A Uniform Approach for Synthesizing Property-Enforcing Supervisors for Partially-Observed Discrete-Event Systems

Xiang Yin, *Student Member, IEEE*, and Stéphane Lafortune, *Fellow, IEEE*

***Abstract*—The problem under consideration in this paper is that of enforcement by supervisory control of a given property on a partially-observed discrete-event system. We present a general methodology that is applicable to a large class of properties previously studied (individually) in the literature. These properties include, but are not restricted to, safety, diagnosability, opacity, detectability, anonymity and attractability. When the given system does not satisfy the considered property, the objective is to synthesize a supervisor that restricts the system's behavior and provably enforces the given property; moreover, it is required that this supervisor be maximally permissive. We consider the general case where the system's events are partitioned into observable and unobservable events, and controllable and uncontrollable events, and we do not make any assumptions about these two partitions; in particular, we do not assume that all controllable events are observable. Our uniform approach first maps the considered property to a suitably-defined information state for the partially-observed system and then develops a supervisor synthesis methodology based on a finite bipartite transition system that embeds all reachable information states and all admissible supervisory control strategies. This transition system is called the All Enforcement Structure (or AES). We present an algorithm for the construction of the AES and discuss its properties. Then we use the AES to develop a synthesis algorithm that constructs a supervisor that is provably property enforcing and maximally permissive. We illustrate the application of our uniform approach to the enforcement of the above-mentioned properties.**

***Index Terms*— Discrete event systems (DES), partial observation, property enforcement, supervisory control.**

I. Introduction

VERIFICATION and synthesis are two important research issues in the study of Discrete Event Systems (DES). In large complex automated systems, we are first interested in *verifying* whether or not the given system satisfies a certain property of interest; when the answer is negative, we wish to *synthesize* a supervisor or controller that provably enforces the property by restricting the system behavior in a manner that maintains maximal permissiveness. In many applications, the system of interest is partially observed due to the limited sensing capabilities. In this paper, we are concerned with the the problem of synthesizing a partial observation supervisor for a DES in order to enforce a certain property on its set of behaviors.

In the context of DES, many properties have been studied. In the standard supervisory control problem [1], the properties under consideration are safety and non-blockingness: safety requires that the system should only execute legal behaviors (modeled in terms of a regular language); non-blockingness requires that the system should always be able to eventually achieve one in a set of desired behaviors. In [2], the property of diagnosability was defined and studied. Diagnosability is related to the problem of fault diagnosis and isolation in automated systems and it requires that any type of fault event be diagnosed unambiguously within a bounded delay. In [3] and [4], a confidentiality property for partially-observed DES called opacity was studied. Opacity captures the plausible deniability of the system's "secret" in the presence of an outside observer that is potentially malicious. Anonymity is a type of opacity that is of interest in the study of privacy. Several other properties have been considered in the DES literature to capture different requirements on the behavior of the system; among them we mention detectability [5], [6] and attractability [7]–[9]. In the computer science literature on verification and reactive synthesis, linear temporal logic or branching time logic are also used to describe desired properties of systems; see, e.g., [10].

When a system does not satisfy a given property of interest, one is interested in enforcing the property via some enforcement mechanism. One of the most commonly-used enforcement mechanisms is to restrict the system behavior by supervisory control. This approach has been extensively studied in the context of DES since the the pioneering work of Ramadge and Wonham [1]. In this setting, the control problem is to synthesize a supervisor that prevents behaviors that violate the property from occurring in the controlled system. A constraint of that synthesis problem is for the supervisor to remain maximally permissive while enforcing the property; such maximal permissiveness is only possible in a local sense, in general, for partially observed DES.

Different approaches have been proposed to synthesize supervisors for different properties. In the standard supervisory control problem, the properties to be enforced are *safety* and

Manuscript received August 6, 2015; accepted September 21, 2015. Date of publication October 1, 2015; date of current version July 22, 2016. This work was partially supported by the National Science Foundation (NSF) grants CCF-1138860 (Expeditions in Computing project ExCAPE: Expeditions in Computer Augmented Program Engineering), CNS-1421122 and CNS-1446298, and by the TerraSwarm Research Center, one of six centers supported by the STARnet phase of the Focus Center Research Program (FCRP) a Semiconductor Research Corporation program sponsored by MARCO and DARPA.

The authors are with the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109 USA (e-mail: xiangyin@umich.edu; stephane@umich.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TAC.2015.2484359

TABLE I
COMPARISON BETWEEN THE PROPOSED UNIFORM APPROACH AND PREVIOUS APPROACHES

Property	Safety	Opacity	Diagnosability	Detectability	Anonymity	Attractability
Previous Works	[11]–[15]	[16]–[19]	[20]	[21]	None	[9]
Previous Assumptions	None	$\Sigma_a \subseteq \Sigma_o$, $\Sigma_c \subseteq \Sigma_o$[1]	$\Sigma_c \subseteq \Sigma_o$	$\Sigma_c \subseteq \Sigma_o$	N/A	$\Sigma_c \subseteq \Sigma_o$
Assumptions in this paper	None	$\Sigma_a = \Sigma_o$	None	None	$\Sigma_a = \Sigma_o$	None

[1] Σ_c, Σ_o and Σ_a are, respectively, the set of events that can be controlled by the supervisor, the set of events that can be observed by the supervisor, and the set of events that can be observed by the external observer.

non-blockingness. This problem was solved in [1] in the case of full observation (e.g., no unobservable events). In the partial observation setting, different solutions methodologies were proposed; see, e.g., the following original references and books [11], [12], [22]–[26]. In [20], an integrated approach to control and diagnosis was studied. Specifically, the authors presented an approach for designing a maximally permissive supervisor that enforces diagnosability. This problem is also referred to as the *active diagnosis* problem. Several approaches have also been proposed in the literature for enforcing *opacity* of a given system that is not opaque at the outset; see, e.g., [16]–[19], [27], [28]. In this context, the control problem is to synthesize a partial-observation supervisor that prevents behaviors that reveal the secret from occurring in the controlled system. In other words, the objective for the opacity-enforcing supervisor is to hide the system's secret in the presence of the external intruder. In [21], the author studied the problem of synthesizing a supervisor that enforces *detectability*. The enforcement of *attractability* was studied in [7] and [8] for the fully-observed case and more recently in [9] under the partial observation assumption.

While there is a wide literature on the enforcement of properties of DES using supervisory control, several open problems remain. First, except for the standard supervisor control problem under partial observation, all other works assume that $\Sigma_c \subseteq \Sigma_o$, where Σ_c and Σ_o are the sets of events that can be controlled and observed by the supervisor, respectively. In other words, the solutions to these property enforcement problems are only available under the assumption that all controllable events are observable. Second, all of the existing literature deals with different property-enforcing problems *separately*, i.e., each enforcement technique developed is only applicable to a specific property. Moreover, the enforcement of some properties, such as anonymity, has not yet been addressed in the literature.

In this paper, we propose a *uniform approach* that is applicable to the enforcement, by supervisory control, of a large class of properties that can be expressed in terms of suitably-defined *information states*. We refer to such properties as *information-state-based* (or IS-based) properties. Roughly speaking, an IS-based property is a property that only depends on the current local information of the system, as available to the supervisor, and does not explicitly depend on the future behavior of the system. The approach that we develop to tackle this problem is significantly different from the previous approaches in the literature, which are also concerned with property enforcement by supervisory control. Specifically, our approach is based on the construction of a finite information structure called the *All Enforcement Structure* and abbreviated as AES. The AES is a game structure between the supervisor and the "environment" (aka system). By construction, the AES embeds in its structure *all* property-enforcing supervisors. Therefore, it can serve as the basis for solving the synthesis problem. The AES was inspired by works in [29] and [30] for dynamic sensor activation problems and by our recent work in [15], [28], [31], and [32], which solves the standard safety supervisory control problem and the opacity enforcement problem by using the same type of transition system capturing the possible moves of the system and the set of admissible supervisors. Note that the *control problem* considered in this paper and the *sensor activation problem* considered in [29] and [30] are incomparable. Activating/deactivating sensors can only affect *knowledge* about the system, whereas control actions can affect both knowledge about the system and *actual behavior* of the system. Consequently, some properties that can be enforced by control may not be enforceable by sensor activation, e.g., safety or attractability, since sensor activations cannot change the actual behavior of the system.

In this paper, we generalize the methodologies in [15] and [28] and develop a single *uniform* solution methodology that is applicable not only to safety and opacity, but to any property that can be expressed as an IS-based property. This includes, but is not restricted to, safety, diagnosability, opacity, detectability, anonymity and attractability. There are properties that cannot be formulated as IS-based properties; the prime example is non-blockingness. In Table I, we compare our proposed approach with previous work. To the best of our knowledge, the problem of synthesizing a maximally permissive supervisor that enforces anonymity has not yet been considered in the literature; this property can be enforced by our general methodology. Moreover, we relax the assumption made by previous works that all controllable events should be observable. We show that, in this more general setting, uniqueness of a maximally permissive solution is lost. Hence, our focus is on the synthesis of solutions that are provably (locally) maximally permissive.

This paper is organized as follows. In Section II, we present the model of the system to be analyzed. In Section III, we formulate the information-state-based property enforcement problem that we solve in this paper. In Section IV, we define a class of bipartite transition systems that is used for solving the property enforcement problem. In Section V, we define the structure called AES, the key notion for the approach investigated in this paper. We then present a general-purpose synthesis algorithm that returns a maximally-permissive partial-observation supervisor based on the AES in Section VI. In Section VII, we show how the proposed uniform approach can be applied to enforce different specific properties. Finally, we conclude the paper in Section VIII.

II. Preliminary

Let Σ be a finite set of events and denote by Σ^* the set of all finite strings over Σ, including the empty string ϵ. A language $L \subseteq \Sigma^*$ is a subset of Σ^*. The prefix closure of language L is the set $\overline{L} = \{t \in \Sigma^* : \exists u \in \Sigma^* \text{ s.t. } tu \in L\}$. We say that a language is *prefix-closed* if $L = \overline{L}$. Given language L and string $s \in L$, we denote the active (event) set at s in L by $\Delta_L(s) = \{\sigma \in \Sigma : s\sigma \in L\}$ and use $L/s = \{t \in \Sigma^* : \text{st} \in L\}$ to denote the set of continuations of s in L. For any $\sigma \in \Sigma, s \in \Sigma^*$, we use $\sigma \in s$ to denote that σ occurs at least once in s.

The DES of interest is modeled as a deterministic finite-state automaton $G = (X, \Sigma, \delta, x_0)$, where X is the finite set of states, Σ is the finite set of events, $\delta : X \times \Sigma \to X$ is the partial transition function, where $\delta(x, \sigma) = y$ means that there is a transition labeled by event σ from state x to state y, and $x_0 \in X$ is the initial state. The transition function δ is extended to $X \times \Sigma^*$ in the usual manner (see, e.g., [23]). For brevity, we write $\delta(x_0, s)$ as $\delta(s)$. The language generated by G is defined by $\mathcal{L}(G) := \{s \in \Sigma^* : \delta(x_0, s)!\}$, where ! means "is defined."

In the framework of supervisory control [1], the system G is controlled by a *supervisor* that dynamically enables/disables events of the system such that some specification is provably achieved. The event set Σ is partitioned into two disjoint subsets: $\Sigma = \Sigma_c \dot\cup \Sigma_{uc}$, where Σ_c is the set of controllable events and Σ_{uc} is the set of uncontrollable events. We say that a control decision $\gamma \in 2^\Sigma$ is admissible if $\Sigma_{uc} \subseteq \gamma$, namely, uncontrollable events can never be disabled. We define $\Gamma = \{\gamma \in 2^\Sigma : \Sigma_{uc} \subseteq \gamma\}$ as the set of admissible control decisions. When the system is partially observed [11], [12], Σ is also partitioned into two disjoint sets: $\Sigma = \Sigma_o \dot\cup \Sigma_{uo}$, where Σ_o is the set of observable events and Σ_{uo} is the set of unobservable events. The natural projection $P : \Sigma^* \to \Sigma_o^*$ is defined by

$$P(\epsilon) = \epsilon \text{ and } P(s\sigma) = \begin{cases} P(s)\sigma & \text{if } \sigma \in \Sigma_o \\ P(s) & \text{if } \sigma \in \Sigma_{uo}. \end{cases} \tag{1}$$

The inverse projection $P^{-1} : \Sigma_o^* \to 2^{\Sigma^*}$ is defined by $P^{-1}(t) := \{s \in \Sigma^* : P(s) = t\}$. Since a supervisor can only make decisions based on its observations, a partial-observation supervisor is a function $S_P : P(\mathcal{L}(G)) \to \Gamma$. We use the notation S_P/G to represent the controlled system and the language generated by S_P/G, denoted by $\mathcal{L}(S_P/G)$, is defined recursively in the usual manner (see, e.g., [23]).

Given a prefix-closed language $\overline{K} = K$, we say that K is *controllable* (w.r.t. G and Σ_c) if $(\forall s \in K, \sigma \in \Sigma_{uc})(s\sigma \in \mathcal{L}(G) \Rightarrow s\sigma \in K)$; we say that K is *observable* (w.r.t. G, Σ_c and Σ_o) if $(\forall s, s' \in K, \sigma \in \Sigma_c)(P(s) = P(s') \wedge s\sigma \in K \wedge s'\sigma \in \mathcal{L}(G) \Rightarrow s'\sigma \in K)$; we say that K is *normal* (w.r.t. G and Σ_o) if $\overline{K} = P^{-1}[P(\overline{K})] \cap \mathcal{L}(G)$. It is well known that there exists a supervisor S_P such that $\mathcal{L}(S_P/G) = K$ if and only if K is controllable and observable [11], [12]. In general, observability is not preserved under union, unless additional assumptions are made. For instance, it was shown in [33] that if $\Sigma_c \subseteq \Sigma_o$, then controllability and observability together imply normality, which is preserved under union. However, this assumption is not required in this paper.

We define several operators that will be used in this paper. The set of *all possible states* in G reachable from the initial state x_0 via some string in sublanguage $L \subseteq \mathcal{L}(G)$ with the same projection as $s \in L$, is given by

$$R_G(s, L) := \{x \in X : \exists t \in L \text{ s.t. } P(t) = P(s) \wedge x = \delta(x_0, t)\}. \tag{2}$$

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

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

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

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

III. Problem Formulation

In this section, we define the class of information-state-based properties and formulate the *Property Enforcement Problem* that we solve in this paper.

In the framework of the supervisory control theory initiated by Ramadge and Wonham [1], the supervisor restricts the behavior of the system such that the controlled system is safe and non-blocking. In this paper, we do not consider the non-blockingness property; this is why the definition of G in Section II did not include a set of "marked" states. Instead, we consider the (weaker) *liveness* property. Liveness is an important property in many cyber and cyber-physical systems, e.g., software systems [34] and flexible manufacturing systems [35]. Formally, we say that a language L is live if for any $s \in L$, we have $\Delta_L(s) \neq \emptyset$. We say that system G is live if its generated language $\mathcal{L}(G)$ is live. Since the definitions of many properties, e.g., diagnosability and detectability, are based on the assumption that the system under consideration is live, hereafter, we assume that G is live and we must ensure that the controlled system is also live. The liveness assumption on G is without essential loss of generality, since it can be relaxed by adding observable self-loops at terminal states, as is done in [2]. (Essentially, this means that system deadlock is observable.)

In addition to liveness, in many applications, we are also interested in enforcing *additional* properties on the system behavior. In general, a property on a language (or a language-based property) is a function $\varphi : 2^{\Sigma^*} \to \{0, 1\}$, where for any language L, $\varphi(L) = 1$ means that L satisfies property φ. We write that $L \models \varphi$ if φ is a language-based property and $\varphi(L) = 1$. For example, safety is a typical property of interest. Let $K \in 2^{\Sigma^*}$ be a specification language. Then the safety property $\varphi : 2^{\Sigma^*} \to \{0, 1\}$ can be defined by: for any $L \in 2^{\Sigma^*}$, $\varphi(L) = 1 \Leftrightarrow L \subseteq K$.

To enforce a given property on the system, we need to synthesize a supervisor that restricts the system behavior to a sublanguage that satisfies the property; moreover, it is desired that this sublanguage be as large as possible w.r.t. set inclusion. In other words, the supervisor should only disable an event if it is necessary to do so. By considering a general property instead of only safety, the standard supervisory control problem under

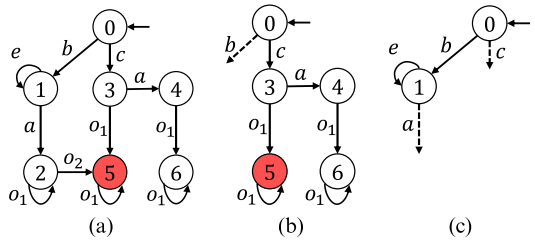

Fig. 1. For G: $\Sigma_c = \{a, b, c\}, \Sigma_o = \{o_1, o_2\}$, and $X_S = \{5\}$. (a) System G. (b) Solution G_1. (c) Solution G_2.

partial observation [11], [12] is generalized to the *Maximally Permissive Property Enforcement Problem* defined as follows.

Definition III.1. (Maximally Permissive Property Enforcement Problem): Given system G and language-based property $\varphi : 2^{\Sigma^*} \to \{0, 1\}$, synthesize a partial observation supervisor $S_P : P(\mathcal{L}(G)) \to \Gamma$, such that

1) $\mathcal{L}(S_P/G)$ is live;
2) $\mathcal{L}(S_P/G) \models \varphi$;
3) For any S'_P satisfying 1) and 2), we have that $\mathcal{L}(S_P/G) \not\subset \mathcal{L}(S'_P/G)$.

In the formulation of the above problem, the property of interest is defined over languages; hence, it may not be possible to bound *a priori* the memory needed for its verification. To simplify our problem, hereafter, we will investigate a particular class of properties called *Information-State-based* (IS-based) properties. Since we are dealing with partially observed systems, we define the notion of an *information state* as a subset $IS \subseteq X$ of states of G and denote by $I = 2^X$ the set of all information states. IS-based properties are defined as follows.

Definition III.2. (IS-Based Property): Given an automaton G, an IS-based property φ w.r.t. G is a function $\varphi : I \to \{0, 1\}$, where $\forall\, i \in 2^X$, $\varphi(i) = 1$ means that i satisfies this property. We say that sublanguage $L \subseteq \mathcal{L}(G)$ satisfies φ w.r.t. G, which is denoted by $L \models_G \varphi$, if $\forall\, s \in L : \varphi(R_G(s, L)) = 1$.

We will show later in Section VII that by some proper state space refinements, many important properties in the DES literature, e.g., safety, diagnosability, opacity, detectability and attractability, can be formulated as IS-based properties.

Example III.1: Let us consider the system G in Fig. 1(a), where the set of observable events is $\Sigma_o = \{o_1, o_2\}$. Consider the subset of states $X_S = \{5\}$. We define the IS-based property $\varphi : I \to \{0, 1\}$ by

$$\varphi(i) = 0 \Leftrightarrow i \subseteq X_S \tag{5}$$

$\forall\, i \in I$. We will show later in Section VII that the IS-based property defined above essentially captures a security property called *current-state opacity*. One may interpret X_S as the set of secret states that the system wants to hide from a potentially malicious external observer, referred to as the intruder. We say that property φ holds if the intruder can never determine unambiguously that the secret has occurred based on its observation capabilities. If the intruder's observable set is $\Sigma_o = \{o_1, o_2\}$, then the system language $\mathcal{L}(G)$ does not satisfy φ, since $R_G(bao_2, \mathcal{L}(G)) = \{5\} \subseteq X_S$, i.e., upon the occurrence of string bao_2, the secret state 5 will be revealed to the intruder. □

Similarly to the property enforcement problem, we formulate the Maximally Permissive IS-Based Property Enforcement Problem (MPIEP) as follows.

Definition III.3. (Maximally Permissive IS-Based Property Enforcement Problem): Given system G and IS-based property $\varphi : 2^X \to \{0, 1\}$ w.r.t. G, synthesize a partial observation supervisor $S_P : \mathcal{L}(G) \to \Gamma$, such that

1) $\mathcal{L}(S_P/G)$ is live;
2) $\mathcal{L}(S_P/G) \models_G \varphi$;
3) For any S'_P satisfying 1) and 2), we have that $\mathcal{L}(S_P/G) \not\subset \mathcal{L}(S'_P/G)$.

Since controllability and observability together provide the necessary and sufficient conditions for the existence of a partial observation supervisor, to solve MPIEP, it suffices to find a *maximal controllable, observable and live sublanguage* of $\mathcal{L}(G)$ satisfying φ. We will show in Section VI that under the assumption that $\Sigma_c \subseteq \Sigma_o$, there always exists a unique *supremal* solution to MPIEP. However, this is not true in general, as illustrated in the following example.

Example III.2: Consider again the system G in Fig. 1. Let the set of controllable events be $\Sigma_c = \{a, b, c\}$, which is incomparable with Σ_o. To enforce property φ defined in Example III.1, we need to find a controllable, observable, and live sublanguage of $\mathcal{L}(G)$ that satisfies φ. It is easy to verify that solutions $\mathcal{L}(G_1)$ and $\mathcal{L}(G_2)$, shown in Fig. 1(b) and (c), respectively, are two maximal controllable and observable solutions satisfying φ. However, the union of these two solutions is not a valid solution, since the system needs to enable event a at state 1 and to disable event a at state 3; but states 1 and 3 are indistinguishable in $\mathcal{L}(G_1) \cup \mathcal{L}(G_2)$. This violates the property of observability. □

IV. Bipartite Transition System

In this section, we define the general notion of *bipartite transition system* (BTS), which was originally investigated in [32] to solve the standard supervisor control problem for safety and non-blockingness.

Definition IV.1. ([32]): A bipartite transition system T w.r.t. G is a 7-tuple[1]

$$T = \left(Q_Y^T, Q_Z^T, h_{YZ}^T, h_{ZY}^T, \Sigma, \Gamma, y_0^T\right) \tag{6}$$

where

- $Q_Y^T \subseteq I$ is the set of Y-states;
- $Q_Z^T \subseteq I \times \Gamma$ is the set of Z-states and $I(z)$ and $\Gamma(z)$ denote, respectively, the information state and the control decision components of a Z-state z, so that $z = (I(z), \Gamma(z))$;
- $h_{YZ}^T : Q_Y^T \times \Gamma \to Q_Z^T$ is the partial transition function from Y-states to Z-states, which satisfies the following constraint: for any $y \in Q_Y^T$, $z \in Q_Z^T$ and $\gamma \in \Gamma$, we have

$$h_{YZ}^T(y, \gamma) = z \Rightarrow [I(z) = \mathrm{UR}_\gamma(y)] \wedge [\Gamma(z) = \gamma]\,. \tag{7}$$

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

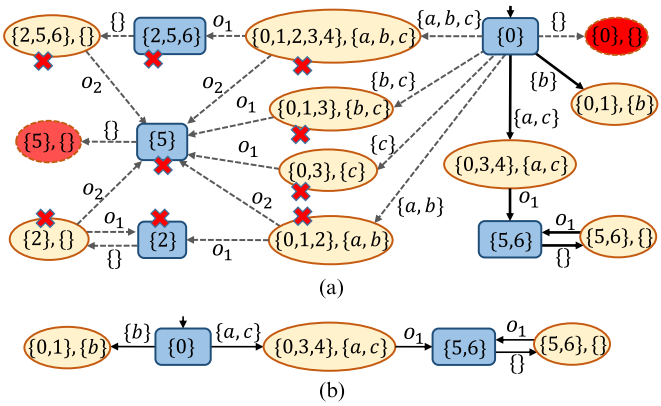

Fig. 2. Example of the construction of the AES. In the diagrams, rectangular (blue) states correspond to Y-states and oval (yellow) states correspond to Z-states. For simplicity, in the diagrams, we omit all uncontrollable events in the control decisions, e.g., decision $\{\}$ represents $\{o_1, o_2\}$, and so forth. (a) The resulting structure after procedure DoDSF. (b) The constructed AES.

- $h^T_{ZY} : Q^T_Z \times \Sigma \to Q^T_Y$ is the partial transition function from Z-states to Y-states, which satisfies the following constraint: for any $y \in Q^T_Y$, $z \in Q^T_Z$ and $e \in \Sigma$, we have

$$h^T_{ZY}(z,e) = y \Rightarrow [e \in \Gamma(z) \cap \Sigma_o] \wedge [y = \text{Next}_e(I(z))]. \quad (8)$$

- Σ is the set of events of G;
- Γ is the set of admissible control decisions of G;
- $y^T_0 \in Q^T_Y$ is the initial Y-state where $y^T_0 = \{x_0\}$.

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

Example IV.1: Consider again the system G in Fig. 1(a). As an example of a BTS, the reader is referred directly to Fig. 2(b), which is a particular type of BTS that we will discuss later in this paper. From the initial Y-state $y_0 = \{0\}$, by making control decision $\gamma = \{a, c, o_1, o_2\}$ (the uncontrollable events o_1 and o_2 are omitted in the figure), we will reach Z-state $z = h^T_{YZ}(y_0, \gamma) = (\{0, 3, 4\}, \{a, c, o_1, o_2\})$. From z, only one observable event, o_1, can happen, and it leads to the next Y-state $y_1 = h^T_{ZY}(z, o_1) = \{5, 6\}$. This BTS includes another control decision at $y_0 = \{0\}$, $\gamma = \{b, o_1, o_2\}$, from which no observation will occur. Finally, at Y-state $\{5, 6\}$, this BTS includes a single control decision, where only the uncontrollable events are included. □

Given two BTSs T_1 and T_2, we say that T_1 is a *subsystem* of T_2, denoted by $T_1 \sqsubseteq T_2$, if $Q^{T_1}_Y \subseteq Q^{T_2}_Y$, $Q^{T_1}_Z \subseteq Q^{T_2}_Z$, and for any $y \in Q^{T_1}_Y$, $z \in Q^{T_1}_Z$, $\gamma \in \Gamma$, and $e \in \Sigma$, we have that

1) $h^{T_1}_{YZ}(y, \gamma) = z \Rightarrow h^{T_2}_{YZ}(y, \gamma) = z$;
2) $h^{T_1}_{ZY}(z, e) = y \Rightarrow h^{T_2}_{ZY}(z, e) = y$.

For example, we see that the BTS in Fig. 2(b) isa subsystem of the BTS in Fig. 2(a).

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

Definition IV.1 provides a general definition for a BTS. However, for the purpose of control, we also want a BTS to satisfy the following two properties:

P1) For any $y \in Q^Y_T$, $C_T(y) \neq \emptyset$;
P2) For any $z \in Q^Z_T$, the following condition holds: $\forall e \in \Gamma(z) \cap \Sigma_o : (\exists x \in I(z) : \delta(x, e)!) \Rightarrow h^T_{ZY}(z, e)!$.

The first property says that for any Y-state, we need to be able to pick at least one control decision. The second property says that for any Z-state, we cannot block any enabled and feasible observable event. This is because we cannot choose which event will occur once we have made a control decision; the system will decide. Properties P1 and P2 together are referred to as the *completeness* property of a BTS.

Given a complete BTS T, it is possible to "decode" supervisors from it, as explained in the following definitions.

Definition IV.2. ([32]): Given a supervisor S_P, $IS^Y_{S_P}(y, s)$ is defined to be the Y-state that results from the occurrence of string s, when starting in Y-state y. This can be computed recursively as follows:

$$IS^Y_{S_P}(y, \epsilon) := y$$

$$IS^Y_{S_P}(y, s\sigma) := \begin{cases} h_{ZY}\left(h_{YZ}\left(IS^Y_{S_P}(y, s), S_P(s)\right), \sigma\right) & \\ & \text{if } \sigma \in \Sigma_o \cap S_P(s) \\ IS^Y_{S_P}(y, s), & \text{if } \sigma \in \Sigma_{uo} \cap S_P(s) \\ \text{undefined}, & \text{otherwise.} \end{cases}$$

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

Also, $IS^Z_{S_P}(z, s)$ is defined analogously by

$$IS^Z_{S_P}(z, \epsilon) := z$$

$$IS^Z_{S_P}(z, s\sigma) := \begin{cases} h_{YZ}\left(h_{ZY}\left(IS^Z_{S_P}(z, s), \sigma\right), S_P(s\sigma)\right), & \\ & \text{if } \sigma \in E_o \cap S_P(s) \\ IS^Z_{S_P}(z, s), & \text{if } \sigma \in E_{uo} \cap S_P(s) \\ \text{undefined}, & \text{otherwise.} \end{cases}$$

For brevity, we define $IS^Z_{S_P}(s) := IS^Z_{S_P}(z_0, s)$, where $z_0 = h_{YZ}(y_0, S_P(\epsilon))$. □

Definition IV.3: A supervisor S_P is said to be included in a complete BTS T if $(\forall s \in \mathcal{L}(S_P/G))[S_P(s) \in C_T(IS^Y_{S_P}(s))]$. $\mathcal{S}(T)$ denotes the set of all supervisors included in T.

Example IV.2: The BTS shown in Fig. 2(b) is a complete BTS. By picking control decision $\{b, o_1, o_2\}$ (shown as $\{b\}$ in the figure) at the initial Y-state $\{0\}$, no future observable behavior can occur, since the only enabled and feasible event e forms an unobservable self-loop at state 1. This leads to a BTS-included supervisor S_P defined by $S_P(\epsilon) = \{b, o_1, o_2\}$. □

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

The next result states that given a supervisor S_P and a string $s \in \mathcal{L}(S_P/G)$, the Z-state defined above is, in fact, equivalent to the set of all possible states the system can be in after observing $P(s)$.

Lemma IV.1: Given a system G and a supervisor S_P, for any string $s \in \mathcal{L}(S_P/G)$, we have

$$I\left(IS_{S_P}^Z(s)\right) = R_G\left(s, \mathcal{L}(S_P/G)\right). \tag{9}$$

Proof: We prove by induction on the length of $P(s)$. For any string s, let $|P(s)| = n$. Let s_k denote the string that consists of the first k events in $P(s)$ for $k = 0, \ldots, n$ and e_k denote the $(k+1)^{th}$ event in $P(s)$ for $k = 0, \ldots, n-1$, so that $s_0 = \epsilon$, $s_1 = e_0$, etc... Define y_0 as usual. For $k = 0, \ldots, n$, let $z_k = h_{YZ}(y_k, S_P(s_k))$, and for $k = 0, \ldots, n-1$, define $y_{k+1} = h_{ZY}(z_k, e_k)$. By definition, we know that

$$R_G\left(s, \mathcal{L}(S_P/G)\right) = \left\{ v \in X : \begin{array}{c} \exists s' \in \mathcal{L}(S_P/G) \text{ s.t.} \\ P(s) = P(s') \wedge v = \delta(x_0, s') \end{array} \right\}. \tag{10}$$

Therefore, the inductive hypothesis is that

$$I(z_k) = \left\{ v \in X : \begin{array}{c} \exists s'_k \in \mathcal{L}(S_P/G) \text{ s.t.} \\ P\left(s'_k\right) = s_k \wedge v = \delta\left(x_0, s'_k\right) \end{array} \right\}. \tag{11}$$

Induction Basis: $s_0 = \epsilon$

$$\begin{aligned} I(z_0) &= \mathrm{UR}_{S_P(\epsilon)}(y_0) \\ &= \{v \in X : \exists t \in (S_P(\epsilon) \cap \Sigma_{uo})^* \text{ s.t. } v = \delta(x_0, t)\} \\ &= \{v \in X : \exists t \in \mathcal{L}(S_P/G) \text{ s.t. } P(t) = \epsilon \wedge v = \delta(x_0, t)\}. \end{aligned}$$

Induction Step:

Assume that the induction hypothesis is true at k. Then

$$\begin{aligned} y_{k+1} &= h_{ZY}(z_k, e_k) \\ &= \{v \in X : \exists u \in I(z_k) \text{ s.t. } v = \delta(u, e_k)\} \\ &= \left\{ v \in X : \begin{array}{c} \exists s'_k \in \mathcal{L}(S_P/G) \text{ s.t.} \\ P(s'_k) = s_k \wedge v = \delta(x_0, s'_k e_k) \end{array} \right\} \\ I(z_{k+1}) &= \mathrm{UR}_{S_P(s_k e_k)}(y_{k+1}) \\ &= \left\{ v \in X : \begin{array}{c} \exists u \in y_{k+1}, \exists t \in (S_P(s_k e_k) \cap \Sigma_{uo})^* \\ \text{s.t. } v = \delta(u, t) \end{array} \right\} \\ &= \left\{ v \in X : \begin{array}{c} \exists s'_{k+1} \in \mathcal{L}(S_P/G) \\ \text{s.t. } P\left(s'_{k+1}\right) = s_k e_k \wedge v = \delta\left(x_0, s'_{k+1}\right) \end{array} \right\}. \end{aligned}$$

This completes the proof. □

V. A Uniform Approach for Enforcing Properties

In this section, we define the All Enforcement Structure, a specific type of BTS that embeds all supervisors that enforce a given IS-based property in its transition structure. (How a BTS embeds supervisors will be defined explicitly later.) We then discuss its properties and its construction.

A. All Enforcement Structure for a Given Property

In Lemma IV.1, we have shown that given a supervisor S_P, for any string $s \in \mathcal{L}(G)$, the Z-state $IS_{S_P}^Z(s)$ reached is the set of all possible states the system could be in after s. Therefore, it is clear that any IS-based property can be formulated as a Z-state property. The following result shows that the liveness property can also be transformed to a Z-state property.

Definition V.1. (Deadlock-Free Z-State): A Z-state z is said to be *deadlock-free* if $(\forall\, x \in I(z))(\exists \sigma \in \Gamma(z))[\delta(x, \sigma)!]$. Otherwise, it is said to be a *deadlock* state.

Lemma V.1: Given a system G and a supervisor S_P, the controlled system is live if and only if any reachable Z-state under S_P is deadlock-free. Mathematically

$$S_P/G \text{ is live} \Leftrightarrow \forall\, s \in \mathcal{L}(S_P/G) : IS_{S_P}^Z(s) \text{ is deadlock-free.}$$

Proof: We proceed by contrapositive. if and only if $\exists s \in \mathcal{L}(S_P/G) : \Delta_{\mathcal{L}(S_P/G)}(s) = \emptyset$, which is equivalent to

$$\exists s \in \mathcal{L}(S_P/G), \forall\, \sigma \in S_P(s) : \delta(x_0, s\sigma)\ \not!. \tag{12}$$

By Lemma IV.1, (12) holds if and only if

$$\exists s \in \mathcal{L}(S_P/G), \exists x \in I\left(IS_{S_P}^Z(s)\right), \quad \forall\, \sigma \in \Gamma\left(IS_{S_P}^Z(s)\right) : \delta(x, \sigma)\ \not!. \tag{13}$$

To see the equivalence between (12) and (13), first suppose that (12) holds. Then (13) holds by taking the same string s in (12) and state $x = \delta(x_0, s)$. If (13) holds, then there exists a string $t \in \mathcal{L}(S_P/G)$ such that $P(s) = P(t)$ and $\delta(x_0, t) = x$. Then (12) holds by taking such a string t. Moreover, by Definition V.1, (13) is equivalent to

$$\exists s \in \mathcal{L}(S_P/G) : IS_{S_P}^Z(s) \text{ is deadlock.} \tag{14}$$

This completes the contrapositive proof. □

As a consequence of Lemmas IV.1 and V.1, if we construct a BTS that is "as large as possible" and in which all reachable Z-states are deadlock-free and satisfy the IS-based property, then the resulting structure should contain all valid property-enforcing supervisors. This leads to the definition of the All Enforcement Structure for a given property.

Definition V.2. (All Enforcement Structure): Given a system G and an IS-based property $\varphi : I \to \{0, 1\}$ w.r.t. G, the All Enforcement Structure (AES) for property φ, denoted by $\mathcal{AES}_\varphi(G) = (Q_Y^{AES}, Q_Z^{AES}, h_{YZ}^{AES}, h_{ZY}^{AES}, \Sigma, \Gamma, y_0^{AES})$, is defined as the largest BTS w.r.t. G such that

1) For any $y \in Q_Y^{AES}$, we have that $|C_{\mathcal{AES}_\varphi(G)}(y)| \geq 1$;
2) For any $z \in Q_Z^{AES}$, we have that
   2.1) $\forall\, e \in \Gamma(z) \cap \Sigma_o : (\exists x \in I(z) : \delta(x, e)!) \Rightarrow h_{ZY}^{AES}(z, e)!$;
   2.2) $\varphi(I(z)) = 1$.
   2.3) z is deadlock-free.

By "largest" subsystem, we mean that for any T satisfying the above conditions, we have that $T \sqsubseteq \mathcal{AES}_\varphi(G)$.

Conditions 1 and 2.1 simply say that the AES is a complete BTS. Note that if T_1 and T_2 are two BTSs that satisfy the above conditions, then it is easy to see that the union of them will still satisfy these conditions, where the union of T_1 and T_2 is a BTS $T_1 \cup T_2$, defined by: 1) $Q_Y^{T_1\cup T_2} = Q_Y^{T_1} \cup Q_Y^{T_2}, Q_Z^{T_1\cup T_2} = Q_Z^{T_1} \cup Q_Z^{T_2}$; and 2) for any $y \in Q_Y^{T_1\cup T_2}, z \in Q_Z^{T_1\cup T_2}, \gamma \in \Gamma$ and $e \in \Sigma$, we have that $h_{YZ}^{T_1\cup T_2}(y,\gamma) = z \Leftrightarrow \exists i \in \{1,2\} : h_{YZ}^{T_i}(y,\gamma) = z$ and $h_{ZY}^{T_1\cup T_2}(z,e) = y \Leftrightarrow \exists i \in \{1,2\} : h_{ZY}^{T_i}(z,e) = y$. Therefore, the notion of "largest BTS" in the definition is well defined. This will also be seen when we present the algorithm for the construction of the AES later.

Example V.1: We return to system G in Fig. 1(a) with the IS-based property in (5). The BTS shown in Fig. 2(b) is, in fact, its AES. For example, at initial Y-state $\{0\}$, we cannot make control decision $\{a,b,c\}$, which would lead us to Z-state $(\{0,1,2,3,4\},\{a,b,c\})$. This is because upon the occurrence of event o_2, Y-state $\{5\}$ would be reached, from which no matter what control decision we take, the secret will be revealed. We will discuss later that how to construct the AES. □

Remark V.1: In Fig. 2(a), we can also take control decision $\{a\}$ at the initial Y-state $y_0 = \{0\}$. However, this control decision is equivalent to decision $\{\}$, since event a will never be executed within the unobservable reach. Formally, we say that a control decision $\gamma \in \Gamma$ is *irredundant* at information state $i \in I$ if, for any $\sigma \in \gamma$, there exists $x \in \text{UR}_\gamma(i)$ such that $f(x,\sigma)$ is defined. From now on, we only consider irredundant control decisions in the AES, which will clearly not affect its properties. Similarly, we say that a supervisor S_P is irredundant if for any $s \in \mathcal{L}(S_P/G)$, control decision $S_P(s)$ is irredundant at information state $IS_{S_P}^Y(s)$. Hereafter, we also assume without loss of generality that any S_P is irredundant.

The following theorem shows that the AES (only) contains valid solutions to the property enforcement problem.

Theorem V.1: Suppose that φ is an IS-based property w.r.t. G. A live supervisor enforces property φ if and only if it is an AES-included supervisor. Mathematically

$$[S_P/G \text{ is live}] \wedge [\mathcal{L}(S_P/G) \models_G \varphi] \Leftrightarrow S_P \in \mathcal{S}(\mathcal{AES}_\varphi(G)). \quad (15)$$

Proof: By Lemmas IV.1 and V.1, we know that the LHS of (15) holds if and only if for any $s \in \mathcal{L}(S_P/G)$

$$\varphi\left(I\left(IS_{S_P}^Z(s)\right)\right) = 1 \wedge IS_{S_P}^Z(s) \text{ is deadlock-free.} \quad (16)$$

Therefore, the "if" part follows immediately from Definitions IV.3 and V.2.

Next, we prove the "only if" part by contradiction. Suppose that S_P is supervisor such that for any $s \in \mathcal{L}(S_P/G)$, (16) holds. We assume that $S_P \notin \mathcal{S}(\mathcal{AES}_\varphi(G))$. First, we know that there exists a complete BTS T such that $S_P \in \mathcal{S}(T)$. Specifically, the complete BTS T can be constructed as follows: $Q_Y^T := \{y \in I : \exists s \in \mathcal{L}(S_P/G) \text{ s.t. } y = IS_{S_P}^Y(s)\}$, $Q_Z^T := \{z \in I \times \Gamma : \exists s \in \mathcal{L}(S_P/G) \text{ s.t. } z = IS_{S_P}^Z(s)\}$ and for any $y \in Q_Y^T$, $C_T(y) := \{\gamma \in \Gamma : \exists s \in \mathcal{L}(S_P/G) \text{ s.t. } y = IS_{S_P}^Y(s) \wedge \gamma = S_P(s)\}$. In other words, any Y or Z-state in T are a Y or Z-state reached by supervisor S_P under some string $t \in \mathcal{L}(S_P/G)$, respectively. By Lemmas IV.1 and V.1, we know that T satisfies both Conditions 1 and 2 in Definition V.2. Since $S_P \notin \mathcal{S}(\mathcal{AES}_\varphi(G))$, we know that there exists a string $s \in \mathcal{L}(S_P/G)$, such that $S_P(s) \notin C_{\mathcal{AES}_\varphi(G)}(IS_{S_P}^Y(s))$. In this case, the union of T and $\mathcal{AES}_\varphi(G)$ is strictly larger than $\mathcal{AES}_\varphi(G)$, since control decision $S_P(s)$ is defined at Y-state $IS_{S_P}^Y(s)$ in $T \cup \mathcal{AES}_\varphi(G)$ but not in $\mathcal{AES}_\varphi(G)$. This is a contradiction since by definition, $\mathcal{AES}_\varphi(G)$ is the largest BTS satisfying Conditions 1 and 2 in Definition V.2. □

Remark V.2: In Problem III.3, we have required that the solution synthesized should be live, since the property under consideration may depend on the liveness assumption in general. However, in the case where the property to be enforced does not depend on the liveness assumption, as is the case with *safety* for instance, this requirement can be relaxed. In this case, we remove Condition 2.3 from Definition V.2. It is straightforward to show that in the resulting modified AES, instead of the result in Theorem V.1, we have instead that $\mathcal{L}(S_P/G) \models_G \varphi \Leftrightarrow S_P \in \mathcal{S}(\mathcal{AES}_\varphi(G))$. In other words, the modified AES will contain all property-enforcing supervisors, resulting in live or non live behavior.

B. Construction of the AES

The construction algorithm for the AES follows directly from its definition and proceed in two steps. First, we construct the BTS that enumerates all possible behaviors for each state by a depth-first search and remove all Z-states that violate either the IS-based property or deadlock-freeness, i.e., Conditions 2.2 and 2.3 in Def. V.2. Second, we prune states that violate Conditions 1 or 2.1 in Def. V.2 from the remaining part of the BTS, until convergence is achieved. In practice, in the depth-first search part, we do not need to search the whole state space and we can stop the search of a branch once a Z-state that violates the IS-based property is encountered.

Algorithm 1: FIND-AES

input : G and φ
output: AES

$AES.Y \leftarrow \{y_0\}, AES.Z \leftarrow \emptyset$ and $AES.h \leftarrow \emptyset$;
DoDFS(G, y_0, AES);
Prune(AES);
$AES \leftarrow$ Accessible(AES);

procedure DoDFS(G, y, AES, φ);
  for $\gamma \in \Gamma$ **do**
    $z \leftarrow h_{YZ}(y,\gamma)$;
    if $\varphi(I(z)) = 1$ ***and*** *z is deadlock-free* **then**
      $AES.h \leftarrow AES.h \cup \{(y,\gamma,z)\}$;
      if $z \notin AES.Z$ **then**
        $AES.Z \leftarrow AES.Z \cup \{z\}$;
        for $e \in \gamma \cap \Sigma_o$ **do**
          $y' \leftarrow h_{ZY}(z,e)$;
          $AES.h \leftarrow AES.h \cup \{(z,e,y')\}$;
          if $y' \notin AES.Y$ **then**
            $AES.Y \leftarrow AES.Y \cup \{y'\}$;
            DoDFS(G, y', AES, φ);

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

The above procedure is formally described in Algorithm FIND-AES whose parameters are as follows: (i) AES represents the AES that we want to construct; (ii) AES.Y and AES.Z are its sets of Y- and Z-states, respectively; and (iii) AES.h is its transition function. Initially, AES.Y is set to be $y_0 = \{x_0\}$. The depth-first search is then started; it is implemented by the procedure DoDFS. Line 7 is used to determine whether the Z-state encountered satisfies both deadlock-freeness and property φ. If not, we terminate the search of this branch. Otherwise, we compute all possible Y-state successors and make a recursive call. This recursive procedure allows us to traverse the whole reachable space of Y- and Z-states. The above procedure may result in Y-states that have no successors. Therefore, we need to iteratively prune: (i) all Y-states that have no successor states; and (ii) all Z-states for which at least one observation is not defined. This step is captured by procedure Prune. Finally, states that are no longer accessible from the initial state of the AES need to be removed before the algorithm returns. Algorithm FIND-AES will terminate in finite steps, since the number of Y- and Z-states is finite.

Example V.2: Consider our running example. We apply Algorithm FIND-AES to construct the corresponding AES. The resulting BTS after running the procedure DoDSF is shown in Fig. 2(a). At the initial Y-state, we cannot take control decision {} since this will lead to a deadlock Z-state ({0},{}). The depth-first search DoDSF terminates at Y-state {5}, since no matter what control decision we take from {5}, a Z-state [marked in red in Fig. 2(a)] that violates the IS-based property (i.e., that reveals the secret) will be encountered. After procedure DoDSF is done, we need to run procedure Prune. This starts by removing Y-state {5}, since no successor state is defined from it. Since Y-state {5} has been removed, all its predecessor Z-states, i.e., $(\{0,3\},\{c\}), (\{0,1,2\},\{a,b\})$ and so forth, must also be removed. Finally, we remove inaccessible states {2, 5, 6} and {2} and obtain the AES shown in Fig. 2(b). □

Theorem V.2: Algorithm FIND-AES correctly constructs the AES.

Proof: Let T be the BTS returned by Algorithm FIND-AES. T satisfies Conditions 1 and 2.1 in Def. V.2, since all states that violate either one of these two conditions have been removed by procedure Prune. Since procedure DoDSF only traverses the state space where all Z-states are deadlock-free and satisfy φ, Conditions 2.2 and 2.3 in Def. V.2 are also satisfied by T. Therefore, it remains to show that T is the largest BTS with the desired properties; for that proof, we proceed by contradiction.

Assume that T' is another BTS satisfying the conditions in Def. V.2 that is strictly larger than T, i.e., $T \sqsubseteq T'$ and $Q_Y^T \cup Q_Z^T \subset Q_Y^{T'} \cup Q_Z^{T'}$. Therefore, in Algorithm FIND-AES, T is obtained by pruning states from some BTS T'' (which may not satisfy the desired properties) such that $T' \sqsubseteq T''$; e.g., T'' can be the resulting BTS after procedure DoDFS. Then any Y- or Z-states in T' satisfying the conditions in Def. V.2 should also satisfy these conditions in T'', since T' is a subsystem of T''. In other words, any Y- or Z-states in T' will not be removed in T'' by procedure Prune. Therefore, Algorithm FIND-AES will converge to a BTS that is strictly lager than T (at least as large as T'). This contradicts the fact that Algorithm FIND-AES converges to T. □

VI. Synthesis of Maximally Permissive Supervisors

In this section, we present a synthesis algorithm that returns a solution to MPIEP. We first discuss the general case, where Σ_c and Σ_o need not be comparable. Then we show that, under the assumption that $\Sigma_c \subseteq \Sigma_o$, there always exists a unique supremal solution to MPIEP.

A. General Case

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

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

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

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

The following theorem establishes the correctness of Algorithm MAX-SYNT:

Algorithm 2: MAX-SYNT

input : $\mathcal{AES}_\varphi(G)$
output: S^*

1 $T.Y \leftarrow \{y_0\}, T.Z \leftarrow \emptyset$ and $T.h \leftarrow \emptyset$;
2 Expand($T, \mathcal{AES}_\varphi(G), y_0$);
3 $S^* \leftarrow S_T$;

procedure Expand($T, \mathcal{AES}_\varphi(G), y$);
4 Find a locally maximal control decision $\gamma \in C_{\mathcal{AES}_\varphi(G)}(y)$ s.t. $\forall \gamma' \in C_{\mathcal{AES}_\varphi(G)}(y) : \gamma \not\subset \gamma'$;
5 $z \leftarrow h_{YZ}(y,\gamma)$;
6 $T.h \leftarrow T.h \cup \{(y,\gamma,z)\}$;
7 **if** $z \notin T.Z$ **then**
8 $T.Z \leftarrow T.Z \cup \{z\}$;
9 **for** $e \in \gamma \cap \Sigma_o$ **do**
10 $y' \leftarrow h_{ZY}(z,e)$;
11 $T.h \leftarrow T.h \cup \{(z,e,y')\}$;
12 **if** $y' \notin T.Y$ **then**
13 $T.Y \leftarrow T.Y \cup \{y'\}$;
14 Expand($T, \mathcal{AES}_\varphi(G), y'$);

Theorem VI.1: Let S^* be a solution returned by Algorithm MAX-SYNT. Then S^* solves MPIEP.

Proof: First, we note that $\mathcal{L}(S_P/G)$ is live and satisfies φ; this follows from Theorem V.1 and the fact that, by construction, S^* is an AES-included supervisor. Therefore, it remains to show that S^* is maximal; for that proof, we proceed by contradiction.

Assume that S^* is not maximal, which means that there exists another AES-included supervisor $S' \in \mathcal{S}(\mathcal{AES}_\varphi(G))$ such that $\mathcal{L}(S^*/G) \subset \mathcal{L}(S'/G)$. Therefore, there exists a string $w \in \mathcal{L}(S^*/G) \subset \mathcal{L}(S'/G)$ such that $S^*(w) \subset S'(w)$ and $S^*(w') = S'(w'), \forall\, w' \in \overline{\{w\}} \setminus \{w\}$. We know that $IS^Y_{S^*}(w) = IS^Y_{S'}(w)$; let us call this Y-state y. But this means that the control decision $S^*(w)$ at y violates the locally maximal construction rule, i.e., there should not exist a control decision $\gamma \in C_{\mathcal{AES}_\varphi(G)}(y) : S^*(w) \subset \gamma$. This is a contradiction. Hence no such S' exists. □

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

Corollary VI.1: MPIEP is solvable if and only if the AES is non-empty.

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

Corollary VI.2: For any IS-based property φ, there exists an IS-based supervisor that solves MPIEP iff $\mathcal{AES}_\varphi(G)$ is non-empty.

Example VI.1: We return to our running example. If we pick locally maximal control decision $\{a, c\}$ at the initial Y-state $\{y_0\}$ and pick the unique control decision $\emptyset$ at the reachable Y-state, which means that all controllable events are disabled, then we will obtain the maximal solution that was shown earlier in Fig. 1(b). On the other hand, if we pick control decision $\{b\}$ at $\{0\}$, which is also locally maximal, then no observable behavior can occur thereafter; this corresponds to the maximal solution shown in Fig. 1(c). □

Remark VI.1: The running time of the entire synthesis procedure is $O(2^{2|X|+2|\Sigma_c|})$. First, we need to construct the AES by Algorithm FIND-AES, which consists of two procedures, DoDSF and Prune. The procedure DoDSF may result in a BTS that, in the worst case, has $2^{|X|+|\Sigma_c|} + 2^{|X|}$ states. The complexity of procedure Prune is quadratic in the size of the above BTS. The complexity of Algorithm MAX-SYTN is linear in the size of the AES that, in the worst case, also has $2^{|X|+|\Sigma_c|} + 2^{|X|}$ states. Therefore, our synthesis procedure is exponential in the size of G. However, it was shown in [36] that synthesizing a partial observation safe supervisor, which is a special case of our problem, is NP-hard. Therefore, this exponential complexity seems to be unavoidable and it is due to the partial observation feature of our problem.

B. Case of $\Sigma_c \subseteq \Sigma_o$

It was shown in [33] that, under the assumption that $\Sigma_c \subseteq \Sigma_o$, observability and controllability together imply normality. Therefore, there exists a supremal controllable and observable sublanguage when $\Sigma_c \subseteq \Sigma_o$. It was also reported in [20] (respectively, [17] and [9]) that, under the assumption that $\Sigma_c \subseteq \Sigma_o$, there exists a supremal controlable, observable and diagnosable (respectively, opaque and attractable) sublanguage. In fact, we can prove the corresponding general result for *any IS-based property* in our framework.

The following lemma reveals that, under the assumption that $\Sigma_c \subseteq \Sigma_o$, the information state encountered does not depend on the control policy we take.

Lemma VI.1: Let S^1_P and S^2_P be two supervisors. Under the assumption that $\Sigma_c \subseteq \Sigma_o$, we have that

$$\left(\forall\, s \in \mathcal{L}(S^1_P/G) \cap \mathcal{L}(S^2_P/G)\right) \left[I\left(IS^Z_{S^1_P}(s)\right) = I\left(IS^Z_{S^2_P}(s)\right)\right]. \tag{17}$$

Proof: We prove this lemma by induction on the length of $P(s)$. For any string s, let $|P(s)| = n$. Let s_k and e_k be the same notations defined in the proof of Lemma IV.1. For any $i = 1, 2$, define y^i_0 as usual and for $k = 0, \ldots, n$, let $z^i_k = h_{YZ}(y^i_k, S^i_P(s_k))$, and for $k = 0, \ldots, n-1$, define $y^i_{k+1} = h_{ZY}(z^i_k, e_k)$. Therefore, the inductive hypothesis is that

$$I\left(z^1_k\right) = I\left(z^2_k\right). \tag{18}$$

Induction Basis ($s_0 = \epsilon$):

$$\begin{aligned} I(z^2_0) &= \mathrm{UR}_{S^2_P(\epsilon)}(y_0) \\ &= \left\{ v \in X : \exists t \in \left(S^2_P(\epsilon) \cap E_{uo}\right)^* \text{ s.t. } v = \delta(x_0, t)\right\} \\ &= \left\{ v \in X : \exists t \in \left(S^1_P(\epsilon) \cap E_{uo}\right)^* \text{ s.t. } v = \delta(x_0, t)\right\} \\ &= \mathrm{UR}_{S^1_P(\epsilon)}(y_0) = I\left(z^1_0\right) \end{aligned}$$

where $S^2_P(\epsilon) \cap \Sigma_{uo} = S^1_P(\epsilon) \cap \Sigma_{uo}$ holds because $\Sigma_c \cap \Sigma_{uo} = \emptyset$.

Induction Step:

Assume that the induction hypothesis is true at k. Then

$$\begin{aligned} y^2_{k+1} &= h_{ZY}(z^2_k, e_k) \\ &= \left\{ v \in X : \exists u \in I\left(z^2_k\right) \text{ s.t. } v = \delta(u, e_k)\right\} \\ &= \left\{ v \in X : \exists u \in I\left(z^1_k\right) \text{ s.t. } v = \delta\left(u, e_k\right)\right\} \\ &= y^1_{k+1} \\ I\left(z^2_{k+1}\right) &= \mathrm{UR}_{S^2_P(s_k e_k)}\left(y^2_{k+1}\right) = \mathrm{UR}_{S^2_P(s_k e_k)}\left(y^1_{k+1}\right) \\ &= \mathrm{UR}_{S^1_P(s_k e_k)}\left(y^1_{k+1}\right) \\ &= I\left(z^1_{k+1}\right) \end{aligned}$$

where $\mathrm{UR}_{S^2_P(s'_k e_k)}(y^1_{k+1}) = \mathrm{UR}_{S^1_P(s'_k e_k)}(y^1_{k+1})$ follows from the same argument as in the induction basis.

This completes the proof by induction. □

Consider two different supervisors; under the assumption that $\Sigma_c \subseteq \Sigma_o$, the information state components of the Z-states encountered upon the occurrence of the same string are identical. Therefore, in this scenario, state estimation (from observed events) does not depend on the control policy we take. In [37],

the authors show that for centralized partial observation control problems, a given Z-state (termed as maximal information set in [37]) is independent from the control policy the supervisor will take in the future. (This is not true in general in decentralized control; see again [37].) In essence, Lemma VI.1 extends this result and says that, under the assumption that $\Sigma_c \subseteq \Sigma_o$, control and state estimation are one-way "fully separated," i.e., in addition to the non-dependency of state estimation on the future control actions, the Z-state even does not depend on the past control actions. This separability also leads to the follows theorem, which says that for any IS-based property, under the assumption that $\Sigma_c \subseteq \Sigma_o$, there exists a unique maximal permissive supervisor that enforces the property.

Theorem VI.2: Assume that $\Sigma_c \subseteq \Sigma_o$. Then there exists a unique (supremal) solution to MPIEP.

Proof: By contradiction. Suppose that $\varphi : I \to \{0,1\}$ is the IS-based property that we want to enforce. We assume that S_P^1 and S_P^2 are two different solutions to MPIEP, i.e., $\mathcal{L}(S_P^1/G)$ and $\mathcal{L}(S_P^2/G)$ are two incomparable maximal controllable and observable sublanguages satisfying φ. Under the assumption that $\Sigma_c \subseteq \Sigma_o$ and that the two given languages are controllable and observable, we know that their union will also be controllable and observable. Hence, there exists a partial observation supervisor S_P^* such that $\mathcal{L}(S_P^*/G) = \mathcal{L}(S_P^1/G) \cup \mathcal{L}(S_P^2/G)$. Specifically, for any $s \in \mathcal{L}(S_P^*/G)$ we have $S_P^*(s) = S_P^1(s) \cup S_P^2(s)$.

Next, we show that S_P^* also enforces φ. Let us assume that S_P^* does not enforce property φ, i.e., $\exists s \in \mathcal{L}(S_P^*/G)$ s.t. $\varphi(R_G(s, \mathcal{L}(S_P^*/G))) = 0$. Since $\mathcal{L}(S_P^*/G) = \mathcal{L}(S_P^1/G) \cup \mathcal{L}(S_P^2/G)$, we know that $\exists i \in \{1,2\}$ s.t. $s \in \mathcal{L}(S_P^i/G)$. By Lemma VI.1, we know that $I(IS_{S_P^*}^Z(s)) = I(IS_{S_P^i}^Z(s))$. Moreover, by Lemma IV.1, we know that $R_G(s, \mathcal{L}(S_P^*/G)) = R_G(s, \mathcal{L}(S_P^i/G))$. However, $\varphi(R_G(s, \mathcal{L}(S_P^i/G))) = 1$, since S_P^i enforces property φ. This implies that $\varphi(R_G(s, \mathcal{L}(S_P^*/G))) = 1$, which is a contradiction. Therefore, S_P^* also enforces property φ.

Using the above result, we conclude that $\mathcal{L}(S_P^1/G) \cup \mathcal{L}(S_P^2/G)$ is also a controllable and observable sublanguage satisfying φ, which contradicts the fact that $\mathcal{L}(S_P^1/G)$ and $\mathcal{L}(S_P^2/G)$ are maximal. Therefore, there only exists a unique solution to MPIEP. □

We have shown that Algorithm MAX-SYNT always returns a maximal solution; moreover, under the assumption that $\Sigma_c \subseteq \Sigma_o$, this maximal solution is unique. Therefore, in this scenario, Algorithm MAX-SYNT returns the unique supremal solution.

Corollary VI.3: Let S^* be the solution returned by Algorithm MAX-SYNT. When $\Sigma_c \subseteq \Sigma_o$, S^* is the unique supremal solution to MPIEP.

Remark VI.2: In the standard supervisory control problem, the supremal controllable and observable sublanguage can be obtained under the assumption that $\Sigma_c \subseteq \Sigma_o$ by computing the supremal controllable and normal sublanguage [13]. Since both the supermal normal approach and Algorithm MAX-SYNT take exponential complexity in the size of the system, our approach does not improve upon the complexity of the previous result under this restrictive assumption. Instead, Algorithm MAX-SYNT provides an alternative approach for the computation of supermal controllable and normal sublanguage for this special case.

VII. Applications of the Uniform Approach

In this section, we show that how to apply the uniform approach described in this paper to the enforcement of several specific properties commonly encountered in the study of DES. Our uniform approach comprises three steps:

1) Formulate the property to be enforced as an IS-based property;
2) Construct the AES using Algorithm FIND-AES;
3) Find a maximal solution based on the AES using Algorithm MAX-SYNT.

In Sections V and VI, we have discussed Steps 2 and 3, respectively; it remains to discuss how to formulate a given property as an IS-based property, whenever feasible. As was mentioned earlier, there are properties that cannot be formulated as IS-based properties; one such example is non-blockingness [32]. However, as we will see in this section, many important properties in the DES literature, including but not restricted to safety, opacity, diagnosability, detectbility, anonymity and attractability, can be formulated as IS-based properties. Therefore, all of them can be enforced by using the above three-step methodology.

A. Enforcement of Safety

Given a prefix-closed specification language K, we say that language $L \subseteq \mathcal{L}(G)$ is safe if $L \subseteq K$. When the uncontrolled system is not safe, the standard supervisory control and observation problem [11], [12] asks to synthesize a least restrictive supervisor such that the controlled system is safe. We show that this can be solved by our uniform approach.

Let $K = \mathcal{L}(H)$, for some automaton H. In [13], the authors provide an algorithm to construct refined automata $H_S = (X_{H_S}, \Sigma, \delta_{H_S}, x_{0,H_S})$ and $G_S = (X_{G_S}, \Sigma, \delta_{G_S}, x_{0,G_S})$ such that the following holds: 1) $\mathcal{L}(G_S) = \mathcal{L}(G)$ and $\mathcal{L}(H_S) = \mathcal{L}(H)$; 2) H_S is a sub-automaton of G_S; 3) For any $x, y \in X_{H_S}, \sigma \in \Sigma$, we have $\delta_{G_S}(x,\sigma) = y \Rightarrow \delta_{H_S}(x,\sigma) = y$. For the construction of G_S and H_S, the reader is referred to [13]. The above conditions imply that X_{H_S} captures the legal behaviors, i.e., any string in $\mathcal{L}(G_S)$ that leads to a state in X_{H_S} is safe and any string in $\mathcal{L}(G_S)$ that leads to a state in $X_{G_S} \setminus X_{H_S}$ is unsafe.

For the refined system model G_S, we define the IS-based property $\varphi_{\text{safe}} : 2^{X_{G_S}} \to \{0,1\}$ w.r.t. G_S as follows. For any information state $i \in 2^{X_{G_S}}$, $\varphi_{\text{safe}}(i) = 1 \Leftrightarrow i \subseteq X_{H_S}$. Then we have the following result.

Proposition VII.1: Let H be the specification automaton and G_S be the refined system automaton defined above, then language $L \subseteq \mathcal{L}(G)$ is safe if and only if $L \models_{G_S} \varphi_{\text{safe}}$.

Proof: The proof follows directly from Definition III.2 and Lemma IV.1, since $L \not\models_{G_s} \varphi_{\text{safe}}$ if and only if $\exists s \in L : \delta_{G_S}(x_{0,G_S}, s) \notin X_{H_S}$, which is equivalent to $L \not\subseteq \mathcal{L}(H)$. □

Hence, to solve the safety control problem, it suffices to synthesize a supervisor that enforces the IS-based property φ_{safe} w.r.t. the refined state space of G_S. Therefore, the maximally permissive safety control problem can be solved by our uniform approach.

B. Enforcement of Current-State Opacity

Opacity is a confidentiality property for partially-observed systems. It captures the plausible deniability of the system's "secret" in the presence of an outside observer that is potentially malicious. First, we recall the definition of *current-state opacity*, as it is presented in [38] and [39].

Definition VII.1: Let $G = (X, \Sigma, \delta, x_0)$ be the system automaton. Language $L \subseteq \mathcal{L}(G)$ is said to be *current-state opaque* w.r.t. $X_S \subseteq X, G$ and P if

$$(\forall\, s \in L : \delta(x_0, s) \in X_S)\,(\exists t \in L)$$
$$[P(s) = P(t) \wedge \delta(x_0, t) \notin X_S]\,. \quad (19)$$

Note that we assume in this section that the external observer and the supervisor have the same observation set, Σ_o.

To formulate the current-state opacity enforcement problem in our framework, we define the IS-based current-state opacity property $\varphi_{\rm opa} : 2^X \to \{0, 1\}$ as follows. For any information state $i \in 2^X$, we have

$$\varphi_{\rm opa}(i) = 0 \Leftrightarrow i \subseteq X_S. \quad (20)$$

The following result says that the IS-based property $\varphi_{\rm opa}$ correctly captures the opacity property.

Proposition VII.2: Let G be the system automaton, $X_S \subseteq X$ be the subset of secret states, and $\varphi_{\rm opa}$ be the IS-based property defined above. Language $L \subseteq \mathcal{L}(G)$ is current-state opaque if and only if $L \models_G \varphi_{\rm opa}$.

Proof: We proceed by contrapositive. By definition, G is not current-state opaque if and only if $(\exists s \in L)(\forall\, t \in L)[P(s) = P(t) \Rightarrow \delta(x_0, t) \in X_S]$, which is equivalent to $(\exists s \in L)(\forall\, x \in R_G(s, L))[x \in X_S]$. This is equivalent to $\exists s \in L : \varphi_{\rm opa}(R_G(s, L)) = 0$, i.e., $L \not\models_G \varphi_{\rm opa}$. □

Consequently, the opacity enforcement problem can be solved by using $\varphi_{\rm opa}$ in our uniform approach. Our running example has already shown how to synthesize a maximally permissive supervisor enforcing opacity.

Remark VII.1: It was shown in [39] that several other notions of opacity, e.g., language-based opacity, initial-state opacity, and initial-and-final-state opacity, can be transformed to current-state opacity in polynomial time. Therefore, the enforcement of these notions of opacity can be done by first transforming them to current-state opacity and then enforcing current-state opacity as discussed above.

C. Enforcement of K-Diagnosability

In fault diagnosis problems, $f \in \Sigma_{uo}$ is a fault event whose occurrences must be diagnosed by the diagnoser within a finite number of steps. Suppose L is the language to be diagnosed. We define $\Psi(f, L) = \{sf \in L : s \in \Sigma^*\}$ to be the set of strings that end with the fault event. We say that a language is K-diagnosable if this diagnosis delay is uniformly bounded by a given number K. The formal definition of K-diagnosability is recalled from [2], [29], and [30].

Definition VII.2. (K-Diagnosability): A live language L is said to be K-diagnosable w.r.t. P and $f \in \Sigma_{uo}$ if

$$(\forall\, s \in \Psi(f, L))\,(\forall\, t \in L/s)$$
$$\left[|t| \geq K \Rightarrow \left(\forall\, w \in P^{-1}\left(P(st)\right) \cap L : f \in w\right)\right]. \quad (21)$$

To formulate K-diagnosability as an IS-based property, we need to refine the state space of the original system G, which is similar to the refinement procedure in [30]. Given $G = (X, \Sigma, \delta, x_0)$ and non-negative integer K, we define the new automaton $G_D = (X_D, \Sigma, \delta_D, x_{D,0})$, where

- $X_D \subseteq X \times \{-1, 0, 1, \ldots, K\}$ is the set of states;
- Σ is the set of events (same as defined in G);
- $\delta_D : X_D \times \Sigma \to X_D$ is the partial transition function that is built from δ in G as follows: for any $u = (x, n) \in X_D, \sigma \in \Sigma$

$$\delta_D(u, \sigma) = \begin{cases} (\delta(x, \sigma), -1)\,, & \text{if } \begin{array}{l} n = -1 \text{ and} \\ \sigma \in \Sigma \setminus \{f\} \end{array} \\ (\delta(x, \sigma), n + 1)\,, & \text{if } \begin{array}{l} 0 \leq n < K \text{ or} \\ n = -1 \wedge \sigma = f \end{array} \\ (\delta(x, \sigma), K)\,, & \text{if } n = K. \end{cases} \quad (22)$$

- $x_{D,0} = (x_0, -1) \in X_D$ is the initial state.

By construction, we have that $\mathcal{L}(G) = \mathcal{L}(G_D)$, i.e., G_D is language-equivalent to G but refines its state space. Therefore, we can analyze the (language-based) property of diagnosability based on the refined system G_D. To this end, we define the IS-based property termed K-diagnosability.

Definition VII.3. (IS-Based K-Diagnosability): The property of IS-based K-diagnosability $\varphi_{\rm diag} : 2^{X_D} \to \{0, 1\}$ w.r.t. G_D is defined by: for any $i \in 2^{X_D}$

$$\varphi_{\rm diag}(i) = 0 \Leftrightarrow (\exists u, v \in i)\,[[u]_n = -1 \wedge [v]_n = K] \quad (23)$$

where $[u]_n$ denotes the integer component of state u.

The following result establishes that to enforce K-diagnosablility, it suffices to enforce the property of IS-based K-diagnosability defined above.

Proposition VII.3: A live language $L \subseteq \mathcal{L}(G) = \mathcal{L}(G_D)$ is K-diagnosable w.r.t. P and f if and only if $L \models_{G_D} \varphi_{\rm diag}$.

We proceed by contrapositive

$$\begin{aligned} &L \text{ is not } K - \text{diagnosable} \\ \Leftrightarrow &\exists t_v = t_{v,1}t_{v,2}, t_u \in L \text{ s.t. } t_{v,1} \in \Psi(f, L) \text{ and} \\ &t_{v,2} \geq K \text{ and } \Sigma_F \notin t_u \text{ and } P(t_u) = P(t_v) && \text{Def. VII.2} \\ \Leftrightarrow &\exists t_v, t_u \in L \text{ s.t. } \left[\delta_K(x_{D,0}, t_v)\right]_n = K \text{ and} \\ &\left[\delta_D(x_{D,0}, t_u)\right]_n = -1 \text{ and } P(t_u) = P(t_v) \\ \Leftrightarrow &\exists t_v \in L \text{ s.t. } \varphi_{\rm diag}\left(R_{G_D}(t_v, L)\right) = 0 && \text{Def. VII.3} \\ \Leftrightarrow &L \not\models_{G_D} \varphi_{\rm diag} && \text{Def. III.2.} \end{aligned}$$

The second equivalence is from the definition of G_D. □

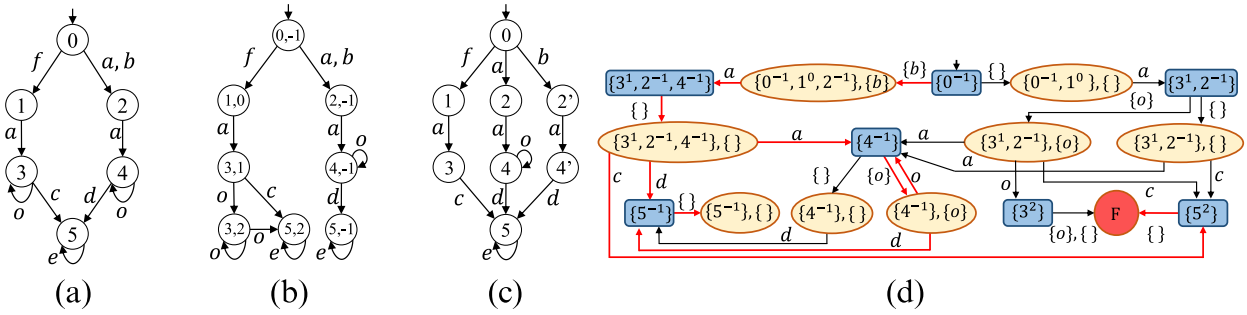

Fig. 3. For G: $\Sigma_c = \{b, o\}, \Sigma_o = \{a, c, d, o\}$, and f is the fault event. For the sake of brevity, in the diagram of the AES, we write state (x, n) in the form of x^n and all uncontrollable events in the control decisions are omitted. We also represent all Z-states z such that $\forall\, x^n \in I(z) : n \geq 0$ as a single state F, since we can diagnose the failure unambiguously at such states. (a) G. (b) G_D. (c) $\mathcal{L}(S^*/G)$. (d) $\mathcal{AES}_{P_{\text{diag}}}(G_D)$.

Example VII.1: Let us consider the system G in Fig. 3(a), where the set of controllable events is $\Sigma_c = \{b, o\}$ and the set of observable events is $\Sigma_o = \{a, c, d, o\}$; these two sets are incomparable. Event f is the unique fault event. Consider a desired diagnosis delay of $K = 2$. The corresponding unfolded system G_D is shown in Fig. 3(b). The corresponding AES $\mathcal{AES}_{P_{\text{diag}}}(G_D)$ for G_D w.r.t. φ_{diag} is given in Fig. 3(d). For the sake of brevity, we write state (x, n) in the form of x^n. For example, at Y-state $\{3^1, 2^{-1}, 4^{-1}\}$, we cannot enable event o, since no matter what control decision we take after the occurrence of o, a Z-state that contains both states 3^2 and 4^{-1} will be encountered, i.e., the IS-based property φ_{diag} will be violated.

By applying Algorithm MAX-SYNT to $\mathcal{AES}_{\varphi_{\text{diag}}}(G_D)$, a maximally permissive supervisor S^* is obtained; we highlight the chosen locally maximal control decision at each reachable Y-state (which in this example is unique) and all feasible observable events at each reachable Z-state in the diagram. The corresponding controlled behavior is given in Fig. 3(c). By Theorem V.1, $\mathcal{L}(S^*/G)$ is a maximal live, controllable, observable and 2-diagnosable sublanguage of $\mathcal{L}(G)$. □

D. Enforcement of Strong Detectability

Detectability is a property arising in state estimation of DES. In [21], the enforcement of strongly detectability is studied under the assumption that $\Sigma_c \subseteq \Sigma_o$. Here, we show that strong detectability with a pre-specified detection delay K, or *strong K-detectability*, can be enforced without such an assumption by using the uniform approach. First, we recall the formal definition of strong K-detectability from [5], [21].

Definition VII.4. (Strong K-Detectability): A live language $L \subseteq \mathcal{L}(G)$ is said to be strongly K-detectable w.r.t. P and G if

$$(\forall\, s \in L)\,[|P(s)| \geq K \Rightarrow |R_G(s, L)| = 1]\,. \qquad (24)$$

Analogous to the enforcement of diagnosability, given an automaton $G = (X, \Sigma, \delta, x_0)$, we can build a new automaton $G_T = (X_T, \Sigma, \delta_T, x_{T,0})$, where

- $X_T \subseteq X \times \{0, 1, \ldots, K\}$ is the set of states;
- Σ is the set of events;
- $\delta_T : X_T \times \Sigma \to X_T$ is the partial transition function and for any $u = (x, n) \in X_T, \sigma \in \Sigma$, δ_T is defined by

$$\delta_T(u, \sigma) = \begin{cases} (\delta(x, \sigma), n)\,, & \text{if } \sigma \in \Sigma_{uo} \wedge n < K \\ (\delta(x, \sigma), n+1)\,, & \text{if } \sigma \in \Sigma_o \wedge n < K \\ (\delta(x, \sigma), K)\,, & \text{if } n = K. \end{cases} \qquad (25)$$

- $x_{T,0} = (x_0, 0)$ is the initial state.

With the refined system G_T, we define IS-based K-detectability as follows.

Definition VII.5. (IS-Based Strong K-Detectability): The property of IS-based K-detectability $\varphi_{\text{det}} : 2^{X_T} \to \{0, 1\}$ w.r.t. G_T is defined by: for any $i \in 2^{X_T}$

$$\varphi_{\text{det}}(i) = 0 \Leftrightarrow (\exists u \in i : [u]_n = K) \wedge |i| > 1 \qquad (26)$$

where $[u]_n$ denotes the integer component of u.

The following result says that to enforce strong K-detectability it suffices to enforce the IS-based property φ_{det} defined above.

Proposition VII.4: A live language L is strongly K-detectable w.r.t. P and G if and only if $L \models_{G_T} \varphi_{\text{det}}$.

Proof: We proceed by contrapositive

L is not strongly $K-$ detectable w.r.t. P and G

$\Leftrightarrow \exists s \in L$ s.t. $|P(s)| \geq K$ and $|R_G(s, L)| > 1$ Def. VII.4

$\Leftrightarrow \exists s \in L$ s.t. $|P(s)| \geq K$ and $|R_{G_T}(s, L)| > 1$

$\Leftrightarrow \exists s \in L$ s.t. $[\delta_T(x_{T,0}, s)]_n = K$ and $|R_{G_T}(s, L)| > 1$

$\Leftrightarrow \exists s \in L$ s.t. $\varphi_{\text{det}}\,(R_{G_T}(s, L)) = 0$

$\Leftrightarrow L \not\models_{G_T} \varphi_{\text{det}}$ Def. III.2.

The third and fourth equivalences follow from the construction of G_T and the definition of φ_{det}, respectively. For the second equivalence, first we have the following observations:

1) $|R_G(s, L)| > 1$ if and only if

$$\exists t \in L : P(s) = P(t) \wedge \delta(x_0, t) \neq \delta(x_0, s). \qquad (27)$$

2) $|R_{G_T}(s,L)|>1$ if and only if

$$\exists t\in L:[P(s)=P(t)]\wedge[\delta(x_0,t)\neq\delta(x_0,s) \text{ or } \max\{|P(t)|,K\}\neq\max\{|P(s)|,K\}]. \quad (28)$$

However, is always true that $\max\{|P(t)|,K\}=\max\{|P(s)|,K\}$ if $P(s)=P(t)$. Therefore, (28) is equivalent to (27), which implies that $|R_G(s,L)|>1\Leftrightarrow|R_{G_T}(s,L)|>1$. □

E. Enforcement of Anonymity

Strong detectability requires that the supervisor eventually be able to determine the exact system state. In security and privacy applications, when the system is monitored by a potentially malicious observer, we may want to enforce the exact opposite, i.e., the exact system state should never be revealed. This is related to the notion of opacity discussed earlier and it is termed *anonymity* [40], which is defined as follows.

Definition VII.6. (Anonymity): Language $L\subseteq\mathcal{L}(G)$ is said to be anonymous w.r.t. P and G if

$$(\forall\, s\in L)(\exists t\in L)\,[P(s)=P(t)\wedge\delta(x_0,s)\neq\delta(x_0,t)]. \quad (29)$$

Anonymity is different from either detectability or opacity. However, anonymity can be easily formulated as an IS-based property, which means that it can enforced by using the uniform approach. To this end, we define the IS-based property $\varphi_{\text{ano}}:2^X\to\{0,1\}$ w.r.t. G by: for any $i\in 2^X$, $\varphi_{\text{ano}}(i)=0\Leftrightarrow|i|=1$. Then we have the following result, which says that enforcing anonymity is equivalent to enforcing IS-based property φ_{ano}.

Proposition VII.5: Language L is anonymous w.r.t. P and G if and only if $L\models_G\varphi_{\text{ano}}$.

Proof: L is not anonymous if and only if $(\exists s\in L)(\forall\, t\in L)[P(s)=P(t)\Rightarrow\delta(x_0,s)=\delta(x_0,t)]$, This is equivalent to $(\exists s\in L)[|R_G(s,L)|=1]$, i.e., $L\not\models_G\varphi_{\text{ano}}$. □

F. Enforcement of Attractability

The last property enforcement problem we study in this section is the state attraction problem. In this problem, the goal is to design a supervisor such that the controlled system will converge to a desired *attractor* in a bounded number of event occurrences. In [9], the state attraction problem under partial observation is studied under the assumption that $\Sigma_c\subseteq\Sigma_o$. We show that this assumption can be relaxed by taking the uniform approach developed in this paper. Hereafter, instead of allowing arbitrary bounded convergence delay, we require that the system converge to the attractor in a pre-specified number of steps, leading to the notion of K-attractability.

Definition VII.7. (K-Attractability): Let $G=(X,\Sigma,\delta,x_0)$ be the system automaton. Language $L\subseteq\mathcal{L}(G)$ is said to be K-attractable w.r.t. G and $A\subseteq X$ if for any $s\in L$, we have

1) $|s|\geq K\Rightarrow\delta(x_0,s)\in A$;
2) $\delta(x_0,s)\in A\Rightarrow\forall\, st\in L:\delta(x_0,st)\in A$.

Remark VII.2: In [9], the authors assume that $A\subseteq X$ is an invariant set, i.e., $(\forall\, x\in A)(\forall\, s\in\Sigma^*)[\delta(x,s)\in A]$. In this case, the second requirement in Definition VII.7 will be satisfied trivially. Therefore, the definition of attractability we consider here is more general.

Given an automaton $G=(X,\Sigma,\delta,x_0)$, to formulate K-attractability as an IS-based property, we first construct the new automaton $G_A=(X_A,\Sigma,\delta_A,x_{A,0})$, where

- $X_A\subseteq X\times\{0,1,\ldots,K\}$ is the set of states;
- Σ is the set of events;
- $\delta_A:X_A\times\Sigma\to X_A$ is the partial transition function and for any $u=(x,n)\in X_A,\sigma\in\Sigma$, δ_T is defined by

$$\delta_A(u,\sigma)=\begin{cases}(\delta(x,\sigma),n+1), & \text{if } n<K\wedge x\notin A\\ (\delta(x,\sigma),K), & \text{if } n<K\wedge x\in A \text{ or } n=K.\end{cases} \quad (30)$$

- $x_{A,0}=(x_0,0)$ is the initial state.

We define IS-based K-attractability as follows.

Definition VII.8. (IS-Based K-Attractability): The property of IS-based K-attractability $\varphi_{\text{att}}:2^{X_A}\to\{0,1\}$ w.r.t. G_A is defined by: for any $i\in 2^{X_A}$

$$\varphi_{\text{att}}(i)=0\Leftrightarrow\exists u\in i:[u]_x\notin A\wedge[u]_n=K \quad (31)$$

where $[u]_x$ and $[u]_n$ are the state component and the integer component of u, respectively.

The following result says that to enforce K-attractability w.r.t. G, it suffices to enforce the IS-based property φ_{att} w.r.t. G_A defined above.

Proposition VII.6: A live language L is K-attractable w.r.t. G if and only if $L\models_{G_A}\varphi_{\text{att}}$.

Proof: We proceed by contrapositive

$$\begin{aligned}
&L \text{ is not } K-\text{attractable w.r.t. } G\\
\Leftrightarrow&\exists s\in L \text{ s.t. } [|s|\geq K\wedge\delta(x_0,s)\notin A] \text{ or}\\
&\quad[\delta(x_0,s)\in A\wedge\exists t\in L/s:\delta(x_0,st)\notin A] \quad \text{Def. VII.7}\\
\Leftrightarrow&\exists w\in L \text{ s.t. } [\delta_A(x_{A,0},w)]_x\notin A \text{ and } [\delta_A(x_{A,0},w)]_n=K\\
\Leftrightarrow&\exists w\in L \text{ s.t. } \varphi_{\text{att}}(R_{G_A}(w,L))=0 \quad \text{Def. VII.8}\\
\Leftrightarrow&L\not\models_{G_A}\varphi_{\text{att}} \quad \text{Def. III.2.}
\end{aligned}$$

For the second equivalence, the proof of the "⇒" direction can be done by taking $w=s$ if the first case holds and $w=st$ if the second case holds. For the "⇐" direction, since $[\delta_A(x_{A,0},w)]_n=K$, by the construction of G_A, we know that (i) $|w|\geq K$ or (ii) $\exists w_1w_2\in\overline{\{w\}}$ s.t. $\delta(x_0,w_1)\in A$ and $\delta(x_0,w_1w_2)\notin A$. These two cases correspond to the two cases after the first equivalence, respectively. □

Remark VII.3: So far, we have discussed the enforcement of K-diagnosability, K-detectability and K-attractability. Since K-diagnosability (respectively, K-detectability and K-attractability) is stronger than diagnosability (respectively,

detectability and attractability), enforcing the former one implies that the latter one is also enforced. Moreover, the uniform approach guarantees the diagnosis (respectively, detection and attraction) delay of the controlled system, which cannot be guaranteed by the previous approaches. In this sense, enforcing these properties with desired delay K is a new feature of the uniform solution rather than a restrictive assumption. However, if one does not care about the diagnosis (respectively, detection and attraction) delay and just wants to enforce diagnosability (respectively, detectability and attractability), then the maximally permissive solution obtained for K-diagnosability (respectively, K-detectability and K-attractability) may not be the maximally permissive solution for diagnosability (respectively, detectability and attractability). Moreover, as K increases, the permissiveness of the solution increases, but the complexity of the synthesis algorithm also increases, since we need to "unfold" the system for more steps. In other words, there is a tradeoff between the permissiveness of the solution and the complexity of synthesis algorithm when there is no delay K required *a priori*. In this case, one may proceed as follows. First, one may start with a solution by choosing a relatively small K. If the permissiveness of this solution satisfies the design requirement, then stop. Otherwise, choose a larger K and repeat the above procedure until a desirable solution is found.

XII. Conclusion

We presented a uniform approach to the problem of synthesizing a maximally permissive supervisor that enforces a certain property for a partially-observed discrete-event system that does not originally satisfy the property. To this end, we defined a class of properties called Information-State-Based properties and a novel information structure called the All Enforcement Structure that embeds all valid supervisors enforcing any IS-based property. Based on the AES, a synthesis algorithm was provided to synthesize a locally maximal solution to this problem, without making any assumptions about the observability properties of the controllable events. In this regard, our approach relaxes the assumption that all controllable events are observable in the existing works on property enforcement by supervisory control. We showed that many important properties in the DES literature can be enforced by the uniform approach described in this paper. Moreover, this approach can be applied to enforce other properties, such as anonymity, for which no synthesis methodologies exist in the current literature. In addition, the AES can be used for solving quantitative optimal property enforcement control problems when a cost structure is imposed on this problem. Since the AES embeds all valid property-enforcing supervisors, it provides a suitable solution space over which to solve such optimal control problems.

As we emphasized in the introduction, this paper only considers liveness and does not take non-blockingness into account. In our recent work [32], we show that non-blockingness is more complicated to deal with and different techniques are required. However, one could combine the general framework proposed in this paper and the techniques in [32] in order to obtain a non-blocking supervisor enforcing some IS-based property.

References

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

[2] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis, "Diagnosability of discrete-event systems," *IEEE Trans. Autom. Control*, vol. 40, no. 9, pp. 1555–1575, 1995.

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

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

[5] S. Shu, F. Lin, and H. Ying, "Detectability of discrete event systems," *IEEE Trans. Autom. Control*, vol. 52, no. 12, pp. 2356–2359, 2007.

[6] S. Shu and F. Lin, "I-detectability of discrete-event systems," *IEEE Trans. Autom. Sci. Eng.*, vol. 10, no. 1, pp. 187–196, 2013.

[7] Y. Brave and M. Heymann, "Stabilization of discrete-event processes," *Int. J. Control*, vol. 51, no. 5, pp. 1101–1117, 1990.

[8] C. M. Özveren, A. S. Willsky, and P. J. Antsaklis, "Stability and stabilizability of discrete event dynamic systems," *J. ACM*, vol. 38, no. 3, pp. 729–751, 1991.

[9] K. W. Schmidt and C. Breindl, "A framework for state attraction of discrete event systems under partial observation," *Inform. Sci.*, vol. 281, no. 10, pp. 265–280, 2014.

[10] E. M. Clarke, O. Grumberg, and D. Peled, *Model checking*. Cambridge, MA: MIT press, 1999.

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

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

[13] H. Cho and S. I. Marcus, "On supremal languages of classes of sublanguages that arise in supervisor synthesis problems with partial observation," *Math. Contr. Sig. Syst.*, vol. 2, no. 1, pp. 47–69, 1989.

[14] N. Ben Hadj-Alouane, S. Lafortune, and F. Lin, "Centralized and distributed algorithms for on-line synthesis of maximal control policies under partial observation," *Discrete Event Dynamic Syst.: Theory Appl.*, vol. 6, no. 4, pp. 379–427, 1996.

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

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

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

[18] M. Ben-Kalefa and F. Lin, "Supervisory control for opacity of discrete event systems," in *Proc. 49th IEEE Annual Allerton Conf. Commun., Control, Comp.*, 2011, pp. 1113–1119.

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

[20] M. Sampath, S. Lafortune, and D. Teneketzis, "Active diagnosis of discrete-event systems," *IEEE Trans. Autom. Control*, vol. 43, no. 7, pp. 908–929, 1998.

[21] S. Shu and F. Lin, "Enforcing detectability in controlled discrete event systems," *IEEE Trans. Autom. Control*, vol. 58, no. 8, pp. 2125–2130, 2013.

[22] R. Kumar and V. K. Garg, *Modeling and Control of Logical Discrete Event Systems*. Norwell, MA: Kluwer, 1995.

[23] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*, 2nd ed. New York: Springer, 2008.

[24] M. Iordache and P. J. Antsaklis, *Supervisory Control of Concurrent Systems: A Petri Net Structural Approach*. New York: Springer Science & Business Media, 2007.

[25] C. Seatzu, M. Silva, and J. H. Van Schuppen, *Control of Discrete-Event Systems. Automata and Petri Net Perspectives*. London, U.K.: Springer London, 2013.

[26] W. M. Wonham, *Supervisory Control of Discrete-Event Systems*. Toronto, Canada, ON: University of Toronto, 2014.

[27] P. Darondeau, H. Marchand, and L. Ricker, "Enforcing opacity of regular predicates on modal transition systems," *Discrete Event Dynamic Syst.: Theory Appl.*, pp. 1–20, 2014.

[28] X. Yin and S. Lafortune, "A new approach for enforcing opacity via supervisory control for partially-observed discrete-event systems," in *Proc. Amer. Control Conf.*, 2015, pp. 377–383.
[29] F. Cassez and S. Tripakis, "Fault diagnosis with static and dynamic observers," *Fund. Inform.*, vol. 88, no. 4, pp. 497–540, 2008.
[30] E. Dallal and S. Lafortune, "On most permissive observers in dynamic sensor activation problems," *IEEE Trans. Autom. Control*, vol. 59, no. 4, pp. 966–981, 2014.
[31] X. Yin and S. Lafortune, "Synthesis of maximally permissive nonblocking supervisors for partially observed discrete event systems," in *Proc. 53rd IEEE Conf. Decision Control*, 2014, pp. 5156–5162.
[32] X. Yin and S. Lafortune, "Synthesis of maximally permissive supervisors for partially observed discrete event systems," *IEEE Trans. Autom. Control*, 2016.
[33] F. Lin and W. M. Wonham, "Supervisory control of timed discrete-event systems under partial observation," *IEEE Trans. Autom. Control*, vol. 40, no. 3, pp. 558–562, 1995.
[34] H. Liao, S. Lafortune, S. Reveliotis, Y. Wang, and S. Mahlke, "Optimal liveness-enforcing control for a class of petri nets arising in multithreaded software," *IEEE Trans. Autom. Control*, vol. 58, no. 5, pp. 1123–1138, 2013.
[35] Z. Li, M. Zhou, and N. Wu, "A survey and comparison of petri net-based deadlock prevention policies for flexible manufacturing systems," *IEEE Trans. Syst., Man, Cybern. C*, vol. 38, no. 2, pp. 173–188, 2008.
[36] J. N. Tsitsiklis, "On the control of discrete-event dynamical systems," *Math. Control, Signals Syst.*, vol. 2, no. 2, pp. 95–107, 1989.
[37] G. Barrett and S. Lafortune, "On the separation of estimation and control in discrete-event systems," in *Proc. 39th IEEE Conf. Decision Control*, 2000, vol. 3, pp. 2258–2259.
[38] F. Lin, "Opacity of discrete event systems and its applications," *Automatica*, vol. 47, no. 3, pp. 496–503, 2011.
[39] Y.-C. Wu and S. Lafortune, "Comparative analysis of related notions of opacity in centralized and coordinated architectures," *Discrete Event Dynamic Syst.*, vol. 23, no. 3, pp. 307–339, 2013.
[40] L. Sweeney, "K-anonymity: A model for protecting privacy," *Int. J. Uncertainty, Fuzziness Knowledge-Based Syst.*, vol. 10, no. 05, pp. 557–570, 2002.

Xiang Yin (S'14) was born in Anhui, China, in 1991. He received the B.Eng. degree in electrical engineering from Zhejiang University, Zhejiang, China, in 2012 and the M.S. degree in electrical engineering from the University of Michigan, Ann Arbor, in 2013, where he is currently pursuing the Ph.D degree.

His research interests include supervisory control of discrete-event systems, model-based fault diagnosis, formal methods, game theory and their applications to cyber and cyber-physical systems.

Stéphane Lafortune (F'99) received the B.Eng. degree from Ecole Polytechnique de Montréal, Montréal, QC. Canada, in 1980, the M.Eng. degree from McGill University, Montréal, QC. Canada, in 1982, and the Ph.D. degree from the University of California at Berkeley in 1986, all in electrical engineering.

Since September 1986, he has been with the University of Michigan, Ann Arbor, where he is a Professor of Electrical Engineering and Computer Science. He is the lead developer of the software package UMDES and co-developer of DESUMA with L. Ricker. He co-authored the textbook *Introduction to Discrete Event Systems—Second Edition* (Springer, 2008). He is Editor-in-Chief of the *Journal of Discrete Event Dynamic Systems: Theory and Applications*. His research interests are in discrete event systems and include multiple problem domains: modeling, diagnosis, control, optimization, and applications to computer and software systems.

Dr. Lafortune received the Presidential Young Investigator Award from the National Science Foundation in 1990 and the George S. Axelby Outstanding Paper Award from the Control Systems Society of the IEEE, in 1994 and 2001, respectively.