Synthesis of Maximally-Permissive Supervisors for the Range Control Problem

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

***Abstract*—We investigate the supervisor synthesis problem for centralized partially-observed discrete event systems subject to safety specifications. It is well known that this problem does not have a unique supremal solution in general. Instead, there may be several incomparable locally maximal solutions. One then needs a mechanism to select one locally maximal solution. Our approach in this paper is to consider a lower bound specification on the controlled behavior, in addition to the upper bound for the safety specification. This leads to a generalized supervisory control problem called the *range control problem*. While the upper bound captures the (prefix-closed) legal behavior, the lower bound captures the (prefix-closed) minimum required behavior. We provide a synthesis algorithm that solves this problem by effectively constructing a maximally-permissive safe supervisor that contains the required lower bound behavior. This is the first algorithm with such properties, as previous works solve either the maximally-permissive safety problem (with no lower bound), or the lower bound containment problem (without maximal permissiveness).**

***Index Terms*—Discrete event systems, maximal permissiveness, partial observation, supervisory control, synthesis.**

I. Introduction

We INVESTIGATE the supervisor synthesis problem for partially-observed Discrete Event Systems (DES) in the framework of supervisory control theory [16]. In this problem, one is interested in synthesizing a supervisor such that the closed-loop system under control satisfies a given safety specification. Formally, let $\mathbf{G}$ be a system and $K \subseteq \mathcal{L}(\mathbf{G})$ be a prefix-closed specification language describing the legal behavior for the controlled system. The goal is to find a supervisor S such that $\mathcal{L}(S/\mathbf{G}) \subseteq K$, where $\mathcal{L}(S/\mathbf{G})$ denotes the language generated by $\mathbf{G}$ under the control of S (closed-loop system). Moreover, we want the supervisor S to be as permissive as possible.

Let $L \subseteq K$ be a sub-language of K. Under the partial observation setting, it is well-known that there exists a supervisor that exactly achieves L if and only if L is *controllable* and *observable* [8], [13]. Since controllability is preserved under union, there exists a supremal controllable sub-language of K; this is the unique supremal solution to the synthesis problem when all events are observable. However, the supervisor synthesis problem is much more challenging under the partial observation setting, since observability is not preserved under union. Therefore, the supervisor synthesis problem may not have a unique supremal solution in general. Instead, there may be several incomparable *locally maximal* solutions.

Several approaches have been proposed in the literature in order to tackle the synthesis problem; see, e.g., [2]–[4], [7], [9], [19], [29]. One approach is to compute the supremal controllable and *normal* solution, which was initially proposed in [8], [13]; see, also [3], [7] 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 [4], [19], two different solutions that are strictly larger than the supremal normal solution were derived. However, the solutions obtained by these approaches are not maximal in general. In [2], an online approach was proposed in order to compute a maximal solution. This approach can only be applied to prefix-closed specifications, since the solution obtained may be blocking in general. The problem of supervisor (or controller) synthesis under partial observation has also been investigated in other frameworks; see, e.g., [1], [6], [12], [21].

In our recent works [27], [28], we proposed a new information structure called the All Inclusive Controller (AIC) in order to solve the supervisor synthesis problem. The AIC is a finite structure that "embeds" all (infinitely many in general) safe supervisors for a given specification. A maximal solution for the prefix-closed case was also obtained based on the AIC. The structure of the AIC was further extended to the non-prefix-closed case, where the Non-Blocking All Inclusive Controller (NB-AIC) was proposed. By using the NB-AIC as a basis, we showed that the problem of synthesizing a maximally-permissive safe and non-blocking supervisor for partially-observed DES is decidable. Moreover, the solution can be represented by a finite structure.

Manuscript received August 24, 2016; revised December 8, 2016; accepted December 16, 2016. Date of publication December 23, 2016; date of current version July 26, 2017. This work was supported in part by the U.S. National Science Foundation grants CCF-1138860 (Expeditions in Computing project ExCAPE: Expeditions in Computer Augmented Program Engineering) and CNS-1446298. This work was done when the first author was at the University of Michigan. Recommended by Associate Editor S. Takai. *(Corresponding author: Xiang Yin.)*

X. Yin is with the Department of Automation, Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: xiangyin@umich.edu).

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.2016.2644867

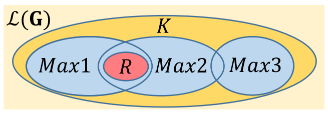


Fig. 1. Let $\mathbf{G}$ be the system, K be the legal behavior and R be the required behavior. $Max1, Max2$ and $Max3$ are three incomparable maximal solutions in K, i.e., $\forall i, j \in \{1,2,3\} : Maxi \not\subseteq Maxj$. However, $Max1$ and $Max2$ contain the required behavior R, while $Max3$ does not contain any string in R.

Although maximal solutions have been reported in the literature in [2], [27], 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. However, no consideration is given to including some minimum *required* behavior in these solutions, a meaningful criterion when choosing among locally maximal solutions. This phenomenon is illustrated in Fig. 1. In fact, none of the synthesis algorithms in [2]–[4], [7], [19], [27], [29] can guarantee that the supervisor synthesized therein contains a given required behavior.

In order to resolve the above issue, we consider in this paper a generalized supervisor synthesis problem called the *Maximally-Permissive Range Control Problem*. In this problem, we not only want to find a locally maximal supervisor, but we also require that the synthesized maximal supervisor contains a given behavior. Namely, we want to find a "meaningful" maximal solution. Note that, as illustrated by Fig. 1, such a solution need not be unique. More specifically, in addition to the safety specification language K, which is also referred to as the upper bound language, we consider a prefix-closed lower bound language $R \subseteq K$, which models the required behavior that the closed-loop system must achieve. To solve the range control problem, we present a new synthesis algorithm based on the two notions of All Inclusive Controller (AIC) and Control Simulation Relation (CSR). The AIC was originally proposed in [27] in order to synthesize an arbitrary maximal solution, with no consideration to a lower bound behavior. Throughout the paper, we only consider prefix-closed languages. Therefore, the issue of non-blockingness cannot be handled. On the other hand, to the best of our knowledge, the maximally-permissive range control problem we solve herein was an open problem even in the prefix-closed case. Note that the range control problem is quite different from the problems studied in [27], [28], where no lower bound specification is considered. Consequently, several new techniques are developed in this paper to tackle the lower bound requirement.

This paper is organized as follows. In Section II, we first introduce some necessary background. Then we formulate the maximally-permissive range control problem that we solve in the paper. The notions of Bipartite Transition Structure (BTS) and AIC are reviewed in Section III. The main contributions of this paper are presented in Sections IV–VI. In Section IV, we first reveal that the notion of strict sub-automaton plays an important role in the range control problem. Then we provide a new constructive approach for computing the infimal safe supervisor that contains the lower bound behavior. In Section V, we define the notion of Control Simulation Relation (CSR). The CSR is used to resolve the future dependency issue, which is the main difficulty in handling maximal permissiveness with the lower bound constraint. In Section VI, we first provide an algorithm to synthesize a maximally-permissive supervisor that contains the required behavior. Then we prove the correctness of the proposed algorithm. We also discuss how to verify whether a given supervisor is maximal or not. Finally, we conclude the paper in Section VII.

Preliminary and partial versions of some of the results in this paper are presented in [25], [26]. The differences between the present paper and its conference versions are as follows. First, the approach in [25] does not solve the range control problem, since it does not guarantee finite convergence. Second, although the algorithm proposed in [26] solves the range control problem, the complexity of the algorithm is double-exponential in general. This is because [26] assumes that the lower bound specification language is controllable and observable; in the worst case, it requires exponential state-space explosion of the original lower bound automaton in order to make this assumption hold. In this paper, we provide a unified and simplified framework that subsumes all results in [25], [26], together with all proofs (only some proofs are presented in [25], [26]). Moreover, we relax the assumption that the lower bound specification language is controllable and observable; this significantly reduces the complexity of the synthesis procedure from double-exponential-time to exponential-time.

II. Problem Formulation

A. Preliminaries

Let Σ be a finite set of events. We denote by Σ^* the set of all 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 is a subset of Σ^*. We denote by $\overline{L}$ the prefix-closure of language L; L is said to be *prefix-closed* if $\overline{L} = L$, i.e., $\overline{L} = \{u \in \Sigma^* : \exists v \in \Sigma^* \text{ s.t. } uv \in L\}$.

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, Σ is the finite set of events, $\delta : X \times \Sigma \rightarrow X$ 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 δ is extended to $X \times \Sigma^*$ in the usual manner; see, e.g., [5]. For brevity, we write $\delta(x, s)$ as $\delta(s)$ if $x = x_0$. We define $\mathcal{L}(\mathbf{G}, x) := \{s \in \Sigma^* : \delta(x, s)!\}$ as the language generated by $\mathbf{G}$ from state x, where ! means "is defined". We write $\mathcal{L}(\mathbf{G}, x)$ as $\mathcal{L}(\mathbf{G})$ if $x = x_0$. 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[1] of $\mathbf{B}$, denoted by $\mathbf{A} \sqsubset \mathbf{B}$, if (i) $\mathbf{A} \sqsubseteq \mathbf{B}$; and (ii) $\forall x, y \in X_A, \forall s \in \Sigma^* : \delta_B(x, s) = y \Rightarrow \delta_A(x, s) = y$. Note that, for any two automata $\mathbf{A}$ and $\mathbf{B}$ such

[1]The definition of strict sub-automaton used in this paper is slightly stronger than the definition used in [7].

that $\mathcal{L}(\mathbf{A}) \subseteq \mathcal{L}(\mathbf{B})$, we can always refine the state spaces of $\mathbf{A}$ and $\mathbf{B}$ and obtain two new automata $\mathbf{A}'$ and $\mathbf{B}'$ such that $\mathcal{L}(\mathbf{A}')=\mathcal{L}(\mathbf{A})$, $\mathcal{L}(\mathbf{B}')=\mathcal{L}(\mathbf{B})$ and $\mathbf{A}' \sqsubset \mathbf{B}'$.

In the supervisory control framework [16], the event set is partitioned into two disjoint sets $\Sigma = \Sigma_c \dot{\cup} \Sigma_{uc}$, where Σ_c is the set of controllable events and Σ_{uc} is the set of uncontrollable events. A control decision $\gamma \in 2^{\Sigma}$ is a set of events with the constraint that $\Sigma_{uc} \subseteq \gamma$, i.e., the supervisor should always enable uncontrollable events. We denote by Γ the set of all control decisions, i.e., $\Gamma := \{\gamma \in 2^{\Sigma} : \Sigma_{uc} \subseteq \gamma\}$. Under the partial observation setting [8], [13], the event set is further partitioned into another pair of disjoint sets $\Sigma = \Sigma_o \dot{\cup} \Sigma_{uo}$, where Σ_o is the set of observable events and Σ_{uo} is the set of unobservable events. The natural projection $P : \Sigma^* \to \Sigma_o^*$ is defined by

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

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 $S : P(\mathcal{L}(\mathbf{G})) \to \Gamma$, i.e., it enables events only based on its observations. We denote by $\mathcal{L}(S/\mathbf{G})$ the language generated by the closed-loop system under control, computed recursively by:

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

Let $K \subseteq \mathcal{L}(\mathbf{G})$. We say that language K is

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

It is well-known that, given a language $K \subseteq \mathcal{L}(\mathbf{G})$, there exists a partial observation supervisor S such that $\mathcal{L}(S/\mathbf{G}) = \overline{K}$, if and only if, K is controllable and observable [8], [13].

When K is not controllable or observable, i.e., $\overline{K}$ cannot be exactly achieved by a supervisor, one is interested in finding a controllable and observable *sub-language* of K that is as "large" as possible in terms of language inclusion; this problem is also referred to as the *synthesis problem*. Since controllability is preserved under union, there exists a *supremal controllable sub-language* of K [16], i.e., the synthesis problem under the full observation setting has a unique optimal solution. However, there does not exist a supremal controllable and observable sub-language of $\mathcal{L}(\mathbf{G})$, since observability is not preserved under union in general, i.e., $L_1 \cup L_2$ may be not observable even if both L_1 and L_2 are observable. Instead, there may be several incomparable *locally maximal* sub-languages; the definition of locally maximal will become clear later.

Although observability is not preserved under union, both controllability and observability are preserved under intersection when the languages are prefix-closed. Therefore, for a given language $R \subseteq \mathcal{L}(\mathbf{G})$, we define the following class of languages

$$\mathcal{CO}(R) = \{L \subseteq \Sigma^* : (L = \overline{L}) \wedge (R \subseteq L \subseteq \mathcal{L}(\mathbf{G})) \wedge (L \text{ is controllable and observable})\} \quad (2)$$

and there exists an infimal element in class $\mathcal{CO}(R)$ defined by $R^{\downarrow CO} := \bigcap_{L \in \mathcal{CO}(R)} L$. We call $R^{\downarrow CO}$ the *infimal prefix-closed controllable and observable super-language* of R. Moreover, it was shown in [11], [17] that $R^{\downarrow CO}$ is a regular language when R is regular; language-based formulas for $R^{\downarrow CO}$ were provided in [11], [17].

Since we consider partially-observed DES, we define an *information state* as a set of states and denote by $I = 2^X$ the set of information states. Let $i \subseteq I$ be an information state, $\gamma \in \Gamma$ be a control decision, and $\sigma \in \Sigma_o$ be an observable event. We define the following two operators:

$$\text{UR}_\gamma(i) = \{x \in X : \exists y \in i, \exists s \in (\Sigma_{uo} \cap \gamma)^* \text{ s.t. } x = \delta(y, s)\} \quad (3)$$

$$\text{Next}_\sigma(i) = \{x \in X : \exists y \in i \text{ s.t. } x = \delta(y, \sigma)\} \quad (4)$$

B. Problem Formulation

In this paper, we consider a generalized supervisory control synthesis problem, called the *range control problem*, where we have two prefix-closed specification languages:

- the upper bound language $K = \overline{K} \subseteq \mathcal{L}(\mathbf{G})$; and
- the lower bound language $R = \overline{R} \subseteq K$.

The upper bound K describes the *legal* behavior of the system and we say that a supervisor S is *safe* if $\mathcal{L}(S/\mathbf{G}) \subseteq K$. We say that a safe supervisor S is *maximally permissive* (or maximal) if there does not exist another safe supervisor S', such that $\mathcal{L}(S/\mathbf{G}) \subset \mathcal{L}(S'/\mathbf{G})$. Note that the maximal supervisor may not be unique and there may be two incomparable maximal supervisors S_1 and S_2 such that $\mathcal{L}(S_1/\mathbf{G}) \not\subset \mathcal{L}(S_2/\mathbf{G})$ and $\mathcal{L}(S_2/\mathbf{G}) \not\subset \mathcal{L}(S_1/\mathbf{G})$. In order to synthesize a "meaningful" maximal solution, we introduce a lower bound language R describing the *required* behavior that the closed-loop system must achieve. Examples of using the range requirement to impose design constraints can be found in [10], [13]–[15]. We now formulate the *Maximally-Permissive Range Control Problem* (MPRCP):

Problem 1. (Maximally-Permissive Range Control Problem) Given system $\mathbf{G}$, lower bound language R and upper bound language K such that $R \subseteq K \subseteq \mathcal{L}(\mathbf{G})$, synthesize a maximally-permissive supervisor $S^* : P(\mathcal{L}(\mathbf{G})) \to \Gamma$ such that $R \subseteq \mathcal{L}(S^*/\mathbf{G}) \subseteq K$.

Remark 1: We make several comments on MPRCP.

1) First, under the assumption that $\Sigma_c \subseteq \Sigma_o$, MPRCP has a unique solution, if one exists. In this case, since controllability and observability together imply normality, the supremal controllable and observable sub-language of K does exist [5] and it coincides with the *supremal controllable and normal sub-language* of K, denoted by $K^{\uparrow CN}$. Therefore, it suffices to compute $K^{\uparrow CN}$, and test whether or not $R \subseteq K^{\uparrow CN}$. If so, then $K^{\uparrow CN}$ is the unique supremal solution; otherwise, there does not exist a solution to MPRCP.
2) Second, when the lower bound requirement is relaxed, i.e., $R = \{\epsilon\}$, MPRCP is solved by the results in [2], [27], since it suffices to synthesize an *arbitrary* maximal supervisor without taking the lower bound into consideration.
3) Finally, if the maximal permissiveness requirement is relaxed, then we just need to compute $R^{\downarrow CO}$, and test

whether or not $R^{\downarrow CO} \subseteq K$. If so, then $R^{\downarrow CO}$ is the *most conservative* solution; otherwise, MPRCP does not have a solution.

Hence, many existing problems solved in the literature are special cases of MPRCP. To the best of our knowledge, MPRCP is still open for the general case, which is clearly more difficult than the above special cases. In fact, the idea of using both the upper bound and the lower bound specifications was originally introduced in [13]. However, the approach proposed in [13] can only find the most restrictive supervisor satisfying the range requirement. ■

Remark 2: Since controllability and observability provide the necessary and sufficient conditions for the existence of a supervisor that achieves a given language, MPRCP is equivalent to the problem of finding a maximal controllable and observable language L such that $R \subseteq L \subseteq K$. This language-based formulation and the supervisor-based formulation are essentially the same. All results developed hereafter will be stated in terms of supervisors rather than languages. ■

Throughout the paper, we use $\mathbf{K} = (X_K, \Sigma, \delta_K, x_{0,K})$ to denote the automaton generating K, and use $\mathbf{R} = (X_R, \Sigma, \delta_R, x_{0,R})$ to denote the automaton generating R. For the sake of simplicity and without loss of generality, we assume that $\mathbf{K} \sqsubset \mathbf{G}$. This assumption essentially says that legality of strings is fully captured by states. Namely, $X \setminus X_K$ is the set of illegal states and $\delta(x_0, s) \notin X_K$ iff $s \notin K$.

III. All Inclusive Controller

In order to solve MPRCP, we use the two structures called Bipartite Transition System (BTS) and All Inclusive Controller (AIC); these were introduced in [27] to solve supervisory control problems. For the sake of completeness of this paper, we review in this section key definitions and results from [27].

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

$$T = (Q_Y^T, Q_Z^T, h_{YZ}^T, h_{ZY}^T, \Sigma_o, \Gamma, y_0) \tag{5}$$

where $Q_Y^T \subseteq I = 2^X$ is the set of Y-states; $Q_Z^T \subseteq I \times \Gamma$ is the set of Z-states and $I(z)$ and $\Gamma(z)$ denote, respectively, the information state and the control decision components of a Z-state z, so that $z = (I(z), \Gamma(z))$; $h_{YZ}^T : Q_Y^T \times \Gamma \to Q_Z^T$ is the partial transition function from Y-states to Z-states, which satisfies the following constraint: for any $y \in Q_Y^T, z \in Q_Z^T$ and $\gamma \in \Gamma$, we have

$$h_{YZ}^T(y, \gamma) = z \Rightarrow [I(z) = \mathrm{UR}_\gamma(y)] \wedge [\Gamma(z) = \gamma]; \tag{6}$$

$h_{ZY}^T : Q_Z^T \times \Sigma_o \to Q_Y^T$ 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 $\sigma \in \Sigma_o$, we have

$$h_{ZY}^T(z, \sigma) = y \Leftrightarrow [\sigma \in \Gamma(z)] \wedge [y = \mathrm{Next}_\sigma(I(z))]; \tag{7}$$

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 $y_0 = \{x_0\}$.

Intuitively, a BTS is a game structure between the system (control decision) and the environment (event occurrence). Each Y-state is a "system state" from which the supervisor makes control decisions. Each Z-state is an "environment state" from which (enabled) observable events occur. Since the supervisor cannot choose which event will occur once it has made a control decision, all enabled and feasible observable events should be defined at a Z-state; this is why we put "$\Leftrightarrow$" in Equation (7). We denote by $C_T(y)$ the set of control decisions defined at $y \in Q_Y^T$ in T, i.e., $C_T(y) = \{\gamma \in \Gamma : h_{YZ}^T(y, \gamma)!\}$. We say that a BTS T is

- *complete*, if $\forall y \in Q_Y^T : C_T(y) \neq \emptyset$; and
- *deterministic*, if $\forall y \in Q_Y^T : |C_T(y)| = 1$.

If T is deterministic, then 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 $y = h_{ZY}^T(z, \sigma)$. Note that, 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 $\gamma_0\sigma_1\gamma_1\sigma_2 \ldots \sigma_n\gamma_n$, where $\gamma_i \in \Gamma, \sigma_i \in \Sigma_o$, a *run*. A run also induces a *sequence*

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

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

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

We denote by $IS_S^Y(s)$ and $IS_S^Z(s)$, the last Y-state and Z-state in $y_0z_0y_1z_2 \ldots z_{n-1}y_nz_n$, 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. Moreover, $IS_S^Y(s)$ and the information state component of $IS_S^Z(s)$ can also be expressed as follows

$$IS_S^Y(s) = \{x \in X : \exists t \in (\Sigma^*\Sigma_o \cup \{\epsilon\}) \cap \mathcal{L}(S/\mathbf{G}) \text{ s.t. } P(t) = s \wedge \delta(x_0, t) = x\}$$

$$I(IS_S^Z(s)) = \{x \in X : \exists t \in \mathcal{L}(S/\mathbf{G}) \text{ s.t. } P(t) = s \wedge \delta(x_0, t) = x\}$$

Example 1: Let us consider the system $\mathbf{G}$ shown in Fig. 2(a). The upper bound automaton $\mathbf{K}$ is obtained by removing the single illegal state 7 from $\mathbf{G}$. Let $\Sigma_c = \{c_1, c_2\}$ and $\Sigma_o = \{a, b\}$. The structure shown in Fig. 2(b) is a complete BTS. From the initial Y-state $y_0 = \{1\}$, we can make control decision $\{\}$, which leads to a Z-state via $\{1\} \xrightarrow{\{\}} (\{1, 2\}, \{\})$. By observing event a from this Z-state, we move to the next Y-state by $(\{1, 2\}, \{\}) \xrightarrow{a} \{3, 4\}$. This BTS is not deterministic since there are three different control decisions defined at Y-state $\{3, 4\}$.

Let us consider supervisor S defined by

$$S(s) = \begin{cases} \Sigma_{uc} & \text{if } s = \epsilon \\ \{c_1\} \cup \Sigma_{uc} & \text{if } s \in \{a, b\} \end{cases} \tag{8}$$

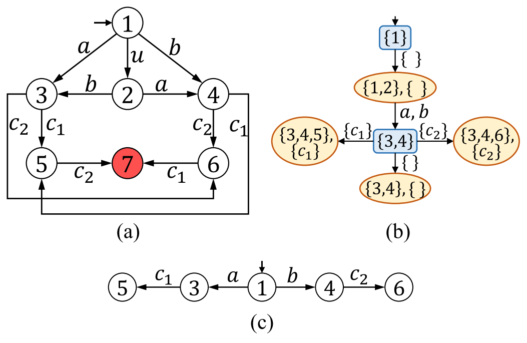


Fig. 2. For $\mathbf{G}$, we have $\Sigma_o = \{a, b\}$ and $\Sigma_c = \{c_1, c_2\}$. In the AIC, rectangular states represent Y-states and oval states represent Z-states. For the sake of simplicity, uncontrollable events a and b are omitted in each control decision in the figure. (a) $\mathbf{K}$ and $\mathbf{G}$, (b) $\mathcal{AIC}(\mathbf{G}, \mathbf{K})$, (c) $\mathbf{R}$.

Then, for string $a \in P(\mathcal{L}(S/\mathbf{G}))$, S induces the sequence

$$\{1\} \xrightarrow{S(\epsilon)} (\{1,2\},\{\}) \xrightarrow{a} \{3,4\} \xrightarrow{S(a)} (\{3,4,5\},\{c_1\})$$

Therefore, we have that $IS_S^Y(a) = \{3,4\}$ and $IS_S^Z(a) = (\{3,4,5\},\{c_1\})$. ■

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

Definition 2: A supervisor S is said to be included in a BTS T if for any observable string $s \in P(\mathcal{L}(S/\mathbf{G}))$, 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.

In [27], the AIC structure is defined as the largest BTS including only safe supervisors.

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

Note that the AIC only depends on the system model $\mathbf{G}$ and the upper bound automaton $\mathbf{K}$; it does not depend on the lower bound automaton $\mathbf{R}$. We refer the reader to [27] for more details and for the construction of the AIC. Here we recall a key property of the AIC from [27].

Theorem 1: ([27]): A supervisor S is safe iff it is included in the AIC, i.e., $S \in \mathbb{S}(\mathcal{AIC}(\mathbf{G},\mathbf{K}))$.

Let $y \in Q_Y^{AIC}$ be a Y-state in the AIC. We say that a control decision $\gamma \in \Gamma$ is *safe* at y if $\gamma \in C_{\mathcal{AIC}(\mathbf{G},\mathbf{K})}(y)$. Then we have the following monotonicity properties.

Proposition 1: (Monotonicity Properties [24]):

1) Any control decision that is safe at Y-state y_1 is also safe at Y-state $y_2 \subseteq y_1$.
2) If control decision γ_1 is safe at Y-state y, then so is any control decision $\gamma_2 \subseteq \gamma_1$.

If a BTS T is deterministic, then the supervisor included in T is unique, since the control decision at each Y-state is unique. In this case, we denote by S_T the unique supervisor included in T, i.e., $\mathbb{S}(T) = \{S_T\}$. Essentially, T is a *realization* of supervisor S_T. However, not all supervisors can be realized by a BTS, since a supervisor may make different control decisions at different visits to the same Y-state. We say that a supervisor S is *information-state-based* (IS-based) if

$$\forall s, t \in P(\mathcal{L}(S/\mathbf{G})) : IS_S^Y(s) = IS_S^Y(t) \Rightarrow S(s) = S(t).$$

Then, a supervisor can be realized by a BTS iff it is IS-based.

Example 2: Let us consider again the system $\mathbf{G}$ and the specification $\mathbf{K}$ in Fig. 2(a). The AIC for this system is in fact the BTS shown in Fig. 2(b). Note that, at Y-state $\{3,4\}$, 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 control decision $\{c_1, c_2\}$ is not defined at $\{3,4\}$ in the AIC. It is easy to verify that supervisor S defined in Equation (8) is included in the AIC, since $S(\epsilon) \in C_{\mathcal{AIC}(\mathbf{G},\mathbf{K})}(\{1\})$ and $S(a), S(b) \in C_{\mathcal{AIC}(\mathbf{G},\mathbf{K})}(\{3,4\})$. Therefore, S is a safe supervisor. Moreover, S is an IS-based supervisor. In particular, if we remove control decisions $\{\}$ and $\{c_2\}$ from Y-state $\{3,4\}$ and call the remaining (deterministic) BTS T, then we have that $\mathbb{S}(T) = \{S\}$, i.e., $S = S_T$. ■

Remark 3: In Fig. 2(b), we can also make control decision $\{c_1\}$ at the initial Y-state $\{1\}$. However, event c_1 is not feasible before the next observable event occurs. Therefore, we treat c_1 as a redundant event and omit it in the control decision. Formally, we say that a control decision γ is *irredundant* at Y-state y if $\forall \sigma \in \gamma, \exists x \in \mathrm{UR}_\gamma(y) : \delta(x, s\sigma)!$. We say that a BTS is irredundant if for any $y \in Q_Y^T$ and for any $\gamma \in C_T(y)$, γ is irredundant at y. Similarly, we say that a supervisor is irredundant if for any $s \in P(\mathcal{L}(S/\mathbf{G}))$, $S(s)$ is irredundant at $IS_S^Y(s)$. Hereafter, we will only consider irredundant BTSs and supervisors; this will not affect their properties. ■

IV. Synthesis of the Infimal Supervisor

In this section, we synthesize a BTS T_R that realizes the infimal supervisor achieving the lower bound, i.e., $\mathcal{L}(S_{T_R}/\mathbf{G}) = R^{\downarrow CO}$. This infimal supervisor will be further used as a basis to solve MPRCP. First, we illustrate the role of strict sub-automaton in this problem. Then, we provide an effective algorithm to construct T_R.

A. The Role of Strict Sub-Automaton

The goal of this section is to construct a BTS T_R such that $\mathcal{L}(S_{T_R}/\mathbf{G}) = R^{\downarrow CO}$. Although we know that $R^{\downarrow CO}$ is a regular language, this fact in itself is not sufficient for the purpose of synthesis. Specifically, we are interested in whether or not $R^{\downarrow CO}$ can be achieved by an IS-based supervisor which can be realized by a BTS. This question is very important, since it essentially asks *what is the right state space in order to realize the infimal supervisor that contains* R. One may conjecture that there always exists a BTS T_R such that $\mathcal{L}(S_{T_R}/\mathbf{G}) = R^{\downarrow CO}$ when $R^{\downarrow CO} \subseteq K$. However, this is not true in general as illustrated by the following example.

Example 3: Let us still consider the system $\mathbf{G}$ and the upper bound automaton $\mathbf{K}$ shown in Fig. 2(a). We consider a lower bound language R which is generated by automaton $\mathbf{R}$ shown in Fig. 2(c). One can easily check that any IS-based supervisor S does not contain R. This is because events a and b lead to the same Y-state $\{3,4\}$ in any BTS and control decision $S(a)$ and $S(b)$ should always be the same in any IS-based supervisor

S. However, we can find a non-IS-based supervisor S', which enables c_1 after observing a and enables c_2 after observing b, such that $R^{\downarrow CO} = \mathcal{L}(S'/\mathbf{G})$. ■

The reason why there may not exist an IS-based supervisor that achieves $R^{\downarrow CO}$ is explained as follows. Suppose that $\mathbf{R}$ is a sub-automaton of $\mathbf{K}$ such that we can match the state space of $\mathbf{R}$ with the state space of $\mathbf{K}$. Let $s \in P(R)$ be an observable string in $P(R)$ and define

$$y_R(s) = \{x \in X_R : \exists t \in R \cap (\Sigma^*\Sigma_o \cup \{\epsilon\}) \text{ s.t. } \delta_R(x_0, t) = x \wedge P(t) = s\}$$

as the "information state" of $\mathbf{R}$ reached upon observing s, which is analogous to a Y-state in a BTS. Then 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 is lost by using the newly reached information state. We call this phenomenon *information merge*. For example, for the lower bound automaton $\mathbf{R}$ shown in Fig. 2(c), we have that $y_R(a) = \{3\}$ and $y_R(b) = \{4\}$. In order to achieve R, in addition to enabling events a and b initially, we also need to enable event u, since it is uncontrollable. Then the two different "information states" $\{3\}$ and $\{4\}$ in $\mathbf{R}$, which are reached by observing a and b, respectively, will be 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 lower bound 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 and suppose that $X_R \subseteq X$. We denote by $y|_\mathbf{R}$ the restriction of y to the state space of $\mathbf{R}$, i.e., $y|_\mathbf{R} = y \cap X_R$. The following result says that the state merging phenomenon described above will not occur when $\mathbf{R}$ is a *strict* sub-automaton of $\mathbf{K}$.

Proposition 2: Assume that $\mathbf{R} \sqsubset \mathbf{K} \sqsubset \mathbf{G}$. Then for any supervisor S such that $R \subseteq \mathcal{L}(S/\mathbf{G})$, we have that

1. $\forall s \in P(R) : IS_S^Y(s)|_\mathbf{R} = y_R(s)$;
2. $\forall s, t \in P(R) : y_R(s) \neq y_R(t) \Rightarrow IS_S^Y(s) \neq IS_S^Y(t)$.

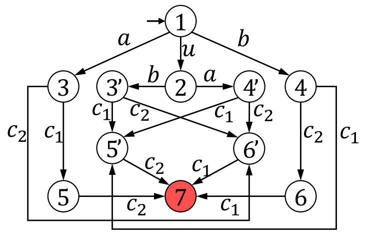
Proof: See the Appendix. ■

Remark 4: The intuition of the above result is as follows. Since $\mathbf{R} \sqsubset \mathbf{K}$, any newly introduced string, namely a string in $\mathcal{L}(\mathbf{K}) \setminus \mathcal{L}(\mathbf{R})$, must lead to a state in $X_K \setminus X_R$. Therefore, if strings s and t lead to two distinct "information states" $y_1 = y_R(s)$ and $y_2 = y_R(t)$, respectively, then the newly reached Y-states y_1' and y_2' under a supervisor whose closed-loop language contains R must be in the form of $y_1' = y_1 \cup \hat{y}_1$ and $y_2' = y_2 \cup \hat{y}_2$, respectively, where $\hat{y}_1, \hat{y}_2 \subseteq X_K \setminus X_R$. Since $y_1 \neq y_2$, we know that $y_1' \neq y_2'$. ■

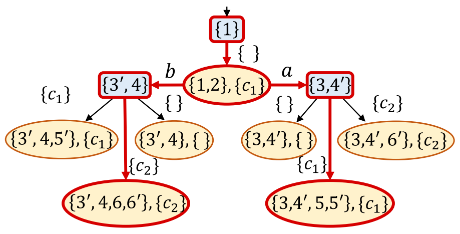
Therefore, we make the following assumption hereafter.

Assumption 1: $\mathbf{R} \sqsubset \mathbf{K} \sqsubset \mathbf{G}$.

Remark 5: Note that the above assumption is without loss of generality: if $\mathbf{R}$, $\mathbf{K}$ and $\mathbf{G}$ do not satisfy this assumption, then we can always refine the state spaces of $\mathbf{R}$, $\mathbf{K}$ and $\mathbf{G}$ by constructing new automata $\mathbf{R}'$, $\mathbf{K}'$ and $\mathbf{G}'$ such that 1) $\mathbf{R}' \sqsubset \mathbf{K}' \sqsubset \mathbf{G}'$; and 2) $\mathcal{L}(\mathbf{R}) = \mathcal{L}(\mathbf{R}')$, $\mathcal{L}(\mathbf{K}) = \mathcal{L}(\mathbf{K}')$ and $\mathcal{L}(\mathbf{G}) = \mathcal{L}(\mathbf{G}')$.



(a)



(b)

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

Such a pre-processing algorithm can be found in [23], which generalizes the procedure in [7] from two automata to three automata. In the worst case, the refined system model $\mathbf{G}'$ contains $|X| \times (|X_K| + 1) \times (|X_R| + 1)$ states. Therefore, only polynomial-space refinement is needed to make Assumption 1 hold; this is different from the state-partition-automata-based refinement in the literature, which has an exponential complexity. This assumption and Proposition 2 play important roles in this paper; they will also be involved several times in our later development. Finally, we remark that the reason why we assume that $\mathbf{K} \sqsubset \mathbf{G}$ and the reason why we assume that $\mathbf{R} \sqsubset \mathbf{K}$ are different. We assume that $\mathbf{K} \sqsubset \mathbf{G}$ to guarantee that legality of strings is fully captured by states. We assume that $\mathbf{R} \sqsubset \mathbf{K}$ to make sure that the information merge phenomenon will not occur. ■

Example 4: Let us return to Example 2. The original automata $\mathbf{R}$ and $\mathbf{K}$ in Figs. 2(c) and (a) 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 $\mathcal{AIC}(\mathbf{G}', \mathbf{K}')$ for the refined system is shown in Fig. 3(b). We see that the original state $\{3, 4\}$ in $\mathcal{AIC}(\mathbf{G}, \mathbf{K})$ splits into two states $\{3', 4\}$ and $\{3, 4'\}$ in $\mathcal{AIC}(\mathbf{G}', \mathbf{K}')$. ■

B. Synthesis Algorithm

We are now ready to show how to compute the supervisor that achieves $R^{\downarrow CO}$. In particular, we show that such a supervisor can be realized by a BTS.

Let $y \subseteq X_R$ be a set of states in $\mathbf{R}$. We first define the following set of events

$$\Gamma_R(y) := \{\sigma \in \Sigma : \exists x \in y, \exists s \in \Sigma_{uo}^* \text{ s.t. } \delta_R(x, s\sigma)!\}$$

The following result reveals that $\Gamma_R(y)$ is indeed the set of events that should be enabled at y in order to achieve R.

Proposition 3: For any supervisor $S : P(\mathcal{L}(\mathbf{G})) \rightarrow \Gamma$, $R \subseteq \mathcal{L}(S/\mathbf{G})$, if and only if,

$$\forall s \in P(\mathcal{L}(S/\mathbf{G})) : IS_S^Y(s)|_{\mathbf{R}} \neq \emptyset \Rightarrow \Gamma_R(IS_S^Y(s)|_{\mathbf{R}}) \subseteq S(s).$$

Proof: See the Appendix. ■

Now, we are ready to present the algorithm that constructs the BTS T_R such that $\mathcal{L}(S_{T_R}/\mathbf{G}) = R^{\downarrow CO}$. Specifically, the BTS T_R is constructed by a depth-first search as follows. Initially, we start from the initial Y-state y_0. For each Y-state y encountered, if $y|_{\mathbf{R}} \neq \emptyset$, we choose $\Gamma_R(y|_{\mathbf{R}}) \cup \Sigma_{uc}$ as the unique control decision defined at y. Note that $y|_{\mathbf{R}} \neq \emptyset$ implies that y can be reached by some string in $P(R)$, i.e., the supervisor is not sure whether or not the system has already gone outside the lower bound language R. Therefore, we choose $\Gamma_R(y|_{\mathbf{R}}) \cup \Sigma_{uc}$ as the control decision since it is the smallest control decision we need in order to contain R. If $y|_{\mathbf{R}} = \emptyset$, then we know for sure that the system has already gone outside R and we just choose Σ_{uc} as the control decision, i.e., all controllable events are disabled. To summarize the above rule, in the constructed BTS T_R, we have that

$$\forall y \in Q_Y^{T_R} : c_{T_R}(y) = \begin{cases} \Gamma_R(y|_{\mathbf{R}}) \cup \Sigma_{uc} & \text{if } y|_{\mathbf{R}} \neq \emptyset \\ \Sigma_{uc} & \text{if } y|_{\mathbf{R}} = \emptyset \end{cases}$$

Based on the above discussion, Algorithm INF-SYNT is proposed to construct T_R. Namely, for each Y-state encountered, we choose *one* control decision based on the above-discussed rules; for each Z-state encountered, we need to consider *all* observable events that are feasible. Such a depth-first search is implemented by the recursive procedure termed DoDFS. Moreover, Algorithm INF-SYNT returns "No Solution" when a Y-state y such that $\Gamma_R(y|_{\mathbf{R}}) \cup \Sigma_{uc} \notin C_{\mathcal{AIC}(\mathbf{G},\mathbf{K})}(y)$ is encountered. This implies that achieving the lower bound R will violate the safety specification, either immediately or unavoidably in the future. In this case, MPRCP has no solution. Of course, $R^{\downarrow CO}$ always exists, but our focus herein is on solving MPRCP. (If the focus is solely on the computation of $R^{\downarrow CO}$, then it suffices to set $K = \mathcal{L}(\mathbf{G})$ in the above development.)

Next, we first illustrate Algorithm INF-SYNT by an example. Then we prove its correctness.

Example 5: Let us return to the running example. The input parameters of Algorithm INF-SYNT are $\mathbf{R}$ and $\mathcal{AIC}(\mathbf{G}', \mathbf{K}')$ shown in Figs. 2(c) and 3(b), respectively. We start procedure DoDFS from the initial Y-state $y_0 = \{1\}$. Since $\{1\}|_{\mathbf{R}} \neq \emptyset$, we take control decision $\Gamma_R(\{1\}) \cup \Sigma_{uc} = \Sigma_{uc}$ (which is depicted as $\{\}$ in Fig. 3(b) for simplicity since all events in it are uncontrollable events), and move to the successor Z-state $(\{1, 2\}, \{\})$. Then we need to consider all possible event occurrences from this Z-state. If a occurs, then Y-state $\{3, 4'\}$ is reached. Since $\{3, 4'\}|_{\mathbf{R}} = \{3\} \neq \emptyset$, we need to take control decision $\Gamma_R(\{3\}) \cup \Sigma_{uc} = \{c_1\} \cup \Sigma_{uc}$. Similarly, we need to take control decision $\{c_2\} \cup \Sigma_{uc}$ if Y-state $\{3', 4\}$ is reached. The above procedure yields deterministic BTS T_R, which is the part highlighted in Fig. 3(b). ■

We now prove the correctness of Algorithm INF-SYNT. First, we show that, under the assumption that $\mathbf{R} \sqsubset \mathbf{K} \sqsubset \mathbf{G}$, Algorithm INF-SYNT will never return "No Solution" when a solution exists.

Algorithm 1: INF-SYNT.

```
input : $\mathbf{R}$ and $\mathcal{AIC}(\mathbf{G}, \mathbf{K})$
output: $T_R$.
1  $Q_Y^{T_R} \leftarrow \{y_0\}, Q_Z^{T_R} \leftarrow \emptyset$;
2  DoDFS($y_0, T_R$);
3  return $T_R$;
procedure DoDFS($y, T_R$);
4  if $y|_{\mathbf{R}} \neq \emptyset$ then
5      if $\Gamma_R(y|_{\mathbf{R}}) \cup \Sigma_{uc} \in C_{\mathcal{AIC}(\mathbf{G},\mathbf{K})}(y)$ then
6          $Act \leftarrow \Gamma_R(y|_{\mathbf{R}}) \cup \Sigma_{uc}$;
       else
7          return "No Solution";
   else
8      $Act \leftarrow \Sigma_{uc}$;
9  $z \leftarrow h_{YZ}(y, Act)$;
10 Add transition $y \xrightarrow{Act} z$ to $h_{YZ}^{T_R}$;
11 if $z \notin Q_Z^{T_R}$ then
12     $Q_Z^{T_R} \leftarrow Q_Z^{T_R} \cup \{z\}$;
13     for $\sigma \in \Sigma_o : h_{ZY}(z, \sigma)!$ do
14         $y' \leftarrow h_{ZY}(z, \sigma)$;
15         Add transition $z \xrightarrow{\sigma} y'$ to $h_{ZY}^{T_R}$;
16         if $y' \notin Q_Y^{T_R}$ then
17             $Q_Y^{T_R} \leftarrow Q_Y^{T_R} \cup \{y'\}$;
18             DoDFS($y', T_R$);
```

Theorem 2: Algorithm INF-SYNT returns "No Solution" if and only if $R^{\downarrow CO} \not\subseteq K$.

Proof: ($\Leftarrow$) By contraposition. Suppose that Algorithm INF-SYNT returns BTS T_R. Since S_{T_R} is an IS-based supervisor, for any $s \in P(\mathcal{L}(S_{T_R}/\mathbf{G}))$ such that $IS_{S_{T_R}}^Y(s)|_{\mathbf{R}} \neq \emptyset$, we have

$$S_{T_R}(s) = c_{T_R}(IS_{S_{T_R}}^Y(s)) = \Gamma_R(IS_{S_{T_R}}^Y(s)|_{\mathbf{R}}) \cup \Sigma_{uc}$$

Therefore, by Proposition 3, we know that $R \subseteq \mathcal{L}(S_{T_R}/\mathbf{G})$. Then, by the definition of $^{\downarrow CO}$, we know that $R^{\downarrow CO} \subseteq \mathcal{L}(S_{T_R}/\mathbf{G})$. Moreover, S_{T_R} is safe since it is an AIC-included supervisor, i.e., $\mathcal{L}(S_{T_R}/\mathbf{G}) \subseteq K$. Overall, we know that $R^{\downarrow CO} \subseteq K$.

($\Rightarrow$) By contradiction. Assume that Algorithm INF-SYNT returns "No Solution" but $R^{\downarrow CO} \subseteq K$. Therefore, we know that there exists a supervisor S such that $R \subseteq \mathcal{L}(S/\mathbf{G}) \subseteq K$ and there exists a sequence in the form of

$$y_0 \xrightarrow{\Gamma_R(y_0|_{\mathbf{R}}) \cup \Sigma_{uc}} z_1 \xrightarrow{\sigma_1} y_1 \ldots \xrightarrow{\Gamma_R(y_{n-1}|_{\mathbf{R}}) \cup \Sigma_{uc}} z_n \xrightarrow{\sigma_n} y_n \quad (9)$$

in procedure DoDFS in Algorithm INF-SYNT such that

1) $\forall i = 0, \ldots, n : y_i|_{\mathbf{R}} \neq \emptyset$; and
2) $\forall i = 0, \ldots, n-1 : \Gamma_R(y_i|_{\mathbf{R}}) \cup \Sigma_{uc} \in C_{\mathcal{AIC}(\mathbf{G},\mathbf{K})}(y_i)$; and
3) $\Gamma_R(y_n|_{\mathbf{R}}) \cup \Sigma_{uc} \notin C_{\mathcal{AIC}(\mathbf{G},\mathbf{K})}(y_n)$.

Next, we show by induction that, for any $i \geq 0$, we have that

$$y_i \subseteq IS_S^Y(\sigma_1 \ldots \sigma_i) \text{ and } y_i|_{\mathbf{R}} = IS_S^Y(\sigma_1 \ldots \sigma_i)|_{\mathbf{R}} \quad (10)$$

Clearly, the induction basis holds for $i = 0$, since $y_0 = IS_S^Y(\epsilon)$. Let us assume that Equation (10) holds for $i = k$; we need to show that Equation (10) holds for $i = k+1$. By definition, we know that

$$\begin{aligned} y_{k+1} = &\{x \in X : \exists x' \in y_k, \exists w \in ((\Gamma_R(y_k|_{\mathbf{R}}) \cup \Sigma_{uc}) \cap \Sigma_{uo})^* \\ &\text{s.t. } \delta(x', w\sigma_{k+1}) = x\} \qquad (11)\\ = &\{x \in X_R : \exists x' \in y_k|_{\mathbf{R}}, \exists w \in ((\Gamma_R(y_k|_{\mathbf{R}}) \cup \Sigma_{uc}) \cap \Sigma_{uo})^* \\ &\text{s.t. } \delta_R(x', w\sigma_{k+1}) = x\} \cup A_{G\setminus R} \qquad (12) \end{aligned}$$

where $A_{G\setminus R} \subseteq X \setminus X_R$. Note that the second equality is a consequence of the assumption that $\mathbf{R} \sqsubset \mathbf{G}$, since any string that leaves the state space of $\mathbf{R}$ must lead to a state in $X \setminus X_R$. Similarly, we can write

$$\begin{aligned} &IS_S^Y(\sigma_1 \dots \sigma_{k+1}) \\ = &\{x \in X : \exists x' \in IS_S^Y(\sigma_1 \dots \sigma_k), \exists w \in (S(\sigma_1 \dots \sigma_k) \cap \Sigma_{uo})^* \\ &\text{s.t. } \delta(x', w\sigma_{k+1}) = x\} \qquad (13)\\ = &\{x \in X_R : \exists x' \in IS_S^Y(\sigma_1 \dots \sigma_k)|_{\mathbf{R}}, \exists w \in (S(\sigma_1 \dots \sigma_k) \cap \Sigma_{uo})^* \\ &\text{s.t. } \delta_R(x', w\sigma_{k+1}) = x\} \cup B_{G\setminus R} \qquad (14)\\ = &\{x \in X_R : \exists x' \in IS_S^Y(\sigma_1 \dots \sigma_k)|_{\mathbf{R}}, \\ &\exists w \in ((\Gamma_R(IS_S^Y(\sigma_1 \dots \sigma_k)|_{\mathbf{R}}) \cup \Sigma_{uc}) \cap \Sigma_{uo})^* \\ &\text{s.t. } \delta_R(x', w\sigma_{k+1}) = x\} \cup B_{G\setminus R} \qquad (15)\\ = &\{x \in X_R : \exists x' \in y_k|_{\mathbf{R}}, \exists w \in ((\Gamma_R(y_k|_{\mathbf{R}}) \cup \Sigma_{uc}) \cap \Sigma_{uo})^* \\ &\text{s.t. } \delta_R(x', w\sigma_{k+1}) = x\} \cup B_{G\setminus R} \qquad (16) \end{aligned}$$

where $B_{G\setminus R} \subseteq X \setminus X_R$. Note that the second equality also comes from the assumption that $\mathbf{R} \sqsubset \mathbf{G}$. The third equality comes from the fact that, for any string

$$w \in ((S(\sigma_1 \dots \sigma_k) \setminus \Gamma_R(IS_S^Y(\sigma_1 \dots \sigma_k)|_{\mathbf{R}})) \cap \Sigma_{uo})^*$$

$\delta_R(x', w\sigma_{k+1})$ is not defined for $x' \in X_R$. Note that the last equality follows from the induction hypothesis that $y_k|_{\mathbf{R}} = IS_S^Y(\sigma_1 \dots \sigma_k)|_{\mathbf{R}}$.

Therefore, by Equations (12) and (16), we know that $y_{k+1}|_{\mathbf{R}} = IS_S^Y(\sigma_1 \dots \sigma_{k+1})|_{\mathbf{R}}$. Moreover, by the induction hypothesis and Proposition 3, we know

$$\Gamma_R(y_k|_{\mathbf{R}}) = \Gamma_R(IS_S^Y(\sigma \dots \sigma_k)|_{\mathbf{R}}) \subseteq S(\sigma_1 \dots \sigma_k). \qquad (17)$$

Since Equation (10) holds for $i = k$, combining Equations (11), (13) and (17) together, we obtain $y_{k+1} \subseteq IS_S^Y(\sigma_1 \dots \sigma_k \sigma_{k+1})$, i.e., Equation (10) holds for $i = k+1$.

Now, let us go back to the sequence in Equation (9). Since $\mathcal{L}(S/\mathbf{G}) \subseteq K$, by Theorem 1, we have $S(\sigma_1 \dots \sigma_n) \in C_{\mathcal{AIC}(\mathbf{G},\mathbf{K})}(IS_S^Y(\sigma_1 \dots \sigma_n))$. We have proved that $y_n \subseteq IS_S^Y(\sigma_1 \dots \sigma_n)$. Then, by Proposition 1, $S(\sigma_1 \dots \sigma_n) \in C_{\mathcal{AIC}(\mathbf{G},\mathbf{K})}(y_n)$. Moreover, since we have shown that $\Gamma_R(y_n|_{\mathbf{R}}) = \Gamma_R(IS_S^Y(\sigma_1 \dots \sigma_n)|_{\mathbf{R}})$, by Proposition 3, we have $\Gamma_R(y_n|_{\mathbf{R}}) \cup \Sigma_{uc} \subseteq S(\sigma_1 \dots \sigma_n)$. Then, by Proposition 1 again, we know that $\Gamma_R(y_n|_{\mathbf{R}}) \cup \Sigma_{uc} \in C_{\mathcal{AIC}(\mathbf{G},\mathbf{K})}(y_n)$. However, this is a contradiction. ■

Remark 6: Note that the "only if" part of the proof of the above theorem relies on the condition that $\mathbf{R} \sqsubset \mathbf{G}$. In fact, if $\mathbf{R} \sqsubset \mathbf{G}$ does not hold, then Algorithm INF-SYNT may return "No Solution" even when $R^{\downarrow CO} \subseteq K$. For example, let us use $\mathbf{R}$ and $\mathcal{AIC}(\mathbf{G},\mathbf{K})$ shown in Figs. 2(c) and (b), respectively, as the input parameters of Algorithm INF-SYNT, where $\mathbf{R}$ is a sub-automaton of $\mathbf{G}$ but not a *strict* sub-automaton. Then, after taking control decision Σ_{uc} at the initial state and observing event a, we will reach Y-state $\{3,4\}$. Since $\Gamma_R(\{3,4\}) \cup \Sigma_{uc} = \{c_1, c_2\} \cup \Sigma_{uc} \notin C_{\mathcal{AIC}(\mathbf{G},\mathbf{K})}(\{3,4\})$, Algorithm INF-SYNT returns "No Solution". However, a solution does exist since $R^{\downarrow CO} \subseteq K$. This highlights our earlier assertion that the strict sub-automaton condition plays an important role in the synthesis algorithm. ■

The next result reveals that the BTS returned by Algorithm INF-SYNT is indeed the one that realizes the infimal safe supervisor.

Theorem 3: Suppose that Algorithm INF-SYNT returns BTS T_R. Then $\mathcal{L}(S_{T_R}/\mathbf{G}) = R^{\downarrow CO}$.

Proof: We prove this by contradiction. Let us assume that $\mathcal{L}(S_{T_R}/\mathbf{G}) \neq R^{\downarrow CO}$. In the proof of Theorem 2, we have shown that $R \subseteq \mathcal{L}(S_{T_R}/\mathbf{G})$. Therefore, there exists a supervisor S' such that $R \subseteq \mathcal{L}(S'/\mathbf{G}) \subset \mathcal{L}(S_{T_R}/\mathbf{G})$. This implies that $\exists s \in P(\mathcal{L}(S'/\mathbf{G})) \cap P(\mathcal{L}(S_{T_R}/\mathbf{G}))$ such that $S'(s) \subset S_{T_R}(s)$. For string s, we have the following two cases.

Case 1: $IS_{S_{T_R}}^Y(s)|_{\mathbf{R}} = \emptyset$.

In this case, by Algorithm INF-SYNT, we know that $S_{T_R}(s) = c_{T_R}(IS_{S_{T_R}}^Y(s)) = \Sigma_{uc}$. However, it contradicts the fact that $S'(s) \subset S_{T_R}(s)$, since $S'(s)$ always contains Σ_{uc}.

Case 2: $IS_{S_{T_R}}^Y(s)|_{\mathbf{R}} \neq \emptyset$.

In this case, by Algorithm INF-SYNT, $S_{T_R}(s) = \Gamma_R(IS_{S_{T_R}}^Y(s)|_{\mathbf{R}}) \cup \Sigma_{uc}$. Since $R \subseteq \mathcal{L}(S'/\mathbf{G})$, by Proposition 3, we know that $\Gamma_R(IS_{S'}^Y(s)|_{\mathbf{R}}) \cup \Sigma_{uc} \subseteq S'(s)$. Moreover, by Proposition 2, we have $IS_{S'}^Y(s)|_{\mathbf{R}} = IS_{S_{T_R}}^Y(s)|_{\mathbf{R}} = y_R(s)$, since both S' and S_{T_R} contains R. This implies that $\Gamma_R(IS_{S_{T_R}}^Y(s)|_{\mathbf{R}}) \cup \Sigma_{uc} \subseteq S'(s)$. However, this also contradicts the fact that $S'(s) \subset S_{T_R}(s)$. ■

Remark 7: Although language-based formulas for $R^{\downarrow CO}$ were provided in [11], [17], the formula-based approach does not tell us directly *what is the right structure* and *how many states are required* to realize the supervisor achieving $R^{\downarrow CO}$. To the best of our knowledge, no constructive approach for $R^{\downarrow CO}$, in terms of supervisor, is provided in the literature. The results in this section not only provide a direct *constructive approach* to compute the infimal prefix-closed controllable and observable super-language, but also provides a new *structural property* about the corresponding infimal supervisor. In particular, we show that, when the state spaces have been properly refined, i.e., $\mathbf{R} \sqsubset \mathbf{K} \sqsubset \mathbf{G}$, 2^X is sufficient enough to represent this supervisor, i.e., the infimal supervisor can be written in the form of $S_{T_R} : 2^X \to \Gamma$. Moreover, the BTS T_R will be further used as a basis to synthesize a maximal safe supervisor containing R. This will be discussed in Section VI. ■

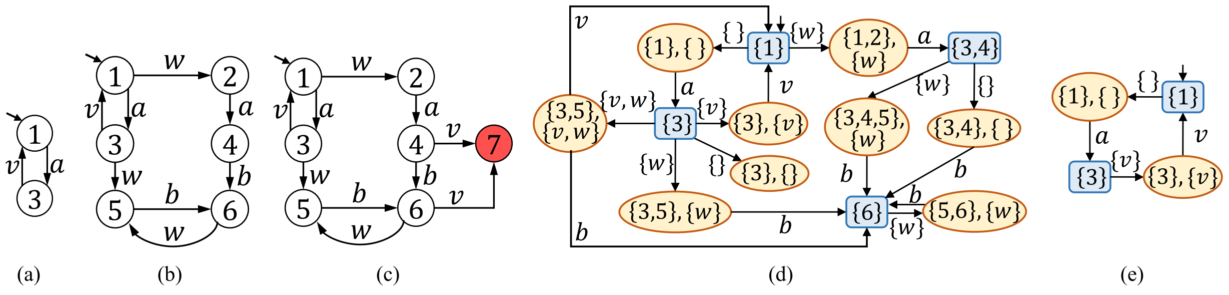


Fig. 4. For $\mathbf{R}$, $\mathbf{K}$ and $\mathbf{G}$: $\Sigma_c = \{v, w\}$ and $\Sigma_o = \{a, b, v\}$. (a) $\mathbf{R}$, (b) $\mathbf{K}$, (c) $\mathbf{G}$, (d) $\mathcal{AIC}(\mathbf{G}, \mathbf{K})$, (e) T_R.

V. Control Simulation Relation

In this section, we first discuss the difficulty that arises in solving the range control problem. Then we define the notion of Control Simulation Relation (CSR) as the tool to overcome the difficulty.

A. Difficulty in Handling the Lower Bound

In order to synthesize a maximal supervisor, the general idea is to guarantee by construction that the control decision made by the supervisor at each instant cannot be improved any further. However, this is not an easy task. Suppose that $y \in Q_Y^{AIC}$ is a Y-state in the AIC; by Theorem 1, we know that any control decision in $C_{\mathcal{AIC}(\mathbf{G},\mathbf{K})}(y)$ is a safe control decision. Therefore, if there is no lower bound requirement and one is only interested in the safety upper bound K, then we can simply pick a "greedy maximal" decision from $C_{\mathcal{AIC}(\mathbf{G},\mathbf{K})}(y)$. This is essentially the strategy we use in [27]; a similar strategy (but not based on the AIC) is used in [2]. However, the following example illustrates how to choose a control decision from $C_{\mathcal{AIC}(\mathbf{G},\mathbf{K})}(y)$ becomes much more complicated when the lower bound specification R has to be considered.

Example 6: Let us consider automata $\mathbf{R}$, $\mathbf{K}$ and $\mathbf{G}$ shown in Figs. 4(a), (b) and (c), respectively, where we have $\mathbf{R} \sqsubset \mathbf{K} \sqsubset \mathbf{G}$. Let $\Sigma_c = \{v, w\}$ and $\Sigma_o = \{a, b, v\}$. The AIC $\mathcal{AIC}(\mathbf{G}, \mathbf{K})$ is shown in Fig. 4(d). By applying Algorithm INF-SYNT, we construct BTS T_R that realizes the infimal supervisor achieving $R^{\downarrow CO}$; T_R is shown in Fig. 4(e). Initially, T_R chooses to disable w, i.e., $c_{T_R}(y_0) = \{\}$, while enabling w is also a safe choice at the initial Y-state according to the AIC. It seems that choosing $\{w\}$ provides more behavior than choosing $\{\}$. However, if we choose $\{w\}$, then upon the occurrence of a, we can only choose to disable v, since we are not sure whether the current state is 3 or 4. This leads to failure to contain the lower bound behavior $(av)^*$, where we need to enable v after observing a. Therefore, the lower bound behavior can only be achieved by choosing $\{\}$ at the beginning rather than choosing $\{w\}$, which is greedy maximal. ■

The above example illustrates the following issue. In some scenario, enabling more events is not a good choice, since it may introduce more information uncertainty. Consequently, to maintain safety, the control decision may become more conservative in the future due to this information uncertainty. This may make the lower bound behavior unachievable. More problematically, we do not know whether or not enabling an event will lead to failure to contain the lower bound behavior, unless we get stuck at some instant in the future, e.g., after observing event a in the previous example. Moreover, we do not know a priori, when or whether or not this phenomenon will occur in the future. In other words, whether or not a decision defined in the AIC is a "good" control decision depends on its effects in the future. This future dependency is the fundamental difficulty of the range control problem and it is in fact the essential difference between MPRCP and the standard supervisor synthesis problem without a lower bound requirement.

B. Definition of the CSR

In order to resolve the future dependency issue discussed above, we propose a simulation-like relation, called the Control Simulation Relation (CSR), to pre-process this future dependency and transform it to local information. The key idea is to compare two BTSs T_1 and T_2 and to establish a formal relationship between states in T_1 and states in T_2. The formal definition of the CSR is presented next.

Definition 4. (Control Simulation Relation) Let T_1 and T_2 be two BTSs w.r.t. the same $\mathbf{G}$. 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_1 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

$$F: 2^{Q_Y^{T_1} \times Q_Y^{T_2}} \cup 2^{Q_Z^{T_1} \times Q_Z^{T_2}} \rightarrow 2^{Q_Y^{T_1} \times Q_Y^{T_2}} \cup 2^{Q_Y^{T_1} \times Q_Y^{T_2}}$$

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 that

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 $(y_1, y_2) \in \Phi_Y$.

The following results reveal how operator F is related to the CSR.

Proposition 4: The operator F has following properties:

1) Φ 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) $\Phi_1 \subseteq \Phi_2 \Rightarrow F(\Phi_1) \subseteq F(\Phi_2)$.

Proof: See the Appendix. ■

The above results have the following implications. First, since $\Phi \subseteq F(\Phi)$ for any CSR Φ, we know that the maximal relation Φ 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 4, we know that F is monotone. Therefore, by Tarski's fixed-point theorem [20], we know that the supremal fixed-point of F, denoted by $\Phi^*(T_1, T_2)$, exists and it can be computed as follows

$$\Phi^*(T_1, T_2) = \lim_{k \to \infty} F^k((Q_Y^{T_1} \times Q_Y^{T_2}) \cup (Q_Z^{T_1} \times Q_Z^{T_2})) \quad (18)$$

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., [18]. Note that the limit in Equation (18) can be achieved within at most $|Q_Y^{T_1}||Q_Y^{T_2}| + |Q_Z^{T_1}||Q_Z^{T_2}|$ iterations.

Example 7: We consider again the AIC $\mathcal{AIC}(\mathbf{G}, \mathbf{K})$ and BTS T_R shown in Figs. 4(e) and 4(d), respectively. We compute the maximal CSR between T_R and $\mathcal{AIC}(\mathbf{G}, \mathbf{K})$; we write $\Phi^*(T_R, \mathcal{AIC}(\mathbf{G}, \mathbf{K})) = \Phi_Y^* \cup \Phi_Z^*$, where $\Phi_Y^* \subseteq Q_Y^{T_R} \times Q_Y^{AIC}$ and $\Phi_Z^* \subseteq Q_Z^{T_R} \times Q_Z^{AIC}$. Then we have

$$\Phi_Y^* = \{(\{1\}, \{1\}), (\{3\}, \{3\}))\}$$
$$\Phi_Z^* = \{((\{1\}, \{\}), (\{1\}, \{\})), ((\{3\}, \{v\}), (\{3\}, \{v\})), ((\{3\}, \{v\}), (\{3, 5\}, \{v, w\}))\}$$

The reason why $(\{3\}, \{3, 4\}) \notin \Phi_Y^*$ is that $\{v\}$ is defined at $\{3\}$ in T_R but there is no decision containing $\{v\}$ defined at $\{3, 4\}$ in the AIC. Consequently, we know that $((\{1\}, \{\}), (\{1, 2\}, \{w\})) \notin \Phi_Z^*$, where $(\{1\}, \{\})$ and $(\{1, 2\}, \{w\})$ are the predecessor Z-states that enter $\{3\}$ and $\{3, 5\}$ with the same event a, respectively. ■

C. Properties of the CSR

Hereafter, we present properties of the CSR that will be used later in the paper. Their proofs are provided in the Appendix.

The first result reveals that the CSR indeed captures whether or not any possible behavior from a state in a BTS can be matched by another BTS from some different state.

Proposition 5: Let T_1 and T_2 be two complete BTSs and $z_1 \in Q_Z^{T_1}$ and $z_1' \in Q_Z^{T_2}$ be two Z-states in T_1 and T_2, respectively. Then $(z_1, z_1') \in \Phi^*(T_1, T_2)$, if and only if, for any sequence

$$z_1 \xrightarrow{\sigma_1} y_1 \xrightarrow{\gamma_1} \dots \xrightarrow{\gamma_{n-1}} z_{n-1} \xrightarrow{\sigma_n} y_n \xrightarrow{\gamma_n} z_n \quad (19)$$

in T_1, there exists a sequence

$$z_1' \xrightarrow{\sigma_1} y_1' \xrightarrow{\gamma_1'} \dots \xrightarrow{\gamma_{n-1}'} z_{n-1}' \xrightarrow{\sigma_n} y_n' \xrightarrow{\gamma_n'} z_n' \quad (20)$$

in T_2, such that $\gamma_i \subseteq \gamma_i', \forall i \geq 0$.

The next result reveals the relationship between the CSR and the closed-loop behavior of the system.

Proposition 6: Let T_1 and T_2 be two deterministic BTSs. Then $\mathcal{L}(S_{T_1}/\mathbf{G}) \subseteq \mathcal{L}(S_{T_2}/\mathbf{G})$, if and only if, $T_1 \preceq T_2$.

The last result reveals that the CSR is transitive.

Proposition 7: Let T_1, T_2 and T_3 be three BTSs such that $T_1 \preceq T_2$ and $T_2 \preceq T_3$. For $i = 1, 2, 3$, let $y_i \in Q_Y^{T_i}$ and $\gamma_i \in C_{T_i}(y_i)$ be a Y-state in T_i and a control decision defined at this state, respectively. Then

$$[(z_1, z_2) \in \Phi^*(T_1, T_2) \wedge (z_2, z_3) \in \Phi^*(T_2, T_3)]$$
$$\Rightarrow [(z_1, z_3) \in \Phi^*(T_1, T_3)]$$

where $z_i = h_{YZ}(y_i, \gamma_i), i = 1, 2, 3$.

VI. Synthesis of a Maximally-Permissive Supervisor

In this section, we first present the main synthesis algorithm that solves MPRCP. Then we prove its correctness.

A. Synthesis Algorithm

As we discussed earlier, to synthesize a maximally permissive supervisor containing R, we need to consider some information in the future. Fortunately, such future information has been transformed to local information by the CSR. The idea of the synthesis algorithm is as follows. First, we construct BTS T_R that includes the infimal supervisor S_{T_R} achieving $R^{\downarrow CO}$. Then we compute the maximal CSR between BTS T_R and the AIC $\mathcal{AIC}(\mathbf{G}, \mathbf{K})$. Next, we construct a new BTS, denoted by T^*, such that $T_R \preceq T^*$, by using a depth-first search procedure. Specifically, suppose that y is a Y-state in T^* at which we need to choose a control decision. First, this control decision should be chosen from $C_{\mathcal{AIC}(\mathbf{G}, \mathbf{K})}(y)$ in order to guarantee safety. In order to take care of the lower bound behavior, we need to make sure that this control decision preserves the CSR. The reason why we consider the CSR between T_R and $\mathcal{AIC}(\mathbf{G}, \mathbf{K})$ is that T_R realizes the infimal supervisor containing R; namely, any BTS whose induced supervisor contains R should "simulate" the behavior of T_R.

In order to formalize the above idea, let $y \in Q_Y^{AIC}$ be a Y-state in the AIC and $\hat{y} \in Q_Y^{T_R}$ be a Y-state that "tracks" y in T_R such that $y|_\mathbf{R}, \hat{y}|_\mathbf{R} \neq \emptyset$. (How $\hat{y}$ "tracks" y will be clear later.) We denote by $\Phi_R^* := \Phi^*(T_R, \mathcal{AIC}(\mathbf{G}, \mathbf{K}))$ the maximal CSR from T_R to $\mathcal{AIC}(\mathbf{G},\mathbf{K})$. Then we define

$$\Xi(y,\hat{y}) := \left\{ \gamma \in \Gamma : \begin{array}{c} \gamma \in C_{\mathcal{AIC}(\mathbf{G},\mathbf{K})}(y) \textbf{ and } \gamma \supseteq c_{T_R}(\hat{y}) \textbf{and} \\ (h_{YZ}(\hat{y}, c_{T_R}(\hat{y})), h_{YZ}(y,\gamma)) \in \Phi_R^* \end{array} \right\}$$

Set $\Xi(y,\hat{y})$ will be the key in the synthesis algorithm. Intuitively, $\gamma \in \Xi(y,\hat{y})$ is a control decision such that:

1) It is safe at y, i.e., $\gamma \in C_{\mathcal{AIC}(\mathbf{G},\mathbf{K})}(y)$; and
2) It contains the corresponding control decision made by S_{T_R} at $\hat{y}$, i.e., $c_{T_R}(\hat{y})$; and
3) Any behavior that can occur from the corresponding Y-state $\hat{y}$ in T_R can still occur from y in the AIC by taking γ.

We are now ready to present the main synthesis algorithm, which is formally presented in Algorithm MAX-RANGE. Let us explain how it works. Initially, we construct T_R and compute the maximal CSR $\Phi^*(T_R, \mathcal{AIC}(\mathbf{G},\mathbf{K}))$ from T_R to $\mathcal{AIC}(\mathbf{G},\mathbf{K})$. Then we construct a new deterministic BTS T^* by a depth-first search as follows. Initially, we start from the initial Y-state y_0. We pick *one* control decision from the AIC for each Y-state encountered (how this control decision is picked will be specified soon) and pick *all* observations for each Z-state encountered. This depth-first search is implemented by recursive procedure DoDFS in Algorithm MAX-RANGE, which traverses the reachable state space of T^*. Moreover, during the construction BTS of T^*, we use T_R to track the sequence that reaches the Y- or Z-state in T^*. Specifically, whenever T^* moves from a Y-state y to a Z-state z (line 10), we need to move from Y-state $\hat{y}$ to its (unique) successor Z-state $\hat{z}$ in T_R (line 12). Similarly, whenever T^* moves from a Z-state z to a Y-state y via observable event σ (line 18), we need to move from Z-state $\hat{z}$ to a successor Y-state $\hat{y}$ in T_R through the same observable event σ (line 20). In other words, Y-state $\hat{y}$ in T_R essentially "tracks" Y-state y in the AIC or in T^*, since they are always reached by sequences that have the same projected string. Note that we use T_R to track T^* only when the current Y-state y encountered in T^* satisfies $y|_{\mathbf{R}} \neq \emptyset$. Whenever $y|_{\mathbf{R}} = \emptyset$, then we just set $\hat{y} = \emptyset$ (line 21). This means that we know for sure that the string is already outside of $\mathcal{L}(\mathbf{R})$.

Now it still remains to discuss how to choose the control decision at each Y-state in T^*. To this end, we need to consider two cases for each Y-state y encountered:

1) Suppose that $y|_{\mathbf{R}} \neq \emptyset$; this means that y must be reached by a sequence $y_0 \xrightarrow{\gamma_1\sigma_1\ldots\gamma_n\sigma_n} y$ such that the projected string in this sequence is in $P(R)$, i.e., $\sigma_1\ldots\sigma_n \in P(R)$. Then we know that there exists a sequence in T_R that "tracks" the above sequence, which means that $\hat{y} \neq \emptyset$. In this case, we choose a locally maximal decision in $\Xi(y,\hat{y})$, since we still need to be able to match any behavior in R in the future. This case is implemented by line 8 of Algorithm MAX-RANGE.

2) Suppose that $y|_{\mathbf{R}} = \emptyset$; this means that y must be reached by a sequence $y_0 \xrightarrow{\gamma_1\sigma_1\ldots\gamma_n\sigma_n} y$ such that $\sigma_1\ldots\sigma_n \notin P(R)$. This also implies that $\hat{y} = \emptyset$. Then we simply chose a locally maximal decision in $C_{\mathcal{AIC}(\mathbf{G},\mathbf{K})}(y)$, since we know for sure that the string is already outside of $\mathcal{L}(\mathbf{R})$. This case is implemented by line 9 of Algorithm MAX-RANGE.

Algorithm 2: MAX-RANGE.

```
input : R and $\mathcal{AIC}(\mathbf{G},\mathbf{K})$.
output: $T^*$.
1   $T_R \leftarrow$INF-SYNT($\mathbf{R}, \mathcal{AIC}(\mathbf{G},\mathbf{K})$) ;
2   $\Phi_R^* \leftarrow \Phi^*(T_R, \mathcal{AIC}(\mathbf{G},\mathbf{K}))$ ;
3   $Q_Y^{T^*} \leftarrow \{y_0\}, Q_Z^{T^*} \leftarrow \emptyset$;
4   DoDFS($y_0, y_0, T^*$);
5   return $T^*$;
6 procedure DoDFS($y, \hat{y}, T^*$);
7   if $y|_{\mathbf{R}} \neq \emptyset$ then
8       Find a locally maximal element $Act$ in
        $\Xi(y,\hat{y})$, i.e., $\forall \gamma \in \Xi(y,\hat{y}) : Act \not\subset \gamma$;
    else
9       Find a locally maximal element $Act$ in
        $C_{\mathcal{AIC}(\mathbf{G},\mathbf{K})}(y)$, i.e., $\forall \gamma \in C_{\mathcal{AIC}(\mathbf{G},\mathbf{K})} : Act \not\subset \gamma$;
10  $z \leftarrow h_{YZ}(y, Act)$;
11  if $\hat{y} \neq \emptyset$ then
12      $\hat{z} \leftarrow h_{YZ}(\hat{y}, c_{T_R}(\hat{y}))$;
    else
13      $\hat{z} \leftarrow \emptyset$;
14  Add transition $y \xrightarrow{Act} z$ to $h_{YZ}^{T^*}$;
15  if $z \notin Q_Z^{T^*}$ then
16      $Q_Z^{T^*} \leftarrow Q_Z^{T^*} \cup \{z\}$;
17      for $\sigma \in \Sigma_o : h_{ZY}(z,\sigma)!$ do
18          $y' \leftarrow h_{ZY}(z,\sigma)$;
19          if $\hat{z} \neq \emptyset$ and $h_{ZY}(\hat{z},\sigma)!$ then
20              $\hat{y}' \leftarrow h_{ZY}(\hat{z},\sigma)$;
            else
21              $\hat{y}' \leftarrow \emptyset$;
22          Add transition $z \xrightarrow{\sigma} y'$ to $h_{ZY}^{T^*}$;
23          if $y' \notin Q_Y^{T^*}$ then
24              $Q_Y^{T^*} \leftarrow Q_Y^{T^*} \cup \{y'\}$;
25              DoDFS($y', \hat{y}', T^*$);
```

Note that, in line 8 of Algorithm MAX-RANGE, the locally maximal element in $\Xi(y,\hat{y})$ may not be unique. If multiple locally maximal elements exist, then we will randomly choose one from them; which one to choose may depend on the specific implementation of the algorithm. However, we will show in Section VI-B that, no matter which maximal element we choose from $\Xi(y,\hat{y})$, our algorithm always guarantees that 1) the solution satisfies the range requirement; and 2) it cannot be improved anymore.

We illustrate Algorithm MAX-RANGE in the next example.

Example 8: Let us return to the system we have considered in Example 6. The inputs are BTS T_R that includes the infimal supervisor and the AIC $\mathcal{AIC}(\mathbf{G},\mathbf{K})$, which are shown in Figs. 4(e) and 4(d), respectively. We first start procedure DoDFS from the pair of initial Y-states, i.e., $y = \hat{y} = y_0 = \{1\}$. Since $((\{1\},\{\}),(\{1,2\},\{w\})) \notin \Phi_R^*$, we know that $\Xi(\{1\},\{1\}) = \{\emptyset\}$. Therefore, the only control decision we can choose is $\{\}$ and we have $z = \hat{z} = h_{YZ}(y,\{\}) = (\{1\},\{\})$. Then upon observing a, we reach new Y-states $y = \hat{y} = \{3\}$. This time we

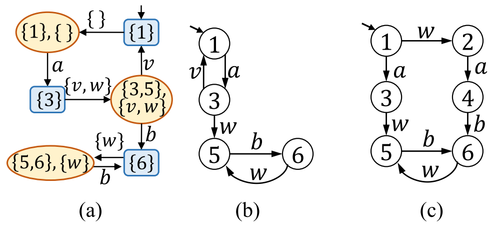


Fig. 5. Figures in Example 8. (a) T^*, (b) $\mathcal{L}(S_{T^*}/\mathbf{G})$, (c) Another maximal solution.

have $\Xi(\{3\},\{3\}) = \{\{w\},\{v,w\}\}$, since $(\{3\},\{v\})$ is related to both $(\{3\},\{v\})$ and $(\{3,5\},\{v,w\})$. Therefore, we choose $\{v,w\}$ at state $\{3\}$ in T^*. Then we move to $z = (\{3,5\},\{v,w\})$ and $\hat{z} = (\{3\},\{v\})$.

Now, from Z-state $(\{3,5\},\{v,w\})$, if event v occurs, T^* moves to Y-state $y = \{1\}$, which has already been visited. If event b occurs, T^* moves to Y-state $y = \{6\}$. However, T_R cannot track this move since b is not defined at $\hat{z} = (\{3\},\{v\})$ in T_R. Therefore, we set $\hat{y} = \emptyset$, which means that the string is already outside of R. Therefore, for Y-state $\{6\}$, we just choose a locally maximal control decision in $C_{\mathcal{AIC}}(\{6\})$, i.e., $\{w\}$, and move to $z = (\{5,6\},\{w\})$ and $\hat{z} = \emptyset$. Finally, by observing b again, T^* moves back to Y-state $\{6\}$ that has been visited. This completes the depth-first search and returns the deterministic BTS T^* shown in Fig. 5(a), which includes a supervisor S_{T^*} such that $R \subseteq \mathcal{L}(S_{T^*}/\mathbf{G}) \subseteq K$, where $\mathcal{L}(S_{T^*}/\mathbf{G})$ is shown in Fig. 5(b). (We will prove later that this supervisor is indeed maximal.) ■

Remark 8: One can verify that the language shown in Fig. 5(c) is a maximal controllable and observable sub-language of K. In fact, this solution is obtained by using the strategies proposed in [2], [27], i.e., we pick a locally maximal decision in $C_{\mathcal{AIC}(\mathbf{G},\mathbf{K})}(y)$ for each Y-state y and disregard the lower bound requirement. However, this solution does not fully contain R although it is maximal. ■

Note that, given arbitrary Y-states y and $\hat{y}$, set $\Xi(y,\hat{y})$ may be empty. For example, in Fig. 4, if we take $y = \{3,4\}$ and $\hat{y} = \{3\}$, then we know that $\Xi(y,\hat{y}) = \emptyset$, since $c_{T_R}(\hat{y}) = \{v\}$ but no control decision defined at y in the AIC contains $\{v\}$. If such a scenario occurs, then Algorithm MAX-RANGE may get stuck before it correctly returns T^*. However, the following result reveals that $\Xi(y,\hat{y})$ is always non-empty for any Y-states y and $\hat{y}$ encountered in Algorithm MAX-RANGE, i.e., the control decision Act in line 8 of Algorithm MAX-RANGE is always well-defined.

Proposition 8: For any Y-state y reached in procedure DoDFS, if $\hat{y} \neq \emptyset$, then $\Xi(y,\hat{y}) \neq \emptyset$. Moreover, $y|_{\mathbf{R}} = \hat{y}|_{\mathbf{R}}$.

Proof: We prove it by induction on the length of the sequence that reaches y in procedure DoDFS.

Induction Basis: The induction basis holds, since for the initial state, we have that $y_0|_{\mathbf{R}} = y_0$, i.e., $c_{T_R}(y_0|_{\mathbf{R}}) \in \Xi(y_0, y_0)$.

Induction Hypothesis: We assume that, for any Y-state reached by sequence in the form of

$$y_0 \xrightarrow{\gamma_0} z_1 \xrightarrow{\sigma_1} y_1 \ldots \xrightarrow{\gamma_{n-1}} z_n \xrightarrow{\sigma_n} y_n$$

in procedure DoDFS, if $\hat{y}_n \neq \emptyset$, we have $\Xi(y_n,\hat{y}_n) \neq \emptyset$ and $y_n|_{\mathbf{R}} = \hat{y}_n|_{\mathbf{R}}$.

Induction Step: To proceed, we show that, for any Y-state reached by sequence in the form of

$$y_0 \xrightarrow{\gamma_0} z_1 \xrightarrow{\sigma_1} y_1 \ldots \xrightarrow{\gamma_{n-1}} z_n \xrightarrow{\sigma_n} y_n \xrightarrow{\gamma_n} z_{n+1} \xrightarrow{\sigma_{n+1}} y_{n+1}$$

in procedure DoDFS, if $\hat{y}_{n+1} \neq \emptyset$, we have that $\Xi(y_{n+1}, \hat{y}_{n+1}) \neq \emptyset$ and $y_{n+1}|_{\mathbf{R}} = \hat{y}_{n+1}|_{\mathbf{R}}$, where $\hat{y}_{n+1}$ is the state reached by the following sequence in T_R

$$y_0 \xrightarrow{c_{T_R}(y_0)} \hat{z}_1 \xrightarrow{\sigma_1} \hat{y}_1 \xrightarrow{c_{T_R}(\hat{y}_1)}$$
$$\ldots \xrightarrow{c_{T_R}(\hat{y}_{n-1})} \hat{z}_n \xrightarrow{\sigma_n} \hat{y}_n \xrightarrow{c_{T_R}(\hat{y}_n)} \hat{z}_{n+1} \xrightarrow{\sigma_{n+1}} \hat{y}_{n+1}$$

First, we show that $y_{n+1}|_{\mathbf{R}} = \hat{y}_{n+1}|_{\mathbf{R}}$. To see this, we write

$$\begin{aligned} y_{n+1}|_{\mathbf{R}} &= \{x \in X : \exists x' \in y_n, \exists w\sigma_{n+1} \in \mathcal{L}(\mathbf{G}) \text{ s.t.}\\ &\qquad w \in (\gamma_n \cap \Sigma_{uo})^* \text{ and } \delta(x', w\sigma_{n+1}) = x\}|_{\mathbf{R}}\\ &= \{x \in X_R : \exists x' \in y_n|_{\mathbf{R}}, \exists w\sigma_{n+1} \in \mathcal{L}(\mathbf{R}) \text{ s.t.}\\ &\qquad w \in (\gamma_n \cap \Sigma_{uo})^* \text{ and } \delta(x', w\sigma_{n+1}) = x\}\\ &= \{x \in X_R : \exists x' \in \hat{y}_n|_{\mathbf{R}}, \exists w\sigma_{n+1} \in \mathcal{L}(\mathbf{R}) \text{ s.t.}\\ &\qquad w \in (c_{T_R}(\hat{y}_n) \cap \Sigma_{uo})^* \text{ and } \delta(x', w\sigma_{n+1}) = x\}\\ &= \{x \in X : \exists x' \in \hat{y}_n, \exists w\sigma_{n+1} \in \mathcal{L}(\mathbf{G}) \text{ s.t.}\\ &\qquad w \in (c_{T_R}(\hat{y}_n) \cap \Sigma_{uo})^* \text{ and } \delta(x', w\sigma_{n+1}) = x\}|_{\mathbf{R}}\\ &= \hat{y}_{n+1}|_{\mathbf{R}} \end{aligned}$$

The second and the fourth equalities follow from the assumption that $\mathbf{R} \sqsubset \mathbf{G}$, since any string that leaves the state space of $\mathbf{R}$ must lead to a state in $X \setminus X_R$. The third equality follows from the induction hypothesis that $y_n|_{\mathbf{R}} = \hat{y}_n|_{\mathbf{R}}$ and the fact that $\Gamma_R(\hat{y}_n|_{\mathbf{R}}) = c_{T_R}(\hat{y}_n) \subseteq \gamma_n$.

Next, we show that $\Xi(y_{n+1},\hat{y}_{n+1}) \neq \emptyset$. According to line 8 in Algorithm MAX-RANGE, γ_n is chosen such that $\gamma_n \in \Xi(y_n,\hat{y}_n)$. Note that $\Xi(y_n,\hat{y}_n)$ is non-empty by the induction hypothesis. Therefore, $c_{T_R}(\hat{y}_n) \subseteq \gamma_n$ and

$$(h_{YZ}(\hat{y}_n, c_{T_R}(\hat{y}_n)), h_{YZ}(y_n,\gamma_n)) \in \Phi_R^*$$

That is, $(\hat{y}_{n+1}, y_{n+1}) \in \Phi_R^*$. Therefore, for any sequence

$$\hat{y}_{n+1} \xrightarrow{c_{T_R}(\hat{y}_{n+1})} \hat{z}_{n+2} \xrightarrow{\sigma_{n+2}} \ldots \xrightarrow{c_{T_R}(\hat{y}_{n+k-1})} \hat{z}_{n+k}$$

in T_R, there exists a sequence

$$y_{n+1} \xrightarrow{\gamma_{n+1}} z_{n+2} \xrightarrow{\sigma_{n+2}} \ldots \xrightarrow{\gamma_{n+k-1}} z_{n+k}$$

in the AIC, such that $c_{T_R}(\hat{y}_{n+i}) \subseteq \gamma_{n+i}, \forall i \geq 1$. Hence, $\gamma_{n+1} \in C_{\mathcal{AIC}(\mathbf{G},\mathbf{K})}(y_{n+1})$ and $(\hat{z}_{n+2}, z_{n+2}) \in \Phi_R^*$, i.e.,

$$(h_{YZ}(\hat{y}_{n+1}, c_{T_R}(\hat{y}_{n+1})), h_{YZ}(y_{n+1},\gamma_{n+1})) \in \Phi_R^* \tag{21}$$

Therefore, $\gamma_{n+1} \in \Xi(y_{n+1},\hat{y}_{n+1})$, i.e., $\Xi(y_{n+1},\hat{y}_{n+1})$ is also non-empty. This completes the induction step. ■

Remark 9: Let us discuss the complexity of Algorithm MAX-RANGE. First, we need to construct the AIC, which takes $O(|X||\Sigma|2^{|X|+|\Sigma|})$ according to [27]. Then Algorithm INF-SYNT takes $O(|X||\Sigma|2^{|X|})$ to construct T_R, since there are at most $2^{|X|}$ Y-states and the same number of Z-states in T_R; for each Y-state it takes $O(|X||\Sigma|)$ to determine its control decision and for each Z-state it takes $O(|\Sigma|)$ to consider all possible observations. Computing the maximal CSR Φ_R^* takes $O(2^{2|X|+2|\Sigma|})$. For procedure DoDFS in Algorithm MAX-RANGE, it takes $O(2^{|\Sigma|})$ to determine control decision Act for each Y-state and it takes $O(|\Sigma|)$ to consider all observations for each Z-state. In the worst case, there are still $2^{|X|}$ Y-states and the same number of Z-states in T^*, which implies that procedure DoDFS takes $O(2^{|X|+|\Sigma|})$ to construct T^*. Therefore, the overall complexity of Algorithm MAX-RANGE is $O(2^{2|X|+2|\Sigma|})$, which is exponential w.r.t. $\mathbf{G}$. However, it is well-known that the supervisor synthesis problem under partial observation is NP-hard even without the lower bound requirement [22]. Therefore, it is highly unlikely that there exists a polynomial-time algorithm for MPRCP. Note that, under the assumption that $\mathbf{K} \sqsubset \mathbf{G}$, the complexity of computing a maximal solution without considering the lower bound is $O(|X||\Sigma|2^{|X|+|\Sigma|})$ [27]. Therefore, we do need to spend additional effort to guarantee the lower bound behavior. Also we note that, if Assumption 1 does not hold, then the refined system automaton may contain at most $|X'| := |X| \times (|X_K| + 1) \times (|X_R| + 1)$ states. In the case, the overall complexity becomes $O(2^{2|X'|+2|\Sigma|})$, which is still single exponential w.r.t. $\mathbf{G}$, $\mathbf{K}$ and $\mathbf{R}$. ■

B. Correctness of the Algorithm

In this section, we establish the correctness of Algorithm MAX-RANGE, i.e., it effectively solves MPRCP.

Hereafter, we still denote by T^* the BTS returned by Algorithm MAX-RANGE and denote by S_{T^*} the supervisor induced by T^*. First, we show that S_{T^*} is a safe supervisor.

Lemma 1: $\mathcal{L}(S_{T^*}/\mathbf{G}) \subseteq K$, i.e., S_{T^*} is safe.

Proof: This follows directly from Theorem 1. Since for each Y-state y encountered, $c_{T^*}(y)$ is chosen from $\Xi(y, \hat{y})$, which is a subset of $C_{\mathcal{AIC}(\mathbf{G},\mathbf{K})}(y)$. Therefore, S_{T^*} is an AIC-included supervisor, which means that it is safe.

Next, we show that language R is contained in $\mathcal{L}(S_{T^*}/\mathbf{G})$.

Lemma 2: $R \subseteq \mathcal{L}(S_{T^*}/\mathbf{G})$.

Proof: We use Proposition 3 to show that S_{T^*} contains R. Let us consider an arbitrary observable string $s \in P(\mathcal{L}(S_{T^*}/\mathbf{G}))$ s.t. $IS^Y_{S_{T^*}}(s)|_{\mathbf{R}} \neq \emptyset$. For simplicity, we denote $y = IS^Y_{S_{T^*}}(s)$. Since $y|_{\mathbf{R}} \neq \emptyset$, when y is reached for the first time in procedure DoDFS of Algorithm MAX-RANGE, i.e., when state y is added, it is reached by a sequence $y_0 \xrightarrow{\gamma_1\sigma_1\ldots\gamma_n\sigma_n} y$ in T^*, where $\sigma_1 \ldots \sigma_n \in P(R)$. Since $\sigma_1 \ldots \sigma_n \in P(R)$, there exists a corresponding sequence $y_0 \xrightarrow{\gamma_1'\sigma_1\ldots\gamma_n'\sigma_n} \hat{y}$ in T_R that tracks the above sequence leading to y in T^*, i.e., $\hat{y}$ is the Y-state that tracks y in the depth-first search. Note that $\sigma_1 \ldots \sigma_n$ need not be equal to s since there may exist multiple sequences that lead to y and the depth-first search just randomly picks one of them. Therefore, $\hat{y}$ may depend on the specific implementation of the depth-first search.

By Algorithm MAX-RANGE, $c_{T^*}(y)$ is chosen such that $c_{T_R}(\hat{y}) \subseteq c_{T^*}(y)$. By Algorithm INF-SYNT, $c_{T_R}(\hat{y})$ is chosen such that $\Gamma_R(\hat{y}|_{\mathbf{R}}) \cup \Sigma_{uc} = c_{T_R}(\hat{y})$. By Proposition 8, we know that $y|_{\mathbf{R}} = \hat{y}|_{\mathbf{R}}$. Moreover, S_{T^*} is an IS-based supervisor, which implies that $S_{T^*}(s) = c_{T^*}(y)$. Overall, we know that

$$\Gamma_R(IS^Y_{S_{T^*}}(s)|_{\mathbf{R}}) = \Gamma_R(\hat{y}|_{\mathbf{R}}) \subseteq c_{T_R}(\hat{y}) \subseteq c_{T_R}(y) = S_{T^*}(s).$$

Recall that s is an arbitrary string in $P(\mathcal{L}(S_{T^*}/\mathbf{G}))$. Therefore, by Proposition 3, we know that $R \subseteq \mathcal{L}(S_{T^*}/\mathbf{G})$. ■

Finally, we show that S_{T^*} is maximal.

Lemma 3: S_{T^*} is a maximally-permissive supervisor, i.e., for any safe supervisor S', $\mathcal{L}(S_{T^*}/\mathbf{G}) \not\subset \mathcal{L}(S'/\mathbf{G})$.

Proof: By contradiction. Assume that S_{T^*} is not maximal. This implies that there exists another safe supervisor S' such that $\mathcal{L}(S_{T^*}/\mathbf{G}) \subset \mathcal{L}(S'/\mathbf{G})$. This implies that

1) $\forall s \in \mathcal{L}(S_{T^*}/\mathbf{G}) : S_{T^*}(P(s)) \subseteq S'(P(s))$; and
2) $\exists s \in \mathcal{L}(S_{T^*}/\mathbf{G}) : S_{T^*}(P(s)) \subset S'(P(s))$.

Let us consider an observable string $t \in P(\mathcal{L}(S_{T^*}/\mathbf{G}))$ such that $S_{T^*}(t) \subset S'(t)$ and $\forall t' \in \overline{\{t\}} \setminus \{t\} : S_{T^*}(t') = S'(t')$. Then we have that $IS^Y_{S_{T^*}}(t) = IS^Y_{S'}(t)$; we call this Y-state y.

We claim that, for the above y and $S'(t)$, we have

$$(h_{YZ}(y, c_{T^*}(y)), h_{YZ}(y, S'(t))) \in \Phi^*(T^*, \mathcal{AIC}(\mathbf{G},\mathbf{K})) \quad (22)$$

Too see this, let us consider an arbitrary sequence

$$y \xrightarrow{c_{T^*}(y)} z_1 \xrightarrow{\sigma_1} y_1 \xrightarrow{c_{T^*}(y_1)} \ldots z_n \xrightarrow{\sigma_n} y_n \xrightarrow{c_{T^*}(y_n)} z_{n+1} \quad (23)$$

in T^*. Since, $\mathcal{L}(S_{T^*}/\mathbf{G}) \subseteq \mathcal{L}(S'/\mathbf{G})$, S' induces the following sequence

$$y \xrightarrow{S'(t)} z_1' \xrightarrow{\sigma_1} y_1' \xrightarrow{S'(t\sigma_1)} \ldots z_n' \xrightarrow{\sigma_n} y_n' \xrightarrow{S'(t\sigma_1\ldots\sigma_n)} z_{n+1}' \quad (24)$$

Since S' is a safe supervisor, by Theorem 1, S' is an AIC-included supervisor. This implies that the above sequence exists in the AIC. Hence, Equation (22) holds by Proposition 5.

Next, we consider two cases for this Y-state y to show the contradiction.

Case 1: $y|_{\mathbf{R}} = \emptyset$.

Since S' is a safe supervisor, by Theorem 1, $S'(t) \in C_{\mathcal{AIC}(\mathbf{G},\mathbf{K})}(y)$. Moreover, $c_{T^*}(y)$ is chosen as a maximal element in $C_{\mathcal{AIC}(\mathbf{G},\mathbf{K})}(y)$. Therefore, we obtain a contradiction immediately since $c_{T^*}(y) \subset S'(t)$ is not possible.

Case 2: $y|_{\mathbf{R}} \neq \emptyset$.

Suppose that Y-state y is reached, for the first time, by the following sequence

$$y_0 \xrightarrow{\gamma_1} z_1 \xrightarrow{\sigma_1} y_1 \xrightarrow{\gamma_2} \ldots \xrightarrow{\gamma_n} z_n \xrightarrow{\sigma_n} y \quad (25)$$

in procedure DoDFS in Algorithm MAX-RANGE. Since $y|_{\mathbf{R}} \neq \emptyset$, we know that $\sigma_1 \ldots \sigma_n \in P(R)$ and the following sequence, which tracks the sequence in Equation (25), is well-defined in T_R

$$y_0 \xrightarrow{c_{T_R}(y_0)} \hat{z}_1 \xrightarrow{\sigma_1} \hat{y}_1 \xrightarrow{c_{T_R}(\hat{y}_1)} \ldots \xrightarrow{c_{T_R}(\hat{y}_n)} \hat{z}_n \xrightarrow{\sigma_n} \hat{y} \quad (26)$$

Since $\mathcal{L}(S_{T_R}/\mathbf{G}) \subseteq \mathcal{L}(S_{T^*}/\mathbf{G})$, by Proposition 6, we know that $T_R \preceq T^*$. Therefore, by the definition of the CSR, Equations (25) and (26) imply that $(y, \hat{y}) \in \Phi^*(T_R, T^*)$. This further implies that

$$(h_{YZ}(\hat{y}, c_{T_R}(\hat{y})), h_{YZ}(y, c_{T^*}(y))) \in \Phi^*(T_R, T^*) \quad (27)$$

Overall, by Eqs. (22) and (27) and by Proposition 7, we get

$$(h_{YZ}(\hat{y}, c_{T_R}(\hat{y})), h_{YZ}(y, S'(t))) \in \Phi^*(T_R, \mathcal{AIC}(\mathbf{G},\mathbf{K}))$$

Note that we also have that $c_{T_R}(\hat{y}) \subseteq c_{T^*}(y) \subset S'(t)$ and $S'(t) \in C_{\mathcal{AIC}(\mathbf{G},\mathbf{K})}(y)$. Therefore, $S'(t) \in \Xi(y, \hat{y})$. However,

$c_{T^*}(y) \subset S'(t)$ is not possible, since $c_{T^*}(y)$ is chosen as a maximal decision in $\Xi(y, \hat{y})$. This is a contradiction. ■

Finally, combining Lemmas 1, 2 and 3 together, we have the following theorem.

Theorem 4: S_{T^*} is a maximally-permissive supervisor such that $R \subseteq \mathcal{L}(S_{T^*}/\mathbf{G}) \subseteq K$, i.e., Algorithm MAX-RANGE effectively solves MPRCP.

Since the resulting supervisor S_{T^*} is realized by BTS T^*, we also have the following corollary.

Corollary 1: S_{T^*} is an IS-based solution, which implies that the closed-loop language $\mathcal{L}(S_{T^*}/\mathbf{G})$ is regular.

Remark 10: We have shown that Algorithm MAX-RANGE solves MPRCP. In fact, it also solves the *maximal-permissiveness verification problem*. Specifically, suppose that there exists a given supervisor $S : P(\mathcal{L}(\mathbf{G})) \rightarrow \Gamma$ and we want to *verify* whether it is maximal or not. In this case, we can just set $R = \mathcal{L}(S/\mathbf{G})$ as the lower bound requirement and apply Algorithm MAX-RANGE to find a maximal safe supervisor S^* that contains R. If $\mathcal{L}(S/\mathbf{G}) = \mathcal{L}(S^*/\mathbf{G})$, then we know that the given supervisor S is already maximally permissive, since we cannot improve it any further. Otherwise, if $\mathcal{L}(S/\mathbf{G}) \subset \mathcal{L}(S^*/\mathbf{G})$, then we know that S is not maximal. To the best of our knowledge, the maximality verification problem was open in the literature; it is now solved as a special case of the synthesis problem. ■

VII. Conclusion

We have solved a generalized supervisor synthesis problem, called the range control problem, for partially-observed DES. We considered both a standard upper bound specification that describes the legal behavior and a lower bound specification that describes the desired behavior. We provided new information-state-based constructive approaches for computing both infimal and maximal supervisors satisfying these requirements. The proposed approach combines the three notions of AIC, strict sub-automaton, and CSR, in a novel manner; each of them plays a different role in the synthesis problem. This results in a "meaningful" maximally-permissive safe supervisor that contains a given behavior. An interesting future direction is to extend the results in this paper to the non-prefix-closed case.

Appendix

A. Proofs Not Contained in Main Body

Proof of Proposition 2

Proof: First, we show the first statement. By the definition of IS_S^Y, we have that

$$\begin{aligned} IS_S^Y(s) &= \{x \in X : \exists w \in \{\epsilon\} \cup (\mathcal{L}(S/\mathbf{G}) \cap \Sigma^*\Sigma_o) \\ &\qquad \text{s.t. } \delta(x_0, w) = x \wedge P(w) = s\} \\ &= \{x \in X_R : \exists w \in \{\epsilon\} \cup (R \cap \Sigma^*\Sigma_o) \\ &\qquad \text{s.t. } \delta_R(x_0, w) = x \wedge P(w) = s\} \cup A_{G\setminus R} \\ &= y_R(s) \cup A_{G\setminus R} \end{aligned}$$

where $A_{G\setminus R} = \{x \in X : \exists w \in \{\epsilon\} \cup ((\mathcal{L}(S/\mathbf{G}) \setminus R) \cap \Sigma^*\Sigma_o) \text{s.t. } \delta(x_0, w) = x \wedge P(w) = s\} \subseteq X \setminus X_R$. The reason why we know that $A_{G\setminus R}$ does not contain a state in $\mathbf{R}$ is that we have already assumed that $\mathbf{R}$ is a strict sub-automaton of both $\mathbf{K}$ and $\mathbf{G}$. Hence, any string that goes outside of R will not go back to the state space of $\mathbf{R}$. Therefore, we have that

$$IS_S^Y(s)|_{\mathbf{R}} = y_R(s)|_{\mathbf{R}} \cup A_{G\setminus R}|_{\mathbf{R}} = y_R(s) \qquad (28)$$

Next, we show the second statement. Let us consider two arbitrary strings $s, t \in P(\mathcal{L}(\mathbf{R}))$ such that $y_R(s) \neq y_R(t)$. By the first statement, we can write $IS_S^Y(s)$ in the form of $IS_S^Y(s) = y_R(s) \cup A_{G\setminus R}$, where $A_{G\setminus R} \subseteq X \setminus X_R$. Similarly, we can write $IS_S^Y(t)$ in the form of $IS_S^Y(t) = y_R(t) \cup B_{G\setminus R}$, where $B_{G\setminus R} \subseteq X \setminus X_R$. Note that $y_R(s), y_R(t) \subseteq X_R$. Therefore, since $y_R(s) \neq y_R(t)$, we have $IS_S^Y(s) \neq IS_S^Y(t)$. ■

Proof of Propositon 3

Proof: ($\Rightarrow$) By contradiction. Assume that $\exists s \in P(\mathcal{L}(S/\mathbf{G}))$ such that $IS_S^Y(s)|_{\mathbf{R}} \neq \emptyset$ and $\Gamma_R(IS_S^Y(s)|_{\mathbf{R}}) \not\subseteq S(s)$. Let σ be an event in $\Gamma_R(IS_S^Y(s)|_{\mathbf{R}}) \setminus S(s)$. By the definition of $\Gamma_R(\cdot)$, we have that $\exists x \in IS_S^Y(s)|_{\mathbf{R}}, \exists w \in \Sigma_{uo}^*$ s.t. $\delta_R(x, w\sigma)!$. Since $x \in IS_S^Y(s)|_{\mathbf{R}}$, there exists a string $t \in R$ such that $P(t) = s$ and $\delta_R(x_0, t) = x$, which implies that $tw\sigma \in R$. However, since $\sigma \notin S(s) = S(P(tw))$, we know that $tw\sigma \notin \mathcal{L}(S/\mathbf{G})$. This contradicts the fact that $R \subseteq \mathcal{L}(S/\mathbf{G})$.

($\Leftarrow$) It suffices to show that, $t \in R \Rightarrow t \in \mathcal{L}(S/\mathbf{G})$. We proceed by induction on the length of the projection of t.

Induction Basis: For string $t \in R$ such that $|P(t)| = 0$, we know that $t \in (\Gamma_R(\{x_0\}) \cap \Sigma_{uo})^* \cap R$. Since $\Gamma_R(\{x_0\}) = \Gamma_R(IS_S^Y(\epsilon)) \subseteq S(\epsilon)$, we know that $t \in (S(\epsilon) \cap \Sigma_{uo})^* \cap \mathcal{L}(\mathbf{G}) \subseteq \mathcal{L}(S/\mathbf{G})$, i.e., the induction basis holds.

Induction Hypothesis: Assume that $t \in R \Rightarrow t \in \mathcal{L}(S/\mathbf{G})$ for any t such that $|P(t)| = k$.

Induction Step: To prove the induction step, we show that $v\sigma w \in R \Rightarrow v\sigma w \in \mathcal{L}(S/\mathbf{G})$, where $|P(v)| = k, \sigma \in \Sigma_o$ and $w \in \Sigma_{uo}^*$. Note that any string t such that $|P(t)| = k + 1$ can be written in the above form. Let $v' \in \overline{\{v\}}$ be the longest prefix of v that ends up with an observable event and let $x = \delta(x_0, v') \in X_R$. Since $|P(v')| = |P(v)| = k$, by the induction hypothesis, we know that $v' \in \mathcal{L}(S/\mathbf{G})$. Therefore, $x = \delta(x_0, v') \in IS_S^Y(P(v))$. By the definition of $\Gamma_R(\cdot)$, all events between v' and $v\sigma$ are in $\Gamma_R(\{x\})$. Since $x \in X_R$ and $x \in IS_S^Y(P(v))$, we know that $IS_S^Y(P(v))|_{\mathbf{R}} \neq \emptyset$. Therefore, $\Gamma_R(\{x\}) \subseteq S(P(v))$, which implies that $v\sigma \in \mathcal{L}(S/\mathbf{G})$. Similarly, let $x' = \delta(x_0, v\sigma) \in X_R$. We know that $x' \in IS_S^Y(P(v)\sigma)$ and $IS_S^Y(P(v)\sigma)|_{\mathbf{R}} \neq \emptyset$. Again, since $\Gamma_R(\{x'\}) \subseteq S(P(v)\sigma)$, we have $s\sigma w \in \mathcal{L}(S/\mathbf{G})$. This completes the induction step. ■

Proof of Proposition 4

Proof: The proof is similar to the proof in [18] for the standard simulation relation. Suppose that $\Phi = \Phi_Y \cup \Phi_Z$ is a control simulation relation from T_1 to T_2. Let $(y_1, y_2) \in \Phi_Y$. Since $(\forall y_1 \xrightarrow{\gamma_1}_{T_1} z_1)(\exists y_2 \xrightarrow{\gamma_2}_{T_2} z_2)[\gamma_1 \subseteq \gamma_2 \wedge (z_1, z_2) \in \Phi_Z]$, we know that $(y_1, y_2) \in F(\Phi)$. Similarly, for any $(z_1, z_2) \in \Phi_Z$, since $(\forall z_1 \xrightarrow{\sigma}_{T_1} y_1)(\exists z_2 \xrightarrow{\sigma}_{T_2} y_2)[(y_1, y_2) \in \Phi_Y]$, we know that $(z_1, z_2) \in F(\Phi_Z)$. Therefore, we conclude that $\Phi \subseteq F(\Phi)$ and $(y_0, y_0) \in \Phi$.

Suppose that $\Phi \subseteq F(\Phi)$ and $(y_0, y_0) \in \Phi$. Clearly, the first requirement in Definition 4 is satisfied. For any $(y_1, y_2) \in \Phi_Y$, we know that the first requirement in the definition of F implies that second requirement in Definition 4. Similarly, for any $(z_1, z_2) \in \Phi_Z$, we know that the second requirement in the definition of F implies that third requirement in Definition 4. Hence, we know that Φ is a control simulation relation from T_1 to T_2.

Now we prove the second property. For any $(y_1, y_2) \in F(\Phi_1) \cap (Q_Y^{T_1} \times Q_Y^{T_2})$, we have that $(y_1, y_2) \in \Phi_1$ and

$$(\forall y_1 \xrightarrow{\gamma_1}_{T_1} z_1)(\exists y_2 \xrightarrow{\gamma_2}_{T_2} z_2)[\gamma_1 \subseteq \gamma_2 \wedge (z_1, z_2) \in \Phi_1] \quad (29)$$

Since $\Phi_1 \subseteq \Phi_2$, we know that $(y_1, y_2), (z_1, z_2) \in \Phi_2$. Therefore, Equation (29) implies that $(y_1, y_2) \in F(\Phi_2)$. Similarly, for any $(z_1, z_2) \in F(\Phi_1) \cap (Q_Z^{T_1} \times Q_Z^{T_2})$, we have that $(z_1, z_2) \in \Phi_1$ and $(\forall z_1 \xrightarrow{\sigma}_{T_1} y_1)(\exists z_2 \xrightarrow{\sigma}_{T_2} y_2)[(y_1, y_2) \in \Phi_1]$. Since $\Phi_1 \subseteq \Phi_2$, we also know that $(y_1, y_2), (z_1, z_2) \in \Phi_2$. Therefore, $(z_1, z_2) \in F(\Phi_2)$. ■

Proof of Proposition 5

Proof: The "only if" part is straightforward. For a sequence in Equation (19), we can always construct a sequence in Equation (20) by choosing γ_i' for each $i \geq 0$ such that $\gamma_i \subseteq \gamma_i'$ and $(h_{YZ}(y_i, \gamma_i), h_{YZ}(y_i', \gamma_i')) \in \Phi^*(T_1, T_2)$. The definition of the CSR guarantees the existence of such γ_i' at each y_i' encountered.

Next, we show the "if part" by contraposition. Suppose that $(z_1, z_1') \notin \Phi^*(T_1, T_2)$. Then, by Equation (18), either there exists an event $\sigma_1 \in \Sigma_o : h_{ZY}(z_1, \sigma_1)!$ but $\sigma_1 \in \Sigma_o : h_{ZY}(z_1, \sigma_1)\neg!$, where "$\neg!$" means "is not defined"; or there exists $\Phi_1 \supset \Phi^*(T_1, T_2)$ such that $(z_1, z_1') \in \Phi_1$ but

$$(\exists \sigma_1 \in \Sigma_o)[(y_1, y_1') \notin \Phi_1] \quad (30)$$

where $y_1 = h_{ZY}(z_1, \sigma_1)$ and $y_1' = h_{ZY}(z_1', \sigma_1)$.

For the first case, we know immediately that there exists a sequence $z_1 \xrightarrow{\sigma_1} y_1 \xrightarrow{\gamma_1} z_2$ in T_1, where γ_1 is an arbitrary control decision in $C_{T_1}(y_1)$, such that there does not exist a sequence $z_1' \xrightarrow{\sigma_1} y_1' \xrightarrow{\gamma_1'} z_2'$ in T_2 satisfying $\gamma_1 \subseteq \gamma_1'$. Hereafter, we consider the case where Equation (30) holds. Again, by Equation (18), $(y_1, y_1') \notin \Phi_1$ implies that either

$$(\exists \gamma_1 \in C_{T_1}(y_1))(\forall \gamma_1' \in C_{T_2}(y_1'))[\gamma_1 \not\subseteq \gamma_1'] \quad (31)$$

or there exists $\Phi_2 \supset \Phi_1$ such that $(y_1, y_1') \in \Phi_2$ but

$$(\exists \gamma_1 \in C_{T_1}(y_1)(\forall \gamma_1' \in C_{T_2}(y_1') : \gamma_1 \subseteq \gamma_1')[(z_2, z_2') \notin \Phi_2] \quad (32)$$

where $z_2 = h_{YZ}(y_1, \gamma_1)$ and $z_2' = h_{YZ}(y_1', \gamma_1')$.

Suppose that Equation (31) holds, then we also know immediately that there exists a sequence $z_1 \xrightarrow{\sigma_1} y_1 \xrightarrow{\gamma_1} z_2$ in T_1, such that there does not exist a sequence $z_1' \xrightarrow{\sigma_1} y_1' \xrightarrow{\gamma_1'} z_2'$ in T_2 satisfying $\gamma_1 \subseteq \gamma_1'$. Suppose that Equation (32) holds. Let $\gamma_1 \in C_{T_1}(y_1)$ be a control decision satisfying Equation (32) and let $\gamma_1' \in C_{T_2}(y_1')$ be an arbitrary control decision such that $\gamma_1 \subseteq \gamma_1'$. Note that $\gamma_1 \subseteq \gamma_1'$ implies that $\forall \sigma \in \Sigma_o : h_{ZY}^{T_1}(z_2, \sigma)! \Rightarrow h_{ZY}^{T_2}(z_2', \sigma)!$. Since $(z_2, z_2') \notin \Phi_2$, by Equation (18), $\exists \Phi_3 \supset \Phi_2$ such that $(z_2, z_2') \in \Phi_3$ but

$$(\exists \sigma_2 \in \Sigma_o)[(y_2, y_2') \notin \Phi_3] \quad (33)$$

where $y_2 = h_{ZY}(z_2, \sigma_2)$ and $y_2' = h_{ZY}(z_2', \sigma_2)$.

By iteratively applying the above arguments, suppose that, for some $m \geq 1$, we have that

$$(\exists \gamma_m \in C_{T_1}(y_m))(\forall \gamma_m' \in C_{T_2}(y_m'))[\gamma_m \not\subseteq \gamma_m'] \quad (34)$$

and

$$(\forall 1 \leq i \leq m)(\exists \Phi_{2i-1} \supset \Phi_{2i-2} : (z_i, z_i') \in \Phi_{2i-1})$$
$$(\exists \sigma_i \in \Sigma_o :)[(y_i, y_i') \notin \Phi_{2i-1}] \quad (35)$$

where $y_i = h_{ZY}(z_i, \sigma_i)$, $y_i' = h_{ZY}(z_i', \sigma_i)$ and $\Phi_0 = \Phi^*(T_1, T_2)$; and

$$(\forall 1 \leq i \leq m-1)(\exists \Phi_{2i} \supset \Phi_{2i-1} : (y_i, y_i') \in \Phi_{2i})(\exists \gamma_i \in C_{T_1}(y_i))$$
$$(\forall \gamma_i' \in C_{T_2}(y_i') : \gamma_i \subseteq \gamma_i')[(z_{i+1}, z_{i+1}') \notin \Phi_{2i}] \quad (36)$$

where $z_{i+1} = h_{YZ}(y_i, \gamma_i)$ and $z_{i+1}' = h_{YZ}(y_i', \gamma_i')$. In particular, Equations (34), (35), and (36) are the generalizations of Equations (31), (33) and (32), respectively. Then we know that there exists a sequence $z_1 \xrightarrow{\sigma_1} y_1 \xrightarrow{\gamma_1} \ldots \xrightarrow{\gamma_{m-1}} z_{m-1} \xrightarrow{\sigma_m} y_m \xrightarrow{\gamma_m} z_m$ in T_1 such that, for any sequence $z_1' \xrightarrow{\sigma_1} y_1' \xrightarrow{\gamma_1'} \ldots \xrightarrow{\gamma_{m-1}'} z_{m-1}' \xrightarrow{\sigma_m} y_m' \xrightarrow{\gamma_m'} z_m'$ in T_2, if $\gamma_i \subseteq \gamma_i', i = 0, \ldots, m-1$, then there does not exist a control decision $\gamma_m' \in C_{T_2}(y_m')$ such that $\gamma_m \subseteq \gamma_m'$. This completes the contrapositive proof.

Note that, since $\Phi_1 \subset \Phi_2 \subset \cdots \subset \Phi_{2m}$ is strictly increasing, such a m always exists. To see this, let $\Phi_{2m} = (Q_Y^{T_1} \times Q_Y^{T_2}) \cup (Q_Z^{T_1} \times Q_Z^{T_2})$, which is the largest possible relation. Suppose that for any $i < m$, there exists $\Phi_{2i} \supset \Phi_{2i-1}$ such that $(y_i, y_i') \in \Phi_{2i}$ but $(\exists \gamma_i \in C_{T_1}(y_i))(\forall \gamma_i' \in C_{T_2}(y_i') : \gamma_i \subseteq \gamma_i')[(z_{i+1}, z_{i+1}') \notin \Phi_{2i}]$ Then for Φ_{2m}, it must be $(\exists \gamma_m \in C_{T_1}(y_m))(\forall \gamma_m' \in C_{T_2}(y_m'))[\gamma_m \not\subseteq \gamma_m']$ since there does not exist a relation that is larger than Φ_{2m} anymore. ■

Proof of Proposition 6

Proof: ($\Rightarrow$) By contraposition. Suppose that $T_1 \not\preceq T_2$, which means that $(y_0, y_0) \notin \Phi^*(T_1, T_2)$. Therefore, either (i) $c_{T_1}(y_0) \not\subseteq c_{T_2}(y_0)$; or (ii) $(z_1^1, z_1^2) \notin \Phi^*(T_1, T_2)$, where $z_1^i = h_{YZ}(y_0, c_{T_i}(y_0)), i = 1, 2$. If case (i) holds, then we know immediately that $\mathcal{L}(S_{T_1}/\mathbf{G}) \not\subseteq \mathcal{L}(S_{T_2}/\mathbf{G})$, since $S_{T_1}(\epsilon) = c_{T_1}(y_0) \not\subseteq c_{T_2}(y_0) = S_{T_2}(\epsilon)$. If case (ii) holds, then by Proposition 5, there exists a string $\sigma_1 \ldots \sigma_n \in P(\mathcal{L}(S_{T_1}/\mathbf{G}))$ such that $S_{T_1}(\sigma_1 \ldots \sigma_i) \subseteq S_{T_2}(\sigma_1 \ldots \sigma_i), \forall i = 1, \ldots, n-1$ but $S_{T_1}(\sigma \ldots \sigma_n) \not\subseteq S_{T_2}(\sigma \ldots \sigma_n)$. Therefore, we still have that $\mathcal{L}(S_{T_1}/\mathbf{G}) \not\subseteq \mathcal{L}(S_{T_2}/\mathbf{G})$.

($\Leftarrow$) By contraposition. Suppose that $\mathcal{L}(S_{T_1}/\mathbf{G}) \not\subseteq \mathcal{L}(S_{T_2}/\mathbf{G})$. Then $\exists \sigma_1 \ldots \sigma_n \in P(\mathcal{L}(S_{T_1}/\mathbf{G}))$ such that $S_{T_1}(\sigma_1 \ldots \sigma_{n-1}) \subseteq S_{T_2}(\sigma_1 \ldots \sigma_{n-1})$ but $S_{T_1}(\sigma_1 \ldots \sigma_n) \not\subseteq S_{T_2}(\sigma_1 \ldots \sigma_n)$. Since T_1 is deterministic, the above string $\sigma_1 \ldots \sigma_n$ uniquely determines the following sequence in T_1

$$y_0 \xrightarrow{\gamma_0^1} z_1^1 \xrightarrow{\sigma_1} y_1^1 \xrightarrow{\gamma_1^1} \ldots \xrightarrow{\gamma_{n-1}^1} z_{n-1}^1 \xrightarrow{\sigma_n} y_n^1 \xrightarrow{\gamma_n^1} z_n^1 \quad (37)$$

where $\gamma_i^1 = S_{T_1}(\sigma_1 \ldots \sigma_i)$ is the unique control decision defined at y_i^1. However, there does not exist a sequence

$$y_0 \xrightarrow{\gamma_0^2} z_1^2 \xrightarrow{\sigma_1} y_1^2 \xrightarrow{\gamma_1^2} \ldots \xrightarrow{\gamma_{n-1}^2} z_{n-1}^2 \xrightarrow{\sigma_n} y_n^2 \xrightarrow{\gamma_n^2} z_n^2 \quad (38)$$

in T_2 such that $\gamma_i^1 \subseteq \gamma_i^2, \forall i = 1, \ldots, n$, since the control decision from each y_i^2 in T_2 is uniquely defined and the only control decision defined at y_n^2, i.e., $S_{T_2}(\sigma_1 \dots \sigma_n)$, does not contain $\gamma_n^1 = S_{T_1}(\sigma_1 \dots \sigma_n)$. ■

Proof of Proposition 7

Proof: Let $z_1 \xrightarrow{\sigma_1} y_1^1 \xrightarrow{\gamma_1^1} \dots \xrightarrow{\gamma_{n-1}^1} z_{n-1}^1 \xrightarrow{\sigma_n} y_n^1 \xrightarrow{\gamma_n^1} z_n^1$ be an arbitrary sequence in T_1. Since $(z_1, z_2) \in \Phi^*(T_1, T_2)$, by Proposition 5, there exists a sequence $z_2 \xrightarrow{\sigma_1} y_1^2 \xrightarrow{\gamma_1^2} \dots \xrightarrow{\gamma_{n-1}^2} z_{n-1}^2 \xrightarrow{\sigma_n} y_n^2 \xrightarrow{\gamma_n^2} z_n^2$ in T_2 such that $\gamma_i^1 \subseteq \gamma_i^2, \forall i = 1, \dots, n$. Similarly, since $(z_2, z_3) \in \Phi^*(T_2, T_3)$, by Proposition 5, there exists a sequence $z_3 \xrightarrow{\sigma_1} y_1^3 \xrightarrow{\gamma_1^3} \dots \xrightarrow{\gamma_{n-1}^3} z_{n-1}^3 \xrightarrow{\sigma_n} y_n^3 \xrightarrow{\gamma_n^3} z_n^3$ in T_3 such that $\gamma_i^2 \subseteq \gamma_i^3, \forall i = 1, \dots, n$. Therefore, $(z_1, z_3) \in \Phi^*(T_1, T_3)$ by Proposition 5. ■

REFERENCES

[1] A. Arnold, A. Vincent, and I. Walukiewicz, "Games for synthesis of controllers with partial observation," *Theoretical Computer Science*, vol. 303, no. 1, pp. 7–34, 2003.

[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 Dynamic Systems: Theory & Applications*, 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. Autom. Control*, vol. 60, no. 3, pp. 659–670, 2015.

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

[6] K. Chatterjee, L. Doyen, T. A. Henzinger, and J.-F. Raskin, "Algorithms for omega-regular games with imperfect information," In *Computer Science Logic*, pages 287–302. Springer: New York, 2006.

[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. Autom. Control*, vol. 33, no. 3, pp. 249–260, 1988.

[9] K. Inan, "Nondeterministic supervision under partial observations,". In *11th Int. Conf. Analy. Opt. Syst.: Discr. Event Syst.*, 39–48, 1994.

[10] S. Jiang, R. Kumar, S. Takai, and W. Qiu, "Decentralized control of discrete-event systems with multiple local specifications," *IEEE Trans. Autom. Sci. Eng.*, vol. 7, no. 3, pp. 512–522, 2010.

[11] R. Kumar and M. Shayman, "Formulae relating controllability, observability, and co-observability," *Automatica*, vol. 34, no. 2, pp. 211–215, 1998.

[12] O. Kupferman and M. Vardi, "Synthesis with incomplete informatio,". In *Advances in Temporal Logic*, pages 109–127. Springer, 2000.

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

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

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

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

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

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

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

[20] A. Tarski, "A lattice-theoretical fixpoint theorem and its applications," *Pacific J. Math.*, vol. 5, no. 2, pp. 285–309, 1955.

[21] J. G. Thistle and H. M. Lamouchi, "Effective control synthesis for partially observed discrete-event systems," *SIAM J. Control Optim.*, vol. 48, no. 3, pp. 1858–1887, 2009.

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

[23] X. Yin and S. Lafortune, "Supplement material," http://www-personal.umich.edu/˜xiangyin/TAC-range-sup.pdf.

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

[25] X. Yin and S. Lafortune, "On maximal permissiveness in partially-observed discrete event systems: Verification and synthesis," In *Proc. 13th Int. Workshop Discrete Event Syst.*, pages 1–7, 2016.

[26] X. Yin and S. Lafortune, "On maximally permissive range control problem in partially-observed discrete event systems," In *Proc. 55th IEEE Conf. Decision Control*, 2016.

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

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

[29] T.-S. Yoo and S. Lafortune, "Solvability of centralized supervisory control under partial observation," *Discrete Event Dynamic Systems: Theory & Applications*, vol. 16, no. 4, pp. 527–553, 2006.

Xiang Yin (S'16) was born in Anhui, China, in 1991. He received the B.Eng. degree from Zhejiang University, Zhejian, China, in 2012, the M.S. degree from the University of Michigan, Ann Arbor, in 2013, and the Ph.D. degree from the University of Michigan, Ann Arbor, in 2017, all in electrical engineering.

He is now with the Department of Automation, Shanghai Jiao-Tong University. His research interests include supervisory control of discrete-event systems, model-based fault diagnosis, formal methods, security and their applications to cyber and cyber-physical systems.

Dr. Yin received the Outstanding Reviewer Award from Automatica in 2016 and the IEEE Conference on Decision and Control (CDC) Best Student Paper Award Finalist in 2016. He is the co-chair of the IEEE CSS Technical Committee on Discrete Event Systems.

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

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

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