Efficient Synthesis of Edit Functions for Opacity Enforcement Using Bisimulation-Based Abstractions

Sahar Mohajerani, Yiding Ji and Stéphane Lafortune

***Abstract*— This paper investigates the synthesis of edit functions for opacity enforcement using abstraction methods to reduce computational complexity. Edit functions are used to alter system outputs by erasing or inserting events in order to prevent violations of opacity. We introduce two abstraction methods, called *opaque observation equivalence* and *opaque bisimulation*, that are used to abstract the original system and its observer before calculating edit functions. We present a set of results on abstraction for opacity and its enforcement by edit functions that prove that edit functions synthesized from abstracted models are "equivalent" to ones synthesized from original ones. Our approach leverages the technique of edit function synthesis using the *All Edit Structure* from prior works.**

***Index Terms*— Finite-state automata, abstraction, opacity, edit function.**

I. INTRODUCTION

Opacity is a security property that characterizes whether the integrity of system *secrets* can be preserved from the inference of an outside intruder, potentially with malicious purposes. The intruder is modeled as an observer with knowledge of the system's structure. A system is called opaque if the intruder is unable to infer any of the system's secrets from its observations using model-based inferencing. For systems modeled as finite state automata or Petri nets, various notions of opacity have been studied and the survey paper [1] summarizes recent results on opacity in discrete event systems (DES). In this paper, we consider automata models and current-state opacity [2].

When opacity does not hold, it is natural to study its enforcement. Several mechanisms have been considered for this purpose; see [1]. We consider the technique of opacity enforcement using *insertion* or *edit* functions, initially proposed in [3] and further developed in [4]–[7]. An edit function works as an interface between the system's output and the intruder's observations; it may manipulate the output of the system by erasing events of by inserting fictitious events to obfuscate the intruder.

Recently, the work [8] revisited the synthesis of edit functions using a three-player game-like discrete structure called the All Edit Structure (AES), which provably embeds all opacity-enforcing edit functions. This structure is constructed as a so-called "three-player observer" from the system model and the opacity enforcement problem is viewed as a three-player game and solved within the AES. The AES of [8] is the structure that we will abstract in this paper, by using bisimulation-based abstractions of the system model and of its observer.

To construct the AES, the observer and the *desired observer* of the system need be calculated [3]. The desired observer is obtained by removing the states of the observer that are subsets of secret states. Calculating the observer can potentially be computationally costly. To mitigate this issue, we investigate abstraction methods that can be used to reduce the size of the system before calculating its observer and before calculating the AES. *Bisimulation* and *observation equivalence* [9] are well-known abstraction methods to abstract the state space of an automaton. Bisimulation and observation equivalence in general cannot be used in an opacity setting; some adjustments must be made. Abstraction-based bisimulation was used in [10] to reduce the state space of the system when *verifying* infinite-step opacity, a property that is different from current-state opacity. We introduce special versions of bisimulation called *opaque observation equivalence* and *opaque bisimulation*, which consider the secrecy status of states in the bisimulation setting.

In our methodology, the system is first abstracted by merging some states or removing some transitions using opaque observation equivalence. Next, the observer of the abstracted system is calculated. Since the abstraction procedure reduces the size of the system, the computational complexity of calculating the observer can potentially be reduced significantly. In addition, the observer of the system can be abstracted further using opaque bisimulation, before calculating the desired observer. We show that the abstracted observer and the abstracted desired observer are bisimilar to their original counterparts and that they can be used to obtain an abstracted AES. We prove that the abstracted AES calculated in this manner still contains all the edit functions that can be used to enforce opacity. Therefore, the performed abstractions preserve the full generality of the AES for the purpose of edit function synthesis.

The presentation of our results is organized as follows. Sect. II gives a brief background about edit function and the AES. Next, Sect. III explains how the abstraction-based AES can be obtained. Then, Sect. IV defines the abstraction methods that are used and shows how they are leveraged for AES synthesis. Finally, some concluding remarks are given in Sect. V. The proof of Theorems 2 and 4 have been omitted due to space constraints.

The work of the first author was supported by the Swedish Research Council. The work of the second and third authors was supported in part by US NSF grant CNS-1421122 and CNS-1738103.

Sahar Mohajerani, Yiding Ji and Stéphane Lafortune are with the Department of Electrical Engineering and Computer Science at the University of Michigan, Ann Arbor, Michigan, USA. {saharm;jiyiding;stephane@umich.edu}

II. OPACITY ENFORCEMENT BY EDIT FUNCTIONS

We need to introduce a good amount of material about opacity enforcement before we can present our contributions in the following sections. We consider discrete event systems modeled by deterministic or nondeterministic automata.

Definition 1: A (nondeterministic) finite-state automaton is a tuple $G = \langle \Sigma_{tot}, Q, \rightarrow, Q^\circ \rangle$, where Σ_{tot} is a finite set of events, Q is a finite set of states, $\rightarrow \subseteq Q \times \Sigma_{tot} \times Q$ is the *state transition relation*, and $Q^\circ \subseteq Q$ is the set of *initial states*. G is *deterministic*, if $|Q^\circ| \leq 1$ and $x \xrightarrow{\sigma} y_1$ and $x \xrightarrow{\sigma} y_2$ always implies $y_1 = y_2$.

We assume that the intruder can only partially observe G. Thus, Σ_{tot} is partitioned into two disjoint subsets: Σ the set of *observable* events and Σ_{uo} the set of *unobservable* events, $\Sigma_{tot} = \Sigma \dot\cup \Sigma_{uo}$. Moreover, in opacity problems, the set of states is also partitioned into two disjoint subset: Q^S the set of *secret states* and $Q^{NS} = Q \setminus Q^S$ the set of *non-secret states*.

Σ^* is the set of all finite traces of events from Σ, including the *empty trace* ε. The *natural projection* $P\colon \Sigma_{tot}^* \to \Sigma^*$ is the operation that removes from traces $t \in \Sigma_{tot}^*$ all events not in Σ, which means unobservable events. The transition relation is written in infix notation $x \xrightarrow{\sigma} y$, and is extended to strings in Σ_{tot}^* by letting $x \xrightarrow{\varepsilon} x$ for all $x \in Q$, and $x \xrightarrow{t\sigma} z$ if $x \xrightarrow{t} y$ and $y \xrightarrow{\sigma} z$ for some $y \in Q$. Furthermore, $x \xrightarrow{t}$ means that $x \xrightarrow{t} y$ for some $y \in Q$, and $x \to y$ means that $x \xrightarrow{t} y$ for some $t \in \Sigma_{tot}^*$. These notations also apply to state sets, $X \xrightarrow{t} Y$ for $X, Y \subseteq Q$ means that $x \xrightarrow{t} y$ for some $x \in X$ and $y \in Y$, and to automata, $G \xrightarrow{t}$ means that $Q^\circ \xrightarrow{t}$, etc. For brevity, $p \overset{s}{\Rightarrow} q$, with $s \in \Sigma^*$, denotes the existence of a string $t \in \Sigma_{tot}^*$ such that $P(t) = s$ and $p \xrightarrow{t} q$. Similarly, $p \Rightarrow q$ means that there exists $t \in \Sigma_{uo}$ such that $p \xrightarrow{t} q$.

The *language* of an automaton G is $\mathscr{L}(G) = \{ s \in \Sigma^* \mid G \overset{s}{\Rightarrow} \}$ and the language generated by G from $q \in Q$ is $\mathscr{L}(G, q) = \{ s \in \Sigma^* \mid q \overset{s}{\Rightarrow} \}$. For nondeterministic automaton $G = \langle \Sigma_{tot}, Q, \rightarrow, Q^\circ \rangle$, the set of *unobservably reached states* of $B \in 2^Q$, is $UR(B) = \bigcup \{ C \subseteq Q \mid B \Rightarrow C \}$. The *observer automaton* $det(G) = \langle \Sigma, X_{obs}, \rightarrow_{obs}, X_{obs}^\circ \rangle$ is a deterministic automaton, where $X_{obs}^\circ = UR(Q^o)$ and $X_{obs} \subseteq 2^Q$, and $X \rightarrow_{obs} Y$, where $X, Y \subseteq X_{obs}$, if and only if $Y = \bigcup \{ UR(y) \mid x \xrightarrow{\sigma} y \text{ for some } x \in X \text{ and } y \in Q \}$. By convention, in this paper, only reachable states from X_{obs}° under $\rightarrow_{obs}$ are considered in X_{obs}.

One common automaton operation is the *quotient* modulo an equivalence relation on the state set.

Definition 2: Let Z be a set. A relation $\sim \subseteq Z \times Z$ is called an *equivalence relation* on Z if it is reflexive, symmetric, and transitive. Given an equivalence relation $\sim$ on Z, the *equivalence class* of $z \in Z$ is $[z] = \{ z' \in Z \mid z \sim z' \}$, and $\tilde{Z} = \{ [z] \mid z \in Z \}$ is the set of all equivalence classes modulo $\sim$.

Definition 3: Let $G = \langle \Sigma_{tot}, Q, \rightarrow, Q^\circ \rangle$ be an automaton and let $\sim \subseteq Q \times Q$ be an equivalence relation. The *quotient automaton* of G modulo $\sim$ is

$$\tilde{G} = \langle \Sigma_{tot}, \tilde{Q}, \rightarrow/\sim, \tilde{Q}^\circ \rangle \ , \tag{1}$$

where $\rightarrow/\sim = \{ ([x], \sigma, [y]) \mid x \xrightarrow{\sigma} y \}$ and $\tilde{Q}^\circ = \{ [x^\circ] \mid x^\circ \in Q^\circ \}$.

Bisimulation is a widely-used notion of abstraction that merges states with the same future behaviour.

Definition 4: [9] Assume $G_1 = \langle \Sigma_{tot,1}, Q_1, \rightarrow_1, Q_1^\circ \rangle$ and $G_2 = \langle \Sigma_{tot,2}, Q_2, \rightarrow_2, Q_2^\circ \rangle$ are two automata. A relation $\approx \subseteq Q_1 \times Q_2$ is called a bisimulation between G_1 and G_2 if, for all $x_1 \in Q_1$ and $x_2 \in Q_2$ and for all $\sigma \in \Sigma$ such that $x_1 \approx x_2$,

$$\text{if } x_1 \xrightarrow{\sigma} y_1 \text{ then } \exists y_2 \in Q_2 \text{ such that } x_2 \xrightarrow{\sigma} y_2 \text{ and } y_1 \approx y_2,$$
$$\text{if } x_2 \xrightarrow{\sigma} y_2 \text{ then } \exists y_1 \in Q_1 \text{ such that } x_1 \xrightarrow{\sigma} y_1 \text{ and } y_1 \approx y_2.$$

G_1 and G_2 are bisimilar if there exists a bisimulation $\approx$ between G_1 and G_2 such that $Q_1^\circ \approx Q_2^\circ$.

When def. 4 is applied on a single automaton (i.e., over state space $Q \times Q$), the bisimulation seeks to merge sates with the same outgoing transitions into equivalence classes, including outgoing unobservable events. If the unobservable events are disregarded in bisimulation, a more general abstraction method called *weak bisimulation* or *observation equivalence* is obtained.

Definition 5: [9] Let $G = \langle \Sigma_{tot}, Q, \rightarrow, Q^\circ \rangle$, where $\Sigma_{tot} = \Sigma \dot\cup \Sigma_{uo}$, be a nondeterministic automaton. An equivalence relation $\sim \subseteq Q \times Q$ is called an *observation equivalence* on G, if the following holds for all $x_1, x_2 \in Q$ such that $x_1 \sim x_2$: if $x_1 \overset{s}{\Rightarrow} y_1$ for some $s \in \Sigma^*$, then there exists $y_2 \in Q$ such that $x_2 \overset{s}{\Rightarrow} y_2$, and $y_1 \sim y_2$.

A system is opaque if an intruder cannot determine with certainty, from the observed behavior, if the system has entered a secret state. Different notions of opacity have been introduced in literature [1]. In this paper only current-state opacity is considered [2].

Definition 6: A nondeterminitic automaton G with set of observable events $\Sigma_{tot} = \Sigma \dot\cup \Sigma_{uo}$ and set of secret states Q^S is *current-state opaque* with respect to Q^S if and only if

$$(\forall q_0 \in Q^o, \forall s \in \mathscr{L}(G, q^\circ) \ : \ q^\circ \overset{s}{\Rightarrow} Q^S)$$
$$\textit{then } (\exists q'^\circ \in Q^o) \textit{ such that } [\ q'^\circ \overset{s}{\Rightarrow} Q^{NS}]$$

The system is current-state opaque if for any string reaching a secret state, there is a string with the same sequence of observable event reaching a non-secret state.

It is well-known [1] that current-state opacity can be verified by building the observer automaton of G.

Definition 7: Let $G = \langle \Sigma_{tot}, Q, \rightarrow, Q^\circ \rangle$ be a nondeterministic automaton with set of secret state Q^S. Let $det(G) = \langle \Sigma, X_{obs}, \rightarrow_{obs}, X_{obs}^\circ \rangle$ be the observer of G. Then G is current-state opaque with respect to Q^S if and only if $[det(G) \rightarrow_{obs} [s]X$ implies that $X \not\subseteq Q^S]$.

If all the states of the observer $det(G)$ that are violating current-state opacity are removed, the accessible part of the resulting subautomaton of $det(G)$ is called the *desired observer*, denoted by $det_d(G)$. The language generated by the desired observer is referred to as *safe language* (w.r.t. opacity): $L_{safe} = \mathscr{L}(det_d(G))$. Accordingly, we define the *unsafe language*, $L_{unsafe} = \mathscr{L}(G) \setminus L_{safe}$.

If a system is not current-state opaque, then it is possible to add an output interface called an *edit function* to enforce opacity [4], [5], [8]. An edit function can both insert and erase events and the intruder cannot distinguish between

inserted events and their genuine counterparts. We denote by $\Sigma^{\varepsilon} = \{\sigma \to \varepsilon : \sigma \in \Sigma\}$ the set of "event-erasure" events.

Definition 8: A *deterministic edit function* is defined as $f_e : \Sigma^* \times \Sigma \to \Sigma^*\Sigma \cup \{\varepsilon\}$. Given $s \in \mathscr{L}(G)$, $\sigma \in \Sigma$,

$$f_e(s,\sigma) = \begin{cases} s_I\sigma & \text{if } s_I \text{ is inserted before } \sigma \\ \varepsilon & \text{if } \sigma \text{ is erased} \\ s_I & \text{if } s_I \text{ is inserted and } \sigma \text{ is erased} \end{cases}$$

In [5] *private safety* for an edit function is defined, when the intruder does not know about the edit function's implementation.

Definition 9 (Private Safety): [5] Given G and its observer $det_d(G)$, an edit function f_e is privately safe if $\forall s \in \mathscr{L}(G)$, $f_e(s) \in L_{safe}$, i.e. $f_e(\mathscr{L}(G)) \subseteq L_{safe}$.

Recently, an approach to calculate the *All Edit Structure*, or AES, which contains all the opacity-enforcing edit functions, was investigated in [8], building on the work in [3], [5]. In this approach, a so-called *three-player observer* of the system is first calculated, then pruned to obtain the AES. In this paper, we will follow the approach of the three-player observer (TPO) to obtain the AES.

Definition 10 (Three-Player Observer): [8] Given a system G, its observer $det(G)$ and desired observer $det_d(G)$, let $I \subseteq X_{obsd} \times X_{obs}$ be the set of information states. A three-player observer is a tuple of the form $T = (Q_Y, Q_Z, Q_W, \Sigma, \Sigma^{\varepsilon}, \Theta, \to_{yz}, \to_{zz}, \to_{zw}, \to_{wy}, y_0)$, where:

- $Q_Y \subseteq I$ is the set of Y states.
- $Q_Z \subseteq I \times \Sigma$ is the set of Z states. Let $I(z)$, $E(z)$ denote the information state component and observable event component of a Z state z respectively, so that $z = (I(z), E(z))$.
- $Q_W \subseteq I \times (\Sigma \cup \Sigma^{\varepsilon})$ is the set of W-states. Let $I(w)$, $A(w)$ denote the information state component and action component of a W state w respectively, so that $w = (I(w), A(w))$.
- $\Sigma \subseteq \Sigma_{tot}$ is the set of observable events.
- Σ^{ε} is the set of event-erasure events.
- $\Theta \subseteq \Sigma \cup \{\varepsilon\} \cup \Sigma^{\varepsilon}$ is the set of edit decisions at Z states.

(i) $\to_{yz}$: $Q_Y \times \Sigma \times Q_Z$ is the transition function from Y states to Z states. For $y = (x_d, x_f) \in Q_Y$, $e_o \in \Sigma$, we have: $y \xrightarrow{e_o}_{yz} z \Rightarrow [x_f \xrightarrow{e_o}_{obs}] \wedge [I(z) = y] \wedge [E(z) = e_o]$.

(ii) $\to_{zz}$: $Q_Z \times \Theta \times Q_Z$ is the transition function from Z states to Z states. For $z = ((x_d, x_f), e_o) \in Q_Z$, $\theta \in \Theta$, we have: $z \xrightarrow{\theta}_{zz} z' \Rightarrow [\theta \in \Sigma] \wedge [I(z') = (x'_d, x_f)] \wedge [x_d \xrightarrow{\theta}_{det_d} x'_d] \wedge [E(z') = e_o]$.

(iii) $\to_{zw1}$: $Q_Z \times \Theta \times Q_W$ is the ε insertion transition function from Z states to W states. For $z = ((x_d, x_f), e_o) \in Q_Z$, $\theta \in \Theta$ we have: $z \xrightarrow{\theta}_{zw1} w \Rightarrow [\theta = \varepsilon] \wedge [I(w) = I(z)] \wedge [A(w) = e_o] \wedge [x_d \xrightarrow{e_o}_{det_d}] \wedge [x_f \xrightarrow{e_o}_{obs}]$.

(iv) $\to_{zw2}$: $Q_Z \times \Theta \times Q_W$ is the event erasure transition function from Z states to W states. For $z = ((x_d, x_f), e_o) \in Q_Z$, $\theta \in \Theta$, we have: $z \xrightarrow{\theta}_{zw2} w \Rightarrow [\theta = e_o \to \varepsilon] \wedge [I(w) = I(z)] \wedge [A(w) = e_o \to \varepsilon] \wedge [x_f \xrightarrow{e_o}_{obs}]$.

(v) $\to_{wy1}$: $Q_W \times \Sigma \times Q_Y$ is the transition function from W states whose action component is in Σ to Y states. For $w = ((x_d, x_f), e_o) \in Q_W$, we have: $w \xrightarrow{e_o}_{wy1} y \Rightarrow [y = (x'_d, x'_f)] \wedge [x_d \xrightarrow{e_o}_{det_d} x'_d] \wedge [x_f \xrightarrow{e_o}_{obs} x'_f]$.

(vi) $\to_{wy2}$: $Q_W \times \Sigma \times Q_Y$ is the transition function from W states whose action component is in Σ^{ε} to Y states. For $w = ((x_d, x_f), e_o \to \varepsilon) \in Q_W$, we have: $w \xrightarrow{e_o}_{wy2} y \Rightarrow [y = (x_d, x'_f)] \wedge [x_f \xrightarrow{e_o}_{obs} x'_f]$.

- $y_0 \in Q_Y$ is the initial Y state where $y_0 = (x_{obsd,0}, x_{obs,0})$. $x_{obsd,0}$ and $x_{obs,0}$ are initial states of $det_d(G)$ and $det(G)$, respectively.

In order to characterize the information flow in a TPO, the notion of *run* is defined [8].

Definition 11 (Run): [8] In a three-player observer T, a run is defined as: $\omega = y_0 \xrightarrow{e_0} z_0^1 \xrightarrow{\theta_0^1} z_0^2 \xrightarrow{\theta_0^2} \cdots \xrightarrow{\theta_0^{m_0-1}} z_0^{m_0} \xrightarrow{\theta_0^{m_0}} w_0 \xrightarrow{e_0} y_1 \xrightarrow{e_1} z_1^1 \xrightarrow{\theta_1^1} z_1^2 \xrightarrow{\theta_1^2} \cdots z_1^{m_1} \xrightarrow{\theta_1^{m_1}} w_1 \xrightarrow{e_1} y_2 \cdots \xrightarrow{e_n} z_n^1 \xrightarrow{\theta_n^1} \cdots z_n^{m_n} \xrightarrow{\theta_n^{m_n}} w_n \xrightarrow{e_n} y_{n+1}$, where y_0 is the initial state of T, $e_i \in \Sigma$, $\theta_i^j \in \Theta(z_i^j)$, $\forall 0 \le i \le n$, $1 \le j \le m_i$ and $n \in \mathbb{N}$, $m_i \in \mathbb{N}^+$.

For simplicity, similar notation as for automata are used for TPOs and thus $T \xrightarrow{\omega} x$ denotes existence of a run in T. Next, we need to define *edit projection* and *string generated by a run* before we can state the key results we will leverage.

Definition 12 (Edit Projection): [8] Given a run $\omega = y_0 \xrightarrow{e_0} z_0^1 \xrightarrow{\theta_0^1} z_0^2 \xrightarrow{\theta_0^2} \cdots \xrightarrow{\theta_0^{m_0-1}} z_0^{m_0} \xrightarrow{\theta_0^{m_0}} w_0 \xrightarrow{e_0} y_1 \xrightarrow{e_1} z_1^1 \xrightarrow{\theta_1^1} z_1^2 \xrightarrow{\theta_1^2} \cdots z_1^{m_1} \xrightarrow{\theta_1^{m_1}} w_1 \xrightarrow{e_1} y_2 \cdots \xrightarrow{e_n} z_n^1 \xrightarrow{\theta_n^1} \cdots z_n^{m_n} \xrightarrow{\theta_n^{m_n}} w_n \xrightarrow{e_n} y_{n+1}$, edit projection $P_e : \Omega \to P[\mathscr{L}(G)]$ is defined such that $P_e(\omega) = e_0 e_1 \cdots e_n$.

Definition 13 (String Generated by a Run): [8] Given a run $\omega = y_0 \xrightarrow{e_0} z_0^1 \xrightarrow{\theta_0^1} z_0^2 \xrightarrow{\theta_0^2} \cdots \xrightarrow{\theta_0^{m_0-1}} z_0^{m_0} \xrightarrow{\theta_0^{m_0}} w_0 \xrightarrow{e_0} y_1 \xrightarrow{e_1} z_1^1 \xrightarrow{\theta_1^1} z_1^2 \xrightarrow{\theta_1^2} \cdots z_1^{m_1} \xrightarrow{\theta_1^{m_1}} w_1 \xrightarrow{e_1} y_2 \cdots \xrightarrow{e_n} z_n^1 \xrightarrow{\theta_n^1} \cdots z_n^{m_n} \xrightarrow{\theta_n^{m_n}} w_n \xrightarrow{e_n} y_{n+1}$, the string generated by ω is defined as: $l(\omega) = \theta_0^1 \theta_0^2 \cdots \theta_0^{m_0-1} \theta_0^{m_0} e_0 \theta_1^1 \cdots \theta_1^{m_1} e_1 \cdots e_{n-1} \theta_n^1 \cdots \theta_n^{m_n} e_n$, where $\forall i \le n$, $\theta_i^{m_i} e_i = \varepsilon$ if $\theta_i^{m_i} = e_i \to \varepsilon$.

Definition 14 (Edit Function Embedded in TPO): [8] Given TPO T, a deterministic edit function f_e is embedded in T, denoted by $f_e \in T$, if $\forall s \in P[\mathscr{L}(G)]$, $\exists \omega \in \Omega_T$, s.t. $P_e(\omega) = s$ and $l(\omega) = f_e(s)$.

In this sequel, the only TPO T we will consider is the *largest* TPO that satisfies the above definition; i.e., all defined transitions are included at each state. This structure is well defined in terms of graph union [8]. After calculating the largest three-player observer T, the next step is to remove *deadlocking* Z-states and W-states, where no transition is defined [8]. Those deadlocking states are due to infeasible insertion choices as well as edit decisions that are not allowed by *edit constraints* [8]. The three-player observer with no deadlocking W or Z states is called *complete*. Since in this paper we do not have any constraint and the edit function is always allowed to erase events in its operation, there are no deadlocking states in the largest three-player observer, which is identical to the All Edit Structure.

Definition 15 (All Edit Structure): [8] Given system G, observer $det(G)$, desired estimator $det_d(G)$, the All Edit Structure (AES) is the largest complete three-player observer.

It is shown in [8] that all the edit functions that satisfy private safety are embedded in the AES. This is analogous to the result in [3] for insertion functions (i.e., no erasures).

Theorem 1: [4], [5] Given a system and its observer, an edit function f_e is privately safe if and only if $f_e \in AES$.

Henceforth, our goal is to build the AES starting from suitable abstractions of G and $det(G)$.

III. ABSTRACTED ALL EDIT STRUCTURE

This section describes the abstraction-based All Edit Structure. We show in Corollary 3 below that AESs of bisimilar observers and desired observers embed the same edit functions. To establish this, Theorem 2 first establishes that bisimilar observers and desired observers have AESs with the same runs.

Theorem 2: Let $G = \langle \Sigma_{tot}, Q, \rightarrow, Q^{\circ} \rangle$, where $\Sigma_{tot} = \Sigma \cup \Sigma_{uo}$, be a nondeterministic automaton with set of secret states $Q^S \subseteq Q$ and set of non-secret states $Q^{NS} = Q \setminus Q^S$. Let H_{det} and H_{des} be two deterministic automata such that $det(G) \approx H_{det}$ and $det_d(G) \approx H_{des}$, where $\approx$ is a bisimulation relation. Let AES be the All Edit Structure of G and let AES' be the All Edit Structure of H_{det} and H_{des}. Then $AES \xrightarrow{\omega} q$ if and only if $AES' \xrightarrow{\omega} \tilde{q}$.

As was reviewed earlier, the AES is the largest three-player observer under our assumption, and it contains all the edit functions that can be used to enforce opacity. The following corollary shows that AESs of bisimilar observers and bisimilar desired observer contain the same edit functions.

Corollary 3: Let $G = \langle \Sigma_{tot}, Q, \rightarrow, Q^{\circ} \rangle$, where $\Sigma_{tot} = \Sigma \cup \Sigma_{uo}$, be a nondeterministic automaton with set of secret states $Q^S \subseteq Q$ and set of non-secret states $Q^{NS} = Q \setminus Q^S$. Let H_{det} and H_{des} be two deterministic automata such that $det(G) \approx H_{det}$ and $det_d(G) \approx H_{des}$, where $\approx$ is a bisimulation relation. Let AES be the All Edit Structure obtained from $det(G)$ and $det_d(G)$ and let and AES' be the All Edit Structure obtained from H_{det} and H_{des}. Then $f_e \in AES$ if and only if $f_e \in AES'$.

Proof ($\Rightarrow$) Assume $f_e \in AES$. From $f_e \in AES$ it holds that $\exists s \in \mathscr{L}(G)$, $\exists \omega \in AES$ such that $P_e(\omega) = s$ and $l(\omega) = f_e(s)$. Then based on Theorem 2 it holds that $\omega \in AES'$ and $P_e(\omega) = s$ and $l(\omega) = f_e(s)$. Thus, $f_e \in AES'$.

($\Leftarrow$) The same argument as ($\Rightarrow$) holds. ■

The following example is provided to clarify how the AES is calculated. It will be re-used later.

Example 1: Consider automaton G with secret state $Q^S = 2$, $\Sigma = \{\alpha, \beta, \gamma\}$ and $\Sigma_{uo} = \{\tau\}$. Automaton G and its observer are shown in Fig. 1. The system is not current-state opaque as by executing event γ the intruder will know that system is in the secret state 2. Thus, an edit function needs to be calculated to enforce opacity. State 2 in $det(G)$ is violating current-state opacity, $\{2\} \subseteq Q^S$, and it is removed when calculating the desired observer, $det_d(G)$. After removing $\{2\}$ state $\{4\}$ becomes unreachable and should also be removed. The desired observer is shown in Fig. 1. The next step is to calculate the largest three-player observer, which is shown in Fig. 1. In the figure Y, Z and W states are shown by rectangular, oval and diamond, respectively. For simplicity

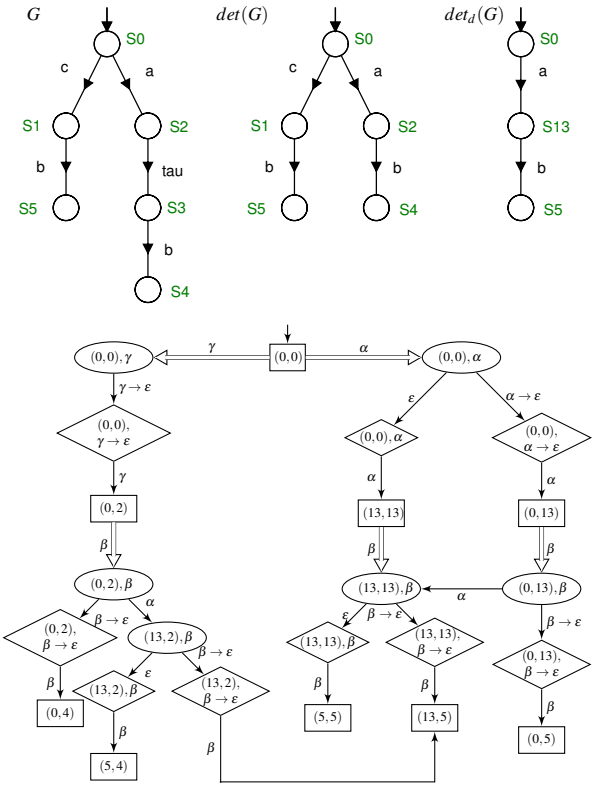

Fig. 1. Automata of Example 1 and its corresponding AES.

the observer states in the three-player observer are shown by their elements, i.e, 0 refers to the state $\{0\}$, 13 refers $\{1,3\}$, etc. The initial state of the three-player observer is $y_0 = (0,0)$. At $(0,0)$ the dummy player execute event γ or α. After executing α, Z state $((0,0),\alpha)$ is reached. At state $((0,0),\alpha)$ the edit function can either erase event α, where the W state $((0,0),\alpha \rightarrow \varepsilon)$ can be reached, or take no action, where the W state $((0,0),\alpha)$ is reached. When the system executes event α at the W state $((0,0),\alpha)$ the Y state $(13,13)$ is reached. The whole structure is interpreted in a similar way. The AES is the same as the calculated largest three-player observer T and can be observed that all the Z and W states have outgoing transitions.

IV. OPAQUE OBSERVATION EQUIVALENCE

In the previous section it was shown that bisimilar observers and desired observers produce AESs with the same edit functions. Following the results of Sect. III, in this section abstraction methods are introduced to abstract the system such that the observer and the desired observer of the abstracted system are bisimilar to their original counterparts. The abstraction methods are based on bisimulation and observation equivalence, which are computationally efficient and can be calculated in polynomial-time. In order to use observation equivalence for abstraction in the setting of opacity, the secrecy status of states needs to be considered. In the following, a restricted version of observation equivalence called *opaque observation equivalence* is defined.

Definition 16: Let $G = \langle \Sigma_{tot}, Q, \rightarrow, Q^{\circ} \rangle$, where $\Sigma_{tot} =$

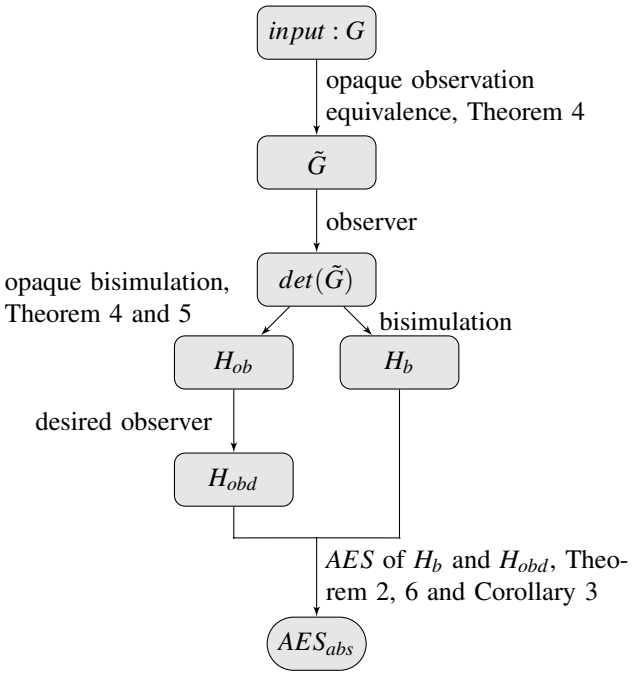

Fig. 2. The steps of calculating an abstracted AES of system G.

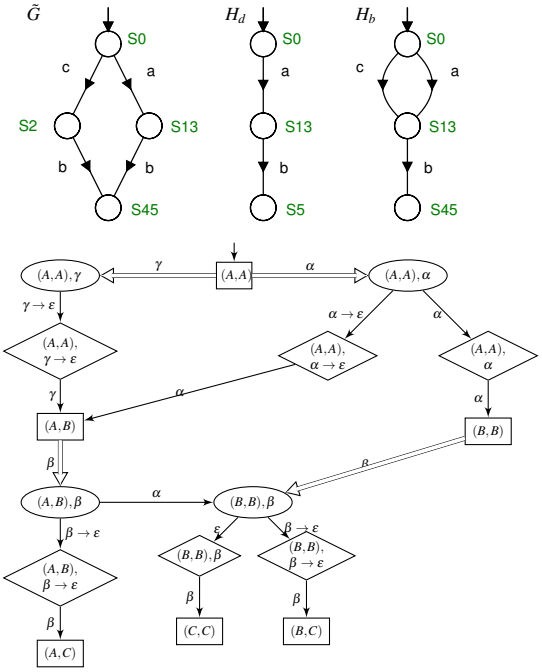

Fig. 3. Automata of Examples 2-3.

$\Sigma \dot\cup \Sigma_{uo}$, be a nondeterministic automaton with set of secret states $Q^S \subseteq Q$ and set of non-secret states $Q^{NS} = Q \setminus Q^S$. An equivalence relation $\sim_o \subseteq Q \times Q$ is called an *opaque observation equivalence* on G with respect to Q^S, if the following holds for all $x_1, x_2 \in Q$ such that $x_1 \sim_o x_2$:

(i) if $x_1 \stackrel{s}{\Rightarrow} y_1$ for some $s \in \Sigma^*$, then there exists $y_2 \in Q$ such that $x_2 \stackrel{s}{\Rightarrow} y_2$, and $y_1 \sim_o y_2$,
(ii) $x_1 \in Q^S$ if and only if $x_2 \in Q^S$.

In this paper bisimulation is used to abstract the observer of a nondeterministic system. Similarly to opaque observation equivalence, *opaque bisimulation* is defined.

Definition 17: Let $G = \langle \Sigma_{tot}, Q, \to, Q^\circ \rangle$, where $\Sigma_{tot} = \Sigma \dot\cup \Sigma_{uo}$, be a nondeterministic automaton with set of secret states $Q^S \subseteq Q$ and set of non-secret states $Q^{NS} = Q \setminus Q^S$. Let $det(G) = \langle \Sigma, X_{obs}, \to_{obs}, X^\circ_{obs} \rangle$ be the observer of G. An equivalence relation $\approx_o \subseteq X_{obs} \times X_{obs}$ is called an *opaque bisimulation equivalence* on $det(G)$ with respect to Q^S, if the following holds for all $X_1, X_2 \in X_{obs}$ such that $X_1 \approx_o X_2$:

(i) if $X_1 \stackrel{s}{\to}_{obs} Y_1$ for some $s \in \Sigma^*$, then there exists $Y_2 \in X_{obs}$ such that $X_2 \stackrel{s}{\to}_{obs} X_2$, and $Y_1 \approx_o Y_2$,
(ii) $X_1 \subseteq Q^S$ if and only if $X_2 \subseteq Q^S$.

Fig. 2 gives an overview of the methodology to construct an abstraction-based AES. The input to the algorithm is a nondetermistic automaton G. The algorithm first abstracts G using opaque observation equivalence. This results in $\tilde{G}$, which has fewer (or the same) states and transitions as compared with G. Since the computational complexity of calculating the observer of G is 2^Q, merging states can potentially reduce the complexity significantly. Next, opaque bisimulation and bisimulation are applied to the observer of $\tilde{G}$, $det(\tilde{G})$, resulting in abstracted deterministic automata H_{ob} and H_b, respectively. Next, H_{ob} is used to calculate the desired observer of the abstracted system, H_{obd}. The final step is to calculate the AES from the abstracted observer H_b and the abstracted desired observer H_{obd}. It will be shown that the abstracted AES embeds all the possible edit functions that enforce opacity of the system.

The first step of the abstraction-based AES algorithm is to abstract the system using opaque observation equivalence. It has been shown in [11] that if two automata are bisimilar, then their observers are also bisimilar. In this paper this result is extended: abstracting a nondeterministic automaton using opaque observation equivalence results in an observer and a desired observer which are bisimilar to the original system's observer and the desired observer, respectively.

Theorem 4: Let $G = \langle \Sigma_{tot}, Q, \to, Q^\circ \rangle$, where $\Sigma_{tot} = \Sigma \dot\cup \Sigma_{uo}$, be a nondeterministic automaton with set of secret states $Q^S \subseteq Q$ and set of non-secret states $Q^{NS} = Q \setminus Q^S$. Let $\sim_0$ be an opaque observation equivalence on G resulting in $\tilde{G}$ and let $\approx$ be a bisimulation. Let $det_d(G)$ and $det_d(\tilde{G})$ be the desired observer of G and $\tilde{G}$. Then $det(G) \approx det(\tilde{G})$ and $det_d(G) \approx det_d(\tilde{G})$.

The need for using opaque observation equivalence and considering the secrecy status of the merged states is essential to guarantee bisimilarity between the abstracted desired observer and the original desired observer.

Example 2: Consider automaton G with set of secret states $Q^S = \{2\}$ and $\Sigma = \{\alpha, \beta, \gamma\}$ and $\Sigma_{uo} = \{\tau\}$, shown in Fig. 1. States 1 and 3 are opaque observation equivalent as they are both non-secret states and state 5 can be reached from both by ignoring the unobservable event τ. Thus, 1 and 3 can be merged. A similar argument holds for states 4 and 5. Merging the equivalent states results in $\tilde{G}$ shown in Fig. 3. Since $\tilde{G}$ is deterministic, then $det(\tilde{G})$ is isomorphic to $\tilde{G}$. Since $\{2\} \subseteq \tilde{Q}^S$ it should be removed when calculating

$det_d(\tilde{G})$. The desired observer $det_d(\tilde{G})$ is shown in Fig. 3 as H_d. It can be observed that H_d is bisimilar, in fact isomorphic, to $det_d(G)$, shown in Fig. 1.

Opaque observation equivalence seeks to merge the states of a nondeterministc automaton before the construction of the observer. After calculating the observer it is possible to abstract the observer further using opaque bisimulation. This can guarantee construction of the smallest observer that generates the same language as the original observer. Then Theorem 5 shows that if opaque bisimulation is used to abstract the observer of an automaton then the abstracted desired observer is bisimilar to the original desired observer.

Theorem 5: Let $G = \langle \Sigma_{tot}, Q, \rightarrow, Q^{\circ} \rangle$, where $\Sigma_{tot} = \Sigma \dot{\cup} \Sigma_{uo}$, be a nondeterministic automaton with set of secret states $Q^S \subseteq Q$ and set of non-secret states $Q^{NS} = Q \setminus Q^S$. Let $\approx_o$ be an opaque bisimulation on $det(G)$ resulting in $\widetilde{det(G)}$. Let $\widetilde{det_d(G)}$ and H_d be the desired observers of $det(G)$ and $\widetilde{det(G)}$, respectively. Then $det_d(G) \approx H_d$, where $\approx$ is a bisimulation relation.

Proof Since $det(G) \approx_o \widetilde{det(G)}$, based on Def. 17 it holds that $det(G) \xrightarrow{s}_{obs} X$ if and only if $\widetilde{det(G)} \xrightarrow{s} [X']$ and $X \in [X']$. Thus, it is enough show that $X \notin X_{obs,det_d(G)}$ if and only if $[X'] \notin X_{obs,H_d}$, where $X \in [X']$.

First assume $X \subseteq Q^S$, which mean $X \notin X_{obs,det_d(G)}$. Then since $X \in [X']$ based on Def. 17 it holds that $\forall X' \in [X']$, $X' \subseteq Q^S$. This means $[X'] \subseteq Q^S$ and consequently $[X'] \notin X_{obs,H_d}$.

Now assume $[X'] \subseteq Q^S$, which mean $[X'] \notin X_{obs,H_d}$. Then since $X \in [X']$ based on Def. 17 it holds that $X \subseteq Q^S$. This means $X \notin X_{obs,det_d(G)}$. ■

So far, we have shown that after opaque observation equivalence, the resulting observer and desired observer are bisimilar to the original observer and desired observer, respectively. Moreover, in Theorem 5, it was proven that opaque bisimulation can be used to abstract an observer and then the abstracted desired observer will be bisimilar to the original desired observer. Therefore, based on Corollary 3, it can be established that opaque observation equivalence and opaque bisimulation can be used to abstract a nondeterministic system and its observer to reduce computational complexity. This result is shown in the following theorem.

Theorem 6: Let G be a nondeterministic automaton with set of secret states Q^S. Let $det(G)$ and $det_d(G)$ be the observer and the desired observer of G, respectively, and let AES be the All Edit Structure of G. Let $\sim_o$ be an opaque observation equivalence on G resulting in $\tilde{G}$. Let $\approx_o$ and $\approx$ be opaque bisimulation and bisimulation on $det(\tilde{G})$, resulting in H_{ob} and H_b, respectively, such that $H_{ob} \approx_o det(\tilde{G})$ and $H_b \approx det(\tilde{G})$. Let H_{obd} be the desired observer obtained from H_{ob} and let AES_{abs} be the All Edit Structure obtained from H_{obd} and H_b. Then $f_e \in AES$ if and only if $f_e \in AES_{abs}$.

Proof The proof follows directly from Theorem 4 and 5 in combination with Corollary 3.

Example 3: Consider automaton G with set of secret states $\tilde{Q}^S = \{2\}$, $\Sigma = \{\alpha, \beta, \gamma\}$ and $\Sigma_{uo} = \{\tau\}$, shown in Fig. 1. As it was shown in Example 2, G can be abstracted using opaque observation equivalence. The resulting abstracted automaton $\tilde{G}$ is shown in Fig. 3. The next step is to calculate the observer of $\tilde{G}$. Since $\tilde{G}$ is determinisitic $det(\tilde{G})$ is isomorphic to $\tilde{G}$. Next, opaque bisimulation is applied to $det(\tilde{G})$, resulting in automaton H_{ob}. States 2 and $[3]$ are bisimilar but not opaque bisimilar as $[3] \not\subseteq \tilde{Q}^S$ and $2 \subseteq \tilde{Q}^S$. Thus, H_{ob} and $det(\tilde{G})$ are the same. Next, the desired observer H_{obd} is obtained by removing state 2 from H_{ob}. H_{obd} is shown in Fig. 3 as H_d. After calculating the desired observer automaton, $det(\tilde{G})$ can be abstracted further using bisimulation, resulting in H_b, shown in Fig. 3. The final step is to calculate AES_{abs} from H_{obd} and H_b. Fig. 3 shows AES_{abs}, and Fig. 1 shows AES, which is the All Edit Structure of the original system. Comparing AES and AES_{abs}, it can be observed that AES_{abs} and AES embed the same edit functions while AES has 24 states and AES_{abs} has 16 states.

V. CONCLUSION

We investigated abstraction-based synthesis of edit functions for opacity enforcement based on the All Edit Structure. The AES is a game-like structure that embeds all valid edit functions that can be used to make a non-opaque system opaque. To mitigate the computational complexity of constructing the AES, we defined two bisimulation-based abstractions that account for the secrecy status of states: opaque observation equivalence and opaque bisimulation. We presented a methodology that employs these abstractions to reduce the size of the original system model and of its observer, in the process of synthesizing a reduced AES that still embeds all valid edit functions. It would be of interest to investigate in the future how abstraction methods should be adapted when there are constraints on edit functions that arise when synthesizing the AES. Moreover, developing scalable benchmarks to quantify the performance gains from these abstraction methods is also of interest.

REFERENCES

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

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

[3] 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.

[4] Y. Ji and S. Lafortune, "Enforcing opacity by publicly known edit functions," in *56th '17*, Dec. 2017, pp. 4866–4871.

[5] 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, pp. 107–131, 2018.

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

[7] Y. Ji, X. Yin, and S. Lafortune, "Opacity enforcement by insertion functions under energy constraints," in *Proceedings of the 14th International Workshop on Discrete Event Systems*, 2018, pp. 291–297.

[8] ——, "Opacity enforcement using nondeterministic publicly-known edit functions," *IEEE Trans. on Auto. Control*, under review, 2018.

[9] R. Milner, *Communication and concurrency*, 1989.

[10] K. Zhang and M. Zamani, "Infinite-step opacity of nondeterministic finite transition systems: A bisimulation relation approach," in *56st 2017*, Dec. 2017, pp. 5615–5619.

[11] J. Rutten, "Automata and coinduction (an exercise in coalgebra)," in *9th '98*, vol. 1466, 1998, pp. 194–218.