# On Maximal Permissiveness in Partially-Observed Discrete Event Systems: Verification and Synthesis

Xiang Yin and Stéphane Lafortune

***Abstract*— The notion of maximal permissiveness plays an important role in synthesis problems in the supervisory control framework. It is well known that the supervisor synthesis problem has a unique supremal solution when all the events are observable. However, under the partial observation setting, no supremal solution exists in general and there may exist several locally maximal solutions. In this paper, we tackle the supervisory control problem under partial observation from a new angle. First, we propose an approach to verify whether a given supervisor is maximal or not. If a supervisor is not maximal, then we provide an algorithm that synthesizes a new supervisor that is strictly more permissive than the given one. To the best of our knowledge, both the verification of maximality and the synthesis of a larger solution were previously open problems; our algorithms are the first ones of their kind.**

## I. Introduction

We consider the supervisor synthesis problem for partially-observed Discrete Event Systems (DES) in the supervisory control framework initialed in [9]. Given a specification language and a plant model, the supervisor *existence* problem asks whether or not the spcecification can be exactly achieved by a supervisor. When a language cannot be exactly achieved, then the *synthesis* problem asks to find a supervisor that achieves a sublanguage of the specification language. It is well known that controllability and observability provide the necessary and sufficient conditions for the existence of a supervisor under the partial observation setting [6], [8]. When all the events are observable, the synthesis problem has a unique supremal solution, namely the supremal controllable sublanguage. However, since observability is not preserved under union in general, the synthesis problem is much more challenging under the partial observation setting. In particular, instead of a unique supremal solution, several incomparable *locally maximal* solutions may exist.

Several approaches have been proposed in the literature in order to tackle the synthesis problem; see, e.g., [1]–[3], [5], [7], [12], [13], [15]–[17]. One approach is to compute the supremal controllable and *normal* solution, which was initially proposed in [6], [8]; see, also [2], [5] for its computation. When all controllable events are observable, normality and controllability coincide with observability and controllability, which implies that the synthesis problem has a supremal solution for this special case. However, the supremal normal solution may be conservative in general, when there are controllable events that are unobservable. In [1], an online approach was proposed in order to compute a maximal solution. In our recent work [14], [15], we showed that the problem of synthesizing a maximally permissive safe and non-blocking supervisor for partially-observed DES is also decidable.

Although maximal solutions have been reported in the literature in [1], [15], the maximal solutions obtained so far are just a particular type of maximal solutions, namely, *greedy maximal* solutions. In a greedy maximal solution, the supervisor tries to enable *as many events as possible* at each control decision instant. Since no supervisor can enable more events at any instant, this greedy strategy yields a locally maximal solution. However, greedy maximal solutions may not be good solutions for the following reasons. First, enabling fewer events at some particular instant may allow to enable more events at later instants, thereby yielding more behaviors in the future post-language. Second, the greedy maximal solution may not contain a given behavior.

In order to resolve the limitations of greedy maximal solutions, we tackle the synthesis problem from a new angle. Specifically, we consider the following problems:

(1) Verify whether a given supervisor is maximal or not.
(2) Synthesize a new supervisor that is strictly more permissive than a given non-maximal supervisor.

We refer to the first problem as the verification problem and to the second problem as the synthesis problem. Clearly, the verification problem is the first step towards the synthesis problems: if the given supervisor is verified to be maximal, then there is no need to solve the synthesis problems. The synthesis problems considered here can also be viewed as synthesis problems with a strict *lower bound* specification, since the given supervisor could model the lower bound behavior that a newly synthesized supervisor must achieve. To the best of our knowledge, the maximality verification problem is still open in the literature. Moreover, the synthesis problem formulated above is also open.

The contributions of this paper are as follows. First, we provide complete solutions to problems (1) and (2) formulated above in the case of prefix-closed specifications. Specifically, we provide an approach to verify whether a given supervisor is maximal or not. Moreover, when the answer to the verification problem is negative, an algorithm is proposed in order to synthesize a new supervisor that strictly contains the behavior generated by the given supervisor. The proposed approaches rely on the AIC investigated in our previous work. However, the new notions of *control*

This work was partially supported by the US National Science Foundation grants CCF-1138860 (Expeditions in Computing project ExCAPE: Expeditions in Computer Augmented Program Engineering) and CNS-1446298.

Xiang Yin and Stéphane Lafortune are with the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109, USA. {xiangyin,stephane}@umich.edu.

*simulation relation* and *control replacement* are proposed in order to solve the new problems.

Due to space constraints, all proofs have been omitted.

## II. Preliminaries

Let $\Sigma$ be a finite set of events. We denote by $\Sigma^*$ the set of all finite strings over $\Sigma$, including the empty string $\epsilon$. A language $L$ is a subset of $\Sigma^*$. We denote by $\overline{L}$ the prefix-closure of language $L \subseteq \Sigma^*$. A DES is modeled as a finite-state automaton $\mathbf{G} = (X, \Sigma, \delta, x_0, X_m)$, where $X$ is the finite set of states, $\Sigma$ 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. The transition function $\delta$ is extended to $X \times \Sigma^*$ in the usual manner and the extended function is still denoted by $\delta$; see, e.g., [4]. For brevity, we write $\delta(x, s)$ as $\delta(s)$ if $x = x_0$. We denote by $\mathcal{L}(\mathbf{G}, x)$ the language generated by $\mathbf{G}$ from state $x$, i.e., $\mathcal{L}(\mathbf{G}, x) := \{s \in \Sigma^* : \delta(x, s)!\}$, where ! means "is defined". We write $\mathcal{L}(\mathbf{G}, x)$ as $\mathcal{L}(\mathbf{G})$ if $x = x_0$; it is the language generated by $\mathbf{G}$. In this paper, we will only deal with prefix-closed languages. Therefore, we assume that $X_m = X$ and denote an automaton by $\mathbf{G} = (X, \Sigma, \delta, x_0)$.

Given two automata $\mathbf{A} = (X_A, \Sigma, \delta_A, x_{A,0})$ and $\mathbf{B} = (X_B, \Sigma, \delta_B, x_{B,0})$, we say that $\mathbf{A}$ is a sub-automaton of $\mathbf{B}$, denoted by $\mathbf{A} \sqsubseteq \mathbf{B}$, if $\delta_A(x_{A,0}, s) = \delta_B(x_{B,0}, s)$ for all $s \in \mathcal{L}(\mathbf{A})$. We say that $\mathbf{A}$ is a *strict* sub-automaton of $\mathbf{B}$, denoted by $\mathbf{A} \sqsubset \mathbf{B}$, if (i) $\mathbf{A} \sqsubseteq \mathbf{B}$; and (ii) $\forall x, y \in X_A, \forall s \in \Sigma^* : \delta_B(x, s) = y \Rightarrow \delta_A(x, s) = y$. For any two automata $\mathbf{A}$ and $\mathbf{B}$ such that $\mathcal{L}(\mathbf{A}) \subseteq \mathcal{L}(\mathbf{B})$, we can always refine the state spaces of $\mathbf{A}$ and $\mathbf{B}$ such that $\mathbf{A} \sqsubset \mathbf{B}$; see, e.g., [5].

In the framework of supervisory control [9], we assume that $\Sigma = \Sigma_c \dot{\cup} \Sigma_{uc}$, where $\Sigma_c$ and $\Sigma_{uc}$ are the set of controllable events and the set of uncontrollable events, respectively. A control decision $\gamma \in 2^{\Sigma}$ is a set of events that a supervisor may enable, with the constraint that $\Sigma_{uc} \subseteq \gamma$. We denote by $\Gamma$ the set of all control decisions, i.e., $\Gamma := \{\gamma \in 2^{\Sigma} : \Sigma_{uc} \subseteq \gamma\}$. Due to limited sensing capabilities, a supervisor may not be able to observe the occurrence of every event. Under the partial observation setting [6], [8], we further assume that $\Sigma = \Sigma_o \dot{\cup} \Sigma_{uo}$, where $\Sigma_o$ and $\Sigma_{uo}$ are the set of observable events and the set of unobservable events, respectively. The natural projection $P : \Sigma^* \to \Sigma_o^*$ is defined in the usual manner; see, e.g., [4]. The projection $P$ is extended to $2^{\Sigma^*}$ by $P(L) = \{t : \exists s \in L \text{ s.t. } t = P(s)\}$ and $P^{-1}$ denotes the inverse projection. A partial-observation supervisor is a function $S : P(\mathcal{L}(\mathbf{G})) \to \Gamma$. We use $S/\mathbf{G}$ to represent the closed-loop system under control. The language generated by $S/\mathbf{G}$ is denoted by $\mathcal{L}(S/\mathbf{G})$; see, e.g., [4].

Let $S \subseteq X$ be a subset of states, $\gamma \in \Gamma$ be a control decision, and $\sigma \in \Sigma_o$ be an observable event. We define the following two operators:
$\text{UR}_\gamma(S) = \{x \in X : \exists u \in S, \exists s \in (\Sigma_{uo} \cap \gamma)^* \text{ s.t. } x = \delta(u, s)\}$;
$\text{Next}_\sigma(S) = \{x \in X : \exists u \in S \text{ s.t. } x = \delta(u, \sigma)\}$.

## III. Problem Formulation

In the supervisor synthesis problem, one is interested in designing a supervisor such that the closed-loop behavior satisfies some specification. In this paper, we only consider prefix-closed languages, i.e., we only consider safety specifications; non-blockingness is not considered. Hence, the specification is given as a prefix-closed sublanguage $K = \overline{K} \subseteq \mathcal{L}(\mathbf{G})$. In words, all strings in $K$ are legal and all strings in $\mathcal{L}(\mathbf{G}) \setminus K$ are illegal. Let $\mathbf{K} = (X_K, \Sigma, \delta_K, x_{0,K})$ be an automaton generating $K$, i.e., $K = \mathcal{L}(\mathbf{K})$. We assume w.l.o.g. that $\mathbf{K} \sqsubset \mathbf{G}$. This assumption essentially says that all states in $X_K$ are legal and all states in $X \setminus X_K$ are illegal.

It is well known that there exists a supervisor $S$ such that $\mathcal{L}(S/\mathbf{G}) = \overline{K}$ iff $K$ is controllable and observable; the reader is referred to [4] for definitions of these properties. When $K$ cannot be exactly achieved, the synthesis problem asks to find a *sublanguage* of $K$ that is "as large as possible". Since observability is not preserved under union in general, there does not exist a supremal controllable and observable sublanguage. This is the fundamental difficulty in the synthesis problem under partial observation. Therefore, instead of computing a supremal controllable and observable sublanguage, one is interested in computing a *locally maximal* controllable and observable sublanguage of $K$. Formally, we say that a supervisor $S$ is *safe* if $\mathcal{L}(S/\mathbf{G}) \subseteq K$ and we say that $S$ is *maximal* if
(i) $S$ is safe; and
(ii) For any safe $S'$, we have $\mathcal{L}(S/\mathbf{G}) \not\subset \mathcal{L}(S'/\mathbf{G})$.

Previous efforts in the literature have only addressed a particular type of maximal solution, namely, a greedy maximal solution. However, as was discussed in the introduction, greedy maximal solutions have drawbacks in general. In this paper, instead of synthesizing a greedy maximal solution, we tackle the synthesis problem from a different angle:
*"Suppose that there exists a supervisor that achieves a desired lower bound behavior. We wish to determine whether or not we can improve this supervisor by synthesizing a new supervisor whose closed-loop behavior is strictly larger than the given one."*
Let us denote by $R \subseteq K \subseteq \mathcal{L}(\mathbf{G})$ the lower bound specification that the supervisor must achieve. Let $\mathbf{R} = (X_R, \Sigma, \delta_R, x_{0,R})$ be an automaton generating $R$, i.e., $\mathcal{L}(\mathbf{R}) = R$. We denote by $S_R$ the supervisor that achieves $R$, i.e., $\mathcal{L}(S_R/\mathbf{G}) = R$. We are interested in whether or not we can improve this lower bound supervisor, namely, find a new supervisor $S'$ such that $R \subset \mathcal{L}(S'/\mathbf{G})$. Note that the existence of $S_R$ achieving $R$ implicitly assumes that $R$ is controllable and observable. This assumption is w.l.o.g. since we can always compute the infimal prefix-closed superlanguage of $R$ if $R$ is not controllable or observable; see, e.g., [10].

In order to solve this problem, we first need to *verify* whether the given supervisor $S_R$ is already maximal or not. If so, then we cannot improve it any further. Therefore, we define the maximality verification problem as follows.

*Problem 1:* Given a supervisor $S_R$ such that $\mathcal{L}(S_R/\mathbf{G}) \subseteq K$, verify whether or not $S_R$ is maximal.

If the answer to the verification problem is negative, then we consider the following synthesis problem.

*Problem 2:* Given a non-maximal supervisor $S_R$ such

that $\mathcal{L}(S_R/\mathbf{G}) \subseteq K$, find a safe supervisor $S$ such that $\mathcal{L}(S_R/\mathbf{G}) \subset \mathcal{L}(S/\mathbf{G})$.

## IV. All Inclusive Controller

In this section, we first review the notion of bipartite transition system (BTS) and the notion of all inclusive controller (AIC) from our earlier work [15]. Then we discuss how to realize a supervisor by using a BTS.

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. First, we recall the definition of BTS from [15].

*Definition 1:* (Bipartite Transition System). A bipartite transition system $T$ w.r.t. $\mathbf{G}$ is a 7-tuple

$$T = (Q_Y^T, Q_Z^T, h_{YZ}^T, h_{ZY}^T, \Sigma_o, \Gamma, y_0) \tag{1}$$

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

$$h_{YZ}^T(y, \gamma) = z \Rightarrow [I(z) = \text{UR}_\gamma(y)] \wedge [\Gamma(z) = \gamma]; \tag{2}$$

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

$$h_{ZY}^T(z, e) = y \Rightarrow e \in \Gamma(z) \wedge y = \text{Next}_e(I(z)); \tag{3}$$

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

Unless otherwise specified, all BTSs in this paper are defined w.r.t. the system model $\mathbf{G}$. Intuitively, a BTS is a game structure between the system (control decision) and the environment (event occurrence). Specifically, each $Y$-state is a system state from which control decisions are made. Similarly, each $Z$-state is an environment state from which observable events occur. The structure is bipartite due to the alternating nature between control and observation. We denote by $C_T(y)$ the set of control decisions defined at $y \in Q_Y^T$ in $T$, i.e., $C_T(y) = \{\gamma \in \Gamma : h_{YZ}^T(y, \gamma)!\}$. For simplicity, hereafter, we also write $y \xrightarrow{\gamma}_T z$ if $z = h_{YZ}^T(y, \gamma)$ and $z \xrightarrow{\sigma}_T y$ if $z = h_{ZY}^T(z, \sigma)$. For two BTSs $T_1$ and $T_2$, we have that $h_{YZ}^{T_1}(y, \gamma) = h_{YZ}^{T_2}(y, \gamma)$ whenever they are defined. Therefore, we will drop the superscript in $h_{YZ}^T(y, \gamma)$ and write it as $h_{YZ}(y, \gamma)$ and $y \xrightarrow{\gamma} z$ if it is defined for some $T$; the same holds for $h_{ZY}$ and $z \xrightarrow{\sigma} y$. We call a sequence $\gamma_0\sigma_1\gamma_1\sigma_2 \dots \sigma_n\gamma_n$, where $\gamma_i \in \Gamma, \sigma_i \in \Sigma_o$, a *run*. Then a BTS essentially generates a set of runs and we denote by $\mathcal{R}(T, y)$ the set of runs generated by $T$ from state $y \in Q_Y^T$, i.e., $\mathcal{R}(T, y) = \{\gamma_0\sigma_1 \dots \sigma_n\gamma_n : y \xrightarrow{\gamma_0}_T y_1 \xrightarrow{\sigma_1}_T z_1 \dots \xrightarrow{\sigma_n}_T z_n \xrightarrow{\gamma_n}_T y_{n+1}\}$.

Definition 1 provides the general definition of a BTS. However, for the purpose of control, we also want a BTS to be *complete*. Formally, we say a BTS $T$ is complete if:

1. $\forall y \in Q_Y^T : [C_T(y) \neq \emptyset]$; and
2. $\forall z \in Q_Z^T, \forall e \in \Gamma(z) \cap \Sigma_o : [(\exists x \in I(z) : \delta(x, e)!) \Rightarrow h_{ZY}^T(z, e)!]$

A supervisor $S$ for $\mathbf{G}$ works as follows. Initially, it makes control decision $S(\epsilon)$. Then new control decision $S(\sigma)$ is made upon the occurrence of (enabled) observable event $\sigma$, and so forth. Let $s = \sigma_1 \dots \sigma_n \in P(\mathcal{L}(S/\mathbf{G}))$ be an observed string. Then the execution of $s$ induces the run

$$\rho_S(s) := \gamma_0\sigma_1\gamma_1 \dots \gamma_{n-1}\sigma_n\gamma_n \tag{4}$$

where $\gamma_0 = S(\epsilon)$ and $\gamma_i = S(\sigma_1 \dots \sigma_i), i \leq n$. The run $\rho_S(s)$ again yields an alternating sequence of $Y$-and $Z$-states

$$y_0 \xrightarrow{\gamma_0} z_0 \xrightarrow{\sigma_1} y_1 \xrightarrow{\gamma_1} \dots \xrightarrow{\gamma_{n-1}} z_{n-1} \xrightarrow{\sigma_n} y_n \xrightarrow{\gamma_n} z_n \tag{5}$$

We denote by $IS_S^Y(s)$ and $IS_S^Z(s)$, the last $Y$-state and $Z$-state in $y_0z_0y_1z_2 \dots z_{n-1}y_nz_n$, respectively, i.e., $IS_S^Y(s) = y_n$ and $IS_S^Z(s) = z_n$. That is, $IS_S^Y(s)$ and $IS_S^Z(s)$ are the $Y$-state and the $Z$-state that result from the occurrence of string $s$ under supervisor $S$, respectively.

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

*Definition 2:* Let $T$ be a complete BTS. A supervisor $S$ is said to be included in $T$ if for any observed string $s \in P(\mathcal{L}(S/\mathbf{G}))$, we have that $S(s) \in C_T(IS_S^Y(s))$. We denote by $\mathbb{S}(T)$ the set of supervisors included in $T$.

In [15], the AIC structure is defined as the largest BTS including only safe supervisors.

*Definition 3:* (All Inclusive Controller). The All Inclusive Controller for $\mathbf{G}$, $\mathcal{AIC}(\mathbf{G}) = (Q_Y^{AICG}, Q_Z^{AICG}, h_{YZ}^{AICG}, h_{ZY}^{AICG}, \Sigma_o, \Gamma, y_0)$, is defined as the largest complete BTS such that $\forall z \in Q_Z^{AICG} : I(z) \subseteq X_K$.

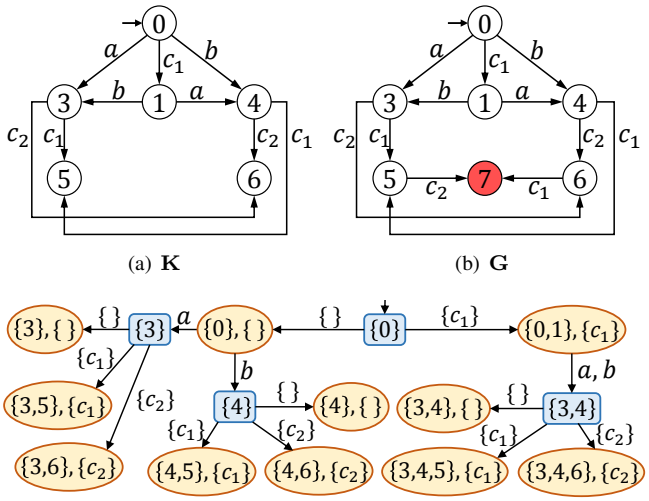
The reader is referred to [15] for more details on the AIC. Here we recall one important property of the AIC from [15].

*Theorem 1:* A supervisor $S$ is safe iff $S \in \mathbb{S}(\mathcal{AIC}(\mathbf{G}))$.

*Example 1:* Consider the system $\mathbf{G}$ and the specification $\mathbf{K}$ in Fig. 1. We have that $\mathbf{K} \sqsubset \mathbf{G}$ and state 7 is the unique illegal state. Let $\Sigma_o = \{a, b\}$ and $\Sigma_c = \{c_1, c_2\}$. Then the AIC for this system is shown in Fig. 1(c), which is also a complete BTS. For the initial $Y$-state $y_0 = \{0\}$, by making control decision $\{c_1\}$, we will reach $Z$-state $z = (\{0, 1\}, \{c_1\})$. From $z$, both observable events $a$ and $b$ can occur. If $a$ occurs, then the next $Y$-state is $y_1 = \{3\}$. At $Y$-state $\{3\}$, we can either choose to enable $c_1$ or $c_2$, but we cannot make control decision $\{c_1, c_2\}$, since it will unobservably lead to illegal state 7. This is why this control decision is not defined at $\{3\}$ in the AIC.

### A. BTS Realization of Supervisor

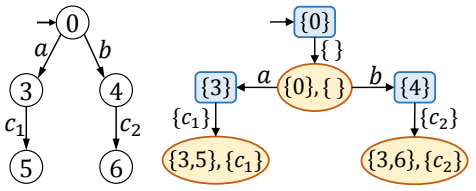
In general, there may be multiple control decisions defined at a $Y$-state in a BTS. We say that a complete BTS $T$ is *deterministic* if $\forall y \in Q_Y^T : |C_T(y)| = 1$. If $T$ is deterministic, then we denote by $c_T(y)$ the unique control decision defined at $y$. A deterministic BTS includes a unique supervisor and we denote by $S_T$ the unique supervisor included in $T$. Then $T$ essentially is a *finite realization* of the supervisor $S_T$.




(a) $\mathbf{K}$ (b) $\mathbf{G}$

(c) The AIC $\mathcal{AIC}(\mathbf{G})$.

Fig. 1. For $\mathbf{G}$, we have $\Sigma_o = \{a, b\}$ and $\Sigma_c = \{c_1, c_2\}$. In the AIC, rectangular states represent $Y$-states and oval states represent $Z$-states. For the sake of simplicity, uncontrollable events, e.g., $a$ and $b$, are omitted in each control decision in the figure. We also omit redundant events in each control decision, namely events that are not feasible within the unobservable reach, e.g., event $c_2$ at $Y$-state $\{0\}$.




(a) $\mathbf{R}$ (b) BTS $T_R$ s.t. $\mathbb{S}(T_R) = \{S_R\}$

Fig. 2. Lower bound supervisor $S_R$ and its BTS realization $T_R$.

We say that a supervisor $S$ is *information-state-based* (IS-based) if $\forall s, t \in P(\mathcal{L}(S/\mathbf{G}))$, we have that $IS_S^Y(s) = IS_S^Y(t) \Rightarrow S(s) = S(t)$. Given the system model $\mathbf{G}$, a supervisor $S$ has a BTS realization, if and only if, $S$ is an IS-based supervisor. Note that not all supervisors are IS-based, since a supervisor may issue different control decisions upon different visits to the same $Y$-state. However, the following result reveals that, by suitable state-space refinement, any finite memory supervisor can be expressed as an IS-based supervisor; therefore, it can be realized by a BTS.

*Proposition 1:* Let $S_R$ be a supervisor and $\mathbf{R}$ be an automaton such that $\mathcal{L}(S_R/\mathbf{G}) = \mathcal{L}(\mathbf{R})$. If $\mathbf{R} \sqsubseteq \mathbf{G}$, then supervisor $S_R$ is an IS-based supervisor.

Based on the above result, hereafter we assume w.l.o.g. that $\mathbf{R} \sqsubseteq \mathbf{G}$. Therefore $S_R$ is an IS-based supervisor and it can be realized by a deterministic BTS, denoted by $T_R$.

*Example 2:* Let us consider the lower bound specification $R$ generated by automaton $\mathbf{R}$ shown in Fig. 2(a). Since $\mathbf{R} \sqsubseteq \mathbf{G}$, we know that supervisor $S_R$ achieving $R$ can be realized by the deterministic BTS $T_R$ shown in Fig. 2(b).

## V. Control Simulation Relation

Let us return to Problem 1, in which $R$ is the lower bound specification, $S_R$ is the corresponding supervisor that achieves $R$, and $T_R$ is the deterministic BTS that realizes (or includes) $S_R$. In order to verify whether or not $S_R$ is maximal, the idea is to "compare" $T_R$ with $\mathcal{AIC}(\mathbf{G})$ and to check whether or not we can find a supervisor included in the AIC that is strictly more permissive than $S_R$. In order to compare $T_R$ with $\mathcal{AIC}(\mathbf{G})$, we need to establish a formal relationship between states in $T_R$ and states in $\mathcal{AIC}(\mathbf{G})$. This is formalized by the notion of control simulation relation (CSR) defined as follows.

*Definition 4:* (Control Simulation Relation). Let $T_1$ and $T_2$ be two BTSs. A relation $\Phi = \Phi_Y \cup \Phi_Z \subseteq (Q_Y^{T_1} \times Q_Y^{T_2}) \cup (Q_Z^{T_1} \times Q_Z^{T_2})$ is said to be a *control simulation relation* from $T_1$ to $T_2$ if the following conditions hold:

1. $(y_0, y_0) \in \Phi$;
2. For every $(y_1, y_2) \in \Phi_Y$ we have that:
   $y_1 \xrightarrow{\gamma_1}_{T_1} z_1$ in $T_1$ implies the existence of $y_2 \xrightarrow{\gamma_2}_{T_2} z_2$ in $T_2$ such that $\gamma_1 \subseteq \gamma_2$ and $(z_1, z_2) \in \Phi_Z$;
3. For every $(z_1, z_2) \in \Phi_Z$ we have that:
   $z_1 \xrightarrow{\sigma}_{T_1} y_1$ in $T_1$ implies the existence of $z_2 \xrightarrow{\sigma}_{T_2} y_2$ in $T_2$ such that $(y_1, y_2) \in \Phi_Y$.

We say that $T_1$ is control-simulated by $T_2$ or that $T_2$ control-simulates $T_1$, denoted by $T_1 \preceq T_2$, if there exists a control simulation relation from $T_1$ to $T_2$.

Intuitively, the control simulation relation captures whether or not $T_2$ is able to match an arbitrary control decision made by $T_1$ by either taking the same control decision or a control decision that is strictly larger than the one made by $T_2$ and maintain this ability for all possible future behaviors.

Given two BTSs $T_1$ and $T_2$, a relevant question is whether or not there exists a CSR from $T_1$ to $T_2$. To answer this question, we define an operator

$$F : 2^{Q_Y^{T_1} \times Q_Y^{T_2}} \cup 2^{Q_Z^{T_1} \times Q_Z^{T_2}} \to 2^{Q_Y^{T_1} \times Q_Y^{T_2}} \cup 2^{Q_Y^{T_1} \times Q_Y^{T_2}} \quad (6)$$

as follows. For any $\Phi = \Phi_Y \cup \Phi_Z \subseteq (Q_Y^{T_1} \times Q_Y^{T_2}) \cup (Q_Z^{T_1} \times Q_Z^{T_2})$, we have

1. $(y_1, y_2) \in F(\Phi_Y)$ if $(y_1, y_2) \in \Phi_Y$ and for any transition $y_1 \xrightarrow{\gamma_1}_{T_1} z_1$ in $T_1$, there exists $y_2 \xrightarrow{\gamma_2}_{T_2} z_2$ in $T_2$ such that $\gamma_1 \subseteq \gamma_2$ and $(z_1, z_2) \in \Phi_Z$.
2. $(z_1, z_2) \in F(\Phi_Z)$ if $(z_1, z_2) \in \Phi_Z$ and for any transition $z_1 \xrightarrow{\sigma}_{T_1} y_1$ in $T_1$, there exists $z_2 \xrightarrow{\sigma}_{T_2} y_2$ in $T_2$ such that $(y_1, y_2) \in \Phi_Y$.

*Proposition 2:* The operation $F$ has following properties

1. $\Phi$ is a control simulation relation from $T_1$ to $T_2$, if and only if, $\Phi \subseteq F(\Phi)$ and $(y_0, y_0) \in \Phi$;
2. $\Phi_1 \subseteq \Phi_2 \Rightarrow F(\Phi_1) \subseteq F(\Phi_2)$.

The above results have the following implications. First, since $\Phi \subseteq F(\Phi)$ for any CSR $\Phi$, we know that the maximal relation $\Phi$ is a fixed-point of operator $F$, i.e., $F(\Phi) = \Phi$. Note that $F(\Phi) \subseteq \Phi$ always holds. By the second property in Proposition 2, we know that $F$ is monotone. Therefore, by Tarski's fixed-point theorem, we know that the supremal fixed-point of $F$, denoted by $\Phi^*(T_1, T_2)$, exists and it can be computed as follows

$$\Phi^*(T_1, T_2) = \lim_{k \to \infty} F^k((Q_Y^{T_1} \times Q_Y^{T_2}) \cup (Q_Z^{T_1} \times Q_Z^{T_2})) \quad (7)$$

In other words, $\Phi^*(T_1, T_2)$ is a maximal control simulation relation from $T_1$ to $T_2$ if $(y_0, y_0) \in \Phi^*(T_1, T_2)$. Otherwise, $T_1 \not\preceq T_2$ if $(y_0, y_0) \notin \Phi^*(T_1, T_2)$. This is similar to the standard simulation relation; see, e.g., [11]. Note that the limit in Equation (7) can be achieved within at most $|Q_Y^{T_1}||Q_Y^{T_2}| + |Q_Z^{T_1}||Q_Z^{T_2}|$ iterations.

## VI. Verification of Maximality

In this section, we show how to verify maximality using the notion of CSR.

First, we introduce the notion of *control replacement*.

*Definition 5:* Let $T$ be a deterministic BTS. Then we say that $\gamma \in \Gamma$ is a control replacement for $T$ at $y \in Q_Y^T$ (or a $(y,T)$-replacement) if $c_T(y) \subset \gamma$ and

$$(\forall c_T(y)\sigma_1\gamma_1 \dots \sigma_n\gamma_n \in \mathcal{R}(T,y)) \quad (8)$$
$$(\exists \gamma\sigma_1\gamma_1' \dots \sigma_n\gamma_n' \in \mathcal{R}(\mathcal{AIC}(\mathbf{G}),y))[\gamma_i \subseteq \gamma_i', \forall i \le n]$$

When $\gamma$ is a $(y,T)$-replacement, this implies that if we replace the original control decision defined at $y$ in $T$, $c_T(y)$, by $\gamma$, then we are still able to (safely) match all behaviors generated by $T$ in the future. It worth noting that not every decision $\gamma \in C_{\mathcal{AIC}(\mathbf{G})}(y)$ such that $c_T(y) \subset \gamma$ is a $(y,T)$-replacement. This is because enabling more events may introduce more uncertainty and the supervisor must be more conservative in the future in order to maintain safety. We see that the definition of control replacement is very similar to that of CSR. In fact, the following theorem reveals that checking whether or not a control decision is a $(y,T)$-replacement can be done by using the CSR.

*Theorem 2:* Let $T$ be a deterministic BTS and $y \in Q_Y^T$ be a $Y$-state. Then $\gamma \in C_{\mathcal{AIC}(\mathbf{G})}(y)$ is a $(y,T)$-replacement, if and only if, $c_T(y) \subset \gamma$ and $(z,z') \in \Phi^*(T, \mathcal{AIC}(\mathbf{G}))$, where $z = h_{YZ}(y, c_T(y))$ and $z' = h_{YZ}(y,\gamma)$.

We are now ready to present the theorem that provides a necessary and sufficient condition for maximality.

*Theorem 3:* Let $S$ be an IS-based supervisor and $T$ be the deterministic BTS s.t. $\mathbb{S}(T) = \{S\}$. Then $S$ is maximal iff

$$(\forall y \in Q_Y^T)(\forall \gamma \in C_{\mathcal{AIC}(\mathbf{G})}(y))[\gamma \text{ is not a } (y,T)\text{-replacement}]$$

*Proof:* (Sketch only)

($\Rightarrow$) By contraposition. Suppose that there exists a $Y$-state $y \in Q_Y^T$ and a control decision $\gamma \in C_{\mathcal{AIC}(\mathbf{G})}(y))$ such that $\gamma$ is a $(y,T)$-replacement. We construct supervisor $S' : P(\mathcal{L}(\mathbf{G})) \to \Gamma$ as follows:

$$S'(t) = \begin{cases} S(t), & \text{if } t \in P(\mathcal{L}(S/\mathbf{G})) \wedge \\ & [IS_S^Y(t) \neq y \vee (\exists s \in \overline{\{t\}} \setminus \{t\} : IS_S^Y(s) = y)] \\ \gamma, & \text{if } t \in P(\mathcal{L}(S/\mathbf{G})) \wedge IS_S^Y(t) = y \\ & \wedge (\forall s \in \overline{\{t\}} \setminus \{t\})[IS_S^Y(s) \neq y] \\ \Sigma_{uc}, & \text{else} \end{cases} \quad (9)$$

For the above constructed $S'$, we can show that $\mathcal{L}(S/\mathbf{G}) \subset \mathcal{L}(S'/\mathbf{G}) \subseteq K$. Therefore, $S$ is not a maximal solution.

($\Leftarrow$) By contraposition. Suppose that $\exists S' \in \mathbb{S}(\mathcal{AIC}(\mathbf{G}))$ such that $\mathcal{L}(S/\mathbf{G}) \subset \mathcal{L}(S'/\mathbf{G})$. Then $\exists t \in P(\mathcal{L}(S/\mathbf{G}))$ such that $S(t) \subset S'(t)$ and $S(t') = S'(t'), \forall t' \in \overline{\{t\}} \setminus \{t\}$. We know that $IS_S^Y(t) = IS_{S'}^Y(t)$; call this $Y$-state $y$. Then we can show that $S'(t)$ is a $(y,T)$-replacement. ■

By Theorems 2 and 3, it is clear that, in order to verify whether or not $S_R$ is maximal, we need to first compute the maximal control simulation relation $\Phi^*(T_R, \mathcal{AIC}(\mathbf{G}))$ from $T_R$ to $\mathcal{AIC}(\mathbf{G})$. Then we need to check, for each $Y$-state $y$ in $T_R$, whether or not there exists a $(y,T)$-replacement by using $\Phi^*(T_R, \mathcal{AIC}(\mathbf{G}))$. If not, then the supervisor $S_R$ is maximal. The above procedure is formally summarized by Algorithm MAX-VER. The overall complexity of Algorithm MAX-VER is $O(2^{2^{|X|+2|\Sigma_c|}})$, since it takes $O(2^{2^{|X|+2|\Sigma_c|}})$ to compute $\Phi^*(T_R, \mathcal{AIC}(\mathbf{G}))$ and $O(2^{|X|} \times 2^{|\Sigma_c|})$ to check all possible control decisions for each $Y$-state.

**Algorithm 1:** MAX-VER($T_R, \mathcal{AIC}(\mathbf{G})$)

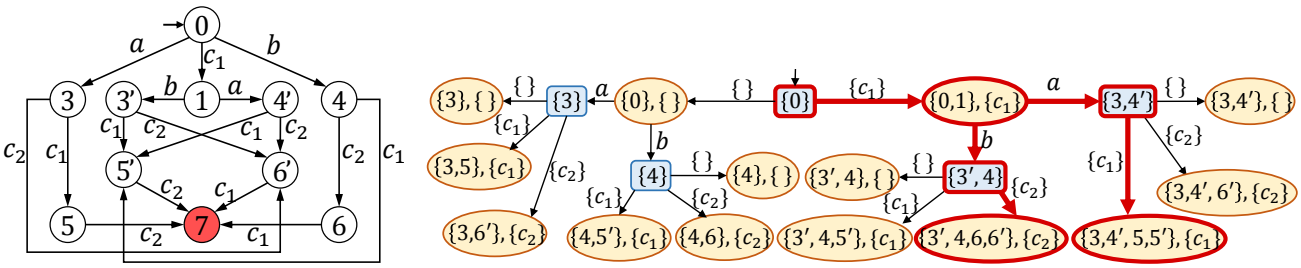
```
Compute Φ := Φ*(T_R, AIC(G));
for y ∈ Q_Y^{T_R} do
    for γ ∈ C_{AIC(G)}(y) : c_{T_R}(y) ⊂ γ do
        if (h_YZ(y, c_{T_R}(y)), h_YZ(y, γ)) ∈ Φ then
            return "No" and y;
return "Yes";
```

*Example 3:* Let us return to the system $\mathbf{G}$ shown in Fig. 1(b) and the lower bound automaton $\mathbf{R}$ shown in Fig. 2(a). The corresponding BTS $T_R$ that realizes $S_R$ and the AIC $\mathcal{AIC}(\mathbf{G})$ have already been shown in Figs. 2(b) and 1(c), respectively. Let us consider the initial $Y$-state $y_0 = \{0\}$ in $T_R$. Clearly, control decision $\{c_1\} \in C_{\mathcal{AIC}(\mathbf{G})}(y_0)$ is a $(y_0, T_R)$-replacement. For example, if we replace the original control decision $c_{T_R}(y_0) = \{\}$ by $\{c_1\}$, then after observing event $a$ $Y$-state $\{3,4\}$ is reached and we can still take control decision $\{c_1\}$ that was originally defined at state $\{3\}$ in order to maintain safety. Similarly, if event $b$ occurs, then $Y$-state $\{3,4\}$ is reached and we can still take control decision $\{c_2\}$ which is defined at $\{4\}$ in $T_R$. Therefore, we know that $S_R$ is not a maximal solution.

## VII. The Role of Strict Sub-automaton

The proof of Theorem 3 is constructive. Specifically, we construct a new supervisor $S'$ such that $\mathcal{L}(S'/\mathbf{G})$ is strictly larger than $\mathcal{L}(S/\mathbf{G})$. However, this is not sufficient for the purpose of synthesis, since $S'$ is defined in terms of languages and it is still not clear how to *realize* this $S'$. One may conjecture that there always exists an IS-based supervisor that is strictly more permissive than a given IS-based supervisor if it is not maximal. However, this is not true. One can check that supervisor $S_R$, which is realized by BTS $T_R$ in Fig. 2(b), is a maximal IS-based supervisor (w.r.t. the state space of $\mathbf{G}$), since we cannot find another BTS $T'$ such that $T_R \preceq T'$. However, we have already shown in Example 3 that $S_R$ is not a maximal supervisor.

The reason why there may not exist an IS-based supervisor that is more permissive than a non-maximal IS-based supervisor is explained as follows. It is possible that two different information states under the original control strategy can be merged as a single information state under the new (more permissive) control strategy. As a consequence, information




(a) $\mathbf{K}'$ and $\mathbf{G}'$ (b) $\mathcal{AIC}(\mathbf{G}')$

Fig. 3. In Figure 3(a), $\mathbf{G}'$ is the entire automaton and $\mathbf{K}'$ is obtained by removing illegal state 7 from $\mathbf{G}'$.

is lost by using the newly reached information state. We call this phenomenon *information merge*. For example, for BTS $T_R$ shown in Fig. 2(b), if we replace the original control decision $\{\}$ defined at the initial state by $\{c_1\}$, then the two different $Y$-states $\{3\}$ and $\{4\}$ in $T_R$, which are reached by observing $a$ and $b$, respectively, are merged as a single state $\{3,4\}$. However, simply knowing state $\{3,4\}$ is not sufficient for making control decisions in order to contain the original behavior. To find an IS-based solution, state $\{3,4\}$ has to be split into two states: one is reached by observing $a$ and the other is reached by observing $b$.

Let $y \in 2^X$ be an information state. We denote by $y|_{\mathbf{R}}$ the restriction of $y$ to the state space of $\mathbf{R}$, i.e., $X_R$. The following result says that the state merging phenomenon described above will not occur when $\mathbf{R} \sqsubset \mathbf{K}$.

*Lemma 1:* Assume that $\mathbf{R} \sqsubset \mathbf{K} \sqsubset \mathbf{G}$. Then for any supervisor $S'$ s.t. $R = \mathcal{L}(\mathbf{R}) \subset \mathcal{L}(S'/\mathbf{G})$, we have that

1. $\forall s \in P(R) : IS^Y_{S'}(s)|_{\mathbf{R}} = IS^Y_{S_R}(s)$;
2. $\forall s,t \in P(R) : IS^Y_{S_R}(s) \neq IS^Y_{S_R}(t) \Rightarrow IS^Y_{S'}(s) \neq IS^Y_{S'}(t)$.

Recall that we have assumed in Section IV-A that $\mathbf{R} \sqsubseteq \mathbf{G}$. Based on Lemma 1, hereafter, we assume w.l.o.g. that $\mathbf{R} \sqsubset \mathbf{K} \sqsubset \mathbf{G}$, which implies that $X_K \subseteq X_G \subseteq X$.

*Example 4:* Let us return to the running example. The automata $\mathbf{R}$ and $\mathbf{K}$ in Fig. 1 do not satisfy the assumption that $\mathbf{R} \sqsubset \mathbf{K}$. Therefore, we refine the state-spaces of $\mathbf{K}$ and $\mathbf{G}$ and obtain new automata $\mathbf{K}'$ and $\mathbf{G}'$ shown in Fig. 3(a) such that $\mathbf{R} \sqsubset \mathbf{K}' \sqsubset \mathbf{G}'$, $\mathcal{L}(\mathbf{K}) = \mathcal{L}(\mathbf{K}')$ and $\mathcal{L}(\mathbf{G}) = \mathcal{L}(\mathbf{G}')$. The AIC for the refined system is shown in Fig. 3(b). We see that the original state $\{3,4\}$ in $\mathcal{AIC}(\mathbf{G})$ splits into two states $\{3',4\}$ and $\{3,4'\}$ in $\mathcal{AIC}(\mathbf{G}')$.

## VIII. Synthesis of a Larger Solution

We tackle synthesis Problem 2 in this section. We continue to denote by $S_R$ the IS-based supervisor that achieves $R$ and by $T_R$ the deterministic BTS that realizes $S_R$. Suppose that $S'$ is a new supervisor such that $\mathcal{L}(S_R/\mathbf{G}) \subset \mathcal{L}(S'/\mathbf{G})$. Let $y = IS^Y_{S'}(s)$ for some $s \in P(\mathcal{L}(S'/\mathbf{G}))$. If $y|_{\mathbf{R}} = \emptyset$, then it implies that $s \notin P(\mathcal{L}(\mathbf{R}))$. In other words, the supervisor knows for sure that the system has already gone outside the lower bound language. If $y|_{\mathbf{R}} \neq \emptyset$, then the supervisor is not sure whether or not the system has already gone outside $R$. In this case, according to Lemma 1, $y|_{\mathbf{R}}$ uniquely determines a corresponding $Y$-state $IS^Y_{S_R}(s)$ under $S_R$.

Based on the above discussion, Algorithm GROW is proposed in order to synthesize a safe supervisor $S'$ such that

**Algorithm 2:** GROW($T_R, \mathcal{AIC}(\mathbf{G})$)

```
if MAX-VER(T_R, AIC(G))="Yes" then
    return T* ← T_R
else
    Let ỹ be the Y-state returned by MAX-VER and
      γ̃_max be a locally maximal
      (ỹ, T_R)-replacement;
    Q^Y_{T*} ← {y_0}, Q^Z_{T*} ← ∅;
    DoDFS(y_0, T*);
    return T*;

procedure DoDFS(y, T*);
if y|_R ≠ ∅ then
    if y = ỹ then
        Act ← γ̃_max;
    else
        Act ← c_{T_R}(y|_R);
else
    Find a locally maximal control decision Act s.t.
      ∀γ ∈ C_{AIC(G)}(y) : Act ⊄ γ;
z ← h_{YZ}(y, Act);
Add transition y --Act--> z to h^{T*}_{YZ};
if z ∉ Q^{T*}_Z then
    Q^{T*}_Z ← Q^{T*}_Z ∪ {z};
    for σ ∈ Σ_o : h_{ZY}(z, σ)! do
        y' ← h_{ZY}(z, σ);
        Add transition z --σ--> y' to h^{T*}_{ZY};
        if y' ∉ Q^{T*}_Y then
            Q^{T*}_Y ← Q^{T*}_Y ∪ {y'};
            DoDFS(y', T*);
```

$\mathcal{L}(S_R/\mathbf{G}) \subset \mathcal{L}(S'/\mathbf{G})$. First, we apply Algorithm MAX-VER to check whether or not $S_R$ is maximal. If not, then MAX-VER will return a $Y$-state, denoted by $\tilde{y}$. Therefore, we can replace the original control decision at $\tilde{y}$ by some larger control decision in $C_{\mathcal{AIC}(\mathbf{G})}(\tilde{y})$. We denote by $\tilde{\gamma}_{max} \in C_{\mathcal{AIC}(\mathbf{G})}(\tilde{y})$ a *locally maximal* $(\tilde{y}, T_R)$-replacement, i.e.,

$$\forall \gamma \in C_{\mathcal{AIC}(\mathbf{G})}(\tilde{y}) : \gamma \text{ is a } (\tilde{y}, T_R)\text{-replacement} \Rightarrow \tilde{\gamma}_{max} \not\subset \gamma$$

Next, we construct a new deterministic BTS $T^*$ in which the original control decision at $\tilde{y}$ is replaced by $\tilde{\gamma}_{max}$. More specifically, $T^*$ is constructed by a depth-first search from

the initial $Y$-state according to the following rules: for each $Y$-state $y$ encountered:

1. If $y|_{\mathbf{R}} = \emptyset$, then we take a locally maximal control decision in the AIC, since we know for sure that the string is already outside of $P(\mathcal{L}(\mathbf{R}))$ and there is no need to match the lower bound behavior.
2. If $y|_{\mathbf{R}} \neq \emptyset$ and $y = \tilde{y}$, then we take control decision $\tilde{\gamma}_{max}$ to replace the original one.
3. If $y|_{\mathbf{R}} \neq \emptyset$ and $y \neq \tilde{y}$, then we take the same control decision taken by $S_R$ at $y|_{\mathbf{R}}$, i.e., $c_{T_R}(y|_{\mathbf{R}})$.

The above three rules and the depth-first search are implemented by procedure DoDFS, in which a recursive call is made in order to to traverse the entire state space. Since there are at most $2^{|X|+|\Sigma_c|} + |\Sigma_o| \times 2^{|X|+|\Sigma_c|}$ transitions in $T^*$, the complexity of procedure DoDFS is $O(|\Sigma_o| \times 2^{|X|+|\Sigma_c|})$. Moreover, it takes $O(2^{2|X|+2|\Sigma_c|})$ to execute Algorithm MAX-VER. Therefore, the entire complexity of Algorithm GROW is $O(2^{2|X|+2|\Sigma|})$.

We illustrate Algorithm GROW by the following example.

*Example 5:* Let us return to the running example. The inputs of Algorithm GROW are $T_R$ and $\mathcal{AIC}(\mathbf{G}')$ shown in Figs. 2(b) and 3(b), respectively. By executing Algorithm MAX-VER, as shown in Example 3, we know that $S_R$ is not maximal and it returns $Y$-state $\tilde{y} = \{0\}$. Then we start procedure DoDFS from the initial $Y$-state $y_0 = \{0\}$. Since $\{0\}|_{\mathbf{R}} \neq \emptyset = \tilde{y}$, we take a locally maximal control replacement, say $\tilde{\gamma}_{max} = \{c_1\}$, and move to $Z$-state $(\{0,1\},\{c_1\})$. Then we need to consider all possible event occurrences from this $Z$-state. If $a$ occurs, then $Y$-state $\{3,4'\}$ is reached. Since $\{3,4'\}|_{\mathbf{R}} = \{3\} \neq \emptyset \neq \tilde{y}$, we need to take control decision $c_{T_R}(\{3\}) = \{c_1\}$ that is originally defined in $T_R$. Similarly, we need to take control decision $\{c_2\}$ if $Y$-state $\{3',4\}$ is reached. The above procedure yields a deterministic BTS $T^*$, which is the part highlighted in Fig. 3(b).

Now we present the correctness of Algorithm GROW.

*Theorem 4:* Let $T^*$ be the BTS returned by Algorithm GROW and $S_{T^*}$ be the unique supervisor included in $T^*$. Then we have that: (i) $S_{T^*}$ is a safe supervisor; and (ii) if $S_R$ is not maximal, then $\mathcal{L}(S_R/\mathbf{G}) \subset \mathcal{L}(S_{T^*}/\mathbf{G})$.

*Remark 1:* By Theorem 4, we know that Problem 2 is solved by Algorithm GROW. However, Problem 2 only requires to synthesize a supervisor that is strictly more permissive than $S_R$ and the synthesized supervisor may still not be maximal. Therefore, one may also be interested in the following problem:

*"Given a non-maximal supervisor $S_R$ such that $\mathcal{L}(S_R/\mathbf{G}) \subseteq K$, find a* maximal *supervisor $S$ such that $\mathcal{L}(S_R/\mathbf{G}) \subset \mathcal{L}(S/\mathbf{G})$."*

The solution to Problem 2 provides a straightforward way to find a solution to the above problem, namely, we can keep growing the behavior generated by the supervisor by iteratively applying Algorithm GROW. If this iteration terminates, then the solution obtained is a maximal solution that contains $R$. However, it is not straightforward to determine if this approach will terminate in a finite number steps. This is because, each time we apply Algorithm GROW, we need to make sure that the assumption that $\mathbf{R} \sqsubset \mathbf{K} \sqsubset \mathbf{G}$ holds. As a consequence, state space refinement of $\mathbf{G}$ may be required and the size of the new AIC may be larger than the previous one. Therefore, the solution space may be unbounded. So far, we have not elucidated if or when finite convergence occurs; this is an important topic for future work.

## IX. Conclusion

We have presented new results on the verification and synthesis of maximally permissive supervisors in partial-observed DES. First, we provided an algorithm to verify whether or not a supervisor is maximal. If the answer to the verification problem is negative, then an algorithm was proposed in order to synthesize a supervisor that strictly contains the behavior generated by the given non-maximal supervisor.

## References


[1] N. Ben Hadj-Alouane, S. Lafortune, and F. Lin. Centralized and distributed algorithms for on-line synthesis of maximal control policies under partial observation. *Discrete Event Dynamic Systems: Theory & Applications*, 6(4):379–427, 1996.

[2] 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.*, 15(2):111–117, 1990.

[3] K. Cai, R. Zhang, and W.M. Wonham. Relative observability of discrete-event systems and its supremal sublanguages. *IEEE Trans. Autom. Control*, 60(3):659–670, 2015.

[4] C.G. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems*. Springer, 2nd edition, 2008.

[5] 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.*, 2(1):47–69, 1989.

[6] R. Cieslak, C. Desclaux, A.S. Fawaz, and P. Varaiya. Supervisory control of discrete-event processes with partial observations. *IEEE Trans. Autom. Control*, 33(3):249–260, 1988.

[7] K. Inan. Nondeterministic supervision under partial observations. In *11th Int. Conf. Analy. Optim. Syst.: Discr. Event Syst.*, pages 39–48. Springer, 1994.

[8] F. Lin and W.M. Wonham. On observability of discrete-event systems. *Inform. Sciences*, 44(3):173–198, 1988.

[9] P.J. Ramadge and W.M. Wonham. Supervisory control of a class of discrete event processes. *SIAM J. Cont. Opt.*, 25(1):206–230, 1987.

[10] K. Rudie and W.M. Wonham. The infimal prefix-closed and observable superlanguange of a given language. *Syst. & Contr. Let.*, 15(5):361–371, 1990.

[11] P. Tabuada. *Verification and Control of Hybrid Systems: A Symbolic Approach*. Springer, 2009.

[12] S. Takai and T. Ushio. Effective computation of an $L_m(G)$-closed, controllable, and observable sublanguage arising in supervisory control. *Syst. & Contr. Let.*, 49(3):191–200, 2003.

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

[14] X. Yin and S. Lafortune. Synthesis of maximally permissive nonblocking supervisors for partially observed discrete event systems. In *53rd IEEE Conf. Decision and Control*, pages 5156–5162, 2014.

[15] X. Yin and S. Lafortune. Synthesis of maximally permissive supervisors for partially observed discrete event systems. *IEEE Trans. Autom. Contr.*, 2016. Doi 10.1109/TAC.2015.2460391.

[16] X. Yin and S. Lafortune. A uniform approach for synthesizing property-enforcing supervisors for partially-observed discrete-event systems. *IEEE Trans. Autom. Contr.*, 2016. Doi 10.1109/TAC.2015.2484359.

[17] T.-S. Yoo and S. Lafortune. Solvability of centralized supervisory control under partial observation. *Discrete Event Dynamic Systems: Theory & Applications*, 16(4):527–553, 2006.