Synthesis of Maximally Permissive Nonblocking Supervisors for the Lower Bound Containment Problem

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

***Abstract*—In this paper, we investigate the nonblocking supervisor synthesis problem for centralized partially observed discrete event systems. The goal is to synthesize a maximally permissive nonblocking supervisor that not only satisfies a class of properties, e.g., safety, but also contains a given lower bound behavior described by a regular language. We show that this synthesis problem can be effectively reduced to a synthesis problem that has been solved in the literature. A new notion, called R-compatibility, is proposed for the purpose of reduction. Our result generalizes existing algorithms for supervisory synthesis of partially observed discrete event systems. This also leads to solutions of several synthesis problems that were open previously, e.g., the nonblocking range control problem.**

***Index Terms*—Discrete event systems, lower bound containment, non-blockingness, partial observation, supervisory control.**

I. Introduction

In this paper, we investigate the supervisor synthesis problem for partially observed discrete event systems (DESs) in the supervisory control framework [16]. The goal is to synthesize a supervisor that restricts the system's behavior such that the closed-loop system satisfies some *properties*. In the standard supervisory control problem [16], the properties considered are safety and nonblockingness, where nonblockingness means that the system does not contain a deadlock or a livelock, while safety requires that all strings executed by the system are legal.

In our recent work, a uniform framework for synthesizing property-enforcing supervisors is proposed [22]–[24]. Instead of considering only safety, a wide class of properties, called information-state-based (IS-based) properties, are considered. These properties include, but are not restricted to, safety, opacity [10], diagnosability [28], and detectability [18]. A finite structure called the *nonblocking all enforcement structure* (NB-AES) that embeds all solutions is proposed, and based on the NB-AES, we have shown how to synthesize a maximally permissive nonblocking supervisor that satisfies an IS-based property.

Manuscript received September 24, 2017; revised January 29, 2018; accepted April 5, 2018. Date of publication April 18, 2018; date of current version December 3, 2018. This work was supported in part by the U.S. National Science Foundation under Grant CCF-1138860 (Expeditions in Computing project ExCAPE: Expeditions in Computer Augmented Program Engineering) and Grant CNS-1446298. Recommended by Associate Editor K. Cai. *(Corresponding author: Xiang Yin.)*

X. Yin is with the Department of Automation and Key Laboratory of System Control and Information Processing, Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: yinxiang@sjtu.edu.cn).

S. Lafortune is with the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109 USA (e-mail: 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.2018.2828098

In general, maximally permissive supervisors are not unique in the partially observation setting; there may exist several incomparable locally maximal supervisors. In order to select a "good" locally maximally permissive supervisor, one approach is to impose a *lower bound constraint* and require that the closed-loop behavior of the synthesized supervisor must contain the given lower bound, which is described by a regular language. This leads to the *lower bound containment* problem. The lower bound requirement can be used to model the *desired behavior* the system must achieve; see, e.g., [13], [15] for applications of the lower bound requirement.

In [25], we have investigated a special lower bound containment problem, called the *range control problem*. Specifically, it only considers safety, which can be described by an upper bound language, rather than the general class of IS-based properties. Also, [25] only considers the prefix-closed case and the supervisor synthesized may be blocking. The approach taken in [25] cannot be extended to the nonprefix-closed case or to arbitrary IS-based properties as it strictly depends on the prefix-closed assumption and the safety specification considered; see Section V-C for a more detailed discussion. Therefore, a new methodology is needed to solve the general lower bound containment problem for both nonblockingness and IS-based properties.

In this paper, we tackle the lower bound containment problem in a more general setting. Specifically, instead of considering a safety specification (upper bound), we consider an arbitrary IS-based property. Moreover, we require that the supervisor synthesized must be *nonblocking*. Our approach is to effectively reduce this synthesis problem to the supervisor synthesis problem for nonblockingness and IS-based properties without a lower bound requirement, which has been solved in [22]–[24]. The general idea of our approach is to first locally check whether or not the lower bound behavior can be fulfilled, and then iteratively remove bad states to guarantee that the lower bound behavior can be fulfilled globally. To this end, a new concept called the $\mathbf{R}$-compatibility is proposed for the purpose of reduction and a new structure called the $\mathbf{R}$-*compatible nonblocking all enforcement structure* ($\mathbf{R}$-compatible NB-AES) is defined. One of the difficulties in this problem is that we need to handle, simultaneously, the lower bound requirement and the nonblockingness requirement, which may be coupled in general. However, our result reveals that these two requirements are actually independent, and we can first restrict the solution space by considering the lower bound requirement before enforcing nonblockingness. Also, we identify that the notion of strict subautomaton plays a crucial role in the correctness of reduction. To the best of our knowledge, there is no algorithm for finding an arbitrary *nonblocking* safe supervisor that contains a given lower bound language. Our paper solves this problem in a more general setting as we are able to find a maximally permissive nonblocking supervisor that contains a given lower bound language subject to any IS-based property.

The rest of this paper is organized as follows. In Section II, we present necessary preliminaries and formulate the lower bound containment

problem. Section III reviews the uniform framework for supervisory control under partial observation proposed in our recent work. The main result of this paper is presented in Section IV. Specifically, a new notion called the $\mathbf{R}$-compatibility is proposed, and we show how to leverage an existing algorithm to solve the lower bound containment problem by using this notion. In Section V, we further compare our new result with existing results in the literature. Finally, we conclude the paper in Section VI.

II. Problem Formulation

A. Preliminaries

Throughout the paper, we use Σ to denote a finite set of events and Σ^* is the set of finite strings over Σ including the empty string ϵ. For any string $s \in \Sigma^*$, $|s|$ denotes its length with $|\epsilon| = 0$. A language $L \subseteq \Sigma^*$ is a set of strings. Notation $\overline{L}$ denotes the prefix-closure of language L, i.e., $\overline{L} = \{u \in \Sigma^* : \exists v \in \Sigma^* \text{ s.t. } uv \in L\}$.

We consider a DES modeled as a finite-state automaton $\mathbf{G} = (X, \Sigma, \delta, x_0, X_m)$, 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, $x_0 \in X$ is the initial state, and $X_m \subseteq X$ is the set of marked states (X_m can be omitted when marking is not considered). The transition function δ is extended to $X \times \Sigma^*$ recursively by the following: For any $s \in \Sigma^*, \sigma \in \Sigma$, $\delta(x, s\sigma) = \delta(\delta(x, s), \sigma)$. For brevity, we will also write $\delta(x, s)$ as $\delta(s)$ if $x = x_0$. The language *generated* by $\mathbf{G}$ is $\mathcal{L}(\mathbf{G}) := \{s \in \Sigma^* : \delta(x_0, s)!\}$, where "!" means "defined"; the language *marked* by $\mathbf{G}$ is $\mathcal{L}_m(\mathbf{G}) := \{s \in \Sigma^* : \delta(x_0, s) \in X_m\}$. Let $\mathbf{A} = (X_A, \Sigma, \delta_A, x_{A,0})$ and $\mathbf{B} = (X_B, \Sigma, \delta_B, x_{B,0})$ be two automata. We say that $\mathbf{A}$ is a strict subautomaton of $\mathbf{B}$, denoted by $\mathbf{A} \sqsubset \mathbf{B}$, if (i) $\forall s \in \mathcal{L}(\mathbf{A}) : \delta_A(s) = \delta_B(s)$; and (ii) $\forall s \in \mathcal{L}(\mathbf{B}) \setminus \mathcal{L}(\mathbf{A}) : \delta_B(s) \notin X_A$. The first condition implies that $\mathbf{A}$ is a "subgraph" of $\mathbf{B}$, while the second condition implies that once a string leaves the state-space of $\mathbf{A}$, it should never come back. Note that, if $\mathcal{L}(\mathbf{A}) \subseteq \mathcal{L}(\mathbf{B})$, then we can always refine the state spaces of $\mathbf{A}$ and $\mathbf{B}$ such that $\mathbf{A} \sqsubset \mathbf{B}$ [7].

In the supervisory control framework [16], the event set Σ is partitioned as follows: (i) $\Sigma = \Sigma_c \dot\cup \Sigma_{uc}$, where Σ_c and Σ_{uc} are the sets of controllable and uncontrollable events, respectively; and (ii) $\Sigma = \Sigma_o \dot\cup \Sigma_{uo}$, where Σ_o and Σ_{uo} are the sets of observable and unobservable events, respectively. We assume that the supervisor cannot disable uncontrollable events. Therefore, we denote by $\Gamma := \{\gamma \in 2^\Sigma : \Sigma_{uc} \subseteq \gamma\}$ the set of all control decisions. Also, we assume that the supervisor can only observe the occurrences of observable events [8], [12]. To this end, we define the natural projection $P : \Sigma^* \to \Sigma_o^*$ 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 projection P is extended to 2^{Σ^*} by $P(L) = \{t \in \Sigma_o^* : \exists s \in L \text{ s.t. } t = P(s)\}$ and P^{-1} denotes the inverse projection.

A supervisor is a function $S : P(\mathcal{L}(\mathbf{G})) \to \Gamma$ that disables events dynamically based on its observations. We denote by $\mathcal{L}(S/\mathbf{G})$ the language generated by the closed-loop system under control, which can be computed recursively by

1) $\epsilon \in \mathcal{L}(S/\mathbf{G})$; and
2) $s\sigma \in \mathcal{L}(S/\mathbf{G}) \Leftrightarrow s \in \mathcal{L}(S/\mathbf{G}) \wedge s\sigma \in \mathcal{L}(\mathbf{G}) \wedge \sigma \in S(P(s))$.

We define $\mathcal{L}_m(S/\mathbf{G}) := \mathcal{L}(S/\mathbf{G}) \cap \mathcal{L}_m(\mathbf{G})$. We say that supervisor S is *nonblocking* if $\mathcal{L}_m(S/\mathbf{G}) = \mathcal{L}(S/\mathbf{G})$.

B. Problem Formulation

Since we consider partially observed systems, we define an *information state* as a set of states and denote by $I = 2^X$ the set of information states. Let S be a supervisor and $s \in P(\mathcal{L}(S/\mathbf{G}))$ be an observable string. We define

$$\mathcal{E}_S(s) := \{x \in X : \exists t \in \mathcal{L}(S/\mathbf{G}) \text{ s.t. } \delta(t) = x \wedge P(t) = s\}$$

as the *state-estimate* upon the occurrence of s, which is the set of states the system could be in by observing s.

The goal of the supervisor is to restrict the system's behavior such that some specification (or property) is satisfied. In this paper, we consider a general class of properties called IS-based properties defined in [24].

Definition 1: ([24]) An IS-based property is a predicate $\varphi : I \to \{0, 1\}$. We say that supervisor S *enforces* φ w.r.t. $\mathbf{G}$, denoted by $S \models_{\mathbf{G}} \varphi$, if $\forall s \in P(\mathcal{L}(S/\mathbf{G})) : \varphi(\mathcal{E}_S(s)) = 1$, i.e., the state-estimate of the system should always satisfy the predicate.

Remark 1: It was shown in [24] that many important properties in the literature, e.g., safety, opacity, and K-diagnosability, can be formulated as IS-based properties. Therefore, for the sake of generality, we investigate this general class of properties rather than studying a specific property. For example, suppose that X_{bad} is a set of *illegal states* and the control objective is to avoid reaching illegal states; this is the safety specification in the standard supervisory control problem [6]. To enforce safety, it suffices to enforce an IS-based property φ_{safe} defined by $\forall i \in I : \varphi_{\text{safe}}(i) = 1 \Leftrightarrow i \cap X_{\text{bad}} = \emptyset$. Throughout the paper, we will use safety as an example of the IS-based property. The reader is referred to [24] for how to formulate more properties in terms of IS-based properties.

In [23] and [24], we have shown how to synthesize a maximally permissive (in terms of set inclusion) nonblocking supervisor that enforces an IS-based property φ. In general, such a supervisor is not unique, i.e., there may exist two maximally permissive, nonblocking, and φ-enforcing supervisors whose closed-loop languages are incomparable. In order to select a 'meaningful' maximally permissive supervisor, one approach is to impose a *lower bound language* [25] and to require that the closed-loop behavior under control has to contain the given lower bound, which is a prefix-close language $R = \overline{R} \subseteq \mathcal{L}(\mathbf{G})$. This leads to the maximally permissive nonblocking lower bound containment problem for the IS-based property (MPLCP-NBIS).

Problem 1: (MPLCP-NBIS). Given system $\mathbf{G}$, an IS-based property $\varphi : I \to \{0, 1\}$ and lower bound language $R \subseteq \mathcal{L}(\mathbf{G})$, synthesize a nonblocking and φ-enforcing supervisor S^* such that

1) $R \subseteq \mathcal{L}(S^*/\mathbf{G})$; and
2) for any nonblocking and φ-enforcing supervisor S' satisfying (i), we have $\mathcal{L}(S^*/\mathbf{G}) \not\subset \mathcal{L}(S'/\mathbf{G})$, i.e., S^* is *maximally permissive*.

Hereafter, we use $\mathbf{R} = (X_R, \Sigma, \delta_R, x_{0,R})$ to denote the automaton generating R. We assume, without loss of generality, that $\mathbf{R} \sqsubset \mathbf{G}$. This assumption is needed for technical reasons; it will become clear in Section IV.

III. All Enforcement Structure

In this section, we review the structure of Bipartite Transition System (BTS) and its variants from our recent work [23], [24]. First, we define the following operators. For any $i \in I$, $\gamma \in \Gamma$, and $\sigma \in \Sigma_o$

$$\text{UR}_\gamma(i) = \{x \in X : \exists y \in i, \exists s \in (\Sigma_{uo} \cap \gamma)^* \text{ s.t. } x = \delta(y, s)\} \tag{2}$$

$$\text{Next}_\sigma(i) = \{x \in X : \exists y \in i \text{ s.t. } x = \delta(y, \sigma)\}. \tag{3}$$

Now, we recall the notion of BTS from [23].

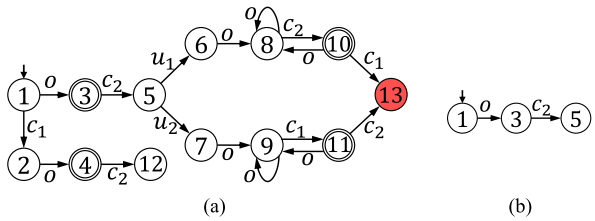

Fig. 1. System with $\Sigma_c = \{c_1, c_2\}$ and $\Sigma_o = \{o\}$. (a) $\mathbf{G}$. (b) $\mathbf{R}$.

Definition 2: A BTS T w.r.t. $\mathbf{G}$ is a 7-tuple

$$T = (Q_Y^T, Q_Z^T, h_{YZ}^T, h_{ZY}^T, \Sigma_o, \Gamma, y_0) \tag{4}$$

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- 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

$$h_{YZ}^T(y, \gamma) = z \Rightarrow [I(z) = \mathrm{UR}_\gamma(y)] \wedge [\Gamma(z) = \gamma] \tag{5}$$

$h_{ZY}^T : Q_Z^T \times \Sigma_o \to Q_Y^T$ is the partial transition function from Z- to Y-states, which satisfies the following constraint: For any $y \in Q_Y^T$, $z \in Q_Z^T$, and $\sigma \in \Sigma_o$, we have

$$h_{ZY}^T(z, \sigma) = y \Leftrightarrow [\sigma \in \Gamma(z)] \wedge [y = \mathrm{Next}_\sigma(I(z))] \tag{6}$$

Σ_o is the set of observable events of $\mathbf{G}$, Γ is the set of control decisions of $\mathbf{G}$, and $y_0 \in Q_Y^T$ is the initial Y-state, where $y_0 = \{x_0\}$.

Let T be a BTS, for each Y-state $y \in Q_Y^T$, we denote by $C_T(y) := \{\gamma \in \Gamma : h_{YZ}^T(y, \gamma)!\}$ the set of control decisions defined at y. We say that T is *complete*, if $\forall y \in Q_Y^T : C_T(y) \neq \emptyset$; we say that T is *deterministic*, if $\forall y \in Q_Y^T : |C_T(y)| = 1$. When T is deterministic, we also use notation $c_T(y)$ to denote the unique control decision defined at $y \in Q_Y^T$. For simplicity, we also write $y \xrightarrow{\gamma}_T z$ if $z = h_{YZ}^T(y, \gamma)$ and $z \xrightarrow{\sigma}_T y$ if $y = h_{ZY}^T(z, \sigma)$; we will drop subscript T when it is clear. We call $\gamma_0 \sigma_1 \gamma_1 \sigma_2 \ldots \sigma_n \gamma_n$ a *run*, where $\gamma_i \in \Gamma, \sigma_i \in \Sigma_o$. A run also induces a *sequence*

$$y_0 \xrightarrow{\gamma_0} z_0 \xrightarrow{\sigma_1} y_1 \xrightarrow{\gamma_1} \cdots \xrightarrow{\gamma_{n-1}} z_{n-1} \xrightarrow{\sigma_n} y_n \xrightarrow{\gamma_n} z_n .$$

We say that a run is generated by T if its induced sequence is defined in T.

Let $S : P(\mathcal{L}(\mathbf{G})) \to \Gamma$ be a supervisor and $s = \sigma_1, \ldots, \sigma_n \in P(\mathcal{L}(S/\mathbf{G}))$ be an observed string. Then, the execution of s induces a well-defined sequence

$$y_0 \xrightarrow{S(\epsilon)} z_0 \xrightarrow{\sigma_1} y_1 \xrightarrow{S(\sigma_1)} \cdots \xrightarrow{\sigma_n} y_n \xrightarrow{S(\sigma_1, \ldots, \sigma_n)} z_n .$$

We denote by $IS_S^Y(s)$ and $IS_S^Z(s)$ the last Y- and Z-states in $y_0 z_0 y_1 z_2 \ldots z_{n-1} y_n z_n$, respectively, i.e., $IS_S^Y(s) = y_n$ and $IS_S^Z(s) = z_n$. In particular, we have that $I(IS_S^Z(s)) = \mathcal{E}_S(s)$, i.e., the information component of $I(IS_S^Z(s))$ is exactly the state-estimate by observing s under supervisor S.

With the above-mentioned notions, we can "decode" supervisors from a BTS as explained in the following definition.

Definition 3: A supervisor S is said to be included in a BTS T if for any $s \in P(\mathcal{L}(S/\mathbf{G}))$, the control decision made by S is defined at the corresponding Y-state, i.e., $S(P(s)) \in C_T(IS_S^Y(s))$. We denote by $\mathbb{S}(T)$ the set of supervisors included in T.

Example 1: Let us consider system $\mathbf{G}$ shown in Fig. 1 (a), where $\Sigma_c = \{c_1, c_2\}$, $\Sigma_o = \{o\}$ and marked states are denoted by double circles. This example will be used as the running example throughout the paper. Suppose that we consider an IS-based property φ defined by $\forall i \in I : \varphi(i) = 1 \Leftrightarrow i \cap \{13\} = \emptyset$, i.e., φ is a safety specification and 13 is the unique illegal state. Then, Fig. 2 (a) shows an example of a complete BTS for $\mathbf{G}$. From the initial Y-state $\{1\}$, there are two control decisions defined, i.e., the BTS is not deterministic. If control decision $\{c_1\}$ is made,[1] then we move to Z-state $(\{1, 2\}, \{c_1\})$, where $\{1, 2\}$ is the set of states the system could be in before the occurrence of the next observable event and $\{c_1\}$ is a copy of the control decision leading to this state. For $(\{1, 2\}, \{c_1\})$, observable event o can occur and we move to Y-state $\{3, 4\}$, which is the set of states the system could be in immediately after observing o, and so forth. For example, let us consider a supervisor S defined by

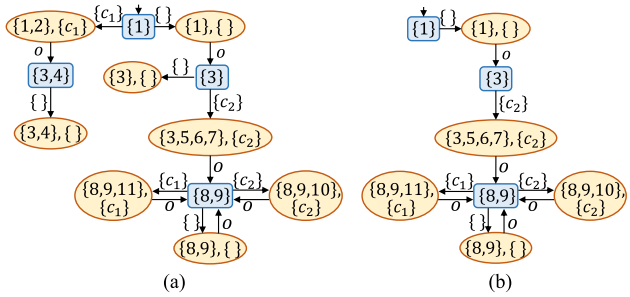

Fig. 2. $\mathbf{R}$-compatible NB-AES. (a) $\mathcal{AES}_\varphi(\mathbf{G})$. (b) $\mathcal{AES}_\varphi^R(\mathbf{G})$.

$$S(\epsilon) = \{c_1\} \text{ and } S(o) = \Sigma_{uc}. \tag{7}$$

Then, we know that S is included in the BTS shown in Fig. 2(a), and we have $IS_S^Y(\epsilon) = \{1\}$, $IS_S^Z(\epsilon) = (\{1, 2\}, \{c_1\})$, $IS_S^Y(o) = \{3, 4\}$, and $IS_S^Z(o) = (\{3, 4\}, \emptyset)$.

Definition 4: Given a BTS T, we say that a Y-state y is *live* in T if, for any $x \in y$, there exist a string $s = \xi_1 \sigma_1 \xi_2, \ldots, \sigma_{n-1} \xi_n$, where $\xi_i \in \Sigma_{uo}^*, \sigma_i \in \Sigma_o$, and a sequence $y \xrightarrow{\gamma_1}_T z_1 \xrightarrow{\sigma_1}_T y_1 \xrightarrow{\gamma_2}_T \cdots \xrightarrow{\sigma_{n-1}}_T y_n \xrightarrow{\gamma_n}_T z_n$ such that $\delta(x, s) \in X_m$ and $\forall i \leq n : \xi_i \in \gamma_i^*$.

Definition 5: A Z-state z is said to be *deadlock-free* if for all $x \in I(z)$ we have either (i) $\exists s \in (\Gamma(z) \cap \Sigma_{uo})^* : \delta(x, s) \in X_m$; or (ii)$\exists s \in (\Gamma(z) \cap \Sigma_{uo})^* (\Gamma(z) \cap \Sigma_o) : \delta(x, s)!$. Otherwise, z is said to be a *deadlock* Z-state.

Definition 6: A BTS T is said to be *nonblocking* if all Y-states in it are live and all Z-states in it are deadlock-free. We say that T satisfies φ if $\forall z \in Q_Z^T : \varphi(I(z)) = 1$.

Intuitively, a Y-state is live if any state in it can reach a marked state under some control decision string allowed by the BTS. Also, a Z-state is deadlock-free if any state in it can either unobservably reach a marked state or can go outside of this Z-state. Nonlive Y-states and deadlock Z-states are states that should be avoided, since both of them cause blocking.

Definition 7: NB-AES for $\mathbf{G}$ and φ, denoted by $\mathcal{AES}_\varphi(\mathbf{G}) = (Q_Y^{AES}, Q_Z^{AES}, h_{YZ}^{AES}, h_{ZY}^{AES}, \Sigma_o, \Gamma, y_0)$, is defined as the largest (in terms of graph-merger) complete and nonblocking BTS satisfying φ.

Example 2: Let us return to the running example. In fact, the BTS shown in Fig. 2(a) is the NB-AES for $\mathbf{G}$. For example, Y-state $\{8, 9\}$ is live, since state 8 can reach marked state 10 under control decision $\{c_2\}$ and state 9 can reach marked state 11 under control decision $\{c_1\}$. Also, all Z-states in it are deadlock-free. Since $\mathcal{AES}_\varphi(\mathbf{G})$ is the largest complete and nonblocking BTS satisfying φ, no control decision can be added at any Y-state. For example, at Y-state $\{3, 4\}$, we cannot make

[1]For the sake of simplicity, in the figure, we omit uncontrollable events in each control decision, which are always enabled. Also, we omit events that are not feasible in each control decision at the current Y-state. For example, we do not consider decisions $\{c_2\}$ and $\{c_1, c_2\}$ at Y-state $\{1\}$ as event c_2 is not feasible within the unobservable reach.

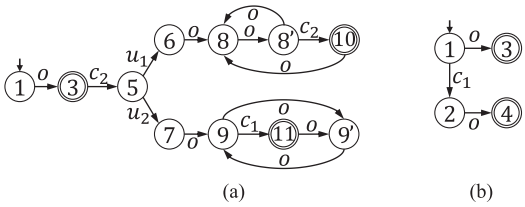

Fig. 3. Closed-loop languages under control. (a) $\mathcal{L}(S^*/\mathbf{G})$. (b) $\mathcal{L}(S/\mathbf{G})$.

control decision $\{c_2\}$; otherwise the system may be blocked at state 12. Also, at Y-state $\{8,9\}$, we cannot make control decision $\{c_1, c_2\}$; otherwise the system may unobservably reach illegal state 13, which will violate φ.

The following result says that we can always synthesize a maximally permissive nonblocking supervisor satisfying φ when there exists a complete and nonblocking BTS T satisfying φ.

Theorem 1: ([22]) For any complete and nonblocking BTS T that satisfies φ, there always exists a maximally permissive, nonblocking, and φ-enforcing supervisor $S \in \mathbb{S}(T)$ included in it, where maximal-permissiveness means that there does not exist another nonblocking and φ-enforcing supervisor $S' \in \mathbb{S}(T)$ such that $\mathcal{L}(S/\mathbf{G}) \subset \mathcal{L}(S'/\mathbf{G})$. Moreover, such a supervisor S can be effectively synthesized by Algorithm NB-SOLU in [22].[2]

To synthesize an arbitrary maximally permissive, nonblocking, and φ-enforcing supervisor, one can use the NB-AES as the input of Algorithm NB-SOLU. However, this does not guarantee the containment of R. In this paper, we will only invoke Algorithm NB-SOLU from [22] in order to extract a maximally permissive, nonblocking, and φ-enforcing supervisor from a complete and nonblocking BTS that satisfies φ. Algorithm NB-SOLU has been implemented in the software tool DPO-SYN[3] [26]. The algorithm itself is rather technical and is beyond the scope of this paper; the reader is referred to [22] for more details on this algorithm.

IV. Handling the Lower Bound

Although Algorithm NB-SOLU in [22] can find a maximally permissive, nonblocking, and φ-enforcing supervisor, it cannot guarantee that the supervisor synthesized contains the desired lower bound language $R = \mathcal{L}(\mathbf{R})$. For example, supervisor S defined in (7) is a maximally permissive nonblocking supervisor for system $\mathbf{G}$ shown in Fig. 1(a); its closed-loop language under control $\mathcal{L}(S/\mathbf{G})$ is shown in Fig. 3(b). In fact, this is the supervisor returned by Algorithm NB-SOLU in [22] when we use the NB-AES as its input. However, suppose that we consider the lower bound language R shown in Fig. 1(b), then this supervisor fails to achieve this lower bound behavior. In this section, we show how to leverage Algorithm NB-SOLU in [22] to synthesize a maximally permissive, nonblocking, and φ-enforcing supervisor that provably contains the given lower bound language.

A. *R*-Compatible NB-AES

Recall that $\mathbf{R} = (X_R, \Sigma, \delta_R, x_{0,R})$ is the automaton generating R, which is assumed to be a strict subautomaton of $\mathbf{G}$. Let $i \in I$ be an information state. First, we define

$$\Gamma_R(i) = \{\sigma \in \Sigma : \exists x \in i \cap X_R, \exists w \in \Sigma_{uo}^* \text{ s.t. } \delta_R(x, w\sigma)!\} \quad (8)$$

as the set of events defined at some state that can be reached unobservably from some state in i.

Then, we introduce the notion of **R**-compatibility.

Definition 8: We say that a BTS T is **R**-*compatible* if

$$\forall y \in Q_T^Y, \forall \gamma \in C_T(y) : \Gamma_R(y) \subseteq \gamma. \quad (9)$$

Intuitively, **R**-compatibility requires that, for any Y-state y, any control decision defined at this Y-state should contain events that are feasible unobservably from a state $x \in y$ in $\mathbf{R}$. Clearly, if two BTSs are **R**-compatible, then their union (in terms of graph merger) is still **R**-compatible. Therefore, the largest complete, nonblocking, and **R**-compatible BTS satisfying φ is well defined, and we term it as the **R**-*compatible NB-AES*.

Definition 9: The **R**-compatible NB-AES for $\mathbf{G}$, $\mathbf{R}$, and φ, denoted by $\mathcal{AES}_\varphi^R(\mathbf{G}) = (Q_Y^R, Q_Z^R, h_{YZ}^R, h_{ZY}^R, \Sigma_o, \Gamma, y_0)$, is defined as the largest complete, nonblocking, and **R**-compatible BTS satisfying φ.

Algorithm 1 shows how to construct the **R**-compatible NB-AES $\mathcal{AES}_\varphi^R(\mathbf{G})$. First, we need to construct the NB-AES $\mathcal{AES}_\varphi(\mathbf{G})$ as $\mathcal{AES}_\varphi^R(\mathbf{G})$ is a subsystem of $\mathcal{AES}_\varphi(\mathbf{G})$ by definition. Then, for each Y-state, we need to remove all control decisions that violate **R**-compatibility; this is implemented by lines 3–5. However, by removing such control decisions, the remaining BTS may not be nonblocking as it is possible that some state in a Y-state may not be able to reach a marked state. Therefore, we need to remove all Y-states that are not live in the new structure and then remove their predecessor Z-states to guarantee the completeness of the structure, and repeat this procedure until the structure is nonblocking and complete. This is implemented by the while-loop from line 6 to line 9.

Algorithm 1:

```
   input : G, R, and φ
   output: AES_φ^R(G)
1  Construct AES_φ(G) by the algorithm in [22];
2  AES_φ^R(G) ← AES_φ(G);
3  for all Y-state y ∈ Q_Z^R do
4      if ∃γ ∈ C_{AES_φ^R(G)}(y) : Γ_R(y) ⊈ γ then
5          Delete γ from C_{AES_φ^R(G)}(y) by removing its
           associated transition;
6  while AES_φ^R(G) is not nonblocking do
7      for all Y-state y ∈ Q_Z^R do
8          if y is not live in AES_φ^R(G) then
9              Delete y and all its predecessor Z-states and
               the associated transitions;
10 if y_0 ∈ Q_Z^{NB} then
11     return AES_φ^R(G) ;
   else
12     return "R-compatible NB-AES does not exist";
```

Let us illustrate the construction of the **R**-compatible NB-AES by the following example.

Example 3: Let us return to our running example. The NB-AES for $\mathbf{G}$ is shown in Fig. 2(a). However, $\mathcal{AES}_\varphi(\mathbf{G})$ is not **R**-compatible. For example, for Y-states $\{3\}$ and $\{3,4\}$, we have $\Gamma_R(\{3\}) = \Gamma_R(\{3,4\}) = \{c_2\}$. However, control decision $\emptyset$ is defined at both Y-states and $\{c_2\} \not\subseteq \emptyset$. Therefore, we need to remove control decision $\emptyset$ from $\{3\}$ and $\{3,4\}$. Since the only control decision is removed from $\{3,4\}$, this Y-state becomes nonlive. Therefore, we need to again remove $\{3,4\}$ and its predecessor Z-state $(\{1,2\}, \{c_1\})$. This results in

[2]Algorithm NB-SOLU was originally presented in [23] for the case of safety specifications only. The new Algorithm NB-SOLU in [22] generalized the original one by combining it with the general IS-based framework proposed in [24].

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

the structure shown in Fig. 2(b). Note that Y-state $\{3\}$ does not need to be removed as control decision $\{c_2\}$, which makes it live, is still defined at this state. Since the structure shown in Fig. 2(b) is nonblocking, we terminate the while loop and this structure is indeed the **R**-compatible NB-AES (which will be formally proved next).

Theorem 2: Algorithm 1 correctly constructs the **R**-compatible NB-AES.

Proof: Let T be the BTS returned by Algorithm 1. First, we know that T is a complete BTS, since $|C_{\mathcal{AES}^R_\varphi(\mathbf{G})}(y)| \neq \emptyset$ by construction, and whenever a Y-state is removed, its predecessor Z-states are removed. Second, we know that T satisfies φ since its state-space is a subset of that of the NB-AES, which satisfies φ. Also, T is nonblocking: (i) any Y-state in it is live by construction; and (ii) any Z-state in it is deadlock-free since it is also in the state-space of the NB-AES and none of its successor Y-states are removed according to line 9. Finally, we know that T is **R**-compatible since at any Y-state, any control decision violating this requirement is removed. Therefore, it remains to show that T is the largest BTS with the above properties. Next, we show this by contradiction.

Let us assume that there exists another complete, nonblocking, and **R**-compatible T' satisfying φ, and T' is strictly larger than T. Let us consider what happens when the "while" loop is executed for the first time. Since T' is a subsystem of $\mathcal{AES}_\varphi(\mathbf{G})$ and T' is nonblocking, then we know that any Y-state in T' is also live in $\mathcal{AES}_\varphi(\mathbf{G})$. Also, since T' is a complete and **R**-compatible, for any Y-state y in T' we have that (i) $|C_{\mathcal{AES}^R_\varphi(\mathbf{G})}(y)| \geq 1$, and (ii) $\forall \gamma \in C_{\mathcal{AES}^R_\varphi(\mathbf{G})}(y) : \Gamma_R(y) \subseteq \gamma$. Therefore, we know that no control decision at any Y-state in T' will be removed. Moreover, since T' is complete, we know that any Z-state in T' can only reach Y-states in T'. Since no Y-state in T' is removed, we know that no Z-state in T' will be removed. Overall, the first execution of the "while" loop can only remove states and transitions outside of T', which means that T' is still a subsystem of the resulting BTS with state removal. Therefore, by the same reason, we know that for the second time the "while" loop is executed, no state in T' will be removed, and so forth. Overall, we know that the "while" loop will converge to a BTS that at least contains T'. This contradicts the fact that T is returned by the algorithm. ■

B. Properties of the **R**-compatible NB-AES

Now, we show how the **R**-compatible NB-AES can be combined with Algorithm NB-SOLU to solve the lower bound containment problem. First, we show that any supervisor included in $\mathcal{AES}^R_\varphi(\mathbf{G})$ contains R.

Lemma 1: Let T be an **R**-compatible BTS. Then, for any $S \in \mathbb{S}(T)$, we have $R \subseteq \mathcal{L}(S/\mathbf{G})$.

Proof: It suffices to show that $\forall s \in R : s \in \mathcal{L}(S/\mathbf{G})$. We prove this by induction on the length of s.

For $|s| = 0$, we know that $\epsilon \in \mathcal{L}(S/\mathbf{G})$ by definition. Let us assume that for any $s \in R$ such that $|s| = k$, we have $s \in \mathcal{L}(S/\mathbf{G})$. We need to show that for any $s\sigma \in R$, where $\sigma \in \Sigma$ and $|s| = k$, we still have $s\sigma \in \mathcal{L}(S/\mathbf{G})$. Let $y = IS^Y_S(P(s))$ be the Y-state reached by $P(s)$ under S. Since $S \in \mathbb{S}(T)$, we know that $S(P(S)) \in C_T(IS^Y_S(P(s)))$. Since T is **R**-compatible, we know that $\Gamma_R(y) \subseteq S(P(s))$. We write string s in the form of $s = s'w$, where s' ends with an observable event and $w \in \Sigma^*_{uo}$. Then, we know that $\delta(s') \in y$. Moreover, since $s' \in R$, we know that $\delta(s') \in X_R$. Since $\delta_R(\delta(s'), w\sigma)!$, we know that $\sigma \in \Gamma_R(y) \subseteq S(P(s))$. Recall that, by the induction hypothesis, $s \in \mathcal{L}(S/\mathbf{G})$. This together with $\sigma \in S(P(s))$ and $s\sigma \in R \subseteq \mathcal{L}(\mathbf{G})$ implies that $s\sigma \in \mathcal{L}(S/\mathbf{G})$, which completes the induction step. ■

Next, we show that the existence of the **R**-compatible BTS is necessary for the existence of a nonblocking and φ-enforcing supervisor that contains R.

Lemma 2: Suppose that there exists a nonblocking and φ-enforcing supervisor S such that $R \subseteq \mathcal{L}(S/\mathbf{G})$. Then, there must exist a nonblocking and **R**-compatible BTS T satisfying φ.

Proof: Let S be a nonblocking and φ-enforcing supervisor such that $R \subseteq \mathcal{L}(S/\mathbf{G})$. Then, we construct a complete BTS T as follows:

1) $Q^T_Y := \{y \in I : \exists s \in P(\mathcal{L}(S/\mathbf{G}))$ s.t. $y = IS^Y_S(s)\}$
2) $Q^T_Z := \{z \in I \times \Gamma : \exists s \in P(\mathcal{L}(S/\mathbf{G}))$ s.t. $z = IS^Z_S(s)\}$
3) For any $y \in Q^T_Y$, $C_T(y) := \{\gamma \in \Gamma : \exists s \in P(\mathcal{L}(S/\mathbf{G}))$ s.t. $y = IS^Y_S(s) \wedge \gamma = S(s)\}$.

By construction, we have $S \in \mathbb{S}(T)$. Note that T need not be deterministic since the control decision of S need not depend on the information-state encountered. Next, we show that, under the assumption that $\mathbf{R} \sqsubset \mathbf{G}$, T is a nonblocking and **R**-compatible BTS satisfying φ.

By the definition of φ-enforcing supervisor, we know that $\forall s \in P(\mathcal{L}(S/\mathbf{G})) : \varphi(I(IS^Z_S(s))) = 1$. Therefore, by the construction of T, we have $\forall z \in Q : \varphi(I(z)) = 1$, i.e., T satisfies φ.

To show that T is nonblocking, we need to show that $\forall y \in Q^T_Y : y$ is live and $\forall z \in Q^T_Z : z$ is deadlock-free. First, let us consider an arbitrary $y \in Q^T_Y$ and an arbitrary $x \in y$. Then, we know that there exists a string $s \in \mathcal{L}(S/\mathbf{G})$ such that $\delta(s) = x$. Since S is a nonblocking supervisor, we know that there exists a string t such that $st \in \mathcal{L}(S/\mathbf{G})$ and $\delta(st) \in X_m$. Then, let $P(t) = \sigma_1, \ldots, \sigma_{|P(t)|}$, then

$$y \xrightarrow{S(P(s))}_T z_1 \xrightarrow{\sigma_1}_T y_1 \xrightarrow{S(P(s)\sigma_1)}_T \cdots \xrightarrow{\sigma_n}_T y_n \xrightarrow{S(P(st))}_T z_n \tag{10}$$

yields a sequence that satisfies liveness, i.e., any $y \in Q^T_Y$ is live. Next, let us consider an arbitrary $z \in Q^T_Z$ and $x \in I(z)$. Still, we know that there exists a string $s \in \mathcal{L}(S/\mathbf{G})$ such that $\delta(s) = x$ and $IS^Z_S(P(s)) = z$. Since S is a nonblocking supervisor, we know that there exists a string t such that $st \in \mathcal{L}(S/\mathbf{G})$ and $\delta(st) \in X_m$. If $t \in \Sigma^*_{uo}$, then we know that $t \in S(P(s))^* = \Gamma(z)^*$ and condition (i) in Definition 5 is satisfied. If $t \notin \Sigma^*_{uo}$, then we know that it must be in the form of $t = w\sigma w'$, where $w \in \Sigma^*_{uo}$ and $\sigma \in \Sigma_o$, and $w\sigma \in S(P(s))^* = \Gamma(z)^*$. Therefore, condition (ii) in Definition 5 is satisfied. Overall, we know that any Z-state in T is deadlock-free, which together with the fact that all Y-states are live imply that T is nonblocking.

Finally, we show that T is **R**-compatible. Let us consider an arbitrary Y-state $y \in Q^Y_T$ and an arbitrary control decision $\gamma \in C_T(y)$ defined at y. By the definition of T, we know that there exists a string $s \in \mathcal{L}(S/\mathbf{G})$ such that $IS^Y_S(P(s)) = y$ and $S(P(s)) = \gamma$. Now, let us assume for the sake of contradiction that $\Gamma_R(y) \not\subseteq \gamma$. Let $\sigma \in \Gamma_R(y) \setminus \gamma$. Then, we know that there exist $x \in y \cap X_R$, $w \in \Sigma^*_{uo}$ such that $\delta_R(x, w\sigma)!$. For this x, there exists $t \in \mathcal{L}(S/\mathbf{G})$ such that $P(s) = P(t)$ and $\delta(t) = x$. Since $x \in X_R$ and $\mathbf{R} \sqsubset \mathbf{G}$, we know that $t \in \mathcal{L}(\mathbf{R})$. However, since $R \subseteq \mathcal{L}(S/\mathbf{G})$, we know that $\sigma \in S(P(t))$; otherwise $tw\sigma \notin \mathcal{L}(S/\mathbf{G})$. This contradicts the fact that $\sigma \notin \gamma = S(P(s)) = S(P(t))$. Therefore, we know that $\forall y \in Q^Y_T, \forall \gamma \in C_T(y) : \Gamma_R(y) \subseteq \gamma$, i.e., T is **R**-compatible. ■

Based on Lemmas 1 and 2, Algorithm 2 is proposed to solve MPLCP-NBIS. First, we need to construct the **R**-compatible NB-AES. If the **R**-compatible NB-AES does not exist, then by Lemma 2, we know that MPLCP-NBIS does not have a solution, i.e., there does not exist a nonblocking and φ-enforcing supervisor whose closed-loop language contains R. When the **R**-compatible NB-AES $\mathcal{AES}^R_\varphi(\mathbf{G})$ exists, since $\mathcal{AES}^R_\varphi(\mathbf{G})$ is a complete and nonblocking BTS, by Theorem 1, we can effectively synthesize a maximally permissive nonblocking supervisor included in it. Moreover, since $\mathcal{AES}^R_\varphi(\mathbf{G})$ is **R**-compatible, by Lemma 1, we know that any supervisor synthesized based on $\mathcal{AES}^R_\varphi(\mathbf{G})$ always achieves the lower bound language R. Therefore, Algorithm 2 effectively solves MPLCP-NBIS in the sense that it is

Algorithm 2:

input: $\mathbf{G}$, $\mathbf{R}$ and φ
output: S
Construct $\mathcal{AES}^R_\varphi(\mathbf{G})$ by Algorithm 1;
if $\mathbf{R}$-compatible NB-AES does not exist**then**
  return MPRCP-NBIS has no solution ;
else
  Use Algorithm NB-SOLU in [22] to synthesize a maximally-permissive nonblocking supervisor $S^* \in \mathcal{AES}^R_\varphi(\mathbf{G})$ included in the $\mathbf{R}$-compatible NB-AES;
  return S^* as the solution to MPRCP-NBIS;

both sound and complete. Moreover, the existence of the $\mathbf{R}$-compatible NB-AES provides the necessary and sufficient condition for the solvability of the lower bound containment problem. The correctness of Algorithm 2 is formally stated in the following theorem.

Theorem 3: Algorithm 2 effectively solves MPLCP-NBIS.

Proof: First, Algorithm 2 is sound, i.e., the nonblocking and φ-enforcing supervisor S^* returned by Algorithm NB-SOLU based on $\mathcal{AES}^R_\varphi(\mathbf{G})$ is indeed a solution to MPRCP-NBIS. Note that Algorithm NB-SOLU already guarantees that S^* is nonblocking and φ-enforcing. Since $\mathcal{AES}^R_\varphi(\mathbf{G})$ is $\mathbf{R}$-compatible and $S^* \in \mathbb{S}(\mathcal{AES}^R_\varphi(\mathbf{G}))$, by Lemma 1, we know that $R \subseteq \mathcal{L}(S^*/\mathbf{G})$. It remains to show that S^* is also maximally permissive. To this end, we assume, for the sake of contradiction, that there exists another nonblocking and φ-enforcing supervisor S' such that

1) $R \subseteq \mathcal{L}(S'/\mathbf{G})$; and
2) $\mathcal{L}(S^*/\mathbf{G}) \subseteq \mathcal{L}(S'/\mathbf{G})$.

By Lemma 2, (i) implies that $S' \in \mathbb{S}(\mathcal{AES}^R_\varphi(\mathbf{G}))$. However, this together with (ii) implies that S^* is not a maximally permissive supervisor included in $\mathcal{AES}^R_\varphi(\mathbf{G})$, which violates the property of Algorithm NB-SOLU.

Next, we show that Algorithm 2 is complete. Suppose that MPRCP-NBIS has a solution. Then, by Lemma 2, we know that there exists a nonblocking and $\mathbf{R}$-compatible BTS T satisfying φ. Therefore, the $\mathbf{R}$-compatible NB-AES exists and Algorithm 2 will not return "no solution" when a solution exists ■

Remark 2: As we claimed earlier, the assumption that $\mathbf{R} \sqsubset \mathbf{G}$ is crucial to guarantee the correctness of Algorithm 2. In particular, this assumption is used in the proof of Lemma 2 to show the existence of the $\mathbf{R}$-compatible NB-AES when a solution to MPRCP-NBIS exists. Therefore, we have to preprocess automata $\mathbf{G}$ and $\mathbf{R}$ to fulfill this assumption before we use Algorithm 2. Otherwise, the synthesis algorithm is only sound but may not be complete, i.e., it may return "no solution" even when a solution exists.

Let us illustrate Algorithm 2 by the following example.

Example 4: Let us still consider system $\mathbf{G}$ shown in Fig. 1(a) and the lower bound language shown in Fig. 1(b), where we have $\mathbf{R} \sqsubset \mathbf{G}$. Its $\mathbf{R}$-compatible NB-AES is shown in Fig. 2(b). By applying Algorithm NB-SOLU with $\mathcal{AES}^R_\varphi(\mathbf{G})$ as its input, we obtain the following supervisor:

$$S^*(s) = \begin{cases} \emptyset & \text{if } s = \epsilon \\ \{c_2\} & \text{if } s = o^{2k+1}, k \geq 0 \\ \{c_1\} & \text{if } s = o^{2k}, k \geq 1 \end{cases} \tag{11}$$

which results in closed-loop language $\mathcal{L}(S^*/\mathbf{G})$ shown in Fig. 3(a). Intuitively, S^* enables c_1 when it visits Y-state $\{8, 9\}$ for $2k+1$ times and it enables c_2 when it visits Y-state $\{8, 9\}$ for $2k$ times, $k \in \mathbb{N}$. Clearly, we see that the lower bound language R is contained in $\mathcal{L}(S^*/\mathbf{G})$. Moreover, by Theorem 3, S^* is indeed a maximally permissive, nonblocking, and φ-enforcing supervisor that achieves R. Recall that supervisor S defined in (7) is also a maximally permissive supervisor, which results in closed-loop language shown in Fig. 3(b). These two supervisors are incomparable. However, S does not contain the lower bound language R, while the synthesized supervisor S^* includes R.

Remark 3: We conclude this section by discussing the complexity of the proposed synthesis algorithm. To construct the $\mathbf{R}$-compatible NB-AES, Algorithm 1 requires quadratic complexity in the size of the NB-AES, which contains at most $2^{|X|} + 2^{|X|+|\Sigma_c|}$ states and $(1 + |\Sigma_o|)2^{|X|+|\Sigma_c|}$ transitions. Algorithm NB-SOLU in [22] is polynomial in the size of the input BTS. However, the input BTS here, $\mathcal{AES}^R_\varphi(\mathbf{G})$, also contains at most $2^{|X|} + 2^{|X|+|\Sigma_c|}$ states and $(1 + |\Sigma_o|)2^{|X|+|\Sigma_c|}$ transitions. Hence, its size is exponential in the size of the original system $\mathbf{G}$, and the overall complexity of Algorithm 2 is exponential in the size of $\mathbf{G}$. This exponential complexity is, in general, unavoidable to synthesize a supervisor under the partial-observation setting [21].

TABLE I
STANDARD SUPERVISORY CONTROL UNDER PARTIAL OBSERVATION

Specification \ Problem	Arbitrary	Max	NB	Max + NB
Safety	[8], [12]	[2]	[27]	[23]
Range	[17]	[25]	This Paper	This Paper

V. DISCUSSION AND COMPARISONS

In this section, we discuss and compare our result with existing results in the literature.

A. Standard Supervisory Control Under Partial Observation

In the standard supervisory control of partially observed DESs, the following problems have been investigated in the literature [8], [12]. Let R and K be two prefix-closed languages such that $R \subseteq K \subseteq \mathcal{L}(\mathbf{G})$. Find a supervisor S that satisfies

1) the *safety* requirement: $\mathcal{L}(S/\mathbf{G}) \subseteq K$; or
2) the *range* requirement: $R \subseteq \mathcal{L}(S/\mathbf{G}) \subseteq K$.

Note that the *safety* requirement is a special case of the IS-based property in this paper, since we can assume w.l.o.g. that language K is realizes by a strict subautomaton of $\mathbf{G}$, and therefore, transforming the safety requirement to an illegal state avoidance requirement discussed in Remark 1.

For both the safety and the range requirements, one can further require that the supervisor synthesized is maximally permissive or nonblocking or both; existing results on all their combinations are listed in Table I. For example, to synthesize an arbitrary safe supervisor, it suffices to compute language $\Sigma^*_{uc} \cap \mathcal{L}(\mathbf{G})$, i.e., we disable all events, and test whether or not this language is in K. To synthesize an arbitrary supervisor satisfying the range requirement (which could be blocking), it suffices to compute language $R^{\downarrow CO}$, the infimal prefix-closed controllable of observable superlanguage, and test whether or not this language is in K. As we can see in Table I, how to synthesize a (maximally permissive) safe and nonblocking supervisor that contains a given lower bound was an open problem. Now, this problem can be solved by the new result in this paper.

B. Comparison With Other Methods

There are many other approaches in the literature for synthesizing safe and nonblocking supervisors under the partial-observation setting (without the lower bound constraint), e.g., the supermal normal

solution [3], [7], [11], [14], the supremal weak normal solution [19], the supremal relatively observable solution [1], [4], [5], and the solutions in [9], [20]. For the nonprefix-closed case, none of the above-mentioned approaches result, in general, a maximally permissive supervisor and it may return "no solution" when a solution does exist.

In fact, for any solution synthesized by any of the above-mentioned methods, we can set it as the lower bound language R in MPRCP-NBIS. Therefore, solving MPRCP-NBIS provides a maximally permissive and nonblocking solution that strictly contains the original solution. For example, we can synthesize a nonblocking supervisor that strictly contains the supremal controllable and normal sublanguage [7] or the supremal controllable and relatively observable sublanguage [4] by setting these languages as the lower bound languages. In other words, the results in this paper can improve existing methods in the literature in general (unless they are already maximally permissive, which is not always guaranteed).

C. Comparison With the Technique in [25]

In [25], we investigated how to synthesize a maximally permissive safe supervisor that contains a given lower bound. However, the approach in [25] can only handle safety rather the general class of IS-based properties. Moreover, nonblockingness is not guaranteed. In fact, the approach [25] takes advantage from the restrictive setting. Specifically, it uses the fact that there exists a unique infimal safe (but potentially blocking) supervisor that contains R (if one exists). Therefore, we can compare NB-AES with the BTS that realizes the unique infimal supervisor via a formal relationship called the *control simulation relation* (CSR). However, this nice property no longer holds when nonblockingness or another IS-based property than safety is considered.

In this paper, we do not try to compare the NB-AES with the infimal supervisor (which does not even exist in general). We "compare" the NB-AES structure directly with the lower bound automaton $\mathbf{R}$ via the new notion of the $\mathbf{R}$-compatibility. Essentially, the $\mathbf{R}$-compatibility captures the computational essence of the CSR, i.e., the iterative computation of the $\mathbf{R}$-compatible NB-AES in Algorithm 1 essentially simulates the fixed-point computation in [25] for the maximal CSR. However, our new approach gets rid of the issue of nonexistence of the infimal supervisor. Moreover, it is conceptually simpler than the method in [25]: It only requires a problem reduction and an existing algorithm in [22] can be leveraged thereafter.

VI. Conclusion

We presented new results on supervisor synthesis for partially observed DESs. A new supervisor synthesis algorithm is proposed to find a maximally permissive nonblocking supervisor that satisfies an arbitrary IS-based property. Moreover, the supervisor synthesized provably contains a given lower bound language. To this end, a new notion called $\mathbf{R}$-compatibility was proposed in order to effectively reduce this problem to a problem previously solved in the literature. Our results further generalize the uniform framework of supervisory control in [23]–[25] by allowing the simultaneous enforcement of the lower bound containment, nonblockingness, and IS-based property.

References

[1] M. V. S. Alves, L. K. Carvalho, and J. C. Basilio, "New algorithms for verification of relative observability and computation of supremal relatively observable sublanguage," *IEEE Trans. Automat. Control*, vol. 62, no. 11, pp. 5902–5908, Nov. 2017.

[2] 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 Dyn. Syst., Theory Appl.*, vol. 6, no. 4, pp. 379–427, 1996.

[3] R. Brandt, V. Garg, R. Kumar, F. Lin, S. I. Marcus, and W. M. Wonham, "Formulas for calculating supremal controllable and normal sublanguages," *Syst. Contr. Lett.*, vol. 15, no. 2, pp. 111–117, 1990.

[4] K. Cai, R. Zhang, and W. M. Wonham, "Relative observability of discrete-event systems and its supremal sublanguages," *IEEE Trans. Automat. Control*, vol. 60, no. 3, pp. 659–670, Mar. 2015.

[5] K. Cai, R. Zhang, and W. M. Wonham, "Characterizations and effective computation of supremal relatively observable sublanguages," *Discrete Event Dyn. Syst., Theory Appl.*, 2017, doi: 10.1007/s10626-017-0250-0.

[6] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*, 2nd ed. Berlin, Germany: Springer, 2008.

[7] 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.

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

[9] M. Heymann and F. Lin, "On-line control of partially observed discrete event systems," *Discrete Event Dyn. Syst., Theory Appl.*, vol. 4, no. 3, pp. 221–236, 1994.

[10] R. Jacob, J.-J. Lesage, and J.-M. Faure, "Overview of discrete event systems opacity: Models, validation, and quantification," *Annu. Rev. Control*, vol. 41, pp. 135–146, 2016.

[11] J. Komenda, T. Masopust, and J. H. Van Schuppen, "Synthesis of controllable and normal sublanguages for discrete-event systems using a coordinator," *Syst. Control Lett.*, vol. 60, no. 7, pp. 492–502, 2011.

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

[13] F. Lin and W. M. Wonham, "Decentralized control and coordination of discrete-event systems with partial observation," *IEEE Trans. Automat. Control*, vol. 35, no. 12, pp. 1330–1337, Dec. 1990.

[14] T. Moor, C. Baier, T.-S. Yoo, F. Lin, and S. Lafortune, "On the computation of supremal sublanguages relevant to supervisory control," in *Proc. 11th IFAC WODES*, 2012, pp. 175–180.

[15] T. Moor and K. W. Schmidt, "Fault-tolerant control of discrete-event systems with lower-bound specifications," in *Proc. 5th Int. Workshop DCDS*, 2015, pp. 161–166.

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

[17] K. Rudie and W. M. Wonham, "The infimal prefix-closed and observable superlanguange of a given language," *Syst. Control Lett.*, vol. 15, no. 5, pp. 361–371, 1990.

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

[19] S. Takai and T. Ushio, "A modified normality condition for decentralized supervisory control of discrete event systems," *Automatica*, vol. 38, no. 1, pp. 185–189, 2002.

[20] S. Takai and T. Ushio, "Effective computation of an $L_m(G)$-closed, controllable, and observable sublanguage arising in supervisory control," *Syst. Control Lett.*, vol. 49, no. 3, pp. 191–200, 2003.

[21] J. N. Tsitsiklis, "On the control of discrete-event dynamical systems," *Math. Control, Signals Syst.*, vol. 2, no. 2, pp. 95–107, 1989.

[22] X. Yin, "Property Enforcement for Partially-Observed Discrete-Event Systems," Ph.D. thesis, Dept. EECS, Univ. Michigan, 2017. [Online]. Available: http://www-personal.umich.edu/ xiangyin/thesis.pdf

[23] X. Yin and S. Lafortune, "Synthesis of maximally permissive supervisors for partially observed discrete event systems," *IEEE Trans. Automat. Control*, vol. 61, no. 5, pp. 1239–1254, May 2016.

[24] X. Yin and S. Lafortune, "A uniform approach for synthesizing property-enforcing supervisors for partially-observed discrete-event systems," *IEEE Trans. Automat. Control*, vol. 61, no. 8, pp. 2140–2154, Aug. 2016.

[25] X. Yin and S. Lafortune, "Synthesis of maximally-permissive supervisors for the range control problem," *IEEE Trans. Automat. Control*, vol. 62, no. 8, pp. 3914–3929, Aug. 2017.

[26] X. Yin, M. Morrison, S.-Y. Sheng, and S. Lafortune, "DPO-SYNT: Discrete control synthesis for partially-observed systems," in *Proc. 20th IFAC World Congr.*, 2017, pp. 6026–6029.

[27] T.-S. Yoo and S. Lafortune, "Solvability of centralized supervisory control under partial observation," *Discrete Event Dyn. Syst., Theory Appl.*, vol. 16, no. 4, pp. 527–553, 2006.

[28] J. Zaytoon and S. Lafortune, "Overview of fault diagnosis methods for discrete event systems," *Annu. Rev. Control*, vol. 37, no. 2, pp. 308–320, 2013.