A compact and uniform approach for synthesizing state-based property-enforcing supervisors for discrete-event systems

Rômulo Meira-Góes, Jack Weitze, Stéphane Lafortune

***Abstract*—In this paper, we are interested in the problem of synthesizing a partial-observation supervisor for a discrete-event system such that it enforces a desired property. We introduce a *compact* and uniform approach to the problem of synthesizing state-based property-enforcing supervisors. A state-based property is a property that only depends on the current estimate of the past behavior of the system and *does not* depend on its future behavior. Previous work has introduced a uniform methodology to solve this problem through the construction of a finite structure called the *All Enforcement Structure* (AES) which captures a game between the supervisor and the environment. Although the AES is a powerful structure that includes all possible property-enforcing supervisors, its construction is computationally challenging since the number of states grows exponentially in the number of states and the number of events of the system. Our contribution is the definition of a compact AES that is equivalent to but computationally more efficient than the original AES. Specifically, the compact AES enjoys the same properties as the original AES with respect to synthesizing maximally permissive supervisors under the assumption of incomparable sets of controllable and observable events. We also provide experimental results to show the benefits of the compact AES over the original AES.**

***Index Terms*—Discrete-event systems, Supervisory control, Property enforcement**

I. Introduction

In the context of Discrete Event Systems (DES), an important research issue is the problem of synthesizing a supervisor that enforces a desired property. This problem first appeared in the seminal work of Ramadge and Wonham in [1]. Since then different variations of this problem have been investigated by the control engineering community where different assumptions on the control loop or different properties are considered. In this paper, we are interested in the problem of synthesizing a partial observation supervisor for a DES such that it enforces a desired property.

The properties under consideration in [1] were safety and non-blockingness, where safety describes the desired legal behavior for the controlled system while non-blockingness describes the condition of always being able to achieve desired behaviors. Moreover, a maximal permissiveness constraint is also imposed on the supervisor, i.e., the supervisor should allow a "maximal" controlled behavior while enforcing the desired properties. This problem setting, termed as the standard supervisory control problem under full observation was solved in [1], while different solutions for the partial observation setting were proposed, see, e.g., [2]–[8]. In the case of enforcement of different properties than the above, different approaches have also emerged, e.g., for diagnosability [9], opacity [10]–[13], and so forth.

A *uniform approach* to the problem of synthesizing property-enforcing supervisors for a large class of properties was presented in [14]. The class of properties investigated in [14] is referred to as *state-based properties*. Intuitively, a state-based property is a property that only depends on the current state estimate of the system and *does not* depend on its future behavior. Properties such as safety, opacity, diagnosability, and detectability are state-based properties while non-blockingness is not a state-based property [14].

The authors are with the Department of EECS, University of Michigan, MI 48109 USA (e-mail:{romulo,jweitze,stephane}@umich.edu). Their work is supported in part by US NSF grants CNS-1738103 and CNS-1801342.

The approach developed in [14] is based on the construction of a finite structure called *All Enforcement Structure* (AES) which captures a game between the supervisor and the environment. Intuitively, the AES embeds *all* property-enforcing supervisors and it serves as the basis for solving the synthesis problem for state-based properties.

Although the AES is a powerful structure that includes all possible property-enforcing supervisors, its construction is computationally challenging since the number of its states grows exponentially in the number of states and the number of events of the system. In this paper, we introduce a compact AES that is equivalent and computationally more efficient than the original AES[1] described in [14]. Based on the set of unobservable events of the system, we introduce an equivalence relation over the set of control decisions a supervisor can select. This equivalence relation allows the construction of a so called *compact* AES, which is equivalent to the original AES for the purpose of *synthesizing maximally permissive supervisors*. Specifically, the compact AES enjoys the same properties as the original AES with respect to synthesizing maximally permissive supervisors under the assumption of incomparable sets of controllable and observable events, but it has a state space that is provably more compact than that of the original AES under general conditions.

Our presentation is organized as follows. Section II introduces necessary background on supervisory control theory. The property-enforcing supervisory control problem is stated in Section III. We briefly review the original AES method in Section IV. In Section V, we present the compact AES and show its soundness and completeness with respect to the property-enforcement problem. Experimental results are presented in Section VI. Finally, we conclude the paper in Section VII.

II. Preliminaries

We consider a deterministic discrete transition system modeled as a finite-state automaton. A finite-state automaton G is defined as a tuple $G = (X_G, \Sigma, \delta_G, x_{0,G})$, where X_G is a finite set of states; Σ is a finite set of events; $\delta_G : X_G \times \Sigma \to X_G$ is a partial transition function; and $x_{0,G} \in X_G$ is the initial state.

The function δ_G is extended in the usual manner to domain $X_G \times \Sigma^*$. The language generated by G is defined as $\mathcal{L}(G) = \{s \in \Sigma^* | \delta_G(x_0, s)!\}$, where ! means "is defined". For $x \in X_G$, we define $En_G(x) = \{e \in \Sigma \mid \delta_G(x, e)!\}$ as the active event set at state x. For $K \subseteq \Sigma^*$, we denote by $pr(K)$ as the set of all prefixes of strings in K and K is said to be prefix-closed if $K = pr(K)$.

In the context of supervisory control theory of DES [1], we consider an uncontrolled system (plant) G that needs to be controlled in order to satisfy given safety specifications. In order to control G, the event set Σ is partitioned into two disjoint sets, which are the set of controllable events Σ_c and the set of uncontrollable events Σ_{uc}. The safety specifications on G are enforced by a supervisor, denoted by S, that dynamically enables/disables controllable events. The resulting controlled behavior is a new DES denoted by S/G, resulting in the closed-loop language $\mathcal{L}(S/G)$, defined in the usual manner (see, e.g., [7], [8]).

[1] We will use the term "original" AES to denote the AES described in [14].

In addition, when the system is partially observed due to the limited sensing capabilities of G, the event set is also partitioned into $\Sigma = \Sigma_o \cup \Sigma_{uo}$, where Σ_o is the set of observable events and Σ_{uo} is the set of unobservable events. Based on this second partition, the *projection* function $P_o : \Sigma^* \to \Sigma_o^*$ is recursively defined as: $P_o(\epsilon) = \epsilon$, $P_o(se) = P_o(s)e$ if $e \in \Sigma_o$, and $P_o(se) = P_o(s)$ otherwise. The inverse projection $P_o^{-1} : \Sigma_o^* \to 2^{\Sigma^*}$ is defined as $P_o^{-1}(t) = \{s \in \Sigma^* | P_o(s) = t\}$.

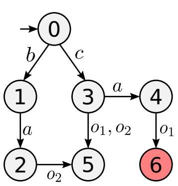
Supervisor S makes its control decisions based on a string of observable events. Formally, a partial-observation supervisor is a (partial) function $S : P_o(\mathcal{L}(G)) \to \Gamma$, where $\Gamma = \{\gamma \in 2^\Sigma : \Sigma_{uc} \subseteq \gamma\}$. The set of all supervisors $S : P_o(\mathcal{L}(G)) \to \Gamma$ is denoted by $\mathbb{S}$.

We also recall the notions of controllability and observability for a prefix-closed language $K \subseteq \mathcal{L}(G)$. We say that the language K is

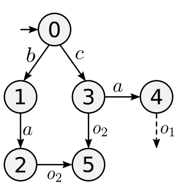
- controllable w.r.t. Σ_c, if $K\Sigma_{uc} \cap \mathcal{L}(G) \subseteq K$;
- observable w.r.t. Σ_o and Σ_c, if $(\forall s \in K, \forall e \in \Sigma_c : se \in K)[P_o^{-1}(P_o(s))e \cap \mathcal{L}(G) \subseteq K]$.

Let $\mathcal{CO}(K)$ be the collection of all prefix-closed controllable and observable sublanguages of K w.r.t. G, Σ_c and Σ_o. Formally, $\mathcal{CO}(K) = \{K' \subseteq \mathcal{L}(G) \mid K' = pr(K') \subseteq K$ s.t. K' is controllable and observable w.r.t. $(\Sigma_c, \Sigma_o)\}$. There does not exist in general a supremal element in $\mathcal{CO}(K)$ [7]. Moreover, there exists a supervisor S for any $L \in \mathcal{CO}(K) \setminus \{\emptyset\}$ such that $\mathcal{L}(S/G) = L$.

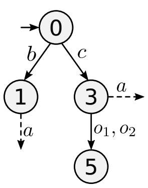
Example II.1. Consider the system G depicted in Fig. 1(a). Let $K = \mathcal{L}(Ac(G, \{6\}))$ be the language specification that a supervised system must enforce, where $Ac(G, X)$ denotes the operation that returns the *accessible* subautomaton of G after deleting states $X \subseteq X_G$. The two automata shown in Fig. 1(b) and Fig. 1(c) belong to $\mathcal{CO}(K)$.



(a) G



(b) G_1



(c) G_2

Fig. 1: System G: $\Sigma_c = \{o_1, o_2, a\}$, $\Sigma_o = \{o_1, o_2\}$, and $X_{uns} = \{6\}$. Dashed lines in G_1 and G_2 are events disabled by a supervisor.

We complete this section by defining useful notation for strings. The transition function δ_G is extended to set of states $X \subseteq X_G$ as $\delta_G(X, e) = \cup_{x \in X}\{\delta_G(x, e)\}$ where $\delta_G(x, e) = \emptyset$ if $\delta_G(x, e)$ is not defined. The unobservable reach of a set of states $X \subseteq X_G$ under the subset of events $\gamma \subseteq \Gamma$ is given by:

$$UR_\gamma(X) = \bigcup_{s \in (\Sigma_{uo} \cap \gamma)^*} \delta_G(X, s) \tag{1}$$

Given $L \subseteq \mathcal{L}(G)$, the set of all possible states in G reachable from its initial state via a string with the same projection as $s \in L$ is given by:

$$Re_G(s, L) = \bigcup_{\substack{t \in L: \\ P_o(s) = P_o(t)}} \{\delta_G(x_{0,G}, t)\} \tag{2}$$

Lastly, for any string $s \in \Sigma^*$, $|s|$ denotes the length of s. We define s^i to be the i^{th} prefix of s where $s^0 = \epsilon$.

III. Property-enforcing problem

In general, a language-based property over events Σ is defined as a function $\varphi : 2^{\Sigma^*} \to \{0, 1\}$, where $L \subseteq \Sigma^*$ satisfies the property if $\varphi(L) = 1$. Safety is a classical property example that can be specified as a language-based property. Given a specification language $K \subseteq \Sigma^*$, the safety property is defined as $\varphi(L) = 1$ if and only if $L \subseteq K$.

Although language-based properties encompass a large class of properties in DES, it may not be possible to bound a priori the memory needed for their verification. To simplify our problem formulation, we investigate a particular class of properties called *state-based properties*. State-based properties are not as general as language-based ones. However, many important properties in the DES literature can be expressed as state-based properties provided that state space refinement of the system may be needed [14]. Safety, diagnosability, and opacity are expressible as state-based properties as described in [14]. Formally, state-based properties are defined as follows.

Definition III.1. (Def. III.2 in [14]) A state-based property φ w.r.t. G is a function $\varphi : 2^{X_G} \to \{0, 1\}$. A sublanguage $L \subseteq \mathcal{L}(G)$ satisfies φ if $\forall s \in L$: $\varphi(Re_G(s, L)) = 1$.

Example III.2. In Example **II.1**, the supervisor must enforce a safety property over the supervised system, i.e., to not reach state 6. The property φ defined for $X \subseteq X_G$ as $\varphi(X) = 1$ if and only if $\{6\} \cap X = \emptyset$ captures this safety property. In general, a safety property over G is defined as $\varphi(X) = 1$ if and only if $X_{uns} \cap X = \emptyset$, where $X \subseteq X_G$ and $X_{uns} \subseteq X_G$ is a set of unsafe states that must not be reached.

We are now ready to present the *maximally permissive state-based property-enforcement problem* (MPSEP).

Problem III.1. Given system G and state-based property φ with respect to G, synthesize a supervisor $S \in \mathbb{S}$ such that

(1) $\mathcal{L}(S/G)$ satisfies φ;
(2) For any $S' \in \mathbb{S}$ satisfying (1), we have $\mathcal{L}(S/G) \not\subset \mathcal{L}(S'/G)$

***Remark* III.1.** In [14], the MPSEP includes an additional liveness constraint in the problem formulation. For simplicity and without loss of generality, we relax this constraint from our problem formulation since our method can be easily extended to include this constraint.

IV. All enforcement structure

In [14], a uniform approach for synthesizing state-based property-enforcing supervisors for partially-observed DES is presented. Their method constructs a finite bipartite transition system (BTS), called All Enforcement Structure (AES), which embeds all admissible supervisors that enforce a desired state-based property. We review the concept of BTS and the construction of the AES as in [14].

Intuitively, a BTS is a bipartite graph where its nodes are divided into two disjoint sets. In a BTS, node types represent the two entities in the supervisory control framework, i.e., one node type represents a supervisor S while the other node type represents the plant G.

Definition IV.2. Given the system G and a set of control patterns $C \subseteq \Gamma$, the bipartite transition system B is defined by $B = (Q_1^B, Q_2^B, h_1^B, h_2^B, \Sigma, C, q_0^B)^2$, where

- $Q_1^B \subseteq 2^{X_G}$ is the set of player 1 states;
- $Q_2^B \subseteq 2^{X_G} \times C$ is the set of player 2 states, $I(q)$ and $\Gamma(q)$ denote, respectively, the projection to the first and second components of $q \in Q_2^B$;
- $h_1^B : Q_1^B \times C \to Q_2^B$ is a partial transition function over C that satisfies:

$$h_1^B(q, \gamma)! \Rightarrow h_1^B(q, \gamma) = (UR_\gamma(q), \gamma) \tag{3}$$

[2] We drop the superscript B whenever B is clear from the context.

- $h_2^B : Q_2^B \times \Sigma_o \to Q_1^B$ is the transition function defined as:
$$h_2^B(q, e) = \begin{cases} \delta_G(I(q), e) & \text{if } e \in \Gamma(q) \cap En_G(I(q)) \\ \text{undefined} & \text{otherwise} \end{cases} \quad (4)$$
- $q_0^B = \{x_{0,G}\} \in Q_1^B$ is the initial state;

The BTS B is the largest BTS with respect to C if h_1^B is complete at every reachable state in Q_1^B. We use T to denote the largest BTS with respect to C. By convention, the sets Q_1^B and Q_2^B are defined to be the accessible states from the initial state and following Eqs. (3-4).

Each element in Q_1 represents the current state estimate of G. In state $q \in Q_1$, player 1 selects a control decision in C and transitions to a state in Q_2. In state $q \in Q_2$, player 2 selects an observable event to occur within the set of enabled events $\Gamma(q)$ and transitions back to a Q_1 state. Therefore, the BTS B exactly simulates the supervisory control framework.

Remark **IV.2.** For convenience, the transition function h_1^T is complete with respect to C in the definition of T. Such definition improves the clarity of our proofs. However, the definition of h_1^T can be further constrained based on the active events of the state estimate $q \in Q_1^T$, i.e., complete only for $\gamma \in C$ such that $\gamma \subseteq En_G(q) \cup \Sigma_{uc}$.

To formalize that T enumerates all possible interactions between supervisors and corresponding event observations of G, we extend h_1^B as H_1^B to be a transition function from Q_1^B to Q_1^B states. The function $H_1^B : Q_1^B \times C \times \Sigma_o \to Q_1^B$ is defined as:

$$H_1^B(q, \gamma, e) = h_2^B(h_1^B(q, \gamma), e) \quad (5)$$

This function can be extended to string $se \in \Sigma_o^* \Sigma_o$ and control decisions $\gamma_0 \dots \gamma_{|s|}$ in a recursive manner, i.e., $H_1^B(q, \gamma_0 \dots \gamma_{|s|}, se) = H_1^B(H_1^B(q, \gamma_0 \dots \gamma_{|s|-1}, s), \gamma_{|s|}, e)$. Next, we define $\mathcal{C}_B(q) = \{\gamma \in \Gamma \mid h_1^B(q, \gamma)!\}$ to be the set of control decisions defined at $q \in Q_1^B$.

Definition IV.3. A supervisor $S \in \mathbb{S}$ is said to be included in B if for any string $s \in P_o(\mathcal{L}(S/G))$, its control decision $S(s)$ is defined at the state $q \in Q_1^B$ reached by string s and control decisions $S(s^0) \dots S(s^{|s|-1})$. Formally,

$$S(s) \in \mathcal{C}_B(H_1^B(q_0^B, S(s^0) \dots S(s^{|s|-1}), s))$$

$\forall s \in P_o(\mathcal{L}(S/G)))$. $\mathcal{S}(B)$ denotes the set of all supervisors included in B.

Definition **IV.3** states that for every string in the supervised system, its control decision is defined at the Q_1^B state reached via this string and its intermediate control decisions. Based on Def. **IV.3**, we can state that any supervisor for system G, control patterns C and observable events Σ_o is included in T.

Lemma IV.1. Any supervisor $S : P_o(\mathcal{L}(G)) \to C$ is included in $\mathcal{S}(T)$.

The result of Lemma **IV.1** is intuitive since $\mathcal{C}_T(q) = C$ for any state $q \in Q_1^T$ and h_2^T is defined for every feasible observable event. Nonetheless, it is based on Lemma **IV.1** that the following results are possible.

In [14], the construction of the AES for a given property φ is done in two steps [Alg. 1 in [14]]. First, the largest BTS T is constructed with respect to $C = \Gamma$. Let $T^{\max} = (Q_1^{\max}, Q_2^{\max}, h_1^{\max}, h_2^{\max}, \Sigma, \Gamma, q_0^{\max})$ denote the largest BTS with respect to Γ. The second step is to prune $T^{\max}$ in order to only obtain control decisions that satisfy φ, i.e., we remove supervisors that violate property φ. This pruning process can be posed as a fully observed supervisory control problem over $T^{\max}$, where $T^{\max}$ is considered as an automaton.

Definition IV.4. Consider $T^{\max}$ with $E_{uc} = \Sigma_o \cup \{\Sigma_{uc}\}$ as the set of uncontrollable events. Let $K = \mathcal{L}(Ac(T^{\max}, M))$ where $M = \{q \in Q_1^{\max} \mid \varphi(q) = 0\} \cup \{q \in Q_2^{\max} \mid \varphi(I(q)) = 0\}$ be the specification over $T^{\max}$. The *All Enforcement Structure* AES_φ is defined as the subautomaton of $T^{\max}$ representation of the supremal controllable sublanguage of K w.r.t. E_{uc} and $T^{\max}$.

We consider $T^{\max}$ as a meta-system with all control decisions, with the exception of control decision Σ_{uc}, to be controllable events. The control decision Σ_{uc} is considered to be uncontrollable since the supervisor should always be able to enable the uncontrollable events. The specification is given based on φ since it removes states in $T^{\max}$ that violate the property φ. It follows that AES_φ is a subautomaton of $T^{\max}$, which implies that AES_φ is also a BTS.

Example IV.3. We return to system G in Fig. 1(a) with the state-based property defined in Example **III.2**. The BTS $T^{\max}$ is depicted in Fig. 2. For convenience, we omit the deadlock states $(\{5\}, \gamma)$, for $\gamma \in \Gamma$, reached by the gray transitions defined in state $\{5\}$. AES_φ is obtained by removing states marked by a red cross and the dashed transitions.

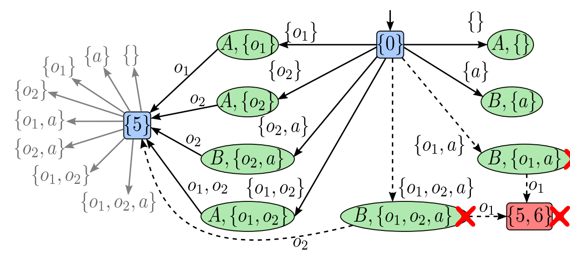


Fig. 2: Example of the construction of $T^{\max}$ and AES_φ. In the diagram, rectangular states correspond to states in $Q_1^{\max}$ and oval states correspond to states in $Q_2^{\max}$. For simplicity, A and B represent state estimates, i.e., $A = \{0, 1, 3\}$ and $B = \{0, 1, 2, 3, 4\}$; and we omit all uncontrollable events in the control decisions, e.g., decision $\{\}$ represents $\{b, c\}$.

To conclude this section, we restate a theorem from [14] that states that AES_φ contains all supervisors that satisfy property φ.

Theorem IV.1. (Theorem V.1. [14]) A supervisor $S \in \mathbb{S}$ enforces a state-based property φ with respect to G if and only if $S \in \mathcal{S}(AES_\varphi)$.

V. Compact All Enforcement Structure

To obtain AES_φ, we must construct $T^{\max}$ which takes into account all possible control decisions in Γ. The set Γ grows exponentially in the number of controllable events, i.e., $|\Gamma| = |2^{\Sigma_c}|$. This exponential growth implies that the number of player 2 states in $T^{\max}$ grows exponentially in the number of controllable events. Therefore, the construction and the manipulation of $T^{\max}$ is computationally expensive.

In this section, we provide a sound, complete and computationally more efficient method for synthesizing property-enforcing supervisors for partially-observed DES. Namely, we construct a *compact* AES_φ from which every supervisor that enforces property φ can be extracted. Our method is based on the original AES_φ but it eliminates recoverable information in its construction method.

A. Control decision equivalence classes

In the largest BTS T, the transition function h_1^T is defined for every control decision in C. In the case of $T^{\max}$, $h_1^{\max}$ is defined for every control decision in Γ. Once player 1 selects a control decision $\gamma \in \Gamma$ in state $q \in Q_1^{\max}$, it transitions to a player 2 state by performing an unobservable reach computation based on the selected control decision and recording this control decision, i.e., $h_1^{\max}(q,\gamma) = (UR_\gamma(q), \gamma)$.

The unobservable reach function depends on the set of unobservable events in the selected $\gamma \in \Gamma$. For this reason, we define an equivalence relation on Γ based on Σ_{uo}. Formally, $R \subseteq \Gamma \times \Gamma$ is defined as:

$$R = \{(\gamma_1, \gamma_2) \mid \gamma_1 \cap \Sigma_{uo} = \gamma_2 \cap \Sigma_{uo}\} \tag{6}$$

It can be shown that R is indeed an equivalence relation. For any $\gamma \in \Gamma$, let $[\![\gamma]\!]$ denote the equivalence class of γ, i.e., $[\![\gamma]\!] = \{\gamma' \in \Gamma \mid (\gamma, \gamma') \in R\}$.

Since the unobservable reach depends on the set of enabled unobservable events, we can state a property of the unobservable reach operation based on the equivalence classes of Γ. For a given set of G states $X \subseteq X_G$, control decisions in the same equivalence class produce the same unobservable reach of X. Formally, we have the following proposition.

Proposition V.1. Let $\gamma \in \Gamma$ and $X \subseteq X_G$, for any $\gamma^* \in [\![\gamma]\!]$ we have that $UR_\gamma(X) = UR_{\gamma^*}(X) = UR_{\gamma \cap \Sigma_{uo}}(X)$.

Proof. The proof follows from the definition of the unobservable reach. $\square$

Proposition **V.1** hints that there exists recoverable information in the construction of $T^{\max}$. Namely, there might be states in $Q_2^{\max}$ that hold information that can be retrieved from other $Q_2^{\max}$ states.

B. Compact bipartite transition system

Proposition **V.1** is the key property used to obtain a BTS that is a compact version of $T^{\max}$. In order to construct this compact BTS, we restrict the set of control decisions based on the equivalence classes of Γ.

Using $T^{\max}$ of Example **IV.3**, we illustrate the above discussion. Figure 3 shows part of $T^{\max}$ of our running example. Starting from $q = \{0\}$, player 1 can select one of the three defined control decisions. Since these three control decisions belong to the same equivalence class, Proposition **V.1** guarantees that $UR_{\{o_1,a\}}(q) = UR_{\{o_2,a\}}(q) = UR_{\{o_1,o_2,a\}}(q) = B = \{0,1,2,3,4\}$.

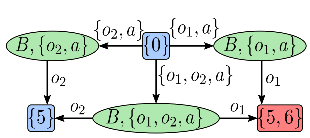


Fig. 3: States with recoverable information in $T^{\max}$

Next, player 2 selects one observable event allowed by the current control decision from each of the $Q_2^{\max}$ states defined in Fig. 3. Since there are only two observable events and the unobservable reaches are identical, the $Q_2^{\max}$ states can only transition to two distinct $Q_1^{\max}$ states. The possible states are states $\{5\}$ and $\{5,6\}$ as depicted in Fig. 3. If we remove player 2 state $(B, \{o_1,o_2,a\})$ from $Q_2^{\max}$, we would not remove any state-based information since $B = \{0,1,2,3,4\}$ and both $\{5\}$, $\{5,6\}$ would remain in the structure. Therefore, the information about player 2 state $(B, \{o_1,o_2,a\})$ can be retrieved from the remaining states. This implies that h_1 can be defined for control decisions $\{o_1,a\}$ and $\{o_2,a\}$ in state $\{0\}$.

In the above illustrative discussion, since the state-based information about $\{o_1,o_2,a\}$ is *retrievable* from $\{o_1,a\}$ and $\{o_2,a\}$, defining h_1 only for control decisions $\{o_1,a\}$ and $\{o_2,a\}$ is sufficient. We define the function C_B^{ext} that finds all retrievable control decisions of a player 1 state in BTS B. Formally,

$$\mathcal{C}_B^{ext}(q) = \{\gamma \in \Gamma \mid (\exists I \subseteq [\![\gamma]\!] \cap \mathcal{C}_B(q))[\gamma = \bigcup_{\gamma^* \in I} \gamma^*]\} \tag{7}$$

In words, γ belongs to $C_B^{ext}(q)$ if it can be reconstructed via the union of control decisions in the same equivalence class defined in state q.

***Remark* V.3.** In Figure 3, defining h_1 only for $\{o_1,o_2,a\}$ also obtains the entire state-based information for this example, i.e., states $B = \{0,1,2,3,4\}$, $\{5\}$, $\{5,6\}$ are going to be defined. However, it is difficult to retrieve the information about $\{o_1,a\}$ and $\{o_2,a\}$ from $\{o_1,o_2,a\}$. Namely, it is simpler to assess the safety of the union of two safe control decisions in the same equivalence class.

If we construct a BTS T^{comp} such that it is possible to retrieve every possible control decision from each Q_1^{comp} state, then this constructed T^{comp} will have the same information as $T^{\max}$. In other words, we must have $\mathcal{C}_{T^{comp}}^{ext}(q) = \Gamma$ for any $q \in Q_1^{comp}$. To obtain this result, we define the set of all control decisions with at most one controllable and observable event as:

$$C^{comp} = \{\gamma \in \Gamma \mid |\gamma \cap \Sigma_o \cap \Sigma_c| \in \{0,1\}\} \tag{8}$$

Let T^{comp} denote the largest BTS constructed based on C^{comp}. Based on the above discussion, we have the following proposition.

Proposition V.2. Given $T^{\max}$ and T^{comp}, then:
(1) $(\forall q \in Q_1^{comp})[\mathcal{C}_{T^{comp}}^{ext}(q) = \Gamma]$;
(2) $Q_1^{\max} = Q_1^{comp}$;
(3) $(\forall q \in Q_2^{\max})(\exists q^* \in Q_2^{comp})[I(q) = I(q^*)]$.

Proof. (1) Let $\gamma \in \Gamma$ and let $obs = \gamma \cap \Sigma_o \cap \Sigma_c$. By definition of C^{comp}, for any $o \in obs$ there exists $\gamma_o \in C^{comp} \cap [\![\gamma]\!]$ such that $o \in \gamma_o$. It follows that $\gamma = \cup_{o \in obs} \gamma_o$. By definition of $\mathcal{C}_T^{ext}$ and of h_1^{comp}, $\gamma \in \mathcal{C}_{T^{comp}}^{ext}(q)$ for any $q \in Q_1^{comp}$.
(2) By the definition of T, it is clear that $Q_1^{comp} \subseteq Q_1^{\max}$ since $C^{comp} \subseteq \Gamma$. The second inclusion follows from (1) and the construction of C^{comp}.
(3) It follows from (2) and C^{comp}. $\square$

Proposition **V.2** states that T^{comp} has the same state-based information as $T^{\max}$. It remains to be proven that T^{comp} also has the same set of included supervisors as $T^{\max}$.

Example V.4. We return to our running example to show T^{comp} and compare it with $T^{\max}$. First, we obtain the set $C^{comp} = \{\{\}, \{a\}, \{o_1\}, \{o_2\}, \{o_1,a\}, \{o_2,a\}\}$ as our compact control decision set. Based on C^{comp}, T^{comp} is constructed and is depicted in Fig. 4. As we expected, T^{comp} has fewer states than $T^{\max}$, i.e., states $(A, \{o_1,o_2\})$, $(B, \{o_1,o_2,a\})$, $(\{5\}, \{o_1,o_2\})$, and $(\{5\}, \{o_1,o_2,a\})$ are not defined in T^{comp}. For convenience, we omit the deadlock states $(\{5\}, \gamma)$, for $\gamma \in C^{comp}$, reached by the gray transitions defined in state $\{5\}$.

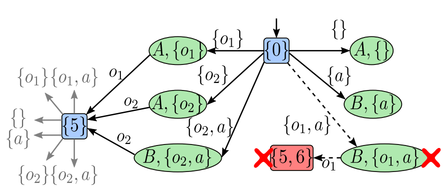


Fig. 4: Example of the construction of T^{comp} and AES_φ^{comp}. For simplicity, A and B represent state estimates, i.e., $A = \{0,1,3\}$ and $B = \{0,1,2,3,4\}$. AES_φ^{comp} is obtained by removing states marked by a red cross and dashed transitions.

According to Lemma **IV.1**, every supervisor for G is included in $\mathcal{S}(T^{\max})$. Although every supervisor $S : P_o(\mathcal{L}(G)) \to C^{comp}$ is included in $\mathcal{S}(T^{comp})$, it is not true that every supervisor $S : P_o(\mathcal{L}(G)) \to \Gamma$ is included in $\mathcal{S}(T^{comp})$. This issue is resolved by extending the definition of supervisor inclusion in B based on the definition of $\mathcal{C}_B^{ext}$.

Definition V.5. A supervisor $S \in \mathbb{S}$ is said to be *included-by-extension* in B if $(\forall s \in P_o(\mathcal{L}(S/G)))(\exists \gamma_i \in [\![S(s^i)]\!])(\forall i \in \{0, \ldots, |s|-1\})[S(s) \in \mathcal{C}_B^{ext}(H_1^B(q_0, \gamma_0 \ldots \gamma_{|s|-1}, s))]$. Let $\mathcal{S}^{ext}(B)$ be the set of all supervisors included-by-extension in B.

Definition **V.5** states that a supervisor is included-by-extension in B if for every string in the supervised system, its control decision is retrievable in the state reached via this string and intermediate equivalent control decisions. It follows that every supervisor for G is included-by-extension in T^{comp}. This statement is formally written in the following lemma.

Proposition V.3. $S \in \mathbb{S}$ if and only if $S \in \mathcal{S}^{ext}(T^{comp})$.

Proof. It follows from Prop. **V.2** and the definition of $\mathcal{S}^{ext}$. □

C. Compact all enforcement structure

As in the definition of AES_φ, denoted hereafter as the original AES, we must prune T^{comp} in order to eliminate states that violate a given property φ. The pruning process is exactly the same as the one in Definition **IV.4** since the state spaces of T^{comp} and $T^{\max}$ are equivalent. Nevertheless, we repeat Definition **IV.4** to explicitly differentiate the AES obtained via $T^{\max}$ from the one obtained from T^{comp}.

Definition V.6. Consider T^{comp} with $E_{uc} = \Sigma_o \cup \{\Sigma_{uc}\}$ as the set of controllable events. Let $K = \mathcal{L}(Ac(T^{comp}, M))$ where $M = \{q \in Q_1^{comp} \mid \varphi(q) = 0\} \cup \{q \in Q_2^{comp} \mid \varphi(I(q)) = 0\}$ be the specification. The *Compact All Enforcement Structure* AES_φ^{comp} is defined as the subautomaton of T^{comp} representation of the supremal controllable sublanguage of K w.r.t. E_{uc} and T^{comp}.

Similarly to Definition **IV.4**, we consider T^{comp} as a meta-system with all control decisions, with the exception of control decision Σ_{uc}, to be controllable events.

Example V.5. Back to our running example, Figure 4 depicts T^{comp} for system G. Considering the state-based property defined in Example **IV.3**, we obtain AES_φ^{comp} following Definition **V.6**. The pruning procedure starts with T^{comp} and eliminates state $\{5, 6\}$ since it violates φ. Next, it prunes state $(B, \{o_1, a\})$ since it violates controllability. The procedure converges and AES_φ^{comp} is depicted in Fig. 4 by states *not* marked by a red cross and the solid transitions. Note that, AES_φ^{comp} has three fewer states than AES_φ.

Intuitively, AES_φ^{comp} includes(-by-extension) the same supervisors as AES_φ since every state-based information in $T^{\max}$ is in T^{comp} and every control decision in Γ is retrievable in every player 1 state of T^{comp}. Therefore, the pruning of T^{comp} only prunes the control decisions that violate the desired property. This statement is formally presented in the following theorem.

Theorem V.2. A supervisor $S \in \mathbb{S}$ enforces a state-based property φ with respect to G if and only if $S \in \mathcal{S}^{ext}(AES_\varphi^{comp})$.

Proof. For notational purposes, we use $\mathcal{C}(q) := \mathcal{C}_{AES_\varphi}(q)$ and $\mathcal{C}^{ext}(q) := \mathcal{C}^{ext}_{AES_\varphi^{comp}}(q)$. Intuitively, the proof shows that every control decision defined in any player 1 state of AES_φ is retrievable from the same state in AES_φ^{comp}, i.e., $\mathcal{C}(q) = \mathcal{C}^{ext}(q)$ for any $q \in Q_1^{AES_\varphi} = Q_1^{AES_\varphi^{comp}}$.

First, it follows from Prop. **V.2**, and Defs. **IV.4** and **V.6** that $Q_1^{AES_\varphi^{comp}} = Q_1^{AES_\varphi}$. Thus, we just need to show that $\mathcal{C}(q) = \mathcal{C}^{ext}(q)$ for any $q \in Q_1^{AES_\varphi}$. After that the result follows from Theorem **IV.1**.

We show that $\mathcal{C}(q) = \mathcal{C}^{ext}(q)$ in four steps.

Step 0: If $\gamma \in \mathcal{C}(q)$, then $\gamma' \in \mathcal{C}(q)$ for any $\gamma' \subseteq \gamma$ and $\gamma' \in [\![\gamma]\!] \cap C^{comp}$. This follows from the pruning process of $T^{\max}$. Any Q_1 state reached from q after γ and $e \in \gamma \cap \Sigma_o$ is also reached via a γ' and e in $T^{\max}$. Since γ survives the pruning, then γ' also remains in AES_φ since the future Q_1 states reached from q after γ' are a subset of the future states reached after γ.

Step 1: Given two control decisions γ_1, γ_2 in the same equivalence class. If $\gamma_1, \gamma_2 \in \mathcal{C}(q)$, then $\gamma_1 \cup \gamma_2 \in \mathcal{C}(q)$. This can be proven similarly as Step 0. Every future state reached after γ_1 or γ_2 can also be reached after $\gamma_1 \cup \gamma_2$. Since both γ_1 and γ_2 remained after the pruning, then $\gamma_1 \cup \gamma_2$ is not pruned.

Step 2: A control decision γ in C^{comp} is in $\mathcal{C}(q)$ if and only if γ is in $\mathcal{C}^{ext}(q)$. It follows from T^{comp} being a subautomaton of $T^{\max}$ and the pruning process.

Step 3: A control decision γ belongs to $\mathcal{C}(q)$ if and only if γ is in $\mathcal{C}^{ext}(q)$. From Step 2, we only need to show this result for $\gamma \in \Gamma \setminus C^{comp}$.

Only if: Assume $\gamma \in \mathcal{C}(q)$. From Step 0, it follows that $\exists I \in [\![\gamma]\!] \cap C^{comp}$ such that $I \subseteq \mathcal{C}(q)$ and $\gamma = \cup_{\gamma' \in I} \gamma'$. From Step 2, we have that $I \subseteq \mathcal{C}^{ext}(q)$ and $\gamma \in \mathcal{C}^{ext}(q)$.

If: Assume $\gamma \in \mathcal{C}^{ext}(q)$. By definition of $\mathcal{C}^{ext}(q)$, $\exists I \in [\![\gamma]\!] \cap C^{comp}$ such that $I \subseteq \mathcal{C}^{ext}(q)$ and $\gamma = \cup_{\gamma' \in I} \gamma'$. From Step 2, $I \subseteq \mathcal{C}(q)$. Finally, Step 1 provides that the union of the elements of I is also an element of $\mathcal{C}(q)$, i.e., $\gamma \in \mathcal{C}(q)$.

This concludes our proof. □

Theorem **V.2** states that AES_φ^{comp} has the same information as AES_φ, i.e., it encodes all property enforcing supervisors. Therefore, our methodology of obtaining all property-enforcing supervisors for system G is sound and complete. Moreover, it is more efficient than the methodology presented in [14]. In Section VI, we compare the scalability of the two methods.

D. Synthesis of maximally permissive supervisors

Given a state-based property, Theorem **V.2** provides a solution space from where property-enforcing supervisors can be extracted. Here, we present a synthesis algorithm for constructing a supervisor S realized with finite memory that solves Problem **III.1**. This algorithm is based on the MAX-SYNT algorithm presented in [14]. However, we modify the algorithm in order to retrieve control decisions that are not defined in AES_φ^{comp}, i.e., $\mathcal{C}^{ext}_{AES_\varphi^{comp}}$. This algorithm starts from $q_0^{AES^{comp}}$ and in a Depth-First Search manner traverses player 1 and player 2 states. In player 1 states it selects a *locally maximal control decision* while in player 2 states it explores all possible observations. Both player 1 and player 2 states are only visited once. This procedure is described in Algorithm 1. The correctness of this algorithm follows from the correctness of the Algorithm MAX-SYNT in [14]. For notational purposes, we use $\mathcal{C}^{ext}(q) := \mathcal{C}^{ext}_{AES_\varphi^{comp}}(q)$.

***Remark* V.4.** The worst-case running time of the entire procedure is $O(2^{2|X_G|+2|\Sigma_c \cap \Sigma_{uo}|})$. The methodology presented in [14] has a worst-case running time of $O(2^{2|X_G|+2|\Sigma_c|})$. Although this exponential complexity in the number of states of G seems to be unavoidable [3], the exponential complexity in the number of events might be reducible. In this paper, we were able to reduce this complexity from $2^{|\Sigma_c|}$ to $2^{|\Sigma_c \cap \Sigma_{uo}|}$ by exploiting the classes of equivalent control decisions. Note that, if $\Sigma_c \subseteq \Sigma_{uo}$ then our methodology has the

Algorithm 1 MAX-SYNT-COMPACT

Input: AES_{φ}^{comp}; **Output:** DFA S;
$X_S \leftarrow \{q_0^{AES^{comp}}\}$, $\delta_S \leftarrow \emptyset$, $x_{0,S} = q_0^{AES^{comp}}$;
Expand($S, AES_{\varphi}^{comp}, q_0^{AES^{comp}}$);
return S
function EXPAND($S, AES_{\varphi}^{comp}, q$)
  select $\gamma \in \mathcal{C}^{ext}(q)$ s.t. $\forall \gamma' \in \mathcal{C}^{ext}(q) : \gamma \not\subset \gamma'$;
  for all $e \in \gamma$ **do**
    if $e \in \Sigma_o$ **then**
      select $\gamma' \in [\![\gamma]\!]$ s.t. $H_1^{AES^{comp}}(q, \gamma', e)!$
      $q' \leftarrow H_1^{AES^{comp}}(q, \gamma', e)$, $\delta_S \leftarrow \delta_S \cup \{q, e, q'\}$;
      if $q' \notin X_S$ **then**
        $X_S \leftarrow X_S \cup \{q'\}$;
        Expand($S, AES_{\varphi}^{comp}, q'$)
    else
      $\delta_S \leftarrow \delta_S \cup \{q, e, q\}$;

same worst case complexity as in [14]. Moreover, if $\Sigma_c \subseteq \Sigma_o$ then our methodology is only exponential in the number of states in G.

VI. Experimental results

A. Methodology

We implemented and compared both the original and the compact AES methods. Moreover, we also compared the compact AES method to the VLP-PO algorithm [6] and a standard algorithm to compute the supremal controllable and normal sublanguage [5]. All methods were explicitly implemented in the MDESops tool[3].

We evaluated these methods based on randomly generated deterministic finite automata, which were generated based on the REGAL tool [15]. The REGAL tool generates DFAs based on parameters $n, m \in \mathbb{N}_+$ that represent the number of states and the number of events, respectively. We further selected parameters $c, o \in [m]$ and randomly select c controllable events and o observable events. Finally, we randomly selected *one* unsafe $x_{uns} \in X_G$. Although the REGAL tool generates a representative set of DFAs, we leave to future work studying industrial or large scale case studies to complement the set of experiments provided in this section.

All experiments were performed on an Intel Xeon CPU at 3.50GHz with 64GB of RAM running Ubuntu 18.04 LTS OS and a single core was used. Our objective is to compare our method against the original AES methodology and other algorithms that solve Problem **III.1**.

B. AES_{φ} versus AES_{φ}^{comp}

We start by comparing the original AES construction versus the compact one. First, we fix all parameters involved to generate the random DFAs but we vary the intersection between controllable and unobservable events, i.e., we vary $\Sigma_c \cap \Sigma_{uo}$. This demonstrates the potential of the compact method over the original one since the compact method takes advantage of the equivalence classes over the set of control decisions.

We fix the parameters as: $n = 40$, $m = 10$, $c = 6$, $o = 6$. We generate 5 sets of samples based on the constraint $|\Sigma_c \cap \Sigma_{uo}| \in \{0, 1, 2, 3, 4\}$, each of which consists of 100 DFAs. Table I summarizes the results of this experiment where all AES_{φ}^{max} and AES_{φ}^{comp} were obtained within 200 seconds. As expected, T^{comp} and AES_{φ}^{comp} have less states than $T^{\max}$ and AES_{φ}. Moreover, the compact structures exhibit a significant reduction in the number of states for smaller values of $|\Sigma_c \cap \Sigma_{uo}|$. Again, this result is expected

[3]https://gitlab.eecs.umich.edu/M-DES-tools/desops

$\lvert\Sigma_c \cap \Sigma_{uo}\rvert$	4	3	2	1	0
$T^{\max}$	221161	76635	19841	3753	567
T^{comp}	167788	39933	6857	844	81
AES_{φ}	51718	22168	6217	1508	127
AES_{φ}^{comp}	40503	13017	2650	471	39

TABLE I: Average number of states of each structure based on 100 DFAs generated.

since the number of states in T^{comp} is proportional to $2^{|\Sigma_c \cap \Sigma_{uo}|}$. We compare the empirical reduction rate between the number of states in T^{comp} and $T^{\max}$ against the theoretical reduction rate. The theoretical maximum reduction is defined based on the difference of Γ and C^{comp} since in the worst case both $T^{\max}$ and T^{comp} could reach the 2^{X_G} possible state estimates. In this manner, the theoretical normalized reduction ratio is given by:

$$r = \frac{2^{|\Sigma_c|} - (1 + |\Sigma_c \cap \Sigma_o|)2^{|\Sigma_c \cap \Sigma_{uo}|}}{2^{|\Sigma_c|}} \tag{9}$$

Figure 5 compares the empirical reduction value against the theoretical reduction value. Since the two curves are close to each other, it shows that our methodology reduces the states of T^{comp} as expected.

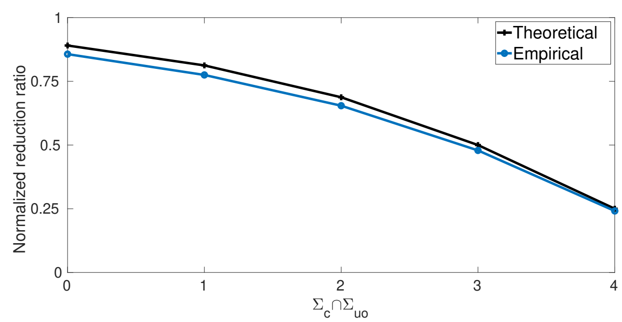


Fig. 5: Empirical and theoretical normalized reduction ratios.

The first experiment demonstrates differences between the compact and the original method but it fails to show how our methodology will scale in comparison to the original one. For this reason, we introduce a second experiment to compare the two methods. In this experiment, we fix the parameter values $m = 12$, $c = 6$, $o = 6$ and randomly generate 200 DFAs for $n \in \{40, 45, \ldots, 100\}$. We do not add any constraint with respect to the intersection $\Sigma_c \cap \Sigma_{uo}$, i.e., the two sets are uniformly randomly selected. We impose a runtime limit of 120 seconds such that we can compare the completion rates of each method. The 120 seconds value is selected based on the runtime of our implementation. These rates are defined as the ratio of completed AES construction under 120 seconds for each $n \in \{40, 45, \ldots, 100\}$. These completion rates are summarized in Fig. 6. As expected, the compact method outperforms the original since it completes at least the same number of problem instances as the original method. We reiterate that our tool is a prototype implementation, and is not optimized for speed.

C. AES_{φ}^{comp} versus standard algorithms

In this section, we design an experiment to compare different algorithms that compute solutions for Problem **III.1** against our compact AES method. As mentioned in Example **II.1**, Problem **III.1** might admit several solutions. For this reason, we focus on the case where the existence of a *unique solution* is guaranteed, i.e., we assume that $\Sigma_c \subseteq \Sigma_o$ which guarantees the existence of the supremal

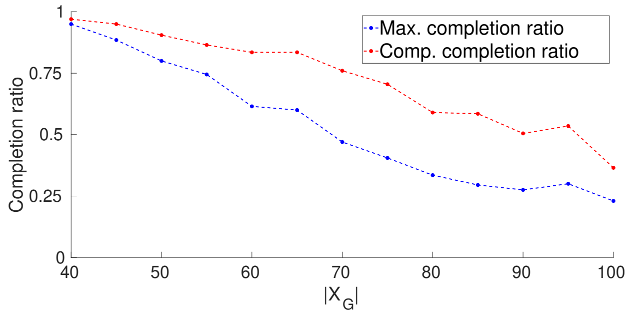


Fig. 6: Completion rates for construction of AES^{max} and AES^{comp} within 120 seconds

controllable and observable sublanguage of a given specification. We compare the compact AES method against the VLP-PO algorithm described in [6] and the algorithm to find the supremal controllable and normal sublanguage (SupCN) described in [5]. This comparison is possible since these three algorithms generate the same unique solution. It is important to remark that while the VLP-PO algorithm solves Problem **III.1**, the SupCN algorithm only solves Problem **III.1** under the assumption that $\Sigma_c \subseteq \Sigma_o$.

In this experiment, we fix the parameter values $m = 12, c = 8, o = 9$, and randomly generate 200 DFAs for each $n \in \{80, 90, \dots, 230\}$ such that $\Sigma_c \subseteq \Sigma_o$. We impose a runtime limit of 30 seconds such that we can compare the completion ratio of each method. Again, this runtime limit is selected based on the time to run these algorithms in the MDESops tool. These ratios are defined as the ratio of successfully extracting a supervisor under 30 seconds for each $n \in \{80, 90, \dots, 230\}$, e.g., after running Algorithm 1 in the case of the AES method. These ratios are summarized in Fig. 7.

As expected, the compact AES method is equivalent to the VLP-PO algorithm since both have a worst-case runtime exponential in the number of states when $\Sigma_c \subseteq \Sigma_o$. Their completion rate difference in Fig. 7 is due to the different way each algorithm obtains the solution. The SupCN algorithm performs badly since it has a computationally expensive preprocessing phase. This preprocessing involves the refinement of G so that it satisfies the State-Partition property, see [5], [16]. Nevertheless, SupCN can be used with non-blockingness properties while VLP-PO and the AES cannot.

Regarding the case when $\Sigma_c \nsubseteq \Sigma_o$, we do not compare the VLP-PO algorithm against the compact AES method since VLP-PO greedily searches for *one* supervisor while the AES method obtains *all* possible property-enforcing supervisors. Therefore, the VLP-PO algorithm outperforms the compact AES method as the VLP-PO has a better worst-case runtime than the compact AES when $\Sigma_c \nsubseteq \Sigma_o$.

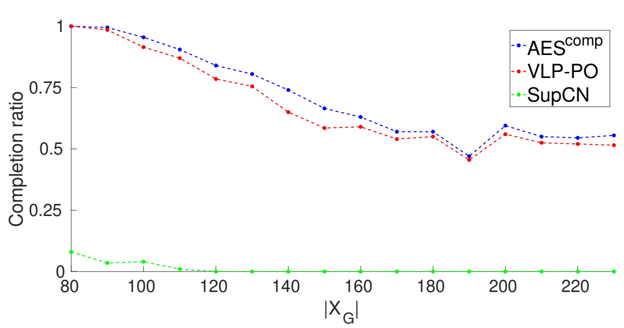


Fig. 7: Completion rates for supervisor construction with AES^{comp}, VLP-PO, and SupCN within 30 seconds

VII. Conclusion

We considered the problem of synthesizing state-based property-enforcing supervisors for partially-observed discrete-event systems. We introduced the compact AES which preserves the same properties as the original AES [14]. Moreover, the compact AES is, in general, computationally more efficient than its original version. We also presented an empirical comparison between both methods that confirms the benefits of the compact AES. In the future, we would like to investigate if the compact method can be used for properties other than state-based properties, e.g., [17]–[19].

References

[1] P. J. 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, Jan. 1987.

[2] R. Cieslak, C. Desclaux, A. S. Fawaz, and P. Varaiya, “Supervisory control of discrete-event processes with partial observations,” *IEEE Transactions on Automatic Control*, vol. 33, no. 3, pp. 249–260, March 1988.

[3] J. N. Tsitsiklis, “On the control of discrete-event dynamical systems,” in *26th IEEE Conference on Decision and Control*, vol. 26, 1987, pp. 419–422.

[4] F. Lin and W. Wonham, “On observability of discrete-event systems,” *Information Sciences*, vol. 44, no. 3, pp. 173 – 198, 1988.

[5] H. Cho and S. I. Marcus, “On supremal languages of classes of sublanguages that arise in supervisor synthesis problems with partial observation,” *Mathematics of Control, Signals and Systems*, vol. 2, no. 1, pp. 47–69, 1989.

[6] N. B. 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 Systems*, vol. 6, no. 4, pp. 379–427, Oct 1996.

[7] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*, 2nd ed. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2008.

[8] W. M. Wonham and K. Cai, *Supervisory Control of Discrete-Event Systems*. Springer International Publishing, 2018.

[9] M. Sampath, S. Lafortune, and D. Teneketzis, “Active diagnosis of discrete-event systems,” *IEEE Transactions on Automatic Control*, vol. 43, no. 7, pp. 908–929, 1998.

[10] E. Badouel, M. Bednarczyk, A. Borzyszkowski, B. Caillaud, and P. Darondeau, “Concurrent secrets,” *Discrete Event Dynamic Systems*, vol. 17, no. 4, p. 425–446, Dec. 2007.

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

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

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

[14] X. Yin and S. Lafortune, “A uniform approach for synthesizing property-enforcing supervisors for partially-observed discrete-event systems,” *IEEE Transactions on Automatic Control*, vol. 61, no. 8, pp. 2140–2154, Aug 2016.

[15] F. Bassino, J. David, and C. Nicaud, “Regal: A library to randomly and exhaustively generate automata,” in *Implementation and Application of Automata*, Berlin, Heidelberg, 2007, pp. 303–305.

[16] G. Jirásková and T. Masopust, “On properties and state complexity of deterministic state-partition automata,” in *Theoretical Computer Science*, Berlin, Heidelberg, 2012, pp. 164–178.

[17] X. Yin and S. Lafortune, “Synthesis of maximally-permissive supervisors for the range control problem,” *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 3914–3929, 2017.

[18] Y. Ji, X. Yin, and S. Lafortune, “Supervisory control under local mean payoff constraints,” in *2019 IEEE 58th Conference on Decision and Control (CDC)*, 2019, pp. 1043–1049.

[19] R. Meira-Goes, S. Lafortune, and H. Marchand, “Synthesis of supervisors robust against sensor deception attacks,” *IEEE Transactions on Automatic Control*, 2021.