of a given language may not exist. However maximal observable sublanguages exist, but are not usually unique.

*Theorem 2.4:* [28] Let $K \subseteq L(G)$ be a prefix-closed nonempty language, $E_C$ the set of controllable events and $E_S$ the set of observable events. There exists a supervisor $Sup$ such that $L(Sup/G) = K$ if and only if $K$ is controllable and observable.

*Definition 2.5 (Supervisory Control and Observation Problem, SCOP):* Given a system $G$, a set of controllable events $E_C$, a set of observable events $E_S$ by the supervisor, and a specification language $K \subseteq L(G)$, find a locally optimal supervisor

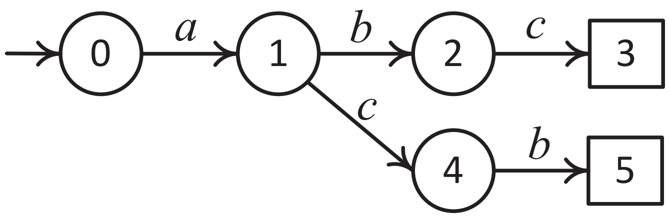

Fig. 2. System $G$ where $E_S = \{a\}$, $E_C = \{a, b, c\}$ and states 3 and 5 should be unreachable.

$Sup$ such that:

1) $L(Sup/G) \subseteq K$
2) $L(Sup/G)$ is maximal, i.e., for any other supervisor $Sup'$,

$$L(Sup/G') \subseteq K \Rightarrow L(Sup/G) \not\subset L(Sup'/G).$$

◇

A SCOP involves the system[1] $G$, the set $E_S$ of events observable by the supervisor, the set $E_C$ of events controllable by the supervisor, and the specification language $K$. To be concise, we call this problem SCOP($G, E_S, E_C, K$).

Since the supremal observable sublanguage may not exist, there may not be the supremal controllable and observable sublanguage of a given language. Consequently, there may be multiple solutions to a SCOP and they are said to be "locally optimal" since under the control of the corresponding supervisors, the behaviors of the controlled system are incomparable.

The SCOP has been considered in the literature and many different methods have been proposed to solve it [31], [32], [33], [34], [35], [36]; in this work we briefly introduce the approach recently presented in [35].

## C. Approach Based on the Total Controller

The authors of [35] propose a structure, called *total controller*, based on which all locally optimal supervisors of the SCOP can be computed. Given a SCOP($G, E_S, E_C, K$) with $K = L(H)$, it is assumed, without loss of generality, that $H = (X_H, E, \delta_H, x_{H,0})$ is a strict sub-automaton[2] of $G$. In other words, the language specification $K$ of a SCOP is reduced to a state specification: a state $x \in X$ is legal iff $x \in X_H$, i.e., $\sigma \in K$ with $x = \delta(x_0, \sigma)$. We denote by $F = X \setminus X_H$ the set of forbidden states. In this subsection, such an approach is introduced through a numerical example.

Consider the system $G = (X, E, \delta, x_0)$ in Fig. 2, where $E_S = \{a\}$ and $E_C = \{a, b, c\}$. The set of forbidden states is $F = \{3, 5\}$. The approach proposed by Yin and Lafortune [35] can be summarized as follows. First, construct a finite structure called a *total controller*, which enumerates all possible control policies of the system. In the total controller there are two types of states: Y-states $\mathcal{Y} \subseteq X$ in rounded boxes and Z-states $\mathcal{Z} = (Z, I)$ in rectangles, where $Z \subseteq X$ and $I$ is a control decision, i.e., it contains the set of events enabled by the supervisor. The initial state of the total controller is $\mathcal{Y}_0 = \{x_0\}$. Y-states are driven to Z-states by control decisions. At each Y-state $\mathcal{Y}$, we enumerate all control decisions[3], and then the successor Z-state corresponding to a control decision is computed: $Z$ is the set of states reachable from $\mathcal{Y}$ by firing unobservable events enabled by the control decision and $I$ is the control decision. For instance, in Fig. 3, from Y-state $\{1\}$, for control decision $\{b\}$ the

[1]Properly speaking, the SCOP concerns the language $L(G)$.

[2]If $H$ is not a strict subautomaton of $G$, the algorithm in [37] can be used to transform both of them to $G'$ and $H'$, respectively, such that $H'$ is a strict subautomaton of $G'$.

[3]For the system in Fig. 2, there is no need to enumerate all control decisions when Y-state is $\{0\}$ or $\{1\}$. Indeed, from state 0, observable event $a$ would never occur before $b$ and $c$, therefore all other control policies are equivalent to $\{a\}$ or $\{\}$. From state 1, event $a$ would never be executed. As a result, control policies containing $a$ are redundant.

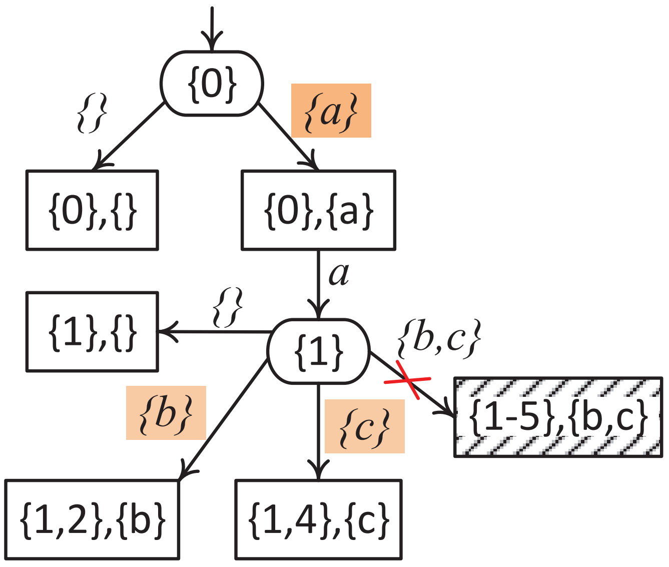

Fig. 3. Total controller of $G$ in Fig. 2. Removing the state in the dashed box, the all inclusive controller is obtained.

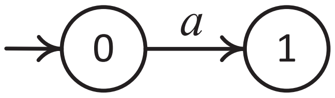

Fig. 4. The automaton structure of the two locally optimal supervisors: for $Sup_1$, $\Psi(0) = \{a\}$ and $\Psi(1) = \{b\}$; for $Sup_2$, $\Psi(0) = \{a\}$ and $\Psi(1) = \{c\}$.

Z-state reached is $(\{1, 2\}, \{b\})$ and for control decision $\{b, c\}$ the Z-state reached is $(\{1 - 5\}, \{b, c\})$. Z-states are driven to Y-states by observable events $e \in E_S$ that are defined at a state in $Z$ and enabled by the control decision $I$. The successor Y-state is the set of states reachable from a state in $Z$ after the occurrence of $e$. For instance, from Z-state $(\{0\}, \{a\})$ event $a$ is enabled at 0 and is allowed by the control decision, therefore the Y-state reached is $\{1\}$.

After the total controller is constructed, removing all the Y-states and Z-states that contain a forbidden state (i.e., 3 and 5 in this case) and the related arcs, the *all inclusive controller* is obtained. In Fig. 3 $(\{1 - 5\}, \{b, c\})$ is such a state and should be removed. The all inclusive controller models all the control policies that enforce the specification language. Finally, after each Y-state we pick a control decision that is not a strict subset of any other decisions. A combination of those local maximal control decisions corresponds to a locally optimal supervisor.

It has been proven that the time complexity of the approach proposed in [35] to solve the SCOP is $\mathcal{O}(|X||E|2^{|X|+|E_C|})$. In Fig. 3, each locally maximal control decision is colored. There are two optimal supervisors $Sup_1$ and $Sup_2$ (see Fig. 4) and the behaviors of the controlled system under different supervisors are $L(Sup/G_1) = \{\varepsilon, a, ab\}$ and $L(Sup/G_2) = \{\varepsilon, a, ac\}$, respectively.

## III. Current-State Opacity and Its Verification

Current-state opacity has been defined in both automata and Petri nets frameworks [6], [9], [13], [38]. In this section, we recall the definition of current-state opacity in finite automata and describe the approach in [38] to checking this property.

Given a system, it is usually assumed that the intruder knows the system's structure $G$ but only the occurrence of some events can be detected by the intruder. Current-state opacity is defined as follows.

*Definition 3.1 (Current-State Opacity):* Given a system $G = (X, E, \delta, x_0)$, a secret $S \subseteq X$, and a set $E_I$ of events observable by the intruder, $G$ is said to be *current-state opaque* (CSO) wrt $S$ and $E_I$ if $\forall \sigma \in L(G)$ such that $\delta(x_0, \sigma) \in S$,

$$\exists \sigma' \in L(G) : P_I(\sigma') = P_I(\sigma) \text{ and } \delta(x_0, \sigma') \notin S.$$

◇

In simple words, for any sequence of events $\sigma$ that leads to a state in the secret, i.e., a secret state, there exists at least one sequence of events that reaches a non-secret state but produces the same observation $P_I(\sigma)$ to the intruder. Therefore, when the intruder observes $P_I(\sigma)$, it cannot conclude whether the current state is contained or not in the secret. Based on the system's structure and its observation, the intruder can estimate the current state.

*Definition 3.2 (Estimate of the Intruder):* Given a system $G = (X, E, \delta, x_0)$ and an observation $w_i$ of the intruder, the *estimate* of the intruder is defined as

$$\mathcal{C}_I(w_i) = \{x \in X | \exists \sigma \in E^* : \delta(x_0, \sigma) = x, P_I(\sigma) = w_i\}.$$

◇

Therefore, if the intruder observes a sequence of events $w_i$, it knows that the current state could be any state in the set $\mathcal{C}_I(w_i)$. Obviously, $\mathcal{C}_I(\varepsilon) = R_I(x_0, \varepsilon)$. The estimate of the intruder after observing $w_i$ can be iteratively computed by

$$\mathcal{C}_I(w_i) = \bigcup_{x \in \mathcal{C}_I(w_i')} R_I(x, e), \tag{1}$$

where $w_i = w_i' e$, $w_i' \in E_I^*$, and $e \in E_I$ [30].

*Theorem 3.3:* [38] Let $G = (X, E, \delta, x_0)$ be the system, $S \subseteq X$ be the secret and $E_I$ be the set of events observable by the intruder. The system is current-state opaque wrt $S$ and $E_I$ if and only if for all $\sigma \in L(G)$,

$$\mathcal{C}_I(w_i) \nsubseteq S,$$

where $w_i = P_I(\sigma)$.

In simple words, to verify if a system is current-state opaque wrt the given secret, one needs to compute the intruder's estimate $\mathcal{C}_I(P_I(\sigma))$ for all words $\sigma \in L(G)$ generated by the system and check whether $\mathcal{C}_I(P_I(\sigma)) \nsubseteq S$ holds. This can be done by constructing the *observer* (defined in Section 2.5.2 of [30]) of the system for the intruder (i.e., wrt $E_I$). The observer captures all state estimates of the intruder. More specifically, the state of the observer reached by $w_i$ is equal to $\mathcal{C}_I(w_i)$. Therefore, we can use the observer to verify current-state opacity. The algorithm to construct the observer can also be found in [30]. Herein, it is not recalled for the sake of brevity.

*Example 3.4:* Consider the system in Fig. 5. Let $E_I = \{o_2\}$ and $S = \{5\}$ (the secret state is in a box). The corresponding observer for the intruder is shown in Fig. 6. Since there exists $w_i = o_2$ such that $\mathcal{C}_I(w_i) = \{5\} \subseteq S$, by Theorem 3.3, the system is not current-state opaque wrt $S$ and $E_I$. ◇

Let us introduce the following notion of opacity that is related to a sublanguage of the generated language of the system and is useful to formalize the result of the work.

*Definition 3.5 (G-opaque Language):* Given a system $G = (X, E, \delta, x_0)$, a secret $S \subseteq X$ and a set $E_I$ of events observable by the intruder, a sublanguage $L \subseteq L(G)$ is said to be $G$-opaque (wrt $S$ and $E_I$) if $\forall \sigma \in L$ such that $\delta(x_0, \sigma) \in S$,

$$\exists \sigma' \in L(G) : \delta(x_0, \sigma') \notin S, \quad P_I(\sigma) = P_I(\sigma').$$

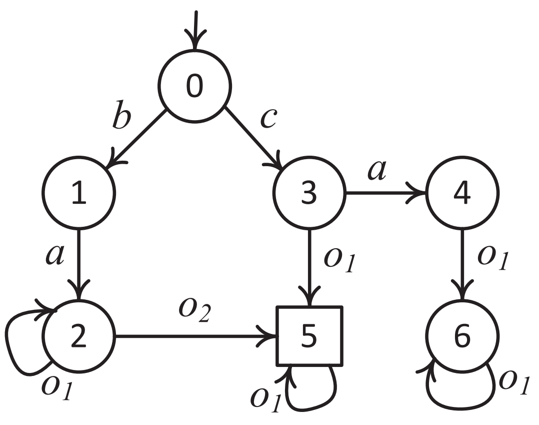

Fig. 5. System $G$ that is not CSO wrt $S = \{5\}$ and $E_I = \{o_2\}$ in Example 3.4.

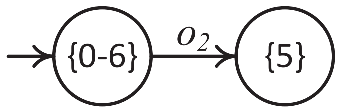

Fig. 6. Observer of the system in Fig. 5 for the intruder.

◇

Clearly, by Definitions 3.1 and 3.5, Corollary 3.6 follows.

*Corollary 3.6:* Given a system $G = (X, E, \delta, x_0)$, a secret $S \subseteq X$ and a set $E_I$ of events observable by the intruder, $G$ is current-state opaque wrt $S$ and $E_I$ if and only if $L(G)$ is $G$-opaque.

In other words, CSO of a system $G$ is equivalent to $G$-opacity of its generated language.

*Proposition 3.7:* Given a system $G$, a secret $S \subseteq X$, a set $E_I$ of events observable by the intruder, and two G-opaque languages $L_1, L_2 \subseteq L(G)$, then it holds:

i) $L_1 \cup L_2$ is $G$-opaque;

ii) $\forall L \subseteq L_1$, $L$ is $G$-opaque.

*Proof:* i) By assumption, $L_i$ (with $i = 1, 2$) is $G$-opaque. By Definition 3.5, for all $\sigma \in L_i, \mathcal{C}_I(P_I(\sigma)) \nsubseteq S$. Therefore, for all $\sigma \in L_1 \cup L_2, \mathcal{C}_I(P_I(\sigma)) \nsubseteq S$, i.e., $L_1 \cup L_2$ is $G$-opaque. ii) Given a subset $L$ of $L_i$, for all $\sigma \in L, \mathcal{C}_I(P_I(\sigma)) \nsubseteq S$, i.e., $L$ is $G$-opaque. ■

The $G$-opacity property of a language is closed under union, and the supremal $G$-opaque sublanguage of a given language exists. Any sublanguage of a $G$-opaque language is still $G$-opaque.

*Proposition 3.8:* Let $Sup/G$ be the controlled system of $G = (X, E, \delta, x_0)$ under a supervisor $Sup$, $E_I \subseteq E$ the set of events observable by the intruder, and $S \subseteq X$ the secret. Given a language $L \subseteq L(G)$, if $L$ is $Sup/G$-opaque wrt $S$ and $E_I$ then $L$ is $G$-opaque wrt $S$ and $E_I$.

*Proof:* Assume that $L$ is $Sup/G$-opaque. Then for all $\sigma \in L$ such that $\delta(x_0, \sigma) \in S$, there exists $\sigma' \in L(Sup/G)$ such that $\delta(x_0, \sigma') \notin S$ and $P_I(\sigma) = P_I(\sigma')$. Since $L(Sup/G) \subseteq L(G)$, there also exists $\sigma' \in L(G)$ such that $\delta(x_0, \sigma') \notin S$ and $P_I(\sigma) = P_I(\sigma')$. Therefore, $L$ is $G$-opaque wrt $S$ and $E_I$. ■

If $L(Sup/G) \subseteq L(G)$ is $Sup/G$-opaque (i.e., $Sup/G$ is CSO) then $L(Sup/G)$ is also $G$-opaque. Note that the converse of Proposition 3.8 is not true. In other words, even if the generated language $L(Sup/G)$ of a controlled system $Sup/G$ is $G$-opaque wrt $S$ and $E_I$, the controlled system $Sup/G$ may not be CSO wrt $S$ and $E_I$. Therefore, CSO of $Sup/G$ generally is a stronger requirement than $L(Sup/G)$ being $G$-opaque.

*Example 3.9:* Consider the system $G$ in Fig. 5 and its controlled system $Sup_2/G$ in Fig. 12(b). Let $S = \{5\}$, $E_I = \{o_1, o_2\}$, $E_S = \{o_1\}$, and $E_C = \{a, b, c\}$. Clearly, $L(Sup_2/G)$ is $G$-opaque wrt $S$ and $E_I$ but not $Sup_2/G$-opaque. Namely, $Sup_2/G$ is not CSO wrt $S$ and $E_I$. Indeed, when the intruder observes $o_1$, if it knows the structure of $Sup_2/G$, its estimate would be $\mathcal{C}_I(o_1) = \{5\} \subseteq S$, i.e., $Sup_2/G$ is not CSO; on the contrary, if the intruder does not know the structure of $Sup_2/G$, its estimate would be $\mathcal{C}_I(o_1) = \{2, 5, 6\} \nsubseteq S$, i.e., the intruder is not able to discover the secret. $\diamond$

Example 3.9 also shows that if the intruder knows the supervisor $Sup$, to guarantee that the intruder does not discover the secret, $L(Sup/G)$ should be $Sup/G$-opaque. On the contrary, if the intruder does not know the supervisor $Sup$, it is sufficient that $L(Sup/G)$ is $G$-opaque. In the latter case, the problem is equivalent to synthesizing a supervisor $Sup$ of $G$ such that $L(Sup/G)$ is $G$-opaque, which is clearly a weaker condition than $Sup/G$ being CSO.

Note that $G$-opacity of $L(Sup/G)$ may guarantee that the intruder cannot infer the secret also in some cases where the intruder knows that there is a supervisor acting on the system but has no sufficient information to determine it exactly. Suppose that the intruder knows that there is a supervisor and has some information on $E_S$ and $E_C$ but not precise. Then the intruder may synthesize an estimated supervisor $Sup'$ on $G$ such that $L(Sup'/G)$ is $Sup'/G$-opaque. However, if $L(Sup/G)$ is $Sup'/G$-opaque, then the intruder is still not able to discover the secret.

*Example 3.10:* Consider Example 3.9 again. Suppose now that the intruder knows that there is a supervisor and believes the supervisor can observe $E'_S = \{a, o_1\}$, and can control $E'_C = \{b, c\}$, which are different from what the supervisor really can observe and control. The estimated supervisor synthesized based on $E'_S$ and $E'_C$ is $Sup'$ which disables event $b$ when observing nothing. Consider the supervisor $Sup_2$ defined in Example 3.9. It is easy to see that, $L(Sup_2/G)$ is $Sup'/G$-opaque wrt $S$ and $E_I$. Therefore, under the control of $Sup_2$ the intruder is still not able to infer the secret. $\diamond$

For simplicity, in the remainder of the paper it is directly assumed that the intruder does not know that a supervisor is controlling the plant to enforce opacity at all. Introducing such an assumption enables us to solve opacity enforcement using supervisory control in an efficient way. Meanwhile, imposing such an assumption is reasonable and meaningful. Indeed, this is realistic in many practical situations. Furthermore, it is interesting from a theoretical point of view since it provides some insights into tackling more general and complicated problems.

## IV. Current-State Opacity Enforcement by Control

Given a system that is not current-state opaque wrt a secret, an interesting question is how to restrict its behavior or how to modify the observation structure such that the secret will never be revealed to the intruder. In this work we address the first issue using supervisory control theory [28]. The supervisor will restrict the system's behavior to prevent the evolutions leaking the secret. In this section, we present a novel approach to designing a locally optimal supervisor enforcing current-state opacity without assuming any containment relationship between $E_S$ and $E_I$, and between $E_C$ and $E_S$.

### A. Problem Formulation

The problem we want to solve in this work can be formalized as follows.

*Definition 4.1 (Current-State Opacity Enforcement Problem, CSOEP):* Given a system $G = (X, E, \delta, x_0)$, a secret $S \subseteq X$, a set $E_I$ of events observable by the intruder, a set $E_S$ of events observable by the supervisor, and a set $E_C$ of controllable events, synthesize a locally optimal supervisor $Sup$ such that

1) $L(Sup/G)$ is $G$-opaque wrt $S$ and $E_I$;

TABLE II
OBSERVABLE AND CONTROLLABLE EVENTS IN EXAMPLE 4.2.

| Events | $E_I$ | $E_S$ | $E_C$ |
|---|---|---|---|
| $o_1$ | $\times$ | $\surd$ | $\times$ |
| $o_2$ | $\surd$ | $\times$ | $\times$ |
| $a$ | $\times$ | $\times$ | $\surd$ |
| $b$ | $\times$ | $\times$ | $\surd$ |
| $c$ | $\times$ | $\times$ | $\surd$ |

2) For any other supervisor $Sup'$ such that $L(Sup'/G)$ is $G$-opaque wrt $S$ and $E_I$ it holds

$$L(Sup/G) \not\subset L(Sup'/G).$$

$\diamond$

A CSOEP involves the system $G$, the set $E_I$ of events observable by the intruder, the secret $S$, the set $E_S$ of events observable by the supervisor and the set $E_C$ of events controllable by the supervisor. To be concise, this problem is denoted as CSOEP($G, E_I, S, E_S, E_C$). A solution to the CSOEP is called a *locally optimal supervisor*.

*Example 4.2:* Consider again the system in Fig. 5. From Example 3.4 we know that the system is not current-state opaque wrt $S = \{5\}$ and $E_I = \{o_2\}$. Now we want to design a locally optimal supervisor $Sup$, so that $L(Sup/G)$ is $G$-opaque. The sets of events observable\controllable by the intruder and the supervisor are shown in Table II. In this case, $E_I$ and $E_S$ are not comparable, i.e., neither $E_I \subseteq E_S$ nor $E_S \subseteq E_I$ holds, and not all controllable events are observable, i.e., $E_C \not\subseteq E_S$. $\diamond$

*Proposition 4.3:* There exists a solution to the CSOEP if and only if there exists a prefix-closed language $K \subseteq L(G)$ such that

1) $K$ is controllable (wrt $L(G)$ and $E_C$) and observable (wrt $L(G)$, $E_S$ and $E_C$);
2) $K$ is $G$-opaque (wrt $S$ and $E_I$);
3) For any other controllable, observable and $G$-opaque language $K' \subseteq L(G)$, $K \not\subset K'$.

*Proof:* By Theorem 2.4, the first item is a necessary and sufficient condition for the existence of a supervisor that restricts the behaviour of the system to $K$. Items 2 and 3 correspond to items 1 and 2, respectively, of Definition 8 that formalizes the requirements that a supervisor has to satisfy for being a locally optimal solution to the CSOEP. ■

Thus, to solve the CSOEP we have to compute a prefix-closed maximal controllable, observable and $G$-opaque sublanguage of $L(G)$. It is known that the supremal observable sublanguage may not exist. Therefore such a maximal controllable, observable and $G$-opaque sublanguage, if it exists, may not be unique. In other words, there may exist a set of locally optimal supervisors.

In the next subsection, we introduce a structure, called *augmented I-observer*, based on which the supremal $G$-opaque sublanguage can be characterized and the optimal supervisors can be designed.

## B. Synthesis of Locally Optimal Supervisors

To design locally optimal supervisors, we have to characterize a maximal controllable and observable behavior of the system such that the secret will never be leaked. To do this, we need to first characterize the supremal $G$-opaque sublanguage of the system as the specification language $K$, and then compute a maximal controllable and observable sublanguage of $K$. Indeed, by Proposition 3.7 if a language is $G$-opaque, any sublanguage of it is still $G$-opaque. Unfortunately, the absence of specific containment relationships between sets $E_I$ and $E_S$ makes the solution via a single structure, as in [18], [21], tricky. In the

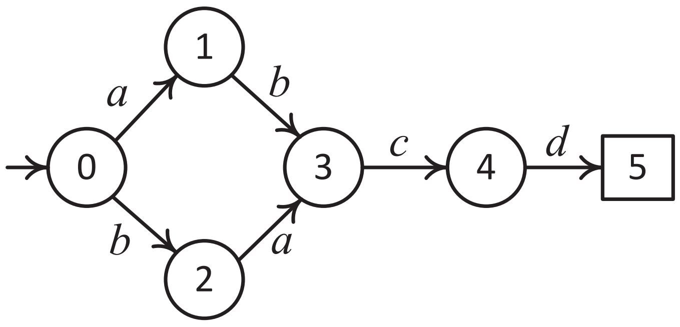

Fig. 7. System $G$ that is not CSO wrt $S = \{5\}$ and $E_I = \{a, d\}$.

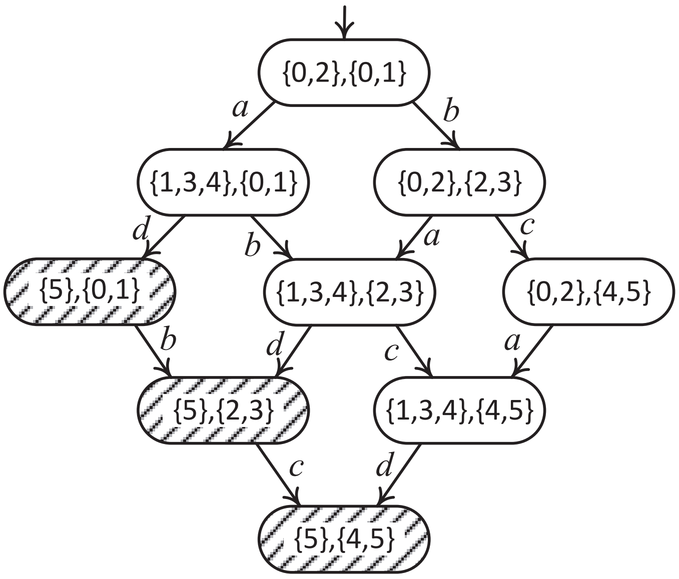

Fig. 8. Parallel composition $N$ of the observers for the intruder and the supervisor.

following we provide an example where the approach in [18] fails since none of the containment relationships $E_I \subseteq E_S$ or $E_S \subseteq E_I$ holds.

*Example 4.4:* Consider the system in Fig. 7. Let $E_I = \{a, d\}$, $E_S = \{b, c\}$, $E_C = \{c\}$, and $S = \{5\}$. Obviously, the system is not opaque wrt $S$ and $E_I$ since when the intruder observes $ad$ it unambiguously knows that the current state is 5. According to [18], observers of the system for the intruder and the supervisor should be constructed first. Then we have to compute the parallel composition $N$ of these two observers to characterize the behavior that would leak the secret and that should be forbidden (see Fig. 8, states in shadow should be unreachable).

Finally, by computing the observer (wrt $E_S$) of the parallel composition structure the optimal supervisor can be obtained. Without the assumption $E_I \subseteq E_S$ or $E_S \subseteq E_I$, the parallel composition between the observers would introduce event sequences (e.g., $\sigma = abd$) not belonging to $P_o(L(G))$, where $P_o : E^* \to (E_I \cup E_S)^*$. In the case at hand, being $E_o = E$, it is $P_o(L(G)) = L(G)$. As a result, the behavior of the system would be over restricted. For instance, sequence $ab$ does not leak the secret. However, it should be disabled: $N$ tells that after uncontrollable event $d$ occurs, sequence $abd$ will lead to a state in shadow. Therefore, the obtained supervisor would not be optimal, or even no such an opacity enforcing supervisor exists (as in the case at hand). Note that assuming $E_C = \{o_1\} = E_S$ the approach in [18] coincidentally works for Example 4.2 though neither $E_I \subseteq E_S$ nor $E_S \subseteq E_I$ holds. $\diamond$

In this work, we show that locally optimal supervisors for the CSOEP can be designed in two phases, without assuming $E_I \subseteq E_S$, or $E_S \subseteq E_I$, or $E_C \subseteq E_S$. First, by introducing a structure, called *augmented I-observer*, the supremal $G$-opaque sublanguage can be computed. Then, applying the method recalled in Section II-C to the augmented I-observer, the locally optimal supervisors can be designed.

The augmented I-observer of system $G = (X, E, \delta, x_0)$ is a DFA denoted as $\mathcal{A} = (Q, E, \delta_a, q_0)$ and consists in the parallel composition of the observer for the intruder and the system $G$. In more detail, a state $q \in Q$ of $\mathcal{A}$ is a pair $(C_I, x)$, where $C_I \subseteq X$ and $x \subseteq X$. The initial state of the augmented I-observer is $q_0 = (R_I(x_0, \varepsilon), x_0)$. Algorithm 1 illustrates the construction of the augmented I-observer. It also computes the set $F = \{q = (C_I, x) \in Q | C_I \subseteq S\}$ that contains the set of states of $\mathcal{A}$ where the estimate $C_I$ of the intruder is a subset of the secret. As proven in the following, set $F$ allows one to identify a necessary and sufficient condition for current-state opacity of $G$, and to define the supremal $G$-opaque sublanguage of $L(G)$.

**Algorithm 1** Computation of the Augmented I-observer

```
Input: A system $G = (X, E, \delta, x_0)$, the sets of events $E_I$ and the secret $S$.
Output: An augmented I-observer $\mathcal{A} = (Q, E, \delta_a, q_0)$ and the subset $F$ of $Q$.
 1: $q_0 := (R_I(x_0, \varepsilon), x_0)$ and assign no tag to it;
 2: $Q := \{q_0\}$;
 3: if $R_I(x_0, \varepsilon) \subseteq S$, then
 4:     $F := \{q_0\}$;
 5: else
 6:     $F := \emptyset$;
 7: end if
 8: while $q = (C_I, x) \in Q$ with no tag exists, do
 9:     for all $e \in E$ such that $\delta(x, e)!$, do
10:         if $e \in E_I$, then
11:             $C'_I := \bigcup_{x \in C_I} R_I(x, e)$;
12:         else
13:             $C'_I := C_I$;
14:         end if
15:         $x' := \delta(x, e)$;
16:         $q' := (C'_I, x')$;
17:         if $q' \notin Q$ then
18:             $Q := Q \cup \{q'\}$;
19:         end if
20:         if $C'_I \subseteq S$, then
21:             $F := F \cup \{q'\}$;
22:         end if
23:         $\delta_a(q, e) := q'$;
24:     end for
25:     Tag $q$ "old";
26: end while
27: Remove all tags;
28: Output $\mathcal{A}$.
```

Now we explain the main ideas behind Algorithm 1. The initial state of the augmented I-observer is $q_0 = (R_I(x_0, \varepsilon), x_0)$, i.e., the pair (set of states estimated by the intruder when observing nothing, initial state of the system). Given a state $q = (C_I, x) \in Q$ and an event $e \in E$ that is defined at $x$ in $G$, using Algorithm 1, the generic state $\delta_a(q, e) = q' = (C'_I, x')$ in the augmented I-observer is computed as follows. $C'_I$ is updated to the new intruder estimate when event $e$ is observed by the intruder; otherwise, $C_I = C'_I$. State $x'$ is reached by the occurrence of $e$ at $x$ in $G$. If $q'$ is a new state, it is added to $Q$,

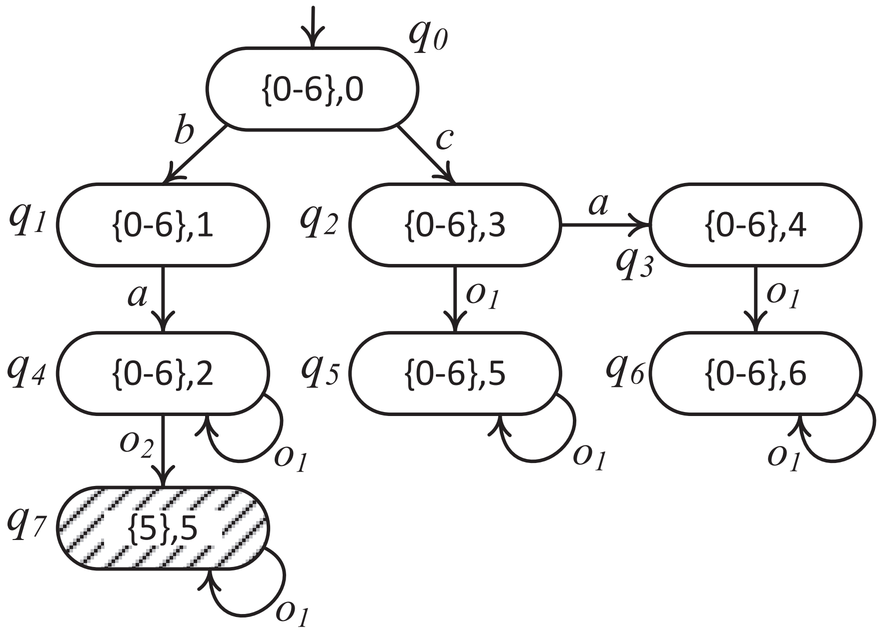

Fig. 9. Augmented I-Observer $\mathcal{A}$ of the system in Example 4.5, where states in $F$ are in dashed boxes.

otherwise $Q$ does not change. The maximum number of states of the augmented I-observer is $|X| \times 2^{|X|}$.

We finally note that the augmented I-observer differs from the parallel composition of observers proposed in [18] and the parallel observer proposed in [29].

*Example 4.5:* Consider the problem in Example 4.2. Using Algorithm 1, the augmented I-observer is constructed and shown in Fig. 9, where states in $F$ are in dashed boxes. ◇

*Proposition 4.6:* Let $G = (X, E, \delta, x_0)$ be a system, $E_I$ the set of events observable by the intruder, and $S$ the secret. The augmented I-observer $\mathcal{A} = (Q, E, \delta_a, q_0)$ constructed using Algorithm 1 has the following properties:

i) $L(\mathcal{A}) = L(G)$;

ii) $\{\sigma \in L(\mathcal{A}) | \delta_a(q_0, \sigma) \in F\} = \{\sigma \in L(G) | \mathcal{C}_I(P_I(\sigma)) \subseteq S\}$.

*Proof:*

i) The statement follows from the fact that Steps 9 and 15 of Algorithm 1 consider all the events (and only them) that are defined at each state of $G$.

ii) Let $q = (C_I, x) = \delta_a(q_0, \sigma)$. By Steps 3 to 7, and 20 to 22 of Algorithm 1, and Eq. (1), $C_I = \mathcal{C}_I(P_I(\sigma))$ holds. Therefore, $\delta_a(q_0, \sigma) \in F$ if and only if $\mathcal{C}_I(P_I(\sigma)) \subseteq S$.

■

Moreover, by Steps 3 to 5 and 20 to 22 of Algorithm 1, there exists $\sigma \in L(\mathcal{A})$ such that $\mathcal{C}_I(P_I(\sigma)) \subseteq S$ if and only if $F \neq \emptyset$. Therefore, we have the following corollary showing that the augmented I-observer can also be used to verify current-state opacity.

*Corollary 4.7:* Given a system $G$, a secret $S$ and the sets of events $E_I$ and $E_S$, let $\mathcal{A} = (Q, E, \delta_a, q_0)$ be the augmented I-observer. $G$ is current-state opaque wrt $S$ and $E_I$ if and only if $F = \emptyset$.

*Proof:* Follows from Steps 3 to 5 and 20 to 22 of Algorithm 1, Proposition 4.6 and Theorem 3.3. ■

The following proposition shows how it is possible to compute the supremal $G$-opaque sublanguage of $G$ using the augmented I-observer.

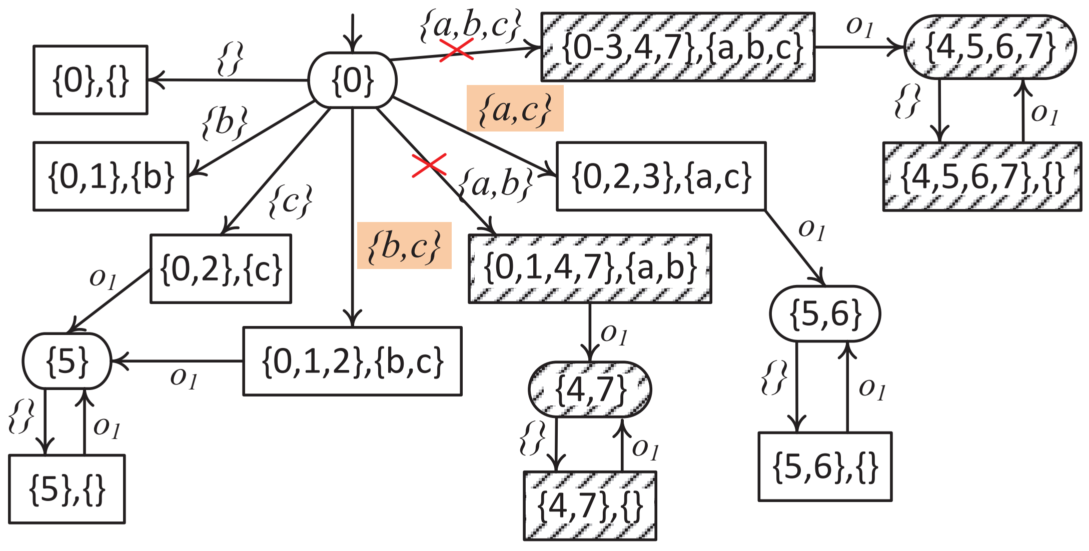

Fig. 10. Total controller of $\mathcal{A}$ in Fig. 9. Removing the states in the dashed boxes, the all inclusive controller is obtained. For simplicity, in the diagrams, we use $i$ (with $i = 0, 1, \ldots, 7$) to denote state $q_i$ in the augmented I-observer and omit all uncontrollable events in the control decisions, e.g., decision $\{\}$ represents $\{o_1, o_2\}$, and so forth.

*Proposition 4.8:* The supremal $G$-opaque sublanguage of $L(G)$ is $K = \{\sigma \in L(\mathcal{A}) | \delta_a(q_0, \sigma) \notin F\}$.

*Proof:* First, we prove that $K$ is $G$-opaque. Let $\sigma \in K, \delta_a(q_0, \sigma) = q = (C_I, C_S)$. Since $q \notin F$, $C_I \nsubseteq S$, i.e., $\mathcal{C}_I(w_i) \nsubseteq S$, where $w_i = P_I(\sigma)$. Therefore, $K$ is $G$-opaque. Now we show that $K$ is the "largest" opaque sublanguage of $L(G)$ and for any other opaque language $L \subseteq L(G)$, $L$ is contained in $K$. Let $\sigma \in L$ and $q = \delta_a(q_0, \sigma) = (C_I, C_S)$. Since $L$ is opaque, $\mathcal{C}_I(P_I(\sigma)) \nsubseteq S$, i.e., $C_I \nsubseteq S$, $q \notin F$ and $\sigma \in K$. Therefore, $L$ is a subset of $K$ and $K$ contains all opaque sublanguages of $G$. ■

Therefore, by means of the augmented I-observer we can compute the supremal opaque sublanguage of $G$, and by Propositions 3.7 and 4.3, the CSOEP can be solved by computing a maximal sublanguage of $K$ that is prefix-closed, controllable and observable. The following theorem states that the CSOEP($G, E_I, S, E_S, E_C$) is equivalent to the SCOP($\mathcal{A}, E_S, E_C, K$), i.e., based on the augmented I-observer locally optimal supervisors can be synthesized to enforce current-state opacity to a system $G$.

*Theorem 4.9:* The set of solutions to the CSOEP($G, E_I, S, E_S, E_C$) coincides with the set of solutions to the SCOP($\mathcal{A}, E_S, E_C, K$), where $\mathcal{A}$ is the augmented I-observer of $G$ and $K = \{\sigma \in L(\mathcal{A}) | \delta_a(q_0, \sigma) \notin F\}$.

*Proof:* We prove this theorem by showing that CSOEP($G, E_I, S, E_S, E_C$) and SCOP($\mathcal{A}, E_S, E_C, K$) define the same supervisory control problem. By Proposition 3.7, any sublanguage of a $G$-opaque language is still $G$-opaque. By Proposition 4.8, $K$ is the supremal $G$-opaque sublanguage of $G$. Therefore, condition 1 in Definition 4.1 can be rephrased as "$L(Sup/G) \subseteq K$", same as condition 1 in Definition 2.5. Moreover, $L(G) = L(\mathcal{A})$. Therefore, the CSOEP($G, E_I, S, E_S, E_C$) and the SCOP($\mathcal{A}, E_S, E_C, K$) define the same supervisory control problem, and thus they share the same set of solutions. Namely, if $Sup$ is a locally optimal supervisor of SCOP($\mathcal{A}, E_S, E_C, K$), then $Sup$ is also a locally optimal supervisor of CSOEP($G, E_I, S, E_S, E_C$), and vice versa. ■

In other words, CSOEP($G, E_I, S, E_S, E_C$) can be solved by synthesizing a locally optimal supervisor of $\mathcal{A}$ with $F$ being the set of forbidden states.

*Example 4.10:* By Theorem 4.9, the CSOEP in Example 4.2 is reduced to the problem of finding a locally optimal supervisor $Sup$ for $\mathcal{A}$ such that state $q_7$ of $\mathcal{A}$ is not reachable in the controlled system. Applying the approach recalled in Section II-B, first

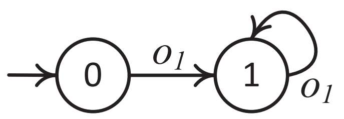

Fig. 11. Supervisors of the CSOEP in Example 4.2: $Sup_1$ and $Sup_2$. They have the same automaton structure. However, for $Sup_1$, $\Psi(0) = \{a, c, o_1, o_2\}$ and $\Psi(1) = \{o_1, o_2\}$; for $Sup_2$, $\Psi(0) = \{b, c, o_1, o_2\}$ and $\Psi(1) = \{o_1, o_2\}$.

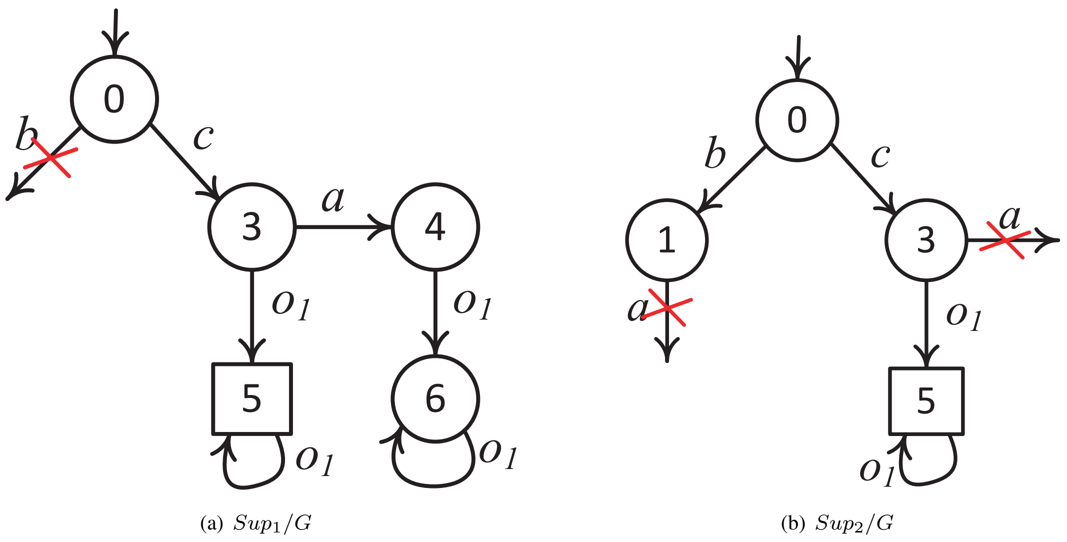

Fig. 12. Controlled system under different locally optimal supervisors in Example 4.10.

we construct the total controller in Fig. 10 and then, after removing all the states that contain forbidden state 7 (i.e., $q_7$ in $\mathcal{A}$), we obtain the all inclusive controller. Finally, at each step we choose a local maximal control decision and all locally optimal supervisors are computed. There are two locally optimal supervisors: $Sup_1$ and $Sup_2$ with the same automaton structure shown in Fig. 11. For $Sup_1$, $\Psi(0) = \{a, c, o_1, o_2\}$ and $\Psi(1) = \{o_1, o_2\}$; for $Sup_2$, $\Psi(0) = \{b, c\}$ and $\Psi(1) = \{o_1, o_2\}$. The controlled systems under $Sup_1$ and $Sup_2$ are shown in Fig. 12. $\diamond$

## C. Computational complexity analysis

According to the previous analysis, in the worst case the number of states of the augmented I-observer is $|X| \times 2^{|X|}$, where $X$ is the set of states of $G$. Since the complexity of solving the SCOP is $\mathcal{O}(|Q||E|2^{|Q|+|E_C|})$, where $Q$ is the set of states of the augmented I-observer, the worst-case complexity of solving the CSOEP is $\mathcal{O}(|X| \times 2^{|X|}|E|2^{|X|\times 2^{|X|}+|E_C|})$, i.e., double exponential in the number of states of $G$. It is clear that one exponential order comes from the construction of the augmented I-observer and the other one comes from the method adopted in this paper to solve the SCOP.

We point out that in some cases (e.g., finding a near optimal supervisor [31], [39], on-line synthesizing the supervisor [34]), the complexity of solving the SCOP may decrease and consequently so would be the complexity of solving CSOEP.

Assuming that the intruder has no knowledge of the supervisor, the proposed approach can solve the same problems in [18], [21], [29] with the same or lower complexity: exponential or double exponential, respectively. Consider the problem in [18] where $E_I \subseteq E_S = E$, $E_C \subseteq E_S$. The augmented I-observer contains all observations of the supervisor (i.e., $P_S(L(\mathcal{A})) = L(\mathcal{A})$). Therefore, the augmented I-observer can be used to synthesize the supervisor directly. Moreover, due to $E_C \subseteq E_S$ the complexity of the proposed approach reduces to $\mathcal{O}(|X| \times 2^{|X|})$ as same as the complexity of the approach in [18]. On the other hand, the complexity of solving the problem in [29] (where $E_S$ and $E_I$ are incomparable but $E_C \subseteq E_S$) using the proposed approach is $\mathcal{O}(2^{(|X|\times 2^{|X|})})$, lower than that of the approach in [29]. In addition, if either $E_S \subseteq E_I$ (or $E_I \subseteq E_S$)

or $E_C \subseteq E_S$ holds, the supervisory synthesis problem considered in the paper cannot be solved using the approaches in [20], [21], [27], [29].

## V. Conclusions and Future Work

In this paper, we proposed a novel approach to solve the problem of current-state opacity enforcement in discrete event systems using finite automata. By constructing the augmented I-observer, all the strings that will leak the secret can be characterized. Based on the augmented I-observer, current-state opacity can be checked and a synthesis algorithm was provided to design a locally optimal supervisor, without assuming the existence of containment relationships between $E_I$ and $E_S$, or between $E_C$ and $E_S$.

There are several directions along which the current research could be extended. First we note that the proposed approach can be applied to systems modeled by nondeterministic finite automata (NFA). In this case the obtained augmented I-observer will be an NFA as well. We also point out that the proposed approach can be extended to Petri nets, a model that is more powerful than finite automata. Moreover, some structural properties of Petri nets may be useful to further reduce the computational complexity and this is one direction for our future research. The other direction is to develop a unified structure that combines the features of augmented I-observer and the total controller so that the complexity of solving CSOEP could decrease.

## Acknowledgement

The authors would like to acknowledge the valuable remarks of an anonymous reviewer who pointed out a way to simplify the approach proposed in an earlier version of the manuscript.

## References

[1] J. A. Goguen and J. Meseguer. Security policies and security models. In *Proceedings of the 2012 IEEE Symposium on Security and Privacy*, pages 11–20, 1982.

[2] N. Busi and R. Gorrieri. A survey on non-interference with Petri nets. In *Lectures on Concurrency and Petri Nets*, pages 328–344. Springer, 2004.

[3] M. K. Reiter and A. D. Rubin. Crowds: Anonymity for web transactions. *ACM Transactions on Information and System Security*, 1(1):66–92, 1998.

[4] V. Shmatikov. Probabilistic analysis of an anonymity system. *Journal of Computer Security*, 12(3):355–377, 2004.

[5] N. B. Hadj-Alouane, S. Lafrance, F. Lin, J. Mullins, and M. M. Yeddes. On the verification of intransitive noninterference in mulitlevel security. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 35(5):948–958, 2005.

[6] J. W. Bryans, M. Koutny, and P. Y. Ryan. Modelling opacity using Petri nets. *Electronic Notes in Theoretical Computer Science*, 121:101–115, 2005.

[7] A. Saboori and C. N. Hadjicostis. Verification of initial-state opacity in security applications of DES. In *Proceedings of the 9th International Workshop on Discrete Event Systems*, pages 328–333, 2008.

[8] F. Cassez, J. Dubreil, and H. Marchand. Dynamic observers for the synthesis of opaque systems. In *Automated Technology for Verification and Analysis*, pages 352–367. Springer, 2009.

[9] Y. C. Wu and S. Lafortune. Comparative analysis of related notions of opacity in centralized and coordinated architectures. *Discrete Event Dynamic Systems*, 23(3):307–339, 2013.

[10] J. W. Bryans, M. Koutny, L. Mazaré, and P. Y. Ryan. Opacity generalised to transition systems. *International Journal of Information Security*, 7(6):421–435, 2008.

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

[12] B. Zhang, S. L. Shu, and F. Lin. Polynomial algorithms to check opacity in discrete event systems. In *Proceedings of the 24th Chinese Control and Decision Conference*, pages 763–769. IEEE, 2012.

[13] Y. Tong, Z. W. Li, C. Seatzu, and A. Giua. Verification of current-state opacity using Petri nets. In *Proceedings of the 2015 American Control Conference*, pages 1935–1940, Chicago, US, 2015. IEEE.

[14] Y. Tong, Z. W. Li, C. Seatzu, and A. Giua. Verification of initial-state opacity in Petri nets. In *Proceedings of the 2015 International Conference on Decision and Control*, pages 344-349, Osaka, Japan, 2015. IEEE.

[15] Y. Tong, Z. Y. Ma, Z. W. Li, C. Seatzu, and A. Giua. Verification of language-based opacity in Petri nets using verifier. In *Proceedings of the 2016 American Control Conference*, pages 757–763, Boston, US. IEEE.

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

[17] S. Takai and Y. Oka. A formula for the supremal controllable and opaque sublanguage arising in supervisory control. *SICE Journal of Control, Measurement, and System Integration*, 1(4):307–311, 2008.

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

[19] M. Ben-Kalefa and F. Lin. Supervisory control for opacity of discrete event systems. In *Proceedings of the 49th Annual Allerton Conference on Communication, Control, and Computing*, pages 1113–1119, Sept 2011.

[20] A. Saboori and C. N. Hadjicostis. Opacity-enforcing supervisory strategies via state estimator constructions. *IEEE Transactions on Automatic Control*, 57(5):1155–1165, 2012.

[21] X. Yin and S. Lafortune. A new approach for synthesizing opacity-enforcing supervisors for partially-observed discrete-event systems. In *Proceedings of the 2015 American Control Conference*, pages 377–383, Chicago, US, 2015. IEEE.

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

[23] Y. C. Wu and S. Lafortune. Synthesis of insertion functions for enforcement of opacity security properties. *Automatica*, 50(5):1336–1348, 2014.

[24] Y. C. Wu and S. Lafortune. Synthesis of opacity-enforcing insertion functions that can be publicly known. In *Proceedings of the 54th IEEE Conference on Decision and Control*, pages 3506–3513, Dec 2015.

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

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

[27] J. Dubreil, P. Darondeau, and H. Marchand. Opacity enforcing control synthesis. In *Proceedings of the 9th International Workshop on Discrete Event Systems,*, pages 28–35. IEEE, 2008.

[28] P. J. G. Ramadge and W. M. Wonham. The control of discrete event systems. *Proceedings of the IEEE*, 77(1):81–98, 1989.

[29] Y. Tong, Z. Ma, Z. Li, C. Seatzu, and A. Giua. Supervisory enforcement of current-state opacity with uncomparable observations. In *Proceedings of the 13th International Workshop on Discrete Event Systems*, pages 313–318, May 2016.

[30] C. G. Cassandras and S. Lafortune. *Introduction to discrete event systems*. Springer, 2008.

[31] M. Heymann and F. Lin. On-line control of partially observed discrete event systems. *Discrete Event Dynamic Systems*, 4(3):221–236, 1994.

[32] N. Hadj-Alouane, S. Lafortune, and F. Lin. Centralized and distributed algorithms for on-line synthesis of maximal control policies under partial observation. *Discrete Event Dynamic Systems*, 6(4):379–427, 1996.

[33] Y. Ru, M. P. Cabasino, A. Giua, and C. N. Hadjicostis. Supervisor synthesis for discrete event systems under partial observation and arbitrary forbidden state specifications. *Discrete Event Dynamic Systems*, 24(3):275–307, 2014.

[34] K. Cai, R. Zhang, and W. Wonham. Relative observability of discrete-event systems and its supremal sublanguages. *IEEE Transactions on Automatic Control*, 60(3):659–670, 2015.

[35] X. Yin and S. Lafortune. Synthesis of maximally permissive supervisors for partially-observed discrete-event systems. *IEEE Transactions on Automatic Control*, 61(5):1239–1254, 2016.

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

[37] Hangju Cho and Steven I Marcus. On supremal languages of classes of sublanguages that arise in supervisor synthesis problems with partial observation. *Mathematics of Control, Signals, and Systems (MCSS)*, 2(1):47–69, 1989.

[38] A. Saboori and C.N. Hadjicostis. Notions of security and opacity in discrete event systems. In *Proceedings of the 46th IEEE Conference on Decision and Control*, pages 5056–5061. IEEE, 2007.

[39] T. Ushio. On-line control of discrete event systems with a maximally controllable and observable sublanguage. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 82(9):1965–1970, 1999.

[40] Y. Tong, Z. W. Li, C. Seatzu, and A. Giua. Decidability of opacity verification problems in labeled Petri net systems. *Automatica*, 80:48–53, 2017.