Preprints of WODES 2018
May 30 - June 1, 2018
Sorrento Coast, Italy

Opacity Enforcement by Insertion Functions under Energy Constraints ⋆

Yiding Ji * **Xiang Yin** ** **Stéphane Lafortune** *

* *Department of Electrical Engineering and Computer Science, the University of Michigan, Ann Arbor, Michigan, USA.*
** *Department of Automation, Shanghai Jiao Tong University, Shanghai, China.*

Abstract:
We investigate the enforcement of opacity by insertion functions, when the system has a certain amount of initial-credit energy for its operation and defense of secrets. The problem is formulated as a two-player game between the insertion function and the system (or environment) with an energy objective and asymmetric partial information. The insertion function must defend the secrets by inserting fictitious output events while guaranteeing that the energy level never drops below zero, under the worst-case scenario of system operation. The insertion function has only partial information about the system, due to unobservable events that affect the energy level. To resolve the partial observation, we construct a game structure called the *Energy Insertion Structure* (or EIS) that provably embeds insertion functions solving the proposed problem.

Keywords: Privacy, opacity enforcement, insertion function, partial observation, energy game

1. INTRODUCTION

Opacity is an information-flow property that characterizes whether the secrets of a system can be inferred by an outside intruder with malicious goals. The outside intruder is typically modeled as an observer with knowledge of the structure of the system whose intention is to infer system secrets when observing the system outputs. The system is called opaque if the intruder is never able to determine any of the system secrets *unambiguously* from its observations.

Since the work by Bryans et al. [2005], opacity has received significant attention for systems modeled as Discrete Event Systems (DES). Several opacity notions have been defined and studied for finite state automata models, e.g., language-based opacity [Lin 2011], current-state opacity [Saboori and Hadjicostis 2007], initial-state opacity [Saboori and Hadjicostis 2013], K-step opacity, and infinite-step opacity [Yin and Lafortune 2017]. Opacity has also been evaluated quantitatively in stochastic settings, e.g., [Keroglou and Hadjicostis 2017], and timed settings, e.g., [Cassez 2009]. The survey paper Jacob et al. [2016] provides a comprehensive review of opacity results in DES.

Violations of opacity give rise to the opacity enforcement problem [Falcone and Marchand 2015], which has been investigated under various mechanisms. Supervisory control can be used to disable non-opaque behaviors, thereby preventing disclosure of secrets [Dubreil et al. 2010, Tong et al. 2017, Takai and Oka 2008, Yin and Lafortune 2016]. Another method is sensor activation [Cassez et al. 2012, Yin and Lafortune 2015], which dynamically changes the observability of certain events but does not intervene with the system's operation. Opacity enforcement using insertion functions was proposed by Wu and Lafortune [2014] and extended in Ji and Lafortune [2017] to a more general method called edit functions. An edit function may insert fictitious events into the system's output or erase events from the system's output to obfuscate the intruder. In this paper, we consider obfuscation by event insertion alone.

In many applications, the execution of system events as well as the obfuscation method may consume quantitative resources of the system, which we refer to as *energy*. We assume that the system may only have a limited amount of energy for its operation and for the defense of its secrets. Motivated by this practical situation, we investigate for the first time opacity enforcement by insertion functions under energy constraints, which require that the energy of the system should never be depleted. Furthermore, we assume that the insertion function is only aware of the occurrence of observable events.

We formulate this problem as opacity enforcement with a quantitative objective and imperfect information. The insertion function aims to enforce opacity under the constraint that the energy of the system should never drop below zero, for all possible system behaviors (worst-case analysis). Then we reduce this problem to a game with perfect information and solve it by constructing a discrete structure called Energy Insertion Structure (EIS). The insertion function plays by inserting events, which consumes energy, while the system plays by executing events, which consumes or gains energy. Therefore the system's energy level dynamically changes. The EIS is a game graph including winning strategies of the insertion function under both qualitative and quantitative requirements.

Our approach is inspired by recent work on energy games, which are two-player quantitative games on weighted

⋆ Research supported in part by the US National Science Foundation under grants CNS-1421122 and CNS-1738103.

graphs, where the weights represent energy gain or consumption. The objective of the first player is to keep the energy above zero, while the other player intends to do the opposite. A special type of energy game is a fixed-initial-credit energy game, where the system has a certain amount of initial energy. In some cases, the first player may have imperfect information about the game [Degorre et al. 2010]. Under certain assumptions, such games are decidable but they are ACK-complete [Pérez 2017]. They are solved by reducing them to reachability games [Degorre et al. 2010, de Alfaro et al. 2007]. The above works have also inspired the work Pruekprasert and Ushio [2017], which studies supervisory control for DES using energy games with partial observation. We adapt some of the methodology in Pruekprasert and Ushio [2017] to the different problem of opacity enforcement by obfuscation, leveraging the approach of Wu and Lafortune [2014]. We believe this paper is the first to investigate opacity enforcement under such types of quantitative energy constraints.

This paper is organized as follows. Section 2 describes our system model. Section 3 formulates the opacity enforcement problem under energy constraints discussed in this paper. Section 4 introduces the Energy Insertion Structure (EIS). Section 5 uses the EIS to solve the proposed problem. And Section 6 concludes the paper.

2. SYSTEM MODEL

We consider opacity in a quantitative DES modeled as a weighted finite-state automaton:

$$G = (X, E, f, x_0, \omega)$$

where X is the finite set of states, E is the finite set of events, $f : X \times E \to X$ is the partial state transition function, and $x_0 \in X$ is the unique initial state. We denote by $X_S \subset X$ the set of *secret* states that should remain opaque. The transition function is extended to domain $X \times E^*$ in the standard manner [Cassandras and Lafortune 2008] and we still denote it by f. The language generated by G is defined as $\mathcal{L}(G) = \{s \in E^* : f(x_0, s)!\}$ where ! means "is defined". The function $\omega : E \to \mathbb{Z}$ assigns a weight to each event in E. The value of the weight reflects the energy gain or cost associated with the occurrence of the event. The function ω is additive and its domain can be extended to E^* by letting $\omega(\epsilon) = 0$, $\omega(se) = \omega(s) + \omega(e)$ where $s \in E^*$, $e \in E$. An *execution* in G is a sequence of states and events: $x_0 e_0 x_1 e_1 \cdots e_n x_n$ where $x_i = f(x_{i-1}, e_i), 1 \leq i \leq n$. An execution contains a *cycle* if $\exists i, j \in \mathbb{N}$, $0 \leq i < j \leq n$, s.t. $x_i = x_j$.

The system has initial-credit energy $v_0 \in \mathbb{N}^+$ and its energy changes with the occurrence of events. Given a string $s = e_0 e_1 \cdots e_{n-1} \in \mathcal{L}(G)$, the *energy level of the system after* s is defined as $V : \mathcal{L}(G) \to \mathbb{Z}$ where $V(s) = v_0 + \sum_{i=0}^{n-1} \omega(e_i)$. In this work, we make the important assumption that the energy level should always be nonnegative.

Assumption 1. $\forall s \in \mathcal{L}(G)$, $V(s) \geq 0$.

We assume that G is partially observable, i.e., $E = E_o \cup E_{uo}$, where E_o is the set of observable events and E_{uo} is the set of unobservable events. Given $t = t'e \in E^*$, its natural projection under $P : E^* \to E_o^*$ is recursively defined as $P(t) = P(t'e) = P(t')P(e)$ where $t' \in E^*$ and $e \in E$. The projection of an event is $P(e) = e$ if $e \in E_o$ and $P(e) = \epsilon$ if $e \in E_{uo} \cup \{\epsilon\}$, where ϵ is the empty string. Then the *observer* of G is defined as $Obs(G) = (X_{obs}, E_o, \delta, x_{obs,0}, \omega_{obs})$ and is obtained following the standard technique in Cassandras and Lafortune [2008]. Here X_{obs} is the state space, δ is the transition function, $x_{obs,0}$ is the initial state and ω_{obs} is the same as ω over the restricted domain E_o. We call the observer state as the *current state estimate (estimate)* of the system.

3. OPACITY AND INSERTION MECHANISM

In this section, we formulate the opacity enforcement problem under energy constraints. We start by reviewing the concept of current-state opacity and then discuss the insertion mechanism for its enforcement.

Definition 1. (Current-State Opacity (CSO)). Given system G, projection P, and secret states X_S, G is CSO if $\forall t \in L_S := \{t \in \mathcal{L}(G) : f(x_0, t) \in X_S\}$, $\exists t' \in L_{NS} := \{t \in \mathcal{L}(G) : f(x_0, t) \in (X \setminus X_S)\}$ such that $P(t) = P(t')$.

A system is current-state opaque if for every string reaching a secret state, there exists another string reaching a non-secret state which shares the same projection, thereby providing deniability of the secret. CSO can be verified by building the observer and checking whether an observer state contains solely secret states. Based on CSO, we define the *safe language*, which is the prefix-closure of the projected non-secret strings: $L_{safe} = [P[\mathcal{L}(G)] \setminus [P[\mathcal{L}(G)] \setminus P(L_{NS})]]E_o^*$. We also define the *unsafe language* $L_{unsafe} = P[\mathcal{L}(G)] \setminus L_{safe}$. The *desired observer* $Obs_d(G) = (X_d, E_o, \delta_d, x_{d,0})$ is obtained by deleting all the observer states composed of only secret states and then taking the accessible part, see Wu and Lafortune [2014]. It generates L_{safe} and we omit the weight function.

Opacity may not always hold and an *insertion function* may be used to enforce it. The insertion function is an interface between the system's output and the external environment including the intruder. It may insert fictitious events into the output stream of the system to obfuscate the intruder, see Wu and Lafortune [2014] for more details.

Definition 2. (Insertion Function). An insertion function is defined as: $f_i : E_o^* \times E_o \to E_o^* E_o$ such that for $l \in E_o^*$ and $e_o \in E_o$, $f_i(l, e_o) = s_I e_o$ where $s_I \in E_o^*$,

By definition, s_I is the inserted string and may be ϵ when nothing is inserted. With a slight abuse of notation, we also define f_i in a string-based manner: $f_i(\epsilon) = \epsilon$ and $f_i(le_o) = f_i(l, e_o) f_i(l)$. An insertion function inserts strings based on the observable behavior. However, unobservable events do occur between two observable events; as a convention, we assume that the inserted string is placed right before the next observable event in an unprojected string.

Convention 1. Given $s = \xi_0 e_0 \cdots \xi_{n-1} e_{n-1} \xi_n \in \mathcal{L}(G)$ where $\forall i \leq n$, $\xi_i \in E_{uo}^*$ and $e_i \in E_o$, if $f_i(e_0 e_1 \cdots e_{i-1}, e_i) = \theta_i e_i$ where $\forall i \leq n$, $\theta_i \in E_o^*$, then s is mapped to $s' = \xi_0 \theta_0 e_0 \cdots \xi_i \theta_i e_i \cdots \xi_n \theta_n e_n$ where $P(s') \in P[\mathcal{L}(G)]$.

Here it is possible that $s' \notin \mathcal{L}(G)$ but only $P(s') \in P[\mathcal{L}(G)]$ since the intruder only observes strings in $P[\mathcal{L}(G)]$.

Based on L_{safe}, we define *private safety* of the insertion function, which characterizes its performance.

Definition 3. (Private Safety). Consider system G with P, and L_{safe}. Insertion function f_i is privately safe if $\forall s \in P[\mathcal{L}(G)]$, $f_i(s) \in L_{safe}$.

Event insertion always costs energy and we define the *insertion cost function* $\omega_i : E_o \to \mathbb{Z} \setminus \mathbb{N}$, which assigns a negative weight to each inserted event. Function ω_i is additive and its domain is extended to E_o^* by letting $\omega_i(\epsilon) = 0$ and $\omega_i(se_o) = \omega_i(s)+\omega_i(e_o)$ for $s \in E_o^*$, $e_o \in E_o$.

Next, we define the *system's energy level after insertion* as $V_m : \mathcal{L}(G) \times E^* \to \mathbb{Z}$. Given $s = \xi_0 e_0 \xi_1 e_1 \cdots \xi_{n-1} e_{n-1} \xi_n \in \mathcal{L}(G)$ where $\forall j \leq n$, $\xi_j \in E_{uo}^*$ and $e_j \in E_o$, suppose s is mapped to $s' = \xi_0 \theta_0 e_0 \xi_1 \theta_1 e_1 \cdots \xi_{n-1} \theta_{n-1} e_{n-1} \xi_n$ by Convention 1 under some insertion function, then $V_m(s, s') = V(s) + \sum_{j=0}^{n-1} \omega_i(\theta_j)$. We will denote s' by s_{f_i} if s is mapped to s' by f_i. We call $V_m(s, s_{f_i})$ as the energy level of the system by string s under insertion function f_i.

Given a non-opaque system G and initial energy v_0, we aim to design an insertion function f_i which enforces opacity but never makes the system's energy level below zero at any time. Thus the operation of the insertion function is constrained by the energy level of the system, i.e., $\forall s \in P[\mathcal{L}(G)]$, $V_m(s, s_{f_i}) \geq 0$. Since insertion costs energy, we make Assumption 1 to guarantee an energy margin for the insertion function. We can now formally formulate the opacity enforcement problem under energy constraints.

Problem 1. (Opacity Enforcement with Energy Constraints). Given system G with initial-credit energy v_0, the opacity enforcement under energy constraints problem is to find an insertion function f_i such that: (1) f_i is privately safe; (2) $\forall s \in \mathcal{L}(G)$, $V_m(s, s_{f_i}) \geq 0$.

Due to the partial observation of the system, we need to properly estimate both the system's current state and the energy level so that the insertion function may make decisions. We will discuss this issue in the next section.

4. ENERGY INSERTION STRUCTURE

In this section, we propose *energy information states* and a bipartite game structure called "Energy Insertion Structure" (EIS). In this way, Problem 1 is transformed into a reachability game of perfect information between the insertion function and the environment. The construction is of the EIS a three-step process: (1) build the *verifier*; (2) build the *safe energy verifier*; (3) build the EIS.

4.1 Build the Safe Energy Verifier

Step (1) was first presented in Wu and Lafortune [2014] and we briefly repeat it here. We first build the desired estimator and the feasible estimator. The feasible estimator is obtained by adding self-loops at each state in $Obs(G)$ for all observable events. We denote the feasible estimator $Obs_f(G) = (X_f, E_o, \delta, \delta_{sl}, x_{f0})$ where $\delta_{sl}(x_f, e_o) = x_f$ for every $x_f \in X_f$ and every $e_o \in E_o$. Thus at a state x_f, there may be two e_o defined, one is for the normal transition δ and the other is for self-loop transition δ_{sl}, where δ stands for the event execution in the observer while δ_{sl} stands for the occurrence of inserted fictitious events.

Then we synchronize $Obs_d(G)$ and $Obs_f(G)$ by the *verifier parallel composition* to obtain the *verifier*, defined as $G_v = (X_v, E_o, \delta_{vd}, \delta_{vs}, x_{v0})$. Here $X_v \subseteq X_d \times X_f$ is the state space, E_o is the set of observable events, $\delta_{vs} : X_v \times E_o \to X_v$ is the transition function corresponding to normal transitions in both $Obs_d(G)$ and $Obs_f(G)$, $\delta_{vd} : X_v \times E_o \to X_v$ is the transition function corresponding to normal transitions in $Obs_d(G)$ with added self-loop transitions in $Obs_f(G)$, and x_{v0} is the initial state. A state $x_v = (x_d, x_f) \in X_v$ has two components: the left one is the intruder's estimate and the right one is the (true) system's estimate. By construction (see Wu and Lafortune [2014]), x_d is not a subset of secret states and does not reveal the system's secrets to the intruder. The two transition functions δ_{vs} and δ_{vd} work as follows from the verifier parallel composition: $\delta_{vs}((x_d, x_f), e_o) = (\delta_d(x_d, e_o), \delta(x_f, e_o))$ if $\delta_d(x_d, e_o)!$ in $Obs_d(G)$ and $\delta(x_f, e_o)!$ in $Obs_f(G)$; $\delta_{vd}((x_d, x_f), e) = (\delta_d(x_d, e), \delta_{sl}(x_f, e)) = (\delta_d(x_d, e), x_f)$ if $\delta_d(x_d, e)!$ in $Obs_d(G)$. We set $\delta_{vd}(x_v, \epsilon) = x_v$ for any $x_v \in X_v$. Intuitively, δ_{vd} indicates potential event insertion and δ_{vs} indicates system's observable event execution.

Next, we come to step (2). Before building the safe energy verifier, we first introduce a measure on vectors. Given two vectors $v_1 = [v_1(1), v_1(2), \cdots, v_1(n)]$, $v_2 = [v_2(1), v_2(2), \cdots, v_2(n)] \in \mathbb{Z}^n$, we denote by $v_1 \leq v_2$ (respectively $v_1 \geq v_2$) if $\forall 1 \leq i \leq n, v_1(i) \leq v_2(i)$ (respectively $v_1(i) \geq v_2(i)$). In order to cope with partial observation as well as to track the system's energy level, we define the *Energy Information State* as follows.

Definition 4. (Energy Information State). An energy information state is: $q^e = ((x_d, x_f), [v(1), \cdots v(|x_f|)]) \in X_v \times \mathbb{Z}^{|x_f|}$. Let $I(q^e)$ and $E_L(q^e)$ denote the verifier state and energy level components; hence, $q^e = (I(q^e), E_L(q^e))$.

Denote by Q^E the set of energy information states, which tracks the system's estimate, the intruder's estimate, and the system's energy level. Each $q^e \in Q^E$ induces a *belief function* $h_{q^e} : X \to \mathbb{Z}$. Specifically, for $q^e \in Q^E$ where $I(q^e) = (x_d, x_f) \in X_v$, $E_L(q^e) = \{h_{q^e}(x) : x \in x_f\}$. We usually put $E_L(q^e)$ in vector form: $[h_{q^e}(x_1), \cdots h_{q^e}(x_{|x_f|})]$. By convention, elements in $E_L(q^e)$ are placed in an increasing order w.r.t. state names in x_f. Our definition is inspired by the belief function in Degorre et al. [2010] and the observation function in Pruekprasert and Ushio [2017].

An energy information state $q^e \in Q^E$ is *energy safe* (or simply *safe*) if $E_L(q^e) \geq 0$ in the point-wise sense. We define an order $\preccurlyeq$ over Q^E: for $q_1^e, q_2^e \in Q^E$, $q_1^e \preccurlyeq q_2^e$ if $I(q_1^e) = I(q_2^e)$ and $E_L(q_1^e) \leq E_L(q_2^e)$. We also say that q_2^e *subsumes* q_1^e if $q_1^e \preccurlyeq q_2^e$, i.e., q_1^e and q_2^e share the same verifier state component but the energy level of q_2^e is no less than that of q_1^e at every possible current state in $I(q_2^e)$. By Dickson's lemma (see Levy [2002]), the order $\leq$ on $\mathbb{N}^k$ is a *well-quasi-ordering* for any $k \in \mathbb{N}$. We further argue that $\preccurlyeq$ on safe energy information states is also a well-quasi-ordering, i.e., for any infinite sequence of states $q_1^e, q_2^e \cdots \in Q^E$, there exists $i, j \in \mathbb{N}$, s.t. $i < j$ and $q_i^e \preccurlyeq q_j^e$.

For $e_o \in E_o$, we say q_2^e is a (e_o, δ_{vd}) *successor* of q_1^e if:

- $I(q_2^e) = (x_{d2}, x_{f1}) = \delta_{vd}(I(q_1^e), e_o)$ where $I(q_1^e) = (x_{d1}, x_{f1})$
- $\forall x \in x_{f1}$, $h_{q_2^e}(x) = h_{q_1^e}(x) + \omega_i(e_o)$

When e_o is inserted, the energy information state is updated from q_1^e to q_2^e. Meanwhile, the system's estimate

x_{f1} does not change and the intruder's estimate changes from x_{d1} to x_{d2}. Besides, the value of $h_{q_1^e}(x)$ decreases by the absolute value of $\omega_i(e_o)$ for every $x \in x_{f1}$ since the insertion of e_o costs energy $\omega_i(e_o)$. We also extend this definition to (θ, δ_{vd}) *successor* for $\theta \in E_o^*$: q_2^e is a (θ, δ_{vd}) successor of q_1^e if $I(q_2^e) = (x_{d2}, x_{f1}) = \delta_{vd}(I(q_1^e), \theta)$ and $\forall x \in x_{f1}$, $h_{q_2^e}(x) = h_{q_2^e}(x) + \omega_i(\theta)$.

Similarly, we say q_2^e is a (e_o, δ_{vs}) *successor* of q_1^e if:

- $I(q_2^e) = (x_{d2}, x_{f2}) = \delta_{vs}(I(q_1^e), e_o)$ where $I(q_1^e) = (x_{d1}, x_{f1})$
- $\forall x' \in x_{f2}, h_{q_2^e}(x') = \min_{\xi \in E_{uo}^*} \{h_{q_1^e}(x) + \omega(e_o) + \omega(\xi) : \exists x \in x_{f1}, \xi \in E_{uo}^*, \text{ s.t. } f(x, e_o\xi) = x'\}$

When e_o is executed by the system, the energy information state is updated from q_{e1} to q_2^e. Meanwhile, both the system's and the intruder's estimates may change. A possible current state x' in x_{f2} may be reached through different strings from some state(s) x in x_{f1}. In this case, $h_{q_2^e}(x')$ indicates the system's *worst case* energy at x' with the occurrence of observable event e_o and unobservable string ξ from some $x \in x_{f1}$ s.t. $x' = f(x, e_o\xi)$.

Then we define the *safe energy verifier* that only contains safe energy information states following Algorithm 1. It is denoted by $G_{ev}^s = (X_{ev}, E_o, \delta_{evs}, \delta_{evd}, x_{ev0}, v_0)$ where $X_{ev} \subseteq Q^E$ is the state space, $\delta_{evs} : X_{ev} \times E_o \to X_{ev}$ is the observable event insertion transition function, $\delta_{evd} : X_{ev} \times E_o \to X_{ev}$ is the event execution transition function, x_{ev0} is the initial state, and v_0 is the initial energy. Procedure $Energy - Unfold$ builds the state space recursively until an unsafe energy information state is reached or an energy information state subsumes an existing one.

Algorithm 1 Construct the Safe Energy Verifier

Input: G_v, v_0, $Obs(G)$
Output: $G_{ev}^s = (X_{ev}, E_o, \delta_{evs}, \delta_{evd}, x_{ev0}, v_0)$

1: $X_{ev} = \{x_{ev0}\}$, $I(x_{ev0}) = (x_{obs,0}, x_{obs,0})$, $\forall x \in x_{obs,0}$, $h_{x_{ev0}}(x) = \min_{\xi \in E_{uo}^*} \{V(\xi) : \exists \xi \in E_{uo}^*, f(x_0, \xi) = x\}$;
2: $G_{ev}^s = Energy - Unfold(x_{ev0}, G_v)$;
3: **procedure** ENERGY-UNFOLD(x_{ev}, G_v)
4: **for** $e \in E_o$, s.t. $\delta_{vd}(I(x_{ev}), e_0)!$ or $\delta_{vs}(I(x_{ev}), e_0)!$ where $I(x_{ev}) = (x_d, x_f)$ **do**
5: **if** $\delta_{vd}(I(x_{ev}), e_0)!$ **then**
6: let state x'_{ev} be an (e_o, δ_{vd}) successor of x_{ev};
7: add transition $x_{ev} \xrightarrow{e_0} x'_{ev}$ to δ_{evd};
8: **else**
9: let state x'_{ev} be an (e_o, δ_{vs}) successor of x_{ev};
10: add transition $x_{ev} \xrightarrow{e_0} x'_{ev}$ to δ_{evs};
11: **if** $x'_{ev} \notin X_{ev}^s$ and x'_{ev} is energy safe **then**
12: $X_{ev}^s = X_{ev}^s \cup \{x'_{ev}\}$;
13: **if** $\exists j < n$, s.t. $x_{ev,j} \preccurlyeq x'_{ev}$ in execution $x_{ev0}e_0x_{ev1}e_1 \cdots x_{ev,n-1}e_{n-1}x'_{ev}$ **then**
14: merge x'_{ev} and $x_{ev,j}$ in G_{ev}^s: remove x'_{ev} and let all transitions reaching x'_{ev} reach $x_{ev,j}$;
15: **else**
16: $Energy - Unfold(x'_{ev}, G_v)$;

An execution in G_{ev}^s is of the form $x_{ev0}e_0x_{ev1}e_1 \cdots e_{n-1}x_{ev,n}$ where $x_{ev,i+1} = \delta_{evs}(x_{ev,i}, e_i)$ or $x_{ev,i+1} = \delta_{evd}(x_{ev,i}, e_i)$ for some $e_i \in E_o$. If $\exists x_{ev,j} \preccurlyeq x_{ev,n}$ for $j < n$, then a cycle is formed in the verifier and the system's energy level is nondecreasing when the cycle is traversed. We merge $x_{ev,j}$ and $x_{ev,n}$ by making all transitions reaching $x_{ev,n}$ end in $x_{ev,j}$ instead since the energy level would forever remain nonnegative on the current branch. Then the following result holds. Similar results are in Degorre et al. [2010], Pruekprasert and Ushio [2017] for different problems.

Theorem 1. The state space of G_{ev}^s is finite.

Proof. Proof by contradiction. Suppose G_{ev}^s is infinite. Since X, E_o and X_v are all finite, the number of outgoing transitions at each state in G_{ev}^s is also finite. By König's lemma (see e.g., Levy [2002]), there exists an infinite execution $x_{ev0}e_0x_{ev1}e_1 \cdots$, where $x_{ev,i+1}$ is a (e_i, δ_{vd}) or (e_i, δ_{vs}) successor of $x_{ev,i}$. From Algorithm 1, any $x_{ev,i}$ is energy safe and it is never the case that $\exists i < j$, s.t. $x_{ev,i} \preccurlyeq x_{ev,j}$. However, this contradicts with well quasi-ordering $\preccurlyeq$ on safe energy information states. □

Given an execution $x_{ev0}e_0x_{ev1}e_1 \cdots e_{n-1}x_{ev,n}$ and $\forall i \leq n$ let $I(x_{ev,i}) = (x_{d,i}, x_{f,i})$, there exists an execution $x_{d0}e_0x_{d1}e_1 \cdots e_{n-1}x_{d,n}$ in the desired estimator by definition, which implies $e_0e_1 \cdots e_n \in \mathcal{L}[Obs_d(G)] = L_{safe}$. Thus $\mathcal{L}(G_{ev}^s) \subseteq L_{safe}$ and private safety is not violated. When we merge states in G_{ev}^s, we do not change the language of the logical part (i.e., disregarding energy) since the logical parts of the states are the same, hence they have the same future logical behavior. Actually, considering energy will only restrict the logical behavior, so $\mathcal{L}(G_{ev}^s)$ may be smaller than L_{safe}. Next, the belief function indicates the minimum energy level of the system by strings with the same observation reaching a state under certain insertion.

Theorem 2. Given $x_{ev} \in X_{ev}$ in the energy verifier, if $I(x_{ev}) = (x_d, x_f)$ and $\exists l \in P[\mathcal{L}(G)]$, s.t. $\delta(x_{obs,0}, l) = x_f$, then $\forall x \in x_f$, $h_{x_{ev}}(x) = \min_{s \in P^{-1}(l) \cap \mathcal{L}(G)} \{V_m(s, s') : f(x_0, s) = x, \delta(x_{obs,0}, P(s)) = x_f, \delta_d(x_{obs,0}, P(s')) = x_d\}$.

The formal proof is omitted here, we give some interpretation instead. Since both δ_{evd} and δ_{evs} transitions track the minimum possible energy level by event insertion or occurrence, we can show this theorem by induction on the length of $l \in P[\mathcal{L}(G)]$. Given $x_{ev} \in X_{ev}$ with $I(x_{ev}) = (x_d, x_f)$, we locate string pairs $\{(s, s') \in \mathcal{L}(G) \times E^* : \delta(x_{obs,0}, P(s)) = x_f, \delta_d(x_{obs,0}, P(s')) = x_d\}$. That is, s is the original string that reaches state x in G and s is mapped to s' after insertion; $P(s)$ reaches the system's estimate x_f in $Obs(G)$ and $P(s')$ reaches the intruder's estimate x_d in $Obs_d(G)$. Let $P(s) = l$, then Theorem 2 shows that $h_{x_{ev}}(x)$ returns the system's *worst case* energy level by strings in $S(l, x_f) = \{s \in \mathcal{L}(G) : s \in P^{-1}(l), \delta(x_{obs,0}, l) = x_f\}$. Those strings have the same projection l but potentially different unobservable substrings and energy levels. Besides, if the minimum energy level by some strings with the same projection is nonnegative, it is true that the energy level by those strings is always nonnegative. It justifies how we update energy information states by (e_o, δ_{vd}) successor and (e_o, δ_{vs}) successor.

Example 1. This example presents a safe energy verifier. Consider the system G in Fig. 1, with $E_o = \{a, b, c\}$, $E_{uo} = \{u_1, u_2, u_3, u_4, u_4, u_5, u_6\}$ and secret states $X_S = \{x_7, x_9\}$. Let $\omega(a) = -8$, $\omega(b) = 1$, $\omega(c) = 2$ and $\omega_i(a) = -3$, $\omega_i(b) = -2$, $\omega_i(c) = -1$. Also, let $\omega(u_1) = 2$, $\omega(u_2) = 1$, $\omega(u_3) = -3$, $\omega(u_4) = -1$, $\omega(u_5) = -2$ and $\omega(u_6) = -1$. Finally, let $v_0 = 9$ be the system's initial energy.

The observer $Obs(G)$ is shown in Fig. 2 with renamed states: $A = \{x_0, x_3, x_4, x_8\}$, $B = \{x_1\}$, $C = \{x_2\}$, $D = \{x_5, x_6\}$, $E = \{x_7\}$, $F = \{x_9\}$. The system is obviously not opaque and we apply insertion functions to enforce opacity. The desired estimator is obtained by removing E and F from $Obs(G)$, while the feasible estimator is obtained by adding self-loops for every event in E_o at every state in $Obs(G)$. We omit them here due to space limitations.

Next we build the verifier in Fig. 3, where dashed lines indicate δ_{vd} transitions and solid lines indicate δ_{vs} transitions. Then we build the safe energy verifier in Fig. 4. States (D, A) and (C, F) are ignored in building G^s_{ev} since by insertion mechanism, every dashed transition should be followed by some solid transition, i.e., a valid string is inserted before observable events. The energy information states are as follows: $x_{ev0} = ((A, A), [9, 10, 7, 10])$, $x_{ev1} = ((B, A), [6, 7, 4, 7])$, $x_{ev2} = ((C, A), [4, 5, 2, 5])$, $x_{ev3} = ((B, F), 6)$, $x_{ev4} = (((B, A)), [3, 4, 1, 4])$, $x_{ev5} = (((C, A)), [1, 2, -1, 2])$, $x_{ev6} = ((C, D), [2, 0])$, $x_{ev7} = ((B, E), 2)$, $x_{ev8} = ((C, E), 0)$, $x_{ev9} = ((B, E), 2)$, $x_{ev10} = ((B, D), [1, -1])$, $x_{ev11} = ((C, D), [5, 3])$, $x_{ev12} = ((B, E), 5)$, $x_{ev13} = ((C, E), 3)$, $x_{ev14} = ((B, E), 5)$, $x_{ev15} = ((B, D), [4, 2])$, $x_{ev16} = ((C, D), [2, 0])$, $x_{ev17} = ((B, E), 2)$, $x_{ev18} = ((C, E), 0)$, $x_{ev19} = ((B, E), 2)$, $x_{ev20} = ((D, D), [8, 6])$, $x_{ev21} = ((B, B), 1)$, $x_{ev22} = ((C, C), 2)$, $x_{ev23} = ((B, B), 4)$, $x_{ev24} = ((B, C), 1)$, $x_{ev25} = ((C, C), -1)$ and $x_{ev26} = ((C, B), -1)$.

$E_L(x_{ev0}) = [h_{x_{ev0}}(x_0), h_{x_{ev0}}(x_3), h_{x_{ev0}}(x_4), h_{x_{ev0}}(x_8)] = [9, 10, 7, 10]$ since the elements in $E_L(x_{ev0})$ are placed in an increasing order w.r.t. state names. For example, $h_{x_{ev0}}(x_0) = v_0 = 9$, $h_{x_{ev0}}(x_3) = \min\{v_0 + \omega(u_1), v_0 + \omega(u_2)\} = 10$, $h_{x_{ev0}}(x_4) = \min\{v_0 + \omega(u_1u_3), v_0 + \omega(u_2u_3)\} = 7$, $h_{x_{ev0}}(x_8) = v_0 + \omega(u_6) = 10$. x_{ev1} is a (a, δ_{vd}) successor of x_{ev0} in Figure 4 and $\forall x \in A$, $h_{x_{ev1}}(x) = h_{x_{ev0}}(x) + \omega_i(a)$. Also $x_{ev,11}$ is a (b, δ_{vs}) successor of x_{ev1}, so $h_{x_{ev,11}}(x_5) = \min\{h_{x_{ev1}}(x_4) + \omega(b), x_{ev1}(x_3) + \omega(b)\} = 5$, $h_{x_{ev,11}}(x_6) = \min\{h_{x_{ev1}}(x_4) + \omega(b) + \omega(u_4), h_{x_{ev1}}(x_4) + \omega(b) + \omega(u_5), h_{x_{ev1}}(x_3) + \omega(b) + \omega(u_4), h_{x_{ev1}}(x_3) + \omega(b) + \omega(u_5)\} = 3$. All the other states are obtained in a similar way. Among them, $h_{x_{ev5}}(x_4) = -1$, $h_{x_{ev,10}}(x_6) = -1$, $h_{x_{ev,25}}(x_2) = -1$ and $h_{x_{ev,26}}(x_1) = -1$. Thus x_{ev5}, $x_{ev,10}$, $x_{ev,25}$ and $x_{ev,26}$ are not safe, they are not included in G^s_{ev}. Also, we merge the states connected with green lines in Fig. 4 since x_{ev7}, $x_{ev,12}$, $x_{ev,17}$, $x_{ev,21}$ subsume x_{ev9}, $x_{ev,14}$, $x_{ev,19}$, $x_{ev,23}$, respectively.

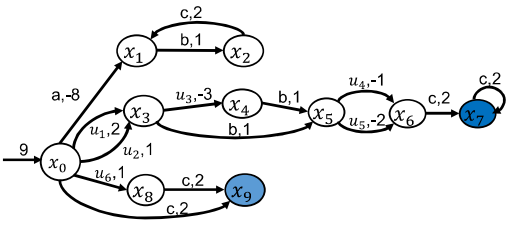

Fig. 1. The original system G

4.2 Build the Energy Insertion Structure

We come to step (3) and construct the Energy Insertion Structure (EIS) from the safe energy verifier. By building the EIS, we form a game with perfect information between the insertion function and the environment.

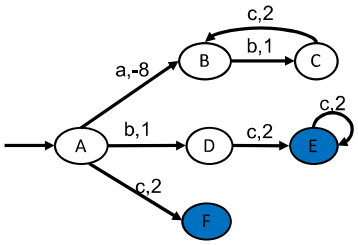

Fig. 2. Observer $Obs(G)$ where $A = \{x_0, x_3, x_4, x_8\}$, $B = \{x_1\}$, $C = \{x_2\}$, $D = \{x_5, x_6\}$, $E = \{x_7\}$, $F = \{x_9\}$

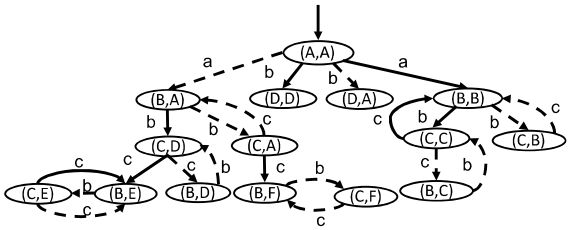

Fig. 3. Verifier G_v

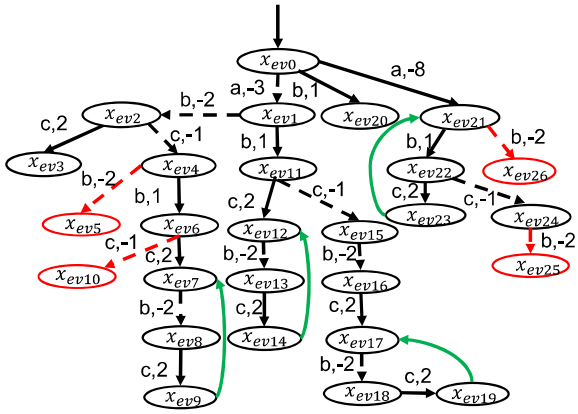

Fig. 4. Safe energy verifier G^s_{ev} (without red states, merge states connected with green lines)

Given system G with initial energy v_0, the Energy Insertion Structure (EIS) is a bipartite structure described by a tuple $(Q^B_Y, Q^B_Z, E_o, \Theta, f^B_{yz}, f^B_{zy}, y_0, v_0)$ where $Q^B_Y \subseteq Q^E$ is the set of energy information states; $Q^B_Z \subseteq Q^E \times E_o$ is the set of energy information states augmented with observable events; $f^B_{yz} : Q^B_Y \times E_o \to Q^B_Z$ is the transition function from Q^B_Y to Q^B_Z; $f^B_{zy} : Q^B_Z \times \Theta \to Q^B_Y$ is the transition function from Q^B_Z to Q^B_Y; E_o is the set of observable events; $\Theta \subseteq E^*_o$ is the set of insertion decisions; $y_0 \in Q^B_Y$ is the initial state where $y_0 = x_{ev0}$; $v_0 \in \mathbb{N}^+$ is the initial energy. The EIS is formally defined by Algorithm 2.

In the EIS, the environment plays at Q^B_Y states (Y-states) and the insertion function plays at Q^B_Z states (Z-states). We call $z \in Q^B_Z$ a *deadlocking* state if $\nexists \theta \in \Theta$, s.t. $f^B_{zy}(z, \theta)!$. Also $y \in Q^B_Y$ is called a *terminating state* if $\nexists e_o \in E_o$, s.t. $f^B_{yz}(y, e_o)!$. Deadlocking states are not allowed and will be pruned away in constructing the EIS.

Next, we construct the EIS from G^s_{ev} in Algorithm 2. In G^s_{ev}, we define its unmodified language, denoted by $\mathcal{L}_u(G^s_{ev})$, as the language generated by G^s_{ev} if we view

all δ_{evd} transitions as ϵ-transitions. If $\mathcal{L}_u(G^s_{ev}) = P[\mathcal{L}(G)]$ does not hold, then we return $\emptyset$ and claim that we cannot find privately safe insertion functions from the EIS, since a privately safe insertion function should be able to map every string in $P[\mathcal{L}(G)]$ to some safe string. On the other hand, if the preceding equality holds, then we may obtain insertion functions from the EIS to solve Problem 1.

Algorithm 2 Construction of the EIS

Input: G^s_{ev}, $Obs(G)$
Output: EIS or $\emptyset$
1: **if** $\mathcal{L}_u(G^s_{ev}) \neq P[\mathcal{L}(G)]$ **then**
2: Return $\emptyset$;
3: **else**
4: $Q^B_Y = \{y_0\} = \{x_{ev0}\}$, $Q^B_Z = \emptyset$;
5: $DoDFS(y_0, G^s_{ev}, Obs(G))$;
6: **while** $\exists z \in Q^B_Z$, s.t. z is deadlocking **do**
7: Remove z and all preceding states $y \in Q^B_Y$, s.t. $f^A_{yz}(y, e_o) = z$ for some $e_o \in E_o$;
8: Take the accessible part of the structure;
9: **procedure** DoDFS($y, G^s_{ev}, Obs(G)$)
10: **for** $e_o \in E_o$, s.t. $\delta(x_f, e_o)!$ in $Obs(G)$ where $I(y) = x_v = (x_d, x_f)$ **do**
11: $z = (y, e_o) = f^B_{yz}(y, e_o)$;
12: add transition $y \xrightarrow{e_o} z$ to f^B_{yz};
13: **if** $z \notin Q^B_Z$ **then**
14: $Q^B_Z = Q^B_Z \cup \{z\}$;
15: **for** $\theta \in \Theta$, s.t. $\exists \tilde{x}_{ev} = \delta_{evd}(y, \theta)$ and $\delta_{evs}(\tilde{x}_{ev}, e_o)!$ **do**
16: $y' = \delta_{evs}(\tilde{x}_{ev}, e_o) = f^B_{zy}(z, \theta)$;
17: add transition $z \xrightarrow{\theta} y'$ to f^B_{zy};
18: **if** $y' \notin Q^B_Y$ **then**
19: $Q^B_Y = Q^B_Y \cup \{y'\}$;
20: $DoDFS(y', G^s_{ev}, Obs(G))$;

Procedure *DoDFS* is a depth-first search process, which adds states and transitions to the EIS. In line 15, θ is a valid insertion choice if string θe_o is well defined in G^s_{ev}. *DoDFS* may cause deadlocking Z-states and we prune them away and remove their preceding Y-states in a recursive manner until no state is to be removed, thus the EIS terminates only at terminating states. If the initial state is pruned, the algorithm also returns nothing. Since G^s_{ev} is finite, Algorithm 2 converges in finite number of steps. The algorithm is similar with the algorithm of building the All Insertion Structure in Wu and Lafortune [2014]. In the following discussion, we assume the EIS is not empty so that it can be used to solve Problem 1.

Example 2. We revisit Example 1 and construct the EIS in Fig. 5 following Algorithm 2. Since $\mathcal{L}_u(G^s_{ev}) = P[\mathcal{L}(G)]$, the EIS is not empty. There are two types of states in the EIS: the square Y-states and the oval Z-states. The transitions from Y-states are events executed by the system and the transitions from Z-states are insertion function's choices. The game is initialized at $y_0 = x_{ev0}$ where events a, b, c are the system's observable outputs. After b is observed, the game reaches state (x_{ev0}, b) and the insertion function begins to play by inserting a or abc before b, where $\omega_i(a) = -3$ and $\omega_i(abc) = -6$. All transitions in the structure are interpreted in a similar manner. After *DoDFS*, there is a deadlocking Z-state $(x_{ev,20}, c)$, which is pruned together with its preceding Y-state $x_{ev,20}$. In the EIS, each time Y-states $x_{ev7}, x_{ev,12}, x_{ev,17}, x_{ev,21}$ are visited, the energy level of the system will increase or stay the same, thus remain nonnegative forever.

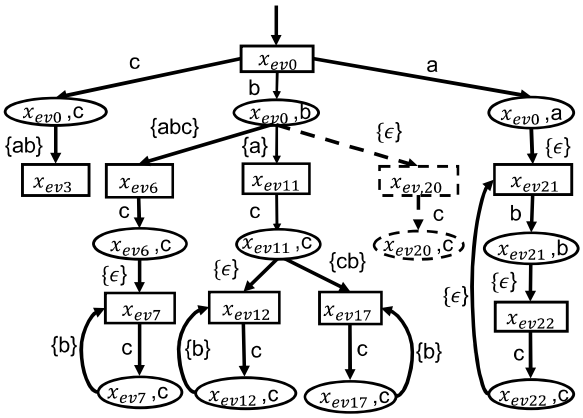

Fig. 5. Energy insertion structure (without the dashed transitions and states)

5. SOLVING OPACITY ENFORCEMENT PROBLEM

In this section, we discuss some properties of the EIS and solve Problem 1 by finding insertion functions in the EIS. Intuitively, since all energy information states in the EIS are safe and every string in $P[\mathcal{L}(G)]$ is mapped to some safe strings under some insertion decisions, then any insertion function obtained from the EIS solves Problem 1.

Definition 5. (Run). A run in the EIS is a sequence of alternating states, observable events and insertion decisions: $r = y_0 \xrightarrow{e_0} z_0 \xrightarrow{\theta_0} y_1 \xrightarrow{e_1} \cdots y_{n-1} \xrightarrow{e_{n-1}} z_{n-1} \xrightarrow{\theta_{n-1}} y_n$ where $n \in \mathbb{N}$, y_0 is the initial state of the EIS, $\forall i < n$, $e_i \in E_o$ and $\theta_i \in \Theta(z_i)$. Denote the set of runs by *Run*.

Given a run r as defined above, the *edit projection* $P_e : Run \to P[\mathcal{L}(G)]$ is such that $P_e(r) = e_0 e_1 \cdots e_n$ and the *generated string* $l_g(r)$ returns $\theta_0 e_0 \theta_1 e_1 \cdots \theta_{n-1} e_{n-1}$.

Definition 6. (Insertion function embedded in the EIS). An insertion function f_i is embedded in the EIS if $\forall l \in P[\mathcal{L}(G)]$, $\exists r \in Run$, s.t. $P_e(r) = l$ and $l_g(r) = f_i(l)$.

If insertion function f_i is embedded in the EIS, given a string $l = e_0 e_1 \cdots e_{n-1} \in P[\mathcal{L}(G)]$, suppose l is edited to $\theta_0 e_0 \theta_1 e_1 \cdots \theta_{n-1} e_{n-1}$, then there is a unique run $r = y_0 \xrightarrow{e_0} z_0 \xrightarrow{\theta_0} y_1 \xrightarrow{e_1} \cdots y_{n-1} \xrightarrow{e_{n-1}} z_{n-1} \xrightarrow{\theta_{n-1}} y_n \in Run$ where each θ_i is defined at a Z-state in the run. Specifically, we denote a run r by $r_{f_i}(l)$ if $P_e(r) = l$ and $l_g(r) = f_i(l)$. f_i works by specifying an outgoing transition at each Z state in the EIS. Insertion functions can be extracted from the EIS and represented as I/O automata, similarly with Algorithm 5 in Wu and Lafortune [2014]; the readers are referred to that work for more details. The following theorem shows the soundness of Algorithm 2.

Theorem 3. Any insertion function embedded in the EIS solves Problem 1.

Proof. Suppose f_i is embedded in the EIS. By the definition of the verifier, $X_v \subseteq X_{obsd} \times X_{obs}$ and the intruder

only observes states in X_{obsd}. Since the state estimate component of each state in the EIS is from X_v, f_i maps every string to a safe one, thus is privately safe.

Besides, for $s \in \mathcal{L}(G)$ with $P(s) = l = e_0e_1\cdots e_{n-1}$, suppose that $f_i(l) = \theta_0e_0\theta_1e_1\cdots\theta_{n-1}e_{n-1}$, then there exists a run $r = y_0 \xrightarrow{e_0} z_0 \xrightarrow{\theta_0} y_1 \xrightarrow{e_1} \cdots y_{n-1} \xrightarrow{e_{n-1}} z_{n-1} \xrightarrow{\theta_{n-1}} y_n$ in the EIS. Denote this run by $r_{f_i}(l)$. Since each y_i in $r_{f_i}(l)$ is energy safe and each energy information state contains the worst case energy level of the system by strings under certain insertions, then from Theorem 2, we know that $\forall s \in P^{-1}(l) \cap \mathcal{L}(G)$, $V_m(s, s_{f_i}) \geq 0$. Also by Algorithm 2 and Definition 6, we know $V_m(s, s_{f_i}) \geq 0$ holds $\forall s \in P[\mathcal{L}(G)]$, therefore f_i solves Problem 1. □

Thus Theorem 3 proves the soundness of Algorithm 2. However, its completeness is still an open problem.

Example 3. We reconsider the EIS in Fig. 5 and obtain an embedded insertion function f_i as follows: $f_i(c) = abc$, $f_i(b) = ab$ and b is inserted as soon as it observes event c occurring from state E in Fig. 2; no other insertions are made. It is easy to verify that f_i and any other insertion function embedded in the EIS solve Problem 1.

6. CONCLUSION

We investigated opacity enforcement by insertion functions in a partially-observed discrete event system under initial energy constraint. To the best of our knowledge, this work is the first one to investigate opacity enforcement under such quantitative constraints. The system's energy level changes dynamically according to event insertion and execution. Our goal is to synthesize an insertion function that enforces opacity as well as guarantees that the system's energy level is never below zero. A bipartite transition structure called Energy Insertion Structure (EIS) was proposed, in order to capture information states and their associated worst-case energy level vectors. The EIS characterizes the game between the insertion functions and the environment. We showed that the EIS embeds insertion functions that solve our proposed problem, thereby providing a valid characterization of the solution space.

ACKNOWLEDGEMENTS

We thank all the reviewers for their useful suggestions for improvement. We are especially indebted to an anonymous reviewer for pointing out an incomplete proof in the submitted version of Theorem 3; at present, the completeness of Algorithm 2 remains an open problem.

REFERENCES

Bryans, J.W., Koutny, M., and Ryan, P. (2005). Modelling opacity using Petri nets. *Electronic Notes in Theoretical Computer Science*, 121, 101–115.

Cassandras, C.G. and Lafortune, S. (2008). *Introduction to discrete event systems – 2nd Edition.* Springer.

Cassez, F. (2009). The dark side of timed opacity. In *International Conference on Information Security and Assurance*, 21–30. Springer.

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

de Alfaro, L., Henzinger, T.A., and Kupferman, O. (2007). Concurrent reachability games. *Theoretical Computer Science*, 386(3), 188–217.

Degorre, A., Doyen, L., Gentilini, R., Raskin, J.F., and Toruńczyk, S. (2010). Energy and mean-payoff games with imperfect information. In *Computer Science Logic*, 260–274. Springer.

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

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

Jacob, R., Lesage, J.J., and Faure, J.M. (2016). Overview of discrete event systems opacity: Models, validation, and quantification. *Annual Reviews in Control.*

Ji, Y. and Lafortune, S. (2017). Enforcing opacity by publicly known edit functions. In *Proceedings of the 56th IEEE Conference on Decision and Control*, 4866–4871.

Keroglou, C. and Hadjicostis, C.N. (2017). Probabilistic system opacity in discrete event systems. *Discrete Event Dynamic Systems: Theory and Applications.*

Levy, A. (2002). *Basic set theory.* Courier Corporation.

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

Pérez, G.A. (2017). The fixed initial credit problem for partial-observation energy games is ack-complete. *Information Processing Letters*, 118, 91–99.

Pruekprasert, S. and Ushio, T. (2017). Supervisory control of partially observed quantitative discrete event systems for fixed-initial-credit energy problem. *IEICE transactions on Information and Systems*, 100(6), 1166–1171.

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

Saboori, A. and Hadjicostis, C.N. (2013). Verification of initial-state opacity in security applications of discrete event systems. *Information Sciences*, 246, 115–132.

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

Tong, Y., Li, Z., Seatzu, C., and Giua, A. (2017). Current-state opacity enforcement in discrete event systems under incomparable observations. *Discrete Event Dynamic Systems: Theory and Applications*, 1–22.

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

Yin, X. and Lafortune, S. (2015). A general approach for solving dynamic sensor activation problems for a class of properties. In *Proceedings of the 54th IEEE Conference on Decision and Control*, 3610–3615.

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

Yin, X. and Lafortune, S. (2017). A new approach for the verification of infinite-step and K-step opacity using two-way observers. *Automatica*, 80, 162–171.