2015 American Control Conference
Palmer House Hilton
July 1-3, 2015. Chicago, IL, USA

A New Approach for Synthesizing Opacity-Enforcing Supervisors for Partially-Observed Discrete-Event Systems

Xiang Yin and Stéphane Lafortune

Abstract— **Opacity is a confidentiality property for partially-observed discrete-event systems relevant to the analysis of security and privacy in cyber and cyber-physical systems. It captures the plausible deniability of the system's "secret" in the presence of an outside observer that is potentially malicious. In this paper, we consider the enforcement of opacity on systems modeled by finite-state automata. We assume that the given system is not opaque and the objective is to restrict its behavior by supervisory control in order to enforce opacity of its secret. We consider the general setting of supervisory control under partial observations where the controllable events need not all be observable. Our approach for the synthesis of an opacity enforcing supervisor is based on the construction of a new transition system that we call the "All Inclusive Controller for Opacity" (or AIC-O). The AIC-O is a finite bipartite transition system that embeds in its transition structure all valid opacity enforcing supervisors. We present an algorithm for the construction of the AIC-O and discuss its properties. We then develop a synthesis algorithm, based on the AIC-O, that constructs a "maximally permissive" opacity-enforcing supervisor. Our approach generalizes previous approaches in the literature for opacity enforcement by supervisory control.**

I. Introduction

Security and privacy are important issues for networked cyber and cyber-physical systems. In this paper, we investigate an important security property called *opacity* that was originally introduced in the computer science literature [1] and since then has been investigated extensively in the framework of Discrete-Event Systems (DES), among other approaches; see, e.g., [2]–[15]. An opacity problem is formulated as follows in the context of DES. The system is modeled as a finite-state automaton; there is a *secret* that the system wants to hide from a potentially malicious external observer, referred to as the *intruder*. The intruder is modeled as an observer that knows explicitly the system's structure but can only observe part of the system's behavior. Given a secret, we say that the system is *opaque* if the intruder can never determine unambiguously that the secret has occurred based on its limited observation capabilities. Specifically, we need that for any behavior of the system that reveals its secret (a "secret behavior"), there must exist another behavior that is observationally equivalent to the secret behavior but does not reveal the secret (a "non-secret behavior"). The secret of the system can take many forms, such as a subset of states (initial or current), a subset of state pairs (initial-final), or a subset of strings. Since most of these notions can be mapped to one another (see [13]), we consider in this paper the property of *current-state opacity*, without essential loss of generality. In current-state opacity, the secret is revealed if the state estimate of the intruder is entirely contained in the set of secret states of the system.

Several approaches have been proposed in the literature for enforcing opacity of a given system that is not opaque at the outset; see, e.g., [6]–[12], [14]–[16]. One of the most commonly-used opacity enforcement mechanism is to use supervisory control to restrict the system behavior; this is the approach investigated in [6]–[11], [16]. In this context, the control problem is to synthesize a partial-observation supervisor that prevents behaviors that reveal the secret from occurring in the controlled system. In [6], the system is assumed to be fully controlled and fully observed by the supervisor and the objective is to hide the system's secret in the presence of multiple intruders. In [7]–[9], E_c and E_o are the sets of events that can be controlled and observed by the supervisor, respectively; the goal is to design a least restrictive supervisor such that the controlled system is opaque with respect to E_a, the set of events that can be observed by the intruder. In these works, it is assumed that $E_c \subseteq E_o$. Besides the supervisory control approach, other enforcement mechanisms have also been investigated in the literature. In [15], the authors propose an enforcement mechanism based on insertion of additional observable events at the system's output; these events are observationally equivalent to genuine system's observable events from the viewpoint of the intruder, thereby creating confusion on its part. Another opacity enforcement mechanism is based on the use of a dynamic observer [12]. Finally, in [14], a runtime enforcement mechanism by using delay to enforce the notion of K-step opacity is proposed.

In this paper, we adopt the enforcement mechanism of supervisory control and solve the following problem: Given a system that is not current-state opaque for a given set of secret states, design a supervisory controller that restricts the behavior of the system such that the controlled system is opaque for the given set of secret states. The approach we develop to tackle this problem is significantly different from that in [7]–[9], which are also concerned with opacity enforcement by supervisory control. Specifically, our approach is based on the construction of a finite information structure called the *All Inclusive Controller for Opacity* and abbreviated as AIC-O. The AIC-O is a game structure

This work was partially supported by NSF grants CCF-1138860 (Expeditions in Computing project ExCAPE: Expeditions in Computer Augmented Program Engineering) and CNS-1421122, and by the TerraSwarm Research Center, one of six centers supported by the STARnet phase of the Focus Center Research Program (FCRP) a Semiconductor Research Corporation program sponsored by MARCO and DARPA.

Xiang Yin and Stéphane Lafortune are with the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109, USA. {xiangyin,stephane}@umich.edu.

between the supervisor and the environment (aka system). By construction, the AIC-O embeds in its structure *all* supervisors that enforce opacity. Therefore, it can serve as the basis for the synthesis problem. The AIC-O was inspired by our recent work in [17], [18], which aims to solve the standard supervisory control problem for safety and non-blockingness [19] by using the same type of transition system capturing the possible moves of the system and the set of admissible supervisors. However, in contrast to [17], [18], the property of interest here is opacity. Moreover, we relax the assumption made in [7]–[9] that all controllable events should be observable. In this more general setting, uniqueness of a maximally permissive solution is lost. Hence, our focus is on the synthesis of solutions that are provably (locally) maximally permissive.

The contributions of this paper are as follows.

- A novel finite information structure, the All Inclusive Controller for Opacity, that embeds all valid opacity-enforcing supervisors, is defined.
- A construction algorithm for the AIC-O is given and its properties are characterized.
- The supervisory control problem for opacity is solved using the AIC-O. The synthesis algorithm that is presented always returns a maximally permissive supervisor, even when $E_c \not\subseteq E_o$.

The remainder of this paper is organized as follows. In Section II, we describe the model of the system. The problem we solve in this paper is formally formulated in Section III. In Section IV, we define a class of bipartite transition systems that is used for solving the opacity enforcement problem. In Section V, we define the structure called AIC-O, the key notion for the approach investigated in this paper. We then present the synthesis algorithm that returns a maximally-permissive partial-observation supervisor based on the AIC-O in Section VI. Finally, we conclude the paper in Section VII. Due to space constraints, all proofs have been omitted and they are available in [20].

II. Preliminaries

The DES of interest is modeled as a deterministic finite-state automaton $G = (X, E, f, X_0)$, where X is the finite set of states, E is the finite set of events, $f : X \times E \to X$ is the partial transition function, where $f(x, \sigma) = y$ means that there is a transition labelled by event σ from state x to state y and $X_0 \subseteq X$ is the set of initial states. The transition function f is extended to $X \times E^*$ in the usual manner (see, e.g., [21]). Note that the initial state of the system G is not unique in general, since X_0 is a set of states. Given a set of states $S \subseteq X$, the language generated by G from S is defined by $\mathcal{L}(G, S) := \{s \in E^* : \exists x \in S \text{ s.t. } f(x, s)!\}$, where ! means "is defined". When $S = X_0$, the prefix-closed language $\mathcal{L}(G, X_0)$ describes the entire system's behavior; we denote it by $\mathcal{L}(G)$ for simplicity.

In the framework of supervisory control [19], the plant G is controlled by a *supervisor* that dynamically enables/disables events of the system such that some specification is provably achieved. The event set E is partitioned into two disjoint subsets: $E = E_c \dot\cup E_{uc}$, where E_c is the set of controllable events and E_{uc} is the set of uncontrollable events. We say that a control decision $\gamma \in 2^E$ is admissible if $E_{uc} \subseteq \gamma$, namely, uncontrollable events can never be disabled. We define $\Gamma = \{\gamma \in 2^E : E_{uc} \subseteq \gamma\}$ as the set of admissible control decisions. When the system is partially-observed [22], [23], E is also partitioned into two disjoint sets: $E = E_o \dot\cup E_{uo}$, where E_o is the set of observable events and E_{uo} is the set of unobservable events. The natural projection $P : E^* \to E_o^*$, is defined by

$$P(\epsilon) = \epsilon \text{ and } P(s\sigma) = \begin{cases} P(s)\sigma & \text{if } \sigma \in E_o \\ P(s) & \text{if } \sigma \in E_{uo} \end{cases} \quad (1)$$

Since a supervisor can only make decisions based on its observations, a partial-observation supervisor is a function $S_P : P(\mathcal{L}(G)) \to \Gamma$. We use the notation S_P/G to represent the controlled system and the language generated by S_P/G, denoted by $\mathcal{L}(S_P/G)$, is defined recursively as follows:

i) $\epsilon \in \mathcal{L}(S_P/G)$; and
ii) $[s \in \mathcal{L}(S_P/G) \wedge s\sigma \in \mathcal{L}(G) \wedge \sigma \in S_P(s)]$
$\Leftrightarrow [s\sigma \in \mathcal{L}(S_P/G)]$.

Given a prefix-closed language K, i.e, $\overline{K} = K$, we say that K is *controllable* (w.r.t. G and E_c) if $(\forall s \in K, \sigma \in E_{uc})(s\sigma \in \mathcal{L}(G) \Rightarrow s\sigma \in K)$; we say that K is *observable* (w.r.t. G, E_c, and E_o) if $(\forall s, s' \in K, \sigma \in E_c)(P(s) = P(s') \wedge s\sigma \in K \wedge s'\sigma \in \mathcal{L}(G) \Rightarrow s'\sigma \in K)$. It is well-known that there exists a supervisor S_P such that $\mathcal{L}(S_P/G) = K$ if and only if K is controllable and observable [22], [23].

We define several operators that will be used in this paper. The set of *all possible reachable states* from a set of states $Q \subseteq X$ under string $s \in \mathcal{L}(G)$, is given by

$$R_G(s, Q) := \{x \in X : \exists q \in Q, \exists t \in \mathcal{L}(G) \text{ s.t. } P(t) = P(s) \wedge x = f(q, t)\} \quad (2)$$

We denote $R_G(s, Q)$ by $R_G(s)$ if $Q = X_0$.

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

$$\text{UR}_\gamma(S) := \{x \in X : (\exists u \in S) (\exists e \in (E_{uo} \cap \gamma)^*) \text{ s.t. } x = f(u, e)\}. \quad (3)$$

The *observable transition* of the subset of states $S \subseteq X$ under observable event $e \in E_o$ is given by

$$\text{Next}_e(S) := \{x \in X : \exists u \in S \text{ s.t. } x = f(u, e)\}. \quad (4)$$

III. Opacity and Problem Formulation

In this section, we formally state the Maximally Permissive Opacity Enforcement Problem (MPOEP) that we solve in this paper. First, we recall the definition of opacity.

As was mentioned in the introduction, we consider *current-state opacity* in our development. In this setting, the state space of the system is partitioned into two disjoint sets: $X = X_S \dot\cup X_{US}$, where X_S is the set of secret states and X_{US} is the set of non-secret states. Hereafter, we set $X_{US} = X \setminus X_S$, for the sake of simplicity. The notion of current-state opacity says that for any string that leads

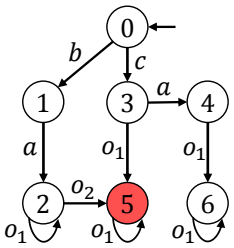
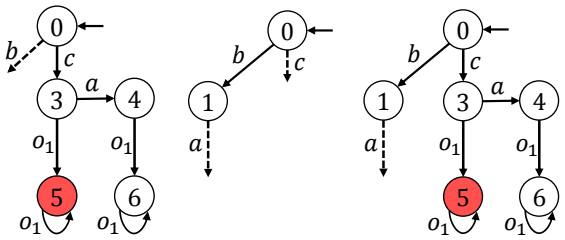


Fig. 1. System G with $E_c = \{a, b, c\}$, $E_o = \{o_1, o_2\}$, and $X_S = \{5\}$



(a) Solution G_1 (b) Solution G_2 (c) G_3: the union of $\mathcal{L}(G_1)$ and $\mathcal{L}(G_2)$

Fig. 2. An example of two incomparable solutions

to a secret state, there must exist a string that leads to a non-secret state and such that the intruder cannot distinguish between these two strings. The formal definition of current-state opacity is as follows (cf. [5], [12], [13]).

Definition 1: (Current-State Opacity). A system $G = (X, E, f, X_0)$ is said to be current-state opaque w.r.t. $X_S \subseteq X$ and $E_o \subseteq E$ if

$$(\forall u \in X_0)(\forall s \in \mathcal{L}(G, u) : f(u, s) \in X_S)(\exists v \in X_0) \\ (\exists t \in \mathcal{L}(v, t))[P(s) = P(t) \wedge f(v, t) \in X \setminus X_S] \quad (5)$$

■

Remark 3.1: It was shown in [13] that several other notions of opacity, specifically language-based opacity, initial-state opacity, and initial-and-final-state opacity, can be transformed to current-state opacity in polynomial time. Therefore, the enforcement algorithm described in this paper, which is based on current-state opacity, can also be applied to these different notions of opacity. This can be done by first transforming them to current-state opacity, as described in [13], and then calling our enforcement algorithm. Without further clarification, our usage of the word "opacity" hereafter means "current-state opacity".

Example 3.1: Consider the system G in Figure 1, where the set of controllable events $E_c = \{a, b, c\}$ and the set of observable events is $E_o = \{o_1, o_2\}$; these two sets are incomparable. Clearly, G is not opaque w.r.t. the single secret state 5, since the intruder can unambiguously know that the system is in state 5 once it observes event o_2. ■

Now, we formulate the *Maximally Permissive Opacity Enforcement Problem* (MPOEP) as follows.

Definition 2: (Maximally Permissive Opacity Enforcement Problem) Given an automaton $G = (X, E, f, X_0)$ with controllable events set E_c, observable events set E_o, and secret states set $X_S \subseteq X$, synthesize a partial observation supervisor $S_P : E_o^* \to \Gamma$, such that

1) $\mathcal{L}(S_P/G)$ is opaque w.r.t. X_S and E_o; and
2) For any S'_P satisfying (1), we have that $\mathcal{L}(S_P/G) \not\subset \mathcal{L}(S'_P/G)$. ■

Since controllability and observability together provide the necessary and sufficient conditions for the existence of a partial observation supervisor, to solve MPOEP, it suffices to find a *maximal controllable, observable and opaque sublanguage* of $\mathcal{L}(G)$. The supremal controllable and opaque sublanguage is studied in [11]. It was shown in [7]–[9] that under the assumption that $E_c \subseteq E_o$, there exists a unique *supremal* solution to MPOEP. However, this is not true in general, as we show in the following example.

Example 3.2: Let us return to Example 3.1. To enforce opacity, we need to find a controllable, observable and opaque sublanguage of $\mathcal{L}(G)$. It easy to verify that solutions $\mathcal{L}(G_1)$ and $\mathcal{L}(G_2)$, shown in Figure 2(a) and Figure 2(b), respectively, are two maximal controllable, observable and opaque solutions. However, the union of these two solutions, which is shown in Figure 2(c), is not a valid solution, since the system needs to enable event a at state 1 and disable event a at state 3; but states 1 and 3 are indistinguishable in $\mathcal{L}(G_3)$. This violates the property of observability. ■

The lack of a supremal solution to MPOEP is due to the fact that the property of observability is not preserved under union. Therefore, in MPOEP, we are looking for a *locally maximal* solution, rather than a supremal solution. This explains condition (2) in the statement of MPOEP.

Remark 3.2: In general, the intruder may have a different set of observable events than the supervisor. For example, in [7], [8], the authors assume that $E_a \subseteq E_o$, where E_a is the set of events that can be observed by the intruder. In this paper, we assume that the supervisor and the intruder observe the same events, i.e., $E_a = E_o$. Our focus is on developing a new approach for the synthesis of opacity-enforcing supervisors that allows to relax the assumption that $E_c \subseteq E_o$, which is made in prior works. We leave the case where $E_a \neq E_o$ for future research.

IV. Bipartite Transition System

In this section, we define the general notion of *bipartite transition system* (BTS), which was originally investigated in [18] to solve the standard supervisor control problem for safety and non-blockingness.

Since we are dealing with partially-observed systems, we define the notion of an information state (abbreviated as IS) as a subset $IS \subseteq X$ of states and denote by $I = 2^X$ the set of all information states. We will use specific types of BTS to capture, in a single structure, the "game" between the controller(s) and the system/environment.

Definition 3: ([18]) A bipartite transition system T w.r.t. G is a 7-tuple[1]

$$T = (Q_Y^T, Q_Z^T, h_{YZ}^T, h_{ZY}^T, E, \Gamma, y_0^T) \quad (6)$$

[1]The superscript refers to T and does not mean transposed.

where

- $Q_Y^T \subseteq I$ is the set of Y-states;
- $Q_Z^T \subseteq I \times \Gamma$ is the set of Z-states and $I(z)$ and $\Gamma(z)$ denote, respectively, the information state and the control decision components of a Z-state z, so that $z = (I(z), \Gamma(z))$;
- $h_{YZ}^T : Q_Y^T \times \Gamma \to Q_Z^T$ is the partial transition function from Y-states to Z-states, which satisfies the following constraint: $h_{YZ}^T(y, \gamma) = z$ only if
  - $I(z) = \text{UR}_\gamma(y)$ and $\Gamma(z) = \gamma$
- $h_{ZY}^T : Q_Z^T \times E_o \to Q_Y^T$ is the partial transition function from Z-states to Y-states, which satisfies the following constraint: $h_{ZY}^T(z, e) = y$ only if
  - $e \in \Gamma(z) \cap E_o$ and $y = \text{Next}_e(I(z))$
- E is the set of events of G;
- Γ is the set of admissible control decisions of G;
- $y_0^T \in Q_Y^T$ is the initial Y-state where $y_0^T = X_0$. ■

The purpose of defining the general notion of BTS is to describe the "game" between the supervisor and the environment (system G). To capture this game, we need a bipartite structure, with two types of nodes (states). A Y-state is an information state, from which the supervisor issues control decisions. A Z-state is an information state augmented with control decisions, from which the system "selects" observable events to occur within the set of enabled events. A transition from a Z-state to a Y-state represents the observable transition, i.e, y in the above definition is the set of states reachable from some state of the information state component of the preceding Z-state through the single observable event. A transition from a Y-state to a Z-state represents the unobservable reach and "remembers" the set of enabled events from the Y-state that leads to it. This means that $I(z)$ is the set of states reachable from some state in the preceding Y-state through some enabled unobservable event strings, and that $\Gamma(z)$ is the control decision made in the preceding Y-state.

Example 4.1: Consider again the system G in Figure 1. As an example of a BTS, the reader is referred directly to Figure 3(b), which is a particular type of BTS that we will discuss later in this paper. For the initial Y-state $y_0 = \{0\}$, by making control decision $\gamma = \{a, c, o_1, o_2\}$ (the uncontrollable events o_1 and o_2 are omitted in the figure), we will reach Z-state $z = h_{YZ}^T(y_0, \gamma) = (\{0, 3, 4\}, \{a, c, o_1, o_2\})$. From z, only one observable event, o_1, can happen, and it leads to the next Y-state $y_1 = h_{ZY}^T(z, o_1) = \{5, 6\}$. ■

In general, the control decision defined at a Y-state may not be unique. Therefore, given a BTS T, we define $C_T(y) := \{\gamma \in \Gamma : h_{YZ}^T(y, \gamma)!\}$, to be the set of control decisions defined at $y \in Q_Y^T$. Since for any two BTS T_1 and T_2, $h_{YZ}^{T_1}(y, \gamma) = h_{YZ}^{T_2}(y, \gamma)$ whenever they are defined, we will drop the superscript in $h_{YZ}^T(y, \gamma)$ and write it as $h_{YZ}(y, \gamma)$ if it is defined for some T; the same holds for h_{ZY}.

Definition 4: ([18]) Given a supervisor S_P, $IS_{S_P}^Y(y, s)$ is defined to be the Y-state that results from the occurrence of string s, when starting in Y-state y. This can be computed recursively as follows:

$$IS_{S_P}^Y(y, \epsilon) := y$$

$$IS_{S_P}^Y(y, s\sigma) := \begin{cases} h_{ZY}(h_{YZ}(IS_{S_P}^Y(y, s), S_P(s)), \sigma), & \text{if } \sigma \in E_o \cap S_P(s) \\ IS_{S_P}^Y(y, s), & \text{if } \sigma \in E_{uo} \cap S_P(s) \\ \text{undefined}, & \text{otherwise} \end{cases}$$

For brevity, we write $IS_{S_P}^Y(y_0, s)$ as $IS_{S_P}^Y(s)$.

Also, $IS_{S_P}^Z(z, s)$ is defined analogously, with $IS_{S_P}^Z(s) := IS_{S_P}^Z(z_0, s)$, where $z_0 = h_{YZ}(y_0, S_P(\epsilon))$. ■

Now, given a BTS T, it is possible to "decode" supervisors from it, as explained in the following definition.

Definition 5: A supervisor S_P is said to be included in the BTS T if

$$(\forall s \in \mathcal{L}(S_P/G))[S_P(s) \in C_T(IS_{S_P}^Y(s))] \tag{7}$$

$\mathcal{S}(T)$ denotes the set of all supervisors included in T. ■

By the above definition, if a BTS T includes some supervisor, then it should satisfy the following two properties:

1) For any $y \in Q_T^Y$, we have $C_T(y) \neq \emptyset$; and
2) For any $z \in Q_T^Z$, we have $\forall e \in \Gamma(z) \cap E_o : (\exists x \in I(z) : f(x, e)!) \Rightarrow h_{ZY}^T(z, e)!$.

The first property simply says that for any Y-state, we need to be able to pick at least one control decision. The second property says that for any Z-state, we cannot block any enabled and feasible observable event. This is because we cannot choose which event will occur once we have made a control decision; the system will decide. These two properties together are also referred to as the *completeness* property of the BTS.

Example 4.2: The BTS shown in Figure 3(b) is a complete BTS. By picking control decision $\{o_1, o_2\}$ (shown as $\{\}$ in the figure) at the initial Y-state $\{0\}$, no future behavior can occur. This leads to a BTS-included supervisor S_P defined by $S_P(\epsilon) = \{o_1, o_2\}$. ■

Remark 4.1: If a BTS T is complete and for any Y-state $y \in Q_Y^T$, we have that $|C_T(y)| = 1$, then it is clear that the set of supervisors included in T is a singleton, since for each information state, the control decision is unique. In this case, we denote the unique supervisor included in T as S_T, i.e., $\mathcal{S}(T) = \{S_T\}$.

The next result states that given a supervisor S_P, the Z-state defined above is, in fact, equivalent to the set of all possible states the system can be in at that point.

Lemma 4.1: Given a system G and a supervisor S_P, for any string $s \in \mathcal{L}(S_P/G)$, we have $I(IS_{S_P}^Z(s)) = R_{S_P/G}(s)$.

V. All Inclusive Controller for Opacity

In this section, we define the All Inclusive Controller for Opacity, a specific type of BTS that embeds all supervisors that enforce opacity in its transition structure.

To begin with, we define the "opacity binary function" that evaluates the opacity property for each information state. An information state $i \in I$ violates current-state opacity if it is a subset of the set of secret states X_S.

Definition 6: (Opacity binary function for information state). The opacity binary function for information states is the function $OP : I \to \{0,1\}$ where:

$$OP(i) = \begin{cases} 1, & \text{if } i \not\subseteq X_S \\ 0, & \text{if } i \subseteq X_S \end{cases} \tag{8}$$

■

Thus, $OP(i) = 1$ if i does not violate current-state opacity. The following result says that the opacity binary function can correctly evaluate the opacity of a system.

Lemma 5.1: A system G is current-state opaque if and only if $(\forall s \in \mathcal{L}(G))[OP(R_G(s)) = 1]$.

Recall that in Lemma 4.1, we have shown that given a supervisor S_P, for any string $s \in \mathcal{L}(G)$, the Z-state $IS^Z_{S_P}(s)$ encountered is set of all possible states the system could be in after s. Consequently, if we construct a BTS that is "as large as possible" and in which all reachable Z-states satisfy the opacity binary function, the resulting structure should contain all valid opacity-enforcing supervisors. This leads to the definition of the All Inclusive Controller for Opacity.

Definition 7: (All Inclusive Controller for Opacity). Given a system G and a set of secret states $X_S \subseteq X$, the All Inclusive Controller for Opacity (AIC-O), denoted by $\mathcal{AICO}(G) = (Q^{AIC}_Y, Q^{AICO}_Z, h^{AICO}_{YZ}, h^{AICO}_{ZY}, E, \Gamma, y^{AICO}_0)$, is defined as the largest BTS such that

1 For any $y \in Q^{AICO}_Y$, we have $|C_{\mathcal{AICO}(G)}(y)| \geq 1$; and
2 For any $z \in Q^{AICO}_Y$, we have
  2.1 $\forall e \in \Gamma(z) \cap E_o : (\exists x \in I(z) : f(x,e)!) \Rightarrow h^{AICO}_{ZY}(z,e)!$;
  2.2 $OP(I(z)) = 1$. ■

Note that if T_1 and T_2 are two BTSs that satisfy the above conditions, then it is easy to see that the union of them will still satisfy these conditions. Therefore, the notion of "largest BTS" in the definition is well defined. This will also be seen when we present the algorithm for the construction of the AIC-O later.

Remark 5.1: In the definition of the AIC-O, opacity is only evaluated at Z-states. This follows from the definition of current-state opacity, Lemma 4.1, and 5.1. However, if the intruder can act much faster than the system's behavior, then even though the system is opaque, the intruder may still be able to know that the secret has occurred *immediately after* observing some observable events. In this case, instead of evaluating opacity on Z-states, we need to require that for any $y \in Q^{AICO}_Y$, we have $OP(y) = 1$. This will lead to a stronger notion of opacity than that we consider in this paper; such an analysis is beyond our scope here, but it can be performed in a straightforward manner using our approach.

Example 5.1: We return to system G in Figure 1. The BTS shown in Figure 3(b) is, in fact, its AIC-O. For example, at initial Y-state $\{0\}$, we cannot make control decision $\{a,b,c\}$, which would lead us to Z-state $(\{0,1,2,3,4\},\{a,b,c\})$. This is because upon the occurrence of event o_2, Y-state $\{5\}$ would be reached, from which no matter what control decision we take, the secret will be revealed. ■

Remark 5.2: In Figure 3(b), we can also take control decision $\{a\}$ at the initial Y-state $y_0 = \{0\}$. However, this control decision is equivalent to decision $\{\}$, since event a will never be executed within the unobservable reach. Formally, we say that a control decision $\gamma \in \Gamma$ is *irredundant* at information state $i \in I$ if, for any $\sigma \in \gamma$, there exists $x \in UR_\gamma(i)$ such that $f(x,\sigma)$ is defined. From now on, we only consider irredundant control decisions in the AIC-O, which will clearly not affect its properties.

The following theorem shows that the AIC-O (only) contains valid solutions to the opacity enforcement problem.

Theorem 1: A supervisor enforces opacity if and only if it is an AIC-O included supervisor:

$$S_P \in \mathcal{S}(\mathcal{AICO}(G)) \Leftrightarrow S_P/G \text{ is opaque} \tag{9}$$

A. Construction of the AIC-O

Algorithm 1: FIND-AIC-O

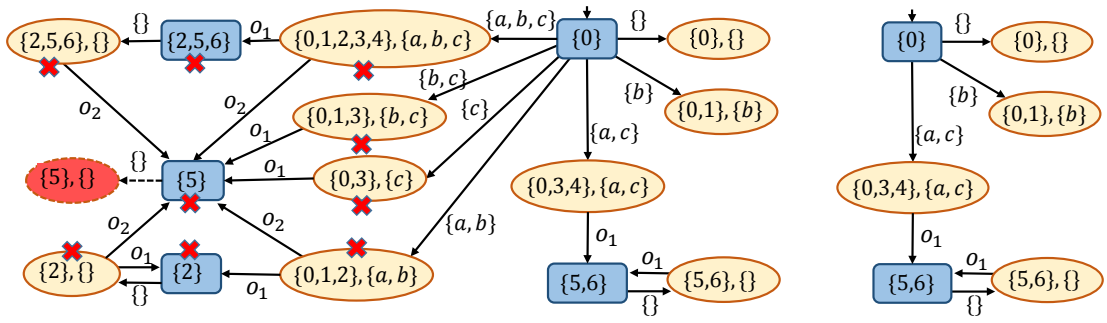
input : G and OP
output: $AICO$

```
1  AICO.Y ← {y0}, AICO.Z ← ∅ and AICO.h ← ∅;
2  DoDFS(G, y0, AICO);
3  Prune(AICO);
4  AICO ← Accessible(AICO);

   procedure DoDFS(G, y, AICO, OP);
5    for γ ∈ Γ do
6      z ← h_YZ(y, γ);
7      if OP(I(z)) = 1 then
8        AICO.h ← AICO.h ∪ {(y, γ, z)};
9        if z ∉ AICO.Z then
10         AICO.Z ← AICO.Z ∪ {z};
11         for e ∈ γ ∩ E_o do
12           y' ← h_ZY(z, e);
13           AICO.h ← AICO.h ∪ {(z, e, y')};
14           if y' ∉ AICO.Y then
15             AICO.Y ← AICO.Y ∪ {y'};
16             DoDFS(G, y', AICO, OP);

   procedure Prune(AICO);
17   while exists Y-state in AICO that has no successor do
18     Delete all such Y-states in AICO and delete all
       their predecessor Z-states;
```

The construction algorithm for the AIC-O follows directly from its definition and proceeds in two steps. First, we construct the BTS that enumerates all possible behaviors by a depth-first search and remove all Z-states that violate the opacity binary function, i.e., condition 2.1 in Def. 7. Second, we prune states that violate conditions 1 or 2.2 in Def. 7 from the remaining part of the BTS, until convergence is achieved. In practice, in the depth-first search part, we do not need to search the whole state space and we can stop the search



(a) The resulting structure after procedure DoDSF (b) The constructed AIC-O

Fig. 3. Example of the construction of the AIC-O. In the diagrams, blue rectangular states correspond to Y-states and yellow oval states correspond to Z-states. For simplicity, in the diagrams, we omit all uncontrollable events in the control decisions, e.g., decision $\{\}$ represents $\{o_1, o_2\}$, and so forth.

of a branch once a Z-state that violates the opacity binary function is encountered.

The above procedure is formally described in Algorithm FIND-AIC-O whose parameters are as follows: (i) AICO represents the AIC-O that we want to construct; (ii) AICO.Y and AICO.Z are its sets of Y and Z-states, respectively; and (iii) AICO.h is its transition function. Initially, AICO.Y is set to be $y_0 = X_0$. The depth-first search is then started; , it is implemented by the procedure DoDFS. Line 7 is used to determine whether the Z-state encountered satisfies opacity. If not, we terminate the search of this branch. Otherwise, we compute all possible Y-state successors and make a recursive call. This recursive procedure allows us to traverse the whole reachable space of Y and Z-states. The above procedure may result in Y-states that have no successors. Therefore, we need to iteratively prune: (i) all Y-states that have no successor states; and (ii) all Z-states for which one or more observation is not defined. This step is captured by procedure Prune. Finally, states that are no longer accessible from the initial state of the AIC-O need to be removed before the algorithm returns. Algorithm FIND-AIC-O will terminate in finite steps, since the number of Y and Z-states is finite.

Example 5.2: We apply Algorithm FIND-AIC-O to construct the AIC-O for the system G shown in Figure 1. The resulting BTS after running the procedure DoDSF is shown in Figure 3(a). The depth-first search DoDSF terminates at Y-state $\{5\}$, since no matter what control decision we take from $\{5\}$, a Z-state (marked by red in Figure 3(a)) that reveals the secret will be encountered. After procedure DoDSF is done, we need to run procedure Prune. This starts by removing Y-state $\{5\}$, since no successor state is defined from it. Since Y-state $\{5\}$ has been removed, all its predecessor Z-states, i.e., $(\{0,3\},\{c\}), (\{0,1,2\},\{a,b\})$, et al., should also be removed. Finally, we remove inaccessible states $\{2,5,6\}$ and $\{2\}$ and obtain the AIC-O shown in Figure 3(b). ■

Theorem 2: Algorithm FIND-AIC-O correctly constructs the AIC-O.

VI. Synthesis of Maximally Permissive Supervisors

In this section, we present a synthesis algorithm that returns a solution to MPOEP.

Theorem 1 provides us with a straightforward procedure for synthesizing an opacity-enforcing supervisor. We can simply start from the initial Y-state and pick *one* control decision defined in the AIC-O; then we pick *all* observations for the successor Z-state, and so forth, until reaching a Z-state that has no successor state. However, this procedure may result in a solution with infinite domain, since we may take different control decisions for different times we visit the same information state. Therefore, we wish to consider a particular type of solution, called an *information-state-based* (IS-based) solution, that can be realized with finite memory. Formally, a supervisor S_P is IS-based if

$$(\forall s,t \in \mathcal{L}(S_P/G))[IS^Y_{S_P}(s) = IS^Y_{S_P}(t) \Rightarrow S_P(s) = S_P(t)]$$

Clearly, if a supervisor is IS-based, then we can redefine it in the form of $S_P : I \to \Gamma$.

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

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

Theorem 3: Let S^* be a solution returned by Algorithm MAX-SYNT. Then S^* solves MPOEP.

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

Corollary 6.1: MPOEP is solvable if and only if the AIC-O is non-empty.

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

Algorithm 2: MAX-SYNT

input : $\mathcal{AICO}(G)$
output: S^*

$T.Y \leftarrow \{y_0\}, T.Z \leftarrow \emptyset$ and $T.h \leftarrow \emptyset$;
Expand($T, \mathcal{AICO}(G), y_0$);
$S^* \leftarrow S_T$;

procedure Expand($T, \mathcal{AICO}(G), y$);
  Find a locally maximal control decision $\gamma \in C_{\mathcal{AICO}(G)}(y)$ such that $\forall \gamma' \in C_{\mathcal{AICO}(G)}(y) : \gamma \not\subset \gamma'$;
  $z \leftarrow h_{YZ}(y, \gamma)$;
  $T.h \leftarrow T.h \cup \{(y, \gamma, z)\}$;
  if $z \notin T.Z$ **then**
    $T.Z \leftarrow T.Z \cup \{z\}$;
    for $e \in \gamma \cap E_o$ **do**
      $y' \leftarrow h_{ZY}(z, e)$;
      $T.h \leftarrow T.h \cup \{(z, e, y')\}$;
      if $y' \notin T.Y$ **then**
        $T.Y \leftarrow T.Y \cup \{y'\}$;
        Expand($T, \mathcal{AICO}(G), y'$);

Corollary 6.2: If the AIC-O is non-empty, then there always exists an IS-based supervisor that solves MPOEP.

Example 6.1: We return to our running example. If we pick locally maximal control decision $\{a, c\}$ at the initial Y-state $\{y_0\}$ and pick the unique control decision $\emptyset$ at the reachable Y-state, which means disable all controllable events, we will obtain the maximal solution that was shown earlier in Figure 2(a). On the other hand, if we pick control decision $\{b\}$ at $\{0\}$, which is another locally maximal decision, then no behavior can occur hereafter; this corresponds to the maximal solution shown in Figure 2(b). ■

VII. Conclusion

We presented a novel approach to the problem of synthesizing a maximally permissive supervisor that enforces opacity for a partially-observed discrete-event system that is not originally opaque. To this end, we defined a novel information structure called the All inclusive Controller for Opacity that embeds all valid supervisors to this problem. Based on the AIC-O, a synthesis algorithm was provided to find a locally maximal solution to this problem, without making any assumptions about the observability properties of the controllable events. In this regard, our approach relaxes the assumption that all controllable events are observable in the previous works on opacity enforcement by supervisory control. In addition, we believe that the AIC-O can be used for solving optimal opacity enforcement control problems where a cost structure is imposed on this problem. Since the AIC-O embeds all valid opacity enforcing supervisors, it provides a suitable solution space over which to solve an optimal control problem for opacity.

Among the many possible directions for future work that would build on the AIC-O, we mention two problems of immediate interest. One is relaxing the assumption made in this paper that the supervisor and the intruder have the same set of observable events. Another one is to consider the synthesis of a maximal solution that provably contains a given particular solution.

References

[1] L. Mazaré, "Using unification for opacity properties," in *Proceedings of WITS*, vol. 4, 2004, pp. 165–176.

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

[3] ——, "Verification of initial-state opacity in security applications of des," in *9th International Workshop on Discrete Event Systems*, 2008, pp. 328–333.

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

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

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

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

[8] M. Ben-Kalefa and F. Lin, "Supervisory control for opacity of discrete event systems," in *49th IEEE Annual Allerton Conference on Communication, Control, and Computing*, 2011, pp. 1113–1119.

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

[10] P. Darondeau, H. Marchand, and L. Ricker, "Enforcing opacity of regular predicates on modal transition systems," *Discrete Event Dynamic Systems: Theory & Applications*, 2014.

[11] S. Takai and Y. Oka, "A formula for the supremal controllable and opaque sublanguage arising in supervisory control," *SICE J. Control, Measurement, and System Integration*, vol. 1, pp. 307–311, 2011.

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

[13] Y.-C. Wu and S. Lafortune, "Comparative analysis of related notions of opacity in centralized and coordinated architectures," *Discrete Event Dyn. Sys.: Theory & Applications*, vol. 23, no. 3, pp. 307–339, 2013.

[14] Y. Falcone, H. Marchand *et al.*, "Enforcement and validation (at runtime) of various notions of opacity," *Discrete Event Dynamic Systems: Theory & Applications*, 2014.

[15] Y.-C. Wu and S. Lafortune, "Synthesis of insertion functions for enforcement of opacity security properties," *Automatica*, vol. 50, no. 5, pp. 1336–1348, 2014.

[16] S. Takai and R. Kumar, "Verification and synthesis for secrecy in discrete-event systems," in *American Control Conference*, 2009, pp. 4741–4746.

[17] X. Yin and S. Lafortune, "A general approach for synthesis of supervisors for partially-observed discrete-event systems," in *19th IFAC World Congress*, 2014, pp. 2422–2428.

[18] ——, "Synthesis of maximally permissive non-blocking supervisors for partially observed discrete event systems," in *53rd IEEE Conference on Decision and Control*, 2014, pp. 5156–5162.

[19] P. Ramadge and W. Wonham, "Supervisory control of a class of discrete event processes," *SIAM J. Control Optim.*, vol. 25, no. 1, pp. 206–230, 1987.

[20] X. Yin and S. Lafortune, "A new approach for synthesizing opacity-enforcing supervisors for partially-observed discrete-event systems," *University of Michigan, Tech. Rep.*, 2014.

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

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

[23] R. Cieslak, C. Desclaux, A. Fawaz, and P. Varaiya, "Supervisory control of discrete-event processes with partial observations," *IEEE Trans. Autom. Control*, vol. 33, no. 3, pp. 249–260, 1988.