IEEE Control Systems Letters paper presented at
2023 62nd IEEE Conference on Decision and Control (CDC)
December 13-15, 2023. Marina Bay Sands, Singapore

Edit Mechanism Synthesis for Opacity Enforcement Under Uncertain Observations

Wei Duan, Ruotian Liu, Maria Pia Fanti, *Fellow, IEEE*, Christoforos N. Hadjicostis, *Fellow, IEEE*, and Zhiwu Li, *Fellow, IEEE*

***Abstract*— This paper addresses the problem of opacity enforcement by using edit functions in discrete event systems modeled as deterministic finite automata under partial observation. The edit function is an output interface of the system that manipulates actual observations to confuse a malicious intruder. We assume that the edit function simply knows whether the intruder observes a larger or smaller set of events than itself, but does not know the exact set of events observed by the intruder. In this *uncertain observation setting*, the edit function aims to confuse the intruder while relying on its own set of observable events, which requires the edit function to be u-*enforcing*. The opacity enforcement problem is then transformed to a two-player game between the system and the edit function under partial information. A so-called *edit mechanism* is proposed in a game scheme to enumerate all possible edited operations following the system behavior. We show that an edit function synthesized from the edit mechanism (if any) can be used to enforce opacity in the system under the uncertain observation setting.**

***Index Terms*— Discrete event systems, Automata, Game theory.**

I. Introduction

Opacity is a confidentiality property that has become an active research topic in discrete event systems (DESs) [1]–[4]. This property captures whether an intruder can infer a secret state of a given system based on its observation of the system behavior and its knowledge of the system's structure. Precisely, a system is said to be opaque if, for any secret behavior, there exists at least another non-secret behavior that appears identical to the intruder; this implies that the intruder can never infer the system's secrets with certainty.

When a system is not opaque, the opacity enforcement problem arises and is extensively addressed via two main approaches. The use of supervisory control theory was first proposed based on the construction of minimally restrictive opacity-enforcing supervisory controllers [5]–[8]. Then, obfuscation mechanisms were used to manipulate observations generated by the system via insertion and edit functions so as to confuse the intruder [9], [10].

We point out that all aforementioned enforcement approaches are carried out under the premise that either (i) the enforcer has the same observation capability as the intruder, i.e., the controllers, the insertion functions, or the edit functions have the same set of observable events as the intruder [6], [9], [10], or (ii) the enforcer knows the exact observations seen by the intruder, i.e., the controllers have knowledge of what the intruder can observe even though they may have different sets of observable events [5], [7], [8]. Such requirements are restrictive since, in reality, the enforcer may have different partial observations of the system compared to the intruder, and may not be aware of the exact observation capability of the intruder.

Inspired from the above considerations, this work addresses the problem of opacity enforcement via edit functions in DESs under a more general scenario, in which the set of observations of the system captured by the intruder, denoted by E_I, is different from that captured by the edit function, denoted by E_D. Moreover, the edit function is assumed to have partial knowledge of the intruder, i.e., it knows the relationship of the observation capability between the intruder and itself, but it is not aware of the exact observations seen by the intruder. Specifically, we consider two cases about the relationship between E_I and E_D: i) $E_I \subseteq E_D$; ii) $E_D \subseteq E_I$. We refer to this general setting as the *uncertain observation* setting.

This paper proposes the notion of u-enforceability to characterize whether the edit function has the ability to enforce opacity of the system under the uncertain observation setting. To this end, we build an edit mechanism within a two-player game between the system and the edit function, so as to enumerate all edited operations associated with corresponding system behavior observed by the edit function. We show that the edit function is u-enforcing if it can be synthesized from the edit mechanism. To the best of our knowledge, this is the first investigation of the opacity enforcement problem in DESs when the enforcer is not aware of the exact observations of the intruder.

II. Preliminaries

Let E be a finite set of events and E^* denote the set of all finite length strings (finite sequences of events) over E, including the empty string ε. The length of a string λ is the number of events in λ, denoted by $|\lambda|$, and the length of the empty string is denoted by $|\varepsilon| = 0$. A language $L \subseteq E^*$ is a subset of finite-length strings. Given strings u and v, uv stands for the concatenation of u and v.

*This work was supported in part by the National R&D Program of China under Grant No. 2018YFB1700104, and the National Natural Science Foundation of China under Grants 61603285 and 61873342.

Wei Duan and Zhiwu Li are with Xidian University, Xi'an, China (e-mails: dwei1024@126.com, zhwli@xidian.edu.cn).

Ruotian Liu and Maria Pia Fanti are with Polytechnic University of Bari, Bari, Italy (emails: {ruotian.liu, mariapia.fanti}@poliba.it).

Christoforos N. Hadjicostis is with University of Cyprus, Nicosia, Cyprus (e-mail: hadjicostis.christoforos@ucy.ac.cy).

In this paper, we model a DES with a deterministic[1] finite automaton (DFA) $G = (X, E, f, x_0)$, where X is a set of states, E is a set of events, $f : X \times E \to X$ is the partial transition function, and $x_0 \in X$ is the initial state. The transition function f can be extended to $X \times E^* \to X$ in the usual manner: $f(x, \alpha\lambda) = f(f(x, \alpha), \lambda)$ for $x \in X$, $\alpha \in E$, and $\lambda \in E^*$. Note that $f(x, \varepsilon) = x$ and $f(x, \alpha\lambda)$ is undefined if $f(x, \alpha)$ is undefined. The generated language of G, denoted by $L(G)$, is defined as $L(G) = \{\lambda \in E^* \mid f(x_0, \lambda) \text{ is defined}\}$.

The event set E in a DFA is partitioned into the set of observable events E_o and the set of unobservable events $E_{uo} = E \setminus E_o$ due to the partial observation of DFA. For a sequence $\lambda \in E^*$, the natural projection with respect to E_o is defined as $P : E^* \to E_o^*$, where $P(\varepsilon) = \varepsilon$; $P(\alpha) = \alpha$ if $\alpha \in E_o$; $P(\alpha) = \varepsilon$ if $\alpha \in E_{uo}$. Moreover, $P(\lambda\alpha) = P(\lambda)P(\alpha)$ where $\lambda \in E^*$ and $\alpha \in E$.

Given a system $G = (X, E, f, x_0)$ and a sequence $\sigma \in E_o^*$, the set of possible states with respect to P starting from a subset of states $Q \subseteq X$ is defined as $R_o(Q, \sigma) = \{x' \in X \mid \exists x \in Q, \exists \lambda \in E^* : P(\lambda) = \sigma, f(x, \lambda) = x'\}$. One can build the system observer to estimate the current states of the system under E_o. The observer is defined as $\mathbf{O}_o = (X_o, E_o, f_o, x_o^0)$, where $X_o \subseteq 2^X$ is the state space, E_o is the set of observable events, $x_o^0 = R_o(x_0, \varepsilon)$ is the set of initial states, and $f_o : X_o \times E_o \to X_o$ is the transition function defined for $x_o \in X_o$ and $\alpha \in E_o$ as $f_o(x_o, \alpha) = R_o(x_o, \alpha)$ ($f_o(x_o, \alpha)$ is taken to be undefined if $R_o(x_o, \alpha)$ is empty).

III. Current-State Opacity Enforcement via Edit Functions under Uncertain Observations

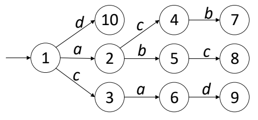
A. Uncertain Observation Setting

Given a system that is not opaque, its opacity can be enforced by implementing an edit mechanism via the use of edit functions [10]. The edit function interfaces at the output of the system by manipulating observations generated by the system with insertion, replacement, and deletion operations.

We assume that the intruder and the edit function have full knowledge of the system's structure but different observations, i.e., the intruder can observe $E_I \subseteq E_o$ whereas the edit function can observe $E_D \subseteq E_o$. Moreover, the edit function does not know the exact set of observable events of the intruder, but knows whether $E_I \subseteq E_D$ or $E_D \subseteq E_I$. The following example is given to illustrate our motivations.

Example 1: Consider the system $G = (X, E, f, x_0)$ in Fig. 1, where the set of observable events is $E_o = \{b, c, d\}$. Assume that $E_I = \{b, c, d\}$ and $E_D = \{b, d\}$; then, the estimate of the system states by the intruder is not necessarily identical to that of the edit function. For example, the intruder can infer the system is in state $\{8\}$ if the system generates sequence abc, whereas the edit function cannot infer this since it can only observe b. Meanwhile, the edit function cannot distinguish the exact inferred states by the intruder in the set of states $\{5, 7, 8\}$. ◇

[1]We focus on the deterministic case to make notations simple, but the results in this paper can be extended to the nondeterministic case in a straightforward manner.



$$\begin{tikzcd}
 & 10 & 4 \arrow[r, "b"] & 7 \\
\arrow[r] & 1 \arrow[u, "d"] \arrow[r, "a"] \arrow[dr, "c"] & 2 \arrow[u, "c"] \arrow[r, "b"] & 5 \arrow[r, "c"] & 8 \\
 & & 3 \arrow[r, "a"] & 6 \arrow[r, "d"] & 9
\end{tikzcd}$$

Fig. 1. System G used in the running example.

Definition 3.1: Given a system G with E_o, E_I, and E_D, the resilient hypothetical set of events E_{RH} captures the set of possible events observed by the intruder from the edit function's point of view. This is defined as

$$E_{RH} = \begin{cases} E_o, & \text{if} \quad E_D \subseteq E_I, \\ E_D, & \text{if} \quad E_I \subseteq E_D. \end{cases}$$

To capture the events observed by the intruder, the edit function needs to deduce the states of the system as captured by the intruder in the worst scenario: in the first case ($E_D \subseteq E_I$), the edit function assumes that the intruder may capture all observable events of the system, i.e., $E_{RH} = E_o$; in the second case ($E_I \subseteq E_D$), the edit function supposes that the intruder has the ability to observe the same events as itself, i.e., $E_{RH} = E_D$.

The resilient hypothetical projection captures observations by the intruder from the edit function's point of view by following sequences generated by the system; it is defined as $P_{RH} : E^* \to E_{RH}^*$ where $P_{RH}(\varepsilon) = \varepsilon$; $P_{RH}(\beta) = \beta$ if $\beta \in E_{RH}$; $P_{RH}(\beta) = \varepsilon$ if $\beta \in E \setminus E_{RH}$, and $P_{RH}(\lambda\alpha) = P_{RH}(\lambda)P_{RH}(\alpha)$ for all $\lambda \in E^*, \alpha \in E$.

Next, the defender projection is introduced to capture observations by the edit function from sequences of observations received by the presumed intruder, which is defined as $P_D : E_{RH}^* \to E_D^*$ where $P_D(\varepsilon) = \varepsilon$; $P_D(\gamma) = \gamma$ if $\gamma \in E_D$; $P_D(\gamma) = \varepsilon$ if $\gamma \in E_{RH} \setminus E_D$, and $P_D(\sigma\beta) = P_D(\sigma)P_D(\beta)$ for all $\sigma \in E_{RH}^*, \beta \in E_{RH}$.

B. Current-State Opacity

By taking into account the set of events observed by the intruder, i.e., $E_I \subseteq E_o$, the intruder projection $P_I : E^* \to E_I^*$ is defined as $P_I(\varepsilon) = \varepsilon$, $P_I(\alpha) = \alpha$ if $\alpha \in E_I$, $P_I(\alpha) = \varepsilon$ if $\alpha \in E \setminus E_I$, and $P_I(\lambda\alpha) = P_I(\lambda)P_I(\alpha)$ for all $\lambda \in E^*, \alpha \in E$. For simplicity, we only focus on the notion of current-state opacity (CSO) in the remainder of the paper.

Definition 3.2: Given a system $G = (X, E, f, x_0)$, the intruder projection P_I and the set of secret states $X_S \subseteq X$, G is said to be CSO with respect to P_I if $\forall t \in L_S := \{t \in L(G) : f(x_0, t) \in X_S\}, \exists t' \in L_{NS} := \{t' \in L(G) : f(x_0, t') \in (X \setminus X_S)\}$ such that $P_I(t) = P_I(t')$.

By Definition 3.2, CSO with respect to P_I requires that the evolution of the system to states in X_S is kept uncertain to the intruder (under the intruder projection map P_I), at least until the current state of the system leaves X_S.

Definition 3.3: Given a system G, the resilient hypothetical projection P_{RH} and the set of secret states $X_S \subseteq X$, G is said to be CSO with respect to P_{RH} if $\forall t \in L_S, \exists t' \in L_{NS}$ such that $P_{RH}(t) = P_{RH}(t')$.

Proposition 3.4: Given a system G, if CSO with respect to P_{RH} holds, then CSO with respect to P_I holds.

Proof: If G is CSO with respect to P_{RH}, it holds for all $t \in L_S$, there exists $t' \in L_{NS}$ such that $P_{RH}(t) = P_{RH}(t')$. By the definition of E_{RH}, we know $E_I \subseteq E_{RH}$ and $E_I^* \subseteq E_{RH}^*$, which implies $P_I(t) = P_I(t')$. Thus, G is CSO with respect to P_I. ■

We use CSO to denote CSO with respect to P_{RH} for short. In the remainder of this paper, we mainly focus on the case $E_D \subseteq E_I$ since the other one ($E_D \subseteq E_I$) can be solved in a manner similar to the techniques in [9], [10].

C. Edit Function

In this paper, the difference is that the edit function may observe a smaller set of events compared to the intruder, i.e., $E_D \subseteq E_I$. With a slight modification on the edit function in [10], an event-based version is defined to make the exposition simpler as follows.

Definition 3.5: An edit function is defined by $f_e : E_{RH} \to E_{RH}^*$ such that for $\beta \in E_{RH} \setminus E_D$, we have $f_e(\beta) = \{\beta\}$; and for $\gamma \in E_D$, we have

$$f_e(\gamma) = \begin{cases} \gamma_I\gamma, \ \gamma_I \in E_D^* \text{ is inserted before } \gamma; \\ \gamma_R, \ \gamma \text{ is replaced with } \gamma_R; \\ \varepsilon, \ \gamma \text{ is erased.} \end{cases}$$

In the following, we assume that no γ_I is of unbounded length. Under the uncertain observation setting, the edit function aims to confuse the intruder based on what the intruder observes. For notational convenience, we let the domain and codomain be E_{RH} to allow the edit function to “react” to all events observed by the intruder. That is, $f_e(\beta) = \beta$ for $\beta \in E_{RH} \setminus E_D$ in the sense that the edit function cannot manipulate events that it does not observe; whereas $f_e(\gamma)$ can implement edited operations (i.e., insertions, replacements, and deletions) for $\gamma \in E_D$. The edit function can be extended to a string-based version in a recursive manner as: $f_e(\varepsilon) = \varepsilon$, $f_e(\sigma\beta) = f_e(\sigma)f_e(\beta)$ for $\sigma \in E_{RH}^*$ and $\beta \in E_{RH}$. Note that each modification of a symbol in E_D has to rely on what the edit function has observed and the edited operations that have been implemented up to that point.

D. U-Enforceability

Due to lack of observations, the edit function cannot respond to every event observed by the intruder. Hence, regardless of what the intruder observes, the specification of u-enforceability aims to guarantee that the edit function is able to modify every event observed by itself such that the intruder cannot deduce the secret states.

Definition 3.6: (U-enforceability) Consider the system G in Definition 3.2 under the projections P_D and P_{RH}. An edit function f_e is u-enforcing if

1) $\forall \sigma \in P_{RH}[L(G)], f_e(\sigma)$ is defined,
2) $\forall \sigma \in P_{RH}[L(G)], \exists \sigma' \in f_e(\sigma)$: $\sigma' \in P_{RH}(L_{NS})$,
3) $\forall \sigma, \sigma' \in P_{RH}[L(G)]$: $P_D(\sigma) = P_D(\sigma') \Rightarrow \{f_e(\sigma), f_e(\sigma')$ are defined such that $P_D(f_e(\sigma)) = P_D(f_e(\sigma'))\}$,
4) $\forall \sigma\beta \in P_{RH}[L(G)]$, $f_e(\sigma\beta) = f_e(\sigma)f_e(\beta)$ is defined such that conditions (1)–(3) hold for σ and $\sigma\beta$.

The first condition requires the edit function to react to each event observed by the intruder (but no modification is allowed if the event is not observed by the edit function). In order to retain CSO of the system, the second condition ensures that the edit function can choose an edited sequence that keeps the secret from being revealed to the intruder. The third condition requires that the edit function should implement the same edited operations to all sequences that have the same defender projection (the edit function cannot react differently since its observations are identical). Finally, the fourth condition is needed to maintain consistency of the edited sequences, i.e., each subsequent edited sequence maintains the previous conditions (1)–(3).

IV. Synthesis of an Edit Mechanism

A. System Estimate by Observers

Since the intruder is assumed to observe the set of events E_{RH} from the perspective of the edit function, one can construct the resilient hypothetical observer in terms of E_{RH} and P_{RH}, where $E_{RH} = E_o$ and $P_{RH} = P$ in the case that $E_D \subseteq E_I$. In this regard, the resilient hypothetical observer is identical to the system observer $\mathbf{O}_o$, and is denoted by $\mathbf{O}_{RH} = (X_{RH}, E_{RH}, f_{RH}, x_{RH}^0)$, where $X_{RH} = X_o$, $E_{RH} = E_o$, $f_{RH} = f_o$, and $x_{RH}^0 = x_o^0$.

Under the uncertain observation setting, the edit function aims to deduce the system estimates possibly realized at the intruder. To do so, based on the resilient hypothetical observer $\mathbf{O}_{RH} = (X_{RH}, E_{RH}, f_{RH}, x_{RH}^0)$ and a sequence $\omega \in E_D^*$, the set of possible states (in the observer $\mathbf{O}_{RH}$) with respect to P_D starting from a set of states $Q_{RH} \subseteq X_{RH}$ is defined as $R_D(Q_{RH}, \omega) = \{x'_{RH} \in X_{RH} \mid \exists x_{RH} \in Q_{RH}, \exists \sigma \in E_{RH}^* : P_D(\sigma) = \omega, f_{RH}(x_{RH}, \sigma) = x'_{RH}\}$. Then, the defender observer, denoted by $\mathbf{O}_D$, is given by the following definition (we allow events in $E_{RH} \setminus E_D$ to perform a self-loop in $\mathbf{O}_D$ for notational convenience).

Definition 4.1: Given a system $G = (X, E, f, x_0)$ with respect to E_{RH} and P_D, the defender observer is defined as $\mathbf{O}_D = (X_D, E_{RH}, f_D, x_D^0)$, where $X_D \subseteq 2^{X_{RH}} \subseteq 2^{2^X}$ is the state space, E_{RH} is the event set, $x_D^0 = R_D(x_{RH}^0, \varepsilon)$ is the set of initial states, and $f_D : X_D \times E_{RH} \to X_D$ is the transition function defined for $x_D \in X_D$ and $\gamma \in E_D$ as $f_D(x_D, \gamma) = R_D(x_D, \gamma)$, and for $\beta \in E_{RH} \setminus E_D$, $f_D(x_D, \beta) = x_D$.

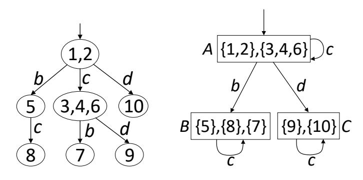
Recall that the edit function confuses the intruder by creating a perturbed output sequence. This is captured by the defended observer that can be constructed by replacing all events in $\mathbf{O}_D$ that are observed by the edit function with the corresponding edited operations.

Definition 4.2: Given a system $G = (X, E, f, x_0)$ with respect to E_D, P_D and f_e, a defended observer is constructed by $\mathbf{O}_T = (X_T, E_{RH}, f_T, x_T^0)$, where $X_T = X_D$ is the state space, $x_T^0 = x_D^0$ is the set of initial states, E_{RH} is the set of events, and f_T is the transition function that implements edited operations as $f_T(x_T, f_e(\gamma)) = f_D(x_T, f_e(\gamma))$ for $x_T \in X_T$ and $\gamma \in E_D$ if $f_e(\gamma)$ is defined; and $f_T(x_T, \beta) = x_T$ for $\beta \in E_{RH} \setminus E_D$.

Since the edit function can only manipulate events in E_D, the transition function $f_T(x_T, f_e(\gamma))$ implements the edited operations if: i) receiving observation $\gamma \in E_D$ for $x_T \in X_T$, and ii) $f_D(x_T, f_e(\gamma))$ is defined. Thus, the transition function $f_T(x_T, f_e(\gamma))$ ensures that the defended observer contains all edited operations to react to every event observed by the edit function; a self-loop is added for all events unobserved by the edit function at each state.

Example 2: Recall the system (shown in Fig. 1) in Example 1. Assume that the edit function knows that $E_D = \{b, d\}$, $E_D \subseteq E_I$, and the set of secret states $X_S = \{8\}$. One can construct the resilient hypothetical observer $\mathbf{O}_{RH}$ with $E_I = E_{RH} = E_o$ as shown in Fig. 2(a). Note that $\mathbf{O}_{RH}$ is identical to the system observer $\mathbf{O}_o$. In a manner similar to Theorem 1 in [4], we can conclude that CSO is violated since there exists a solely secret state $\{8\}$ in $\mathbf{O}_{RH}$.

The defender observer $\mathbf{O}_D$ is constructed according to Definition 4.1 as shown in Fig. 2(b). For the sake of readability, we rename states $[\{1,2\},\{3,4,6\}]$, $[\{5\},\{8\},\{7\}]$, and $[\{9,10\}]$ as A, B, and C, respectively. Notice that each state in $\mathbf{O}_D$ is a subset of states in $\mathbf{O}_{RH}$, which is used to infer the possible estimates of the system captured by the intruder (e.g., state B in $\mathbf{O}_D$ means that the intruder may reach the secret state from the perspective of the edit function).



(a) $\mathbf{O}_o$ or $\mathbf{O}_{RH}$. (b) $\mathbf{O}_D$ or $\mathbf{O}_T$.

Fig. 2. Illustrations of the observers.

The defended observer $\mathbf{O}_T$, constructed following Definition 4.2, is identical to $\mathbf{O}_D$. For instance, from the initial state A in $\mathbf{O}_T$, the new state is state C if the edit function receives observation b and replaces b with d via $f_e(b) = d$. There is no update if the edit function inserts d before b via $f_e(b) = db$ since db is not defined (i.e., we cannot find sequence db from the initial state in $\mathbf{O}_D$). $\diamond$

B. Edit Mechanism

Our objective is to construct an edit mechanism that contains all u-enforcing edit functions. To do so, we first assume that the edit function observes (but cannot react to) events in $E_{RH} \setminus E_D$, and build a two-player game structure, where the players are the system and the edit function, to systematically illustrate how the edit function could execute each allowed edited operation following the system behavior. Then, we remove the above assumption that events in $E_{RH} \setminus E_D$ are observed.

Definition 4.3: Consider a system $G = (X, E, f, x_o)$ with a set of secret states X_S, and its system observer $\mathbf{O}_o = (X_o, E_o, f_o, x_o^0)$, defender observer $\mathbf{O}_D = (X_D, E_{RH}, f_D, x_D^0)$, resilient hypothetical observer $\mathbf{O}_{RH} = (X_{RH}, E_{RH}, f_{RH}, x_{RH}^0)$, and defended observer $\mathbf{O}_T = (X_T, E_{RH}, f_T, x_T^0)$. An edit game structure between the system and the edit function is defined as $\mathcal{EGS} = (V, E_o, \delta, v_0)$, where

1) $V = V_A \cup V_F$, where $V_A = X_o \times X_D \times X_{RH} \times X_T$ is the set of system states and $V_F = (X_o \times X_D \times X_{RH} \times X_T) \times E_o$ is the set of edit function states;
2) $v_0 = (x_o^0, x_D^0, x_{RH}^0, x_T^0) \in V_A$ is the initial state;
3) E_o is the set of actions;
4) $\delta = \delta_{ID} \cup \delta_{DI}$ is the transition function, where
   a) $\delta_{ID} = V_A \times E_o \to V_F$ is the transition function from the system to the edit function defined as: $\forall (x_o, x_D, x_{RH}, x_T) \in V_A, \forall \alpha \in E_o, \delta_{ID}((x_o, x_D, x_{RH}, x_T), \alpha) = ((f_o(x_o, \alpha), f_D(x_D, \alpha), x_{RH}, x_T), \alpha)$ if both $f_o(x_o, \alpha)$ and $f_D(x_D, \alpha)$ are defined;
   b) $\delta_{DI} : V_F \times E_o^{\leq (K+1)} \to V_A$ is the transition function from the edit function to the system defined as: b.1) $\forall ((x_o, x_D, x_{RH}, x_T), \alpha) \in V_F$ where $\alpha \in E_D$, $\forall \omega \in E_D^*$, we have i) $\omega = \alpha' \in E_D \setminus \{\alpha\}$ via $f_e(\alpha) = \omega$ (replacement), ii) $\omega = \varepsilon$ via $f_e(\alpha) = \varepsilon$ (deletion), and iii) $\omega = \omega' \alpha$ via $f_e(\alpha) = \omega' \alpha$ (insertion), such that $\delta_{DI}(((x_o, x_D, x_{RH}, x_T), \alpha), \omega) = ((x_o, x_D, f_{RH}(x_{RH}, \omega), f_T(x_T, \omega))$ if both $f_{RH}(x_{RH}, \omega)$ and $f_T(x_T, \omega)$ are defined; and b.2) $\forall ((x_o, x_D, x_{RH}, x_T), \alpha) \in V_F$ where $\alpha \in E_o \setminus E_D$, we have $\delta_{DI}(((x_o, x_D, x_{RH}, x_T), \alpha), \alpha) = (x_o, x_D, f_{RH}(x_{RH}, \alpha), f_T(x_T, \alpha) = x_T)$.

Note that we use $E_o^{\leq (K+1)} \subset E_o^*$ to denote the finite set of strings of maximum length $K+1$, where $K \in \mathbb{N}$ is the maximum number for insertions. In this regard, $E_o^{\leq (K+1)}$ can represent all edited operations (including replacements, deletions, and finite insertions) during the evolution of the transition function $\delta_{ID} : V_F \times E_o^{\leq (K+1)} \to V_A$. By taking advantage of the construction of $\mathbf{O}_D$, the edit function is assumed to "react" with the same event if the system generates an event unobserved by the edit function. With alternate moves, $\mathcal{EGS}$ is constructed such that it contains all possible actions between the system and the edit function.

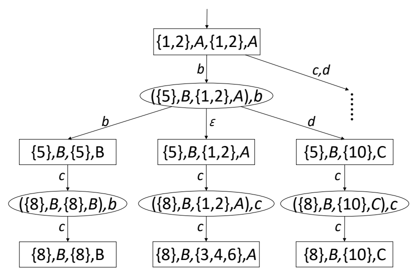


Fig. 3. Part of the edit game structure.

Example 3: Consider again the system G in Fig. 1 and its system observer, defender observer, resilient hypothetical observer, and defended observer in Fig. 2. By Definition 4.3, one can construct the edit game structure $\mathcal{EGS}$. To conserve space, we only show part of $\mathcal{EGS}$ in Fig. 3. From the initial state $(\{1,2\}, A, \{1,2\}, A)$, the $\mathcal{EGS}$ evolves by following the system behavior (i.e., b, c, or d). If event b occurs first, then the initial state will update to $[(\{5\}, B, \{1,2\}, A), b]$ via $f_o(\{1,2\}, b) = \{5\}$, $f_D(A, b) =$

B. Note that event b in $[(\{5\}, B, \{1,2\}, A), b]$ is used to track the last observed event. Then, the edit function has three options to react to b (e.g., replacing b with d such that state $(\{5\}, B, \{10\}, C)$ is reached). Since the edit function can only execute edited operations once it observes a symbol; it cannot do anything if the system generates an event that the edit function cannot observe, i.e., the last component A in state $[(\{8\}, B, \{3,4,6\}, A), c]$ remains unchanged since c is unobservable by the edit function. $\diamond$

Note that $\mathcal{EGS}$ could generate problematic states when 1) the edit function may be not able to react to one or more events generated by the system; 2) the intruder may infer the secrets after the edited operations via the edit function (e.g., state $(\{8\}, B, \{8\}, B)$ in Fig. 3). To capture such problematic states, we first introduce a utility function.

Definition 4.4: The utility function $U : V \to \{0, 1\}$ is defined for each state $v \in V$ such that:

$$U(v) = \begin{cases} 0, & \textit{if } [[(\exists x \in x_T) x \subseteq X_S] \wedge [v \in V_A]] \vee \\ & [(\forall \omega \in E_D^*)[\delta_{DI}(v, \omega) \textit{ is undefined }] \wedge [v \in V_F]], \\ 1, & \textit{otherwise.} \end{cases}$$

The utility function captures two types of problematic states. Specifically, state $(x_o, x_D, x_{RH}, x_T) \in V_A$ is assigned value 0 if there exists $x \in x_T$ such that $x \subseteq X_S$; since state x_T contains subsets of states of the system G (each subset representing a possible set of estimates at the intruder), it means that the intruder may be able to infer the secret states from the edit function's point of view if for some $x \in x_T$, we have $x \subseteq X_S$. A state of the form $[(x_o, x_D, x_{RH}, x_T), \alpha] \in V_F$ is assigned value 0 if for all edited operations in the form $\omega \in E_D^*$, $\delta_{DI}(v, \omega)$ is not defined; this means that the edit function is not able to react to event α.

The process of pruning away all problematic states according to the utility function can be formulated as an instance of a *supervisory control problem without blocking* (SCPB) [1]. In this regard, one can mark all system states $V_A \subseteq V$ and leave all edit function states $V_F \subseteq V$ unmarked. All outgoing actions of V_A (i.e., observable events outputted by the system $\alpha \in E_o$) and all outgoing actions $\alpha \in (E_o \setminus E_D)$ of V_F (i.e., events unobserved by the edit function) are taken to be uncontrollable because the edit function cannot respond to events it cannot observe. All other outgoing actions of V_F (i.e., events edited by the edit function in the form $\omega \in f_e(\alpha)$ for $\alpha \in E_D$) are controllable. The specification for the supervisory control problem is the trimmed version of $\mathcal{EGS}$, denoted by $\mathcal{EGS}_{trim}$, which can be obtained by iteratively pruning away all problematic states captured by the utility function. Then, one can obtain $L_m(\mathcal{EGS}_{trim})^{\uparrow C}$ with respect to $\mathcal{EGS}$ by following the standard $\uparrow C$ algorithm in [1]. In the end, the trimmed game structure $\mathcal{TGS} = (V_T, E_o, \delta, v_0)$ can be obtained as the subautomaton that generates $L_m(\mathcal{EGS}_{trim})^{\uparrow C}$. The reader is referred to [1] for more details; in this case, the trimmed game structure is the supervisor of the edit game structure.

Example 4: Continuing with Example 3, the trimmed game structure $\mathcal{TGS}$ can be constructed from $\mathcal{EGS}$ via supervisory control as follows. One can prune away all problematic states by following the utility function, which results in the specification for the supervisory control problem, i.e., state $[\{8\}, B, \{8\}, B]$ is problematic since the defended observer is at state B so that the intruder may infer the secret state $\{8\}$. Then, edited operation b is disabled at state $[(\{5\}, B, \{1,2\}, A), b]$ to prevent problematic state $[(\{8\}, B, \{8\}, B), b]$ since b is controllable. The resulting trimmed game structure $\mathcal{TGS}$ is shown in Fig. 4. $\diamond$

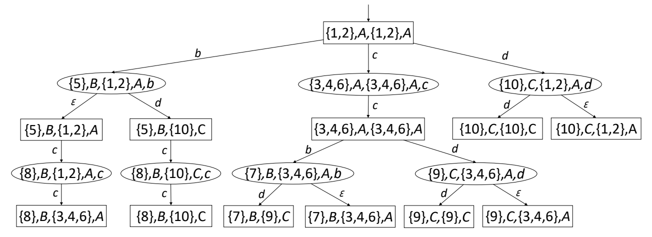


Fig. 4. Trimmed game structure.

Next, the edit mechanism is constructed by merging states from $\mathcal{TGS}$, so as to: (i) incorporate the fact that events in $E_{RH} \setminus E_D$ are not observable to the edit function; (ii) ensure that all edited operations are defined at the merged states (otherwise the intruder will infer the existence of the edit function if the edit function outputs an edited operation that is not recognized from the point view of the intruder. The formal procedure is presented in Algorithm 1.

Algorithm 1: Construction of edit mechanism $\mathcal{EM}$

```
Input: $\mathcal{TGS} = (V_T, E_o, \delta, v_0)$, where $V_T = V'_A \cup V'_F \subseteq V_A \cup V_F$, and $\delta = \delta_{ID} \cup \delta_{DI}$.
Output: $\mathcal{EM} = (V_E, E_D, \delta_I \cup \delta_D, v_0^E)$, where $V_E = V_m \cup V_n$.
1  $v_0^E = \{v_0\} \cup \{v \in V'_A \mid \exists \sigma \in (E_{RH} \setminus E_D)^* : \delta(v_0, \sigma) = v\}$;
2  Initialize $V_E = V_m = \{v_0^E\}$, and $V_n = \emptyset$;
3  for $v \in V_m$ that have not been examined do
4      for $\gamma \in E_D$ do
5          if $\exists z \in v$, $\delta_I(z, \gamma)$ is defined then
6              $\delta_I(v, \gamma) = \bigcup_{z \in v} \delta_I(z, \gamma) = \{y \in V'_F \mid \exists \sigma \in E_{RH}^* : P_D(\sigma) = \gamma \wedge y \in \delta(z, \sigma)\}$ ;
7              Add $\delta_I(v, \gamma)$ to $V_n$;
8  for $v \in V_n$ that have not been examined do
9      for $\omega \in E_D^*$ do
10         if $\forall z \in v$, $\delta_D(z, \omega)$ is defined then
11             $\delta_D(v, \omega) = \bigcup_{z \in v} \delta_D(z, \omega) = \bigcup_{z \in v} \delta_{DI}(z, \omega)$;
12             Add $\delta_D(v, \omega)$ to $V_m$;
13 Go back to line 3 and repeat until all accessible part has been built.
```

We briefly explain how Algorithm 1 works here. We first compute the initial state of the edit mechanism v_0^E in Line 1 as the set of states that can be reached from state v_0 in $\mathcal{TGS}$ via sequences of events unobserved by the edit function. For the sake of explanation, we assume that $v_0^E = \{v_0, v_1\}$. Then, steps 3 to 10 evolve the initial state v_0^E to a new state $v_1^E \in V_m$ via δ_I, i.e., $v_1^E = \{v_0', v_1'\}$ via $\delta(v_0, \sigma) = v_0'$ and $\delta(v_1, \sigma) = v_1'$ when the edit function receives observation $\gamma \in E_D$ from the sequences $\sigma = \sigma_1\gamma\sigma_2$ generated by the system in $\mathcal{TGS}$, where $\sigma_1, \sigma_2 \in (E_{RH} \setminus E_D)^*$. Since the edit function outputs the same event if the system generates an event unobserved by the edit function in $\mathcal{TGS}$, v_0' and v_1' in v_1^E are system states in V_A. Steps 11 to 18 evolve state v_1^E to a new state $v_2^E \in V_n$ via δ_D, i.e., $v_2^E = \{v_0'', v_1''\}$ via $\delta_{DI}(v_0', \omega) = v_0''$ and $\delta_{DI}(v_1', \omega) = v_1''$ when the edit function manipulates observation γ to ω in terms of $f_e(\gamma) = \omega$. In this case, both of states v_0' and v_1' in v_1^E should be defined for γ by δ_{DI}, otherwise the edit function may be inferred by the intruder since the intruder can distinguish between these two states.

Theorem 4.5: An edit function is u-enforcing if and only if it can be synthesized from the edit mechanism $\mathcal{EM}$.

Proof: (If) By contradiction, we assume that the edit function is not u-enforcing. That is, at least one of the conditions in Definition 3.6 does not hold. Thus, it cannot react to every event observed by the edit function or it cannot modify sequences generated by the system to edited sequences such that the intruder cannot infer the secret. In other words, one can find a sequence leading to states that satisfies $U(v) = 0$. However, by construction of $\mathcal{TGS}$, such sequences have been removed. Furthermore, suppose the edit function cannot ensure that every edited sequence can be recognized by the intruder; however, this is not allowed in $\mathcal{EM}$, which is a contradiction. Therefore, the edit function should be u-enforcing. (Only if) Given a u-enforcing edit function, the four conditions in Definition 3.6 hold. Thus, it can be retained in $\mathcal{EGS}$ in terms of the first condition since we build $\mathcal{EGS}$ by following system behavior observed by the edit function. Moreover, we prune away all states violating the utility function when we build $\mathcal{TGS}$ such that the second condition holds. Finally, the transition function in $\mathcal{EM}$ is able to ensure that every edited operation can be recognized by the intruder. Therefore, all edited sequences exist in $\mathcal{EM}$, such that any subsequent edited sequence can maintain the previous three conditions. One can conclude that the edit function can be synthesized from $\mathcal{EM}$. ■

Example 5: Continuing with Example 4, the $\mathcal{TGS}$ can be transformed to the edit mechanism $\mathcal{EM}$ as follows. The initial state in $\mathcal{EM}$ is $v_0^E = \{v_0, v_1\}$ in terms of $v_0^E = \{v_0\} \cup \{v_1 \mid \exists c \in (E_{RH} \setminus E_D)^* : \delta(v_0, c) = v_1\}$, where $v_0 = (\{1, 2\}, A, \{1, 2\}, A)$ and $v_1 = (\{3, 4, 6\}, A, \{3, 4, 6\}, A)$. Subsequently, state $v_1^E = \{[(\{5\}, B, \{1, 2\}, A), b], [(\{7\}, B, \{3, 4, 6\}, A), b]\}$ is reached if the edit function receives observation b in terms of $\delta_I(v_0^E, b) = v_1^E$. Finally, state $v_2^E = \{(\{5\}, B, \{10\}, C), (\{8\}, B, \{10\}, C), (\{7\}, B, \{9\}, C)\}$ is reached if the edit function replaces b with d in terms of $\delta_D(v_1^E, d) = v_2^E$. Note that the edit function is not able to replace b with b since b is not defined at state $[(\{5\}, B, \{1, 2\}, A), b]$ in v_1^E in $\mathcal{TGS}$; if b is used, in such case, the intruder will infer the presence of the edit function. In the end, the edit mechanism in Fig. 5 can be obtained by following Algorithm 1. Given a sequence abc generated by the system, the intruder can infer the secret state $\{8\}$ by following observation $P_I(abc) = bc$ via the system observer $\mathbf{O}_o$. At this point, the edit mechanism reaches state $\{[(\{5\}, B, \{1, 2\}, A), b], [(\{7\}, B, \{3, 4, 6\}, A), b]\}$ via observation b. By following the edit mechanism, the edit function can respond to b via either a deletion operation (outputs ε) or a replacement operation (outputs d). In either case, from the point view of the edit function, the intruder will not be able to infer the secret state. $\diamond$

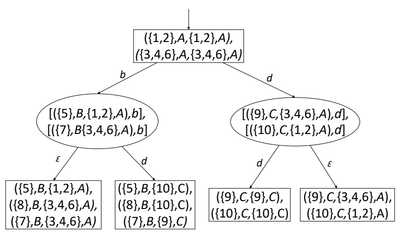


Fig. 5. Edit mechanism under uncertain observations.

V. Conclusions

In this paper, we have considered the problem of CSO enforcement via edit functions under the uncertain observation setting. The notion of u-enforceability has been proposed to characterize the capability of the edit functions to enforce opacity of the system. Then, we construct the edit mechanism via a game scheme to implement all u-enforcing edit functions. Our future work will investigate a more general case where no inclusion relation between E_I and E_D exists.

References

[1] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*. US: Springer, 2010.

[2] C. N. Hadjicostis, *Estimation and Inference in Discrete Event Systems: A Model-Based Approach with Finite Automata*. Switzerland: Springer, 2020.

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

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

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

[6] 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.

[7] X. Yin and S. Lafortune, "A uniform approach for synthesizing property-enforcing supervisors for partially-observed discrete-event systems," *IEEE Transactions on Automatic Control*, vol. 61, no. 8, pp. 2140–2154, 2015.

[8] Y. Tong, Z. Li, C. Seatzu, and A. Giua, "Current-state opacity enforcement in discrete event systems under incomparable observations," *Discrete Event Dynamic Systems*, vol. 28, pp. 161–182, 2018.

[9] 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.

[10] Y. C. Wu, V. Raman, B. C. Rawlings, S. Lafortune, and S. A. Seshia, "Synthesis of obfuscation policies to ensure privacy and utility," *Journal of Automated Reasoning*, vol. 60, no. 3, pp. 107–131, 2018.