2016 IEEE 55th Conference on Decision and Control (CDC)
ARIA Resort & Casino
December 12-14, 2016, Las Vegas, USA

On the Maximally-Permissive Range Control Problem in Partially-Observed Discrete Event Systems

Xiang Yin and Stéphane Lafortune

***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).**

I. Introduction

We investigate the supervisor synthesis problem for partially-observed Discrete Event Systems (DES) in the framework of supervisory control theory [8]. In this problem, one is interested in synthesizing a supervisor such that the closed-loop system under control satisfies some requirement. 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* [6], [7]. Since controllability is preserved 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.

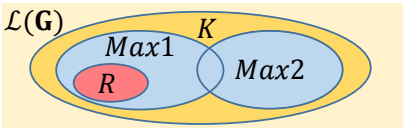


Fig. 1. Let $\mathbf{G}$ be the system, K be the legal behavior and R be the required behavior. $Max1$ and $Max2$ are two incomparable maximal solutions in K, i.e., $Max1 \not\subset Max2$ and $Max2 \not\subset Max1$. However, $Max1$ contains the required behavior R, while $Max2$ does not contain any string in R.

Many different approaches have been proposed in the literature to tackle the supervisor synthesis problem under partial observation; see, e.g., [1]–[3], [5], [10]–[12], [15], [16], [18]. One approach is to compute the supremal controllable and *normal* sub-language of K [2], [5]. However, since normality is stronger than observability, this solution is conservative in general. In [3], [10], two different solutions that are strictly more permissive than the supremal normal solution were proposed, respectively. However, these solutions are not maximal in general. In [1], an online approach was provided to synthesize a maximal solution for case of prefix-closed specifications. Recently, we showed in [15] that synthesizing a maximally-permissive safe and nonblocking supervisor is also decidable.

The algorithms in [1], [15] synthesize particular types of maximal solutions. 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 Figure 1. In fact, none of the synthesis algorithms in [1]–[3], [5], [10], [15], [18] can handle a minimal behavior requirement.

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 contain a given behavior. Namely, we want to find a "meaningful" maximal solution. 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 problem, we present a new synthesis algorithm based on the two notions of All Inclusive Controller (AIC) and Control Simulation Relation (CSR) proposed in our recent works [14], [15], respectively. In [15], the AIC is used to synthesize an arbitrary maximal solution, with no consideration to a lower bound behavior. In [14], the CSR

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

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

is used to *verify* whether a given supervisor is maximal or not. The present paper solves a supervisor synthesis problem, which is fundamentally more difficult than the verification problem studied in [14]. Throughout the paper, we only consider prefix-closed languages, i.e., nonblockingness is not considered. However, 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.

Due to space constraints, all proofs have been omitted and they are available in [17].

II. 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 $\overline{L} = L$. 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, $\delta : X \times \Sigma \to X$ is the partial transition function, $x_0 \in X$ is the initial state, and $X_m \subseteq X$ is the set of marked states. The transition function δ is extended to $X \times \Sigma^*$ in the usual manner; see, e.g., [4]. 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 of $\mathbf{B}$, denoted by $\mathbf{A} \sqsubset \mathbf{B}$, if (i) $\mathbf{A} \sqsubseteq \mathbf{B}$; and (ii) $\forall x, y \in X_A, \forall s \in \Sigma^* : \delta_B(x, s) = y \Rightarrow \delta_A(x, s) = y$.

In the supervisory control framework [8], 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 [6], [7], the event set is further partitioned into another two disjoint sets $\Sigma = \Sigma_o \dot\cup \Sigma_{uo}$, where Σ_o is the set of observable events and Σ_{uo} is the set of unobservable events. The natural projection $P : \Sigma^* \to \Sigma_o^*$ is defined in the usual manner; see, e.g., [4]. 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 which can be computed recursively by: i) $\epsilon \in \mathcal{L}(S/\mathbf{G})$; and ii) $s \in \mathcal{L}(S/\mathbf{G}) \wedge s\sigma \in \mathcal{L}(\mathbf{G}) \wedge \sigma \in S(P(s)) \Leftrightarrow s\sigma \in \mathcal{L}(S/\mathbf{G})$. We refer the reader to [4] for the definitions of *controllability* and *observability* in supervisory control theory.

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 all information states. Let $i \in I$ 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:
$\text{UR}_\gamma(i) = \{x \in X : \exists y \in i, \exists s \in (\Sigma_{uo} \cap \gamma)^* \text{ s.t. } x = \delta(y, s)\}$;
$\text{Next}_\sigma(i) = \{x \in X : \exists y \in i \text{ s.t. } x = \delta(y, \sigma)\}$.

III. 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 supervisor 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. We now formulate the *Maximally Permissive Range Control Problem* (MPRCP):

Problem 1: (Maximally-Permissive Range Control Problem). Given system G, lower bound language R and upper bound language K, 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. Specifically, it suffices to compute the *supremal controllable and normal sub-language* of K, denoted by $K^{\uparrow CN}$, 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 algorithms in [1], [15], since it suffices to synthesize an *arbitrary* maximal supervisor.
3) Finally, if the maximal permissiveness requirement is relaxed, then we just need to compute the *infimal prefix-closed controllable and observable super-language* of R (see, e.g., [9]), denoted by $R^{\downarrow CO}$, 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. However, to the best of our knowledge, MPRCP is still open for the general case, which is clearly more difficult than the above special cases.

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 also make the following assumptions:

A-1 $\mathbf{R} \sqsubset \mathbf{K} \sqsubset \mathbf{G}$; and

A-2 R is controllable and observable.

Assumption A-1 is w.l.o.g. since we can always refine the state spaces of $\mathbf{R}, \mathbf{K}$ and $\mathbf{G}$ such that A-1 holds; see, e.g., [5]. This assumption also says that legality of strings (w.r.t. K) is fully captured by states of $\mathbf{G}$. Namely, $X \setminus X_K$ is the set of illegal states and any string s leads to an illegal state iff $s \notin K$. Assumption A-2 is also w.l.o.g.: if R is not controllable or observable, then it suffices to compute $R^{\downarrow CO}$ to replace R, since any supervisor containing R must contain $R^{\downarrow CO}$. Under this assumption, we know that there exists a supervisor, denoted by S_R, such that $\mathcal{L}(S_R/\mathbf{G}) = R$. We assume that R is controllable and observable only to guarantee that supervisor S_R exists. Finally, let $y \in 2^X$ be an information state. We denote by $y|_{\mathbf{R}}$ the restriction of y to the state space of $\mathbf{R}$, i.e., $y|_{\mathbf{R}} = \{x \in X_R : x \in y\}$.

IV. 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 [13], [15] to solve supervisory control problems. For the sake of completeness of this paper, we review in this section key definitions and results from [15].

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

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

where $Q_Y^T \subseteq I = 2^X$ 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

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

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

$$h_{ZY}^T(z, \sigma) = y \Leftrightarrow \sigma \in \Gamma(z) \wedge y = \text{Next}_\sigma(I(z)); \tag{3}$$

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

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 (3). 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 $y \in Q_Y^T$, i.e., $C_T(y) = \{c_T(y)\}$.

For simplicity, we also write $y \xrightarrow{\gamma}_T z$ if $z = h_{YZ}^T(y, \gamma)$ and $z \xrightarrow{\sigma}_T y$ if $z = h_{ZY}^T(z, \sigma)$. 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 $\gamma_0 \sigma_1 \gamma_1 \sigma_2 \dots \sigma_n \gamma_n$, where $\gamma_i \in \Gamma, \sigma_i \in \Sigma_o$, a *run*. A run also induces a *sequence*

$$y_0 \xrightarrow{\gamma_0} z_0 \xrightarrow{\sigma_1} y_1 \xrightarrow{\gamma_1} \dots \xrightarrow{\gamma_{n-1}} z_{n-1} \xrightarrow{\sigma_n} y_n \xrightarrow{\gamma_n} z_n \tag{4}$$

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. Then for any observed string $s = \sigma_1 \dots \sigma_n \in P(\mathcal{L}(S/\mathbf{G}))$, it induces a well-defined sequence

$$y_0 \xrightarrow{S(\epsilon)} z_0 \xrightarrow{\sigma_1} y_1 \xrightarrow{S(\sigma_1)} \dots \xrightarrow{\sigma_n} y_n \xrightarrow{S(\sigma_1 \dots \sigma_n)} z_n \tag{5}$$

We denote by $IS_S^Y(s)$ and $IS_S^Z(s)$, the last Y-state and Z-state in $y_0 z_0 y_1 z_2 \dots z_{n-1} y_n z_n$, 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.

Next, we define the supervisors included in a BTS.

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.

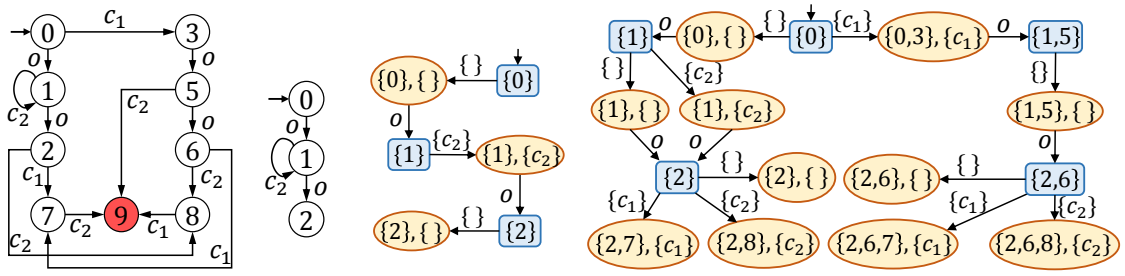
In [15], the AIC structure is defined as the largest BTS including only safe supervisors.

Definition 3: (All Inclusive Controller). The All Inclusive Controller for $\mathbf{G}$, $\mathcal{AIC}(\mathbf{G}) = (Q_Y^{AIC}, Q_Z^{AIC}, h_{YZ}^{AIC}, h_{ZY}^{AIC}, \Sigma_o, \Gamma, y_0)$, 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 [15] for more details and the construction of the AIC. Here we recall the key property of the AIC from [15].

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

Note that, 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 issue different control decisions at different visits to the same Y-state. We say that a supervisor S is *information-state-based* (IS-based) if for any strings $s, t \in P(\mathcal{L}(S/\mathbf{G}))$, we have that $IS_S^Y(s) = IS_S^Y(t) \Rightarrow S(s) = S(t)$. Then, a supervisor can be realized by a BTS iff it is IS-based. In [14], we showed that, under the assumption that $\mathbf{R} \sqsubseteq \mathbf{G}$, S_R is IS-based. Therefore, we know that the supervisor achieving the lower



(a) **G** and **K** (b) **R** (c) T_R (d) $\mathcal{AIC}(\mathbf{G})$

Fig. 2. In the diagrams, rectangular states denote Y-states and oval states denote Z-states. Uncontrollable events are omitted in each control decision. We also omit redundant events in each control decision, namely, events that are not feasible within the unobservable reach, e.g., event c_1 at Y-state $\{1\}$.

bound language can be realized by a BTS; we denote this BTS by T_R, i.e., $\mathbb{S}(T_R) = \{S_R\}$ and $\mathcal{L}(S_R/\mathbf{G}) = R$.

Example 1: Let us consider the system **G** shown in Figure 2(a). The upper bound automaton **K** is obtained by removing the single illegal state 9 from **G**. Let $\Sigma_c = \{c_1, c_2\}$ and $\Sigma_o = \{o\}$. Then the AIC $\mathcal{AIC}(\mathbf{G})$ is shown in Figure 2(d). From the initial Y-state $y_0 = \{0\}$, we can make control decision $\{c_1\}$, which leads to a Z-state via $\{0\} \xrightarrow{\{c_1\}} (\{0,3\},\{c_1\})$. By observing event o from this Z-state, we move to the next Y-state by $(\{0,3\},\{c_1\}) \xrightarrow{o} \{1,5\}$. Note that we cannot make control decision $\{c_2\}$ from Y-state $\{1,5\}$, since this will unobservably lead to illegal state 9. This is why $\{c_2\}$ is not defined at $\{1,5\}$ in $\mathcal{AIC}(\mathbf{G})$. Next, let us consider a lower bound specification R, which is generated by automaton **R** shown in Figure 2(b). We see that $\mathbf{R} \sqsubset \mathbf{K} \sqsubset \mathbf{G}$. The BTS T_R that includes the supervisor achieving R is shown in Figure 2(c). Clearly, both T_R and $\mathcal{AIC}(\mathbf{G})$ are complete. Moreover, T_R is deterministic, since the control decision defined at each Y-state in it is unique.

V. Control Simulation Relation

In [14], the notion of *Control Simulation Relation* (CSR) was defined in order to *verify* whether a supervisor is maximal or not. We will use this tool in order to *synthesize* a maximal solution.

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

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

We denote by $T_1 \preceq T_2$ the case where there exists a CSR from T_1 to T_2.

In order to check whether or not $T_1 \preceq T_2$, we define the following operator

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

by, for any $\Phi = \Phi_Y \cup \Phi_Z \subseteq (Q_Y^{T_1} \times Q_Y^{T_2}) \cup (Q_Z^{T_1} \times Q_Z^{T_2})$,

1. $(y_1, y_2) \in F(\Phi)$ 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. $(z_1, z_2) \in F(\Phi)$ 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 $(y_1, y_2) \in \Phi_Y$.

We showed in [14] that the supremal fixed-point of F, denoted by $\Phi^*(T_1, T_2)$, exists and it can be computed by

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

Therefore, $T_1 \preceq T_2$, if and only if, $(y_0, y_0) \in \Phi^*(T_1, T_2)$ and $\Phi^*(T_1, T_2)$ is the maximal CSR from T_1 to T_2.

It was shown in [14] that the CSR can be used to *verify* maximality, which is stated by the following result.

Theorem 2: ([14]). Let S be an IS-based supervisor and T be the deterministic BTS such that $\mathbb{S}(T) = \{S\}$. Then S is not maximal iff there exists a decision $\gamma \in C_{\mathcal{AIC}(\mathbf{G})}(y)$ such that $c_T(y) \subset \gamma$ and $(z, z') \in \Phi^*(T, \mathcal{AIC}(\mathbf{G}))$, where $z = h_{YZ}(y, c_T(y))$ and $z' = h_{YZ}(y, \gamma)$.

Example 2: Let us consider the BTS T_R and the AIC $\mathcal{AIC}(\mathbf{G})$ shown in Figures 2(c) and 2(d), respectively. Clearly, $(\{1\},\{1,5\}) \notin \Phi^*(T_R, \mathcal{AIC}(\mathbf{G}))$, since decision $\{c_1\}$ is defined at $\{1\}$ but there is no decision containing $\{c_1\}$ defined at $\{1,5\}$. By Theorem 2, we know that S_R is not maximal, since for Y-state $\{2\}$, we can find a control decision $\{c_2\} \in C_{\mathcal{AIC}(\mathbf{G})}(\{2\})$, such that $\{c_2\} \supset c_{T_R}(\{2\}) = \{\}$ and $((\{2\},\{\}),(\{2,8\},\{c_2\})) \in \Phi^*(T_R, \mathcal{AIC}(\mathbf{G}))$.

VI. Synthesis of a Maximal Solution

In this section, we present a new synthesis algorithm that solves MPRCP. First, we briefly discuss the main difficulty that arises in solving the range control problem and our approach to overcome it.

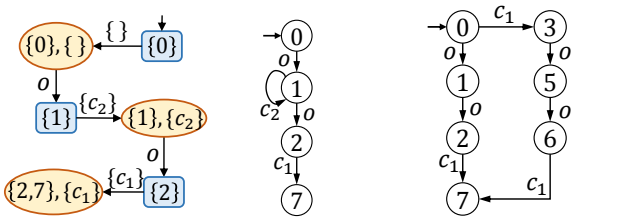
In order to synthesize a maximal supervisor, the general idea is to guarantee by construction that the control decision made by the supervisor at each instance cannot be improved any further. However, this is not an easy task. Suppose that $y \in Q_Y^{AIC}$ is a Y-state in the AIC; by Theorem 1, we know that any control decision in $C_{\mathcal{AIC}(\mathbf{G})}(y)$ 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 $C_{\mathcal{AIC}(\mathbf{G})}(y)$. This is essentially the "greedy" strategy we use in [15]. Suppose that $C_{\mathcal{AIC}(\mathbf{G})}(y)$ only contains two control

decisions and one is strictly larger than the other one, i.e., $C_{\mathcal{AIC}(\mathbf{G})}(y) = \{\gamma_1, \gamma_2\}$ and $\gamma_1 \subset \gamma_2$. If we are just interested in finding a maximal safe supervisor without considering R, we need to choose γ_2, since the resulting supervisor is strictly more permissive than that obtained by choosing γ_1. However, how to choose a control decision from $C_{\mathcal{AIC}(\mathbf{G})}(y)$ becomes much more complicated when the lower bound specification R has to be considered. For example, in our running example shown in Figure 2, there are two choices at the initial Y-state $y_0 = \{0\}$: enable c_1 or disable c_1. It seems that choosing $\{c_1\}$ provides more behavior than choosing $\{\}$. However, if we choose $\{c_1\}$, then upon the occurrence of o, we can only choose to disable c_2, since we are not sure whether the current state is 1 or 5. This fails to keep containing the lower bound behavior R, since we need to enable c_2 after o. Therefore, the lower bound behavior can only be achieved by choosing $\{\}$ at the beginning rather than choosing $\{c_1\}$, which is greedy maximal.

The above discussion 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 instance in the future, e.g., after observing event o 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.

Fortunately, we can use the CSR to pre-process this future dependency and transform it to purely local information. Since it is required that the synthesized supervisor contains the lower bound language R, we first compute the maximal CSR between BTS T_R and the AIC $\mathcal{AIC}(\mathbf{G})$. Recall that T_R is the deterministic BTS that includes the supervisor S_R achieving the lower bound R. Then we construct a new BTS, denoted by T^*, such that $T_R \preceq T^*$. Now, suppose that y is an information state at which we need to choose a control decision. First, this control decision should be chosen from $C_{\mathcal{AIC}(\mathbf{G})}(y)$ 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 key question is: Which CSR we should use, since we are constructing a new BTS? We will show that looking at the CSR between T_R and $\mathcal{AIC}(\mathbf{G})$ is sufficient and there is not need to compute the CSR between the newly constructed BTS and the AIC.

In order to formalize the above idea, let $y \in Q_Y^{AIC}$ be a Y-state such that $y|_{\mathbf{R}} \neq \emptyset$ and $\Phi_R^* := \Phi^*(T_R, \mathcal{AIC}(\mathbf{G}))$ be



(a) T^* (b) $\mathcal{L}(S_{T^*}/\mathbf{G})$ (c) Another maximal solution

Fig. 3. Figures in Example 3.

the maximal CSR from T_R to $\mathcal{AIC}(\mathbf{G})$. Then we define

$$\Xi(y) := \left\{ \gamma \in \Gamma : \begin{array}{l} \gamma \in C_{\mathcal{AIC}(\mathbf{G})}(y) \textbf{ and } \gamma \supseteq c_{T_R}(y|_{\mathbf{R}}) \textbf{ and} \\ (h_{YZ}(y|_{\mathbf{R}}, c_{T_R}(y|_{\mathbf{R}})), h_{YZ}(y, \gamma)) \in \Phi_R^* \end{array} \right\} \tag{8}$$

Intuitively, $\gamma \in \Xi(y)$ is a safe control decision at y such that 1) it contains the corresponding control decision made by S_R, i.e., $c_{T_R}(y|_{\mathbf{R}})$; and 2) any behavior that can occur from the corresponding Y-state $y|_{\mathbf{R}}$ in T_R can still occur by taking γ from y in the AIC.

Based on the above discussion, we present Algorithm MAX-RANGE to solve MPRCP. Initially, we compute the maximal CSR $\Phi^*(T_R, \mathcal{AIC}(\mathbf{G}))$ from T_R to $\mathcal{AIC}(\mathbf{G})$. Then we construct a new deterministic BTS T^* by a depth-first search, which is implemented by the recursive procedure called DoDFS. Procedure DoDFS picks *one* control decision for each Y-state encountered and picks *all* observations for each Z-state encountered. More specifically, for each Y-state y, if $y|_{\mathbf{R}} = \emptyset$, then we just chose a locally maximal decision in $C_{\mathcal{AIC}(\mathbf{G})}$, since we know for sure that the string is already outside of $\mathcal{L}(\mathbf{R})$; if $y|_{\mathbf{R}} \neq \emptyset$, then we choose a locally maximal decision in $\Xi(y)$, since we still need to be able to match any behavior in R in the future. The entire complexity of Algorithm MAX-RANGE is $O(2^{2^{|X|+|\Sigma|}})$, since it takes $O(2^{2^{|X|+|\Sigma|}})$ to compute $\Phi^*(T_R, \mathcal{AIC}(\mathbf{G}))$ and takes $O(2^{|X|+|\Sigma|})$ to implement the depth-first search.

Example 3: Let us return to the running example. The inputs T_R and $\mathcal{AIC}(\mathbf{G}')$ are shown in Figs. 2(c) and 2(d), respectively. We first start procedure DoDFS from the initial Y-state $y_0 = \{0\}$. Since $((\{0\}, \{\}), (\{0, 3\}, \{c_1\})) \notin \Phi_R^*$, we know that $\Xi(\{0\}) = \{\emptyset\}$. Therefore, the only control decision we can choose is $\{\}$. Then upon observing o, we reach Y-state $\{1\}$. This time we still have $\Xi(\{1\}) = \{\{c_2\}\}$, which means that $\{c_2\}$ is still the only choice. Finally, by observing o again, Y-state $\{2\}$ is encountered. At this state, we have $\Xi(\{2\}) = \{\emptyset, \{c_1\}, \{c_2\}\}$. We can choose either $\{c_1\}$ or $\{c_2\}$; let us choose $\{c_1\}$. This completes the depth-first search and returns the deterministic BTS T^* shown in Figure 3(a), which includes 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 Figure 3(b). (We will establish later that this supervisor is indeed maximal.) One can also verify that the solution shown in Figure 3(c) is also a maximal solution. In fact, this solution is obtained by using the "greedy maximal" strategy proposed in [1], [15], i.e., we pick a locally maximal decision in $C_{\mathcal{AIC}}(y)$ for each Y-state y and disregard the lower bound requirement. However, this solution does not fully contain R although it is maximal.

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

1 $Q_Y^{T^*} \leftarrow \{y_0\}, Q_Z^{T^*} \leftarrow \emptyset$;
2 DoDFS(y_0, T^*);
3 **return** T^*;

procedure DoDFS(y, T^*);
4 **if** $y|_{\boldsymbol{R}} \neq \emptyset$ **then**
5 Find a locally maximal element Act in $\Xi(y)$, i.e., $\forall \gamma \in \Xi(y) : Act \not\subset \gamma$;
else
6 Find a locally maximal element Act in $C_{\mathcal{AIC}(\mathbf{G})}(y)$, i.e., $\forall \gamma \in C_{\mathcal{AIC}(\mathbf{G})} : Act \not\subset \gamma$;
7 $z \leftarrow h_{YZ}(y, Act)$;
8 Add transition $y \xrightarrow{Act} z$ to $h_{YZ}^{T^*}$;
9 **if** $z \notin Q_Z^{T^*}$ **then**
10 $Q_Z^{T^*} \leftarrow Q_Z^{T^*} \cup \{z\}$;
11 **for** $\sigma \in \Sigma_o : h_{ZY}(z, \sigma)!$ **do**
12 $y' \leftarrow h_{ZY}(z, \sigma)$;
13 Add transition $z \xrightarrow{\sigma} y'$ to $h_{ZY}^{T^*}$;
14 **if** $y' \notin Q_Y^{T^*}$ **then**
15 $Q_Y^{T^*} \leftarrow Q_Y^{T^*} \cup \{y'\}$;
16 DoDFS(y', T^*);

Note that, given an arbitrary Y-state y, set $\Xi(y)$ may be empty. For example, in Figure 2, we have that $\Xi(\{1,5\}) = \emptyset$, since $c_{T_R}(\{1,5\}|_{\mathbf{R}}) = c_{T_R}(\{1\}) = \{c_2\}$ but the only control decision defined at $\{1,5\}$ in the AIC is $\{\}$. If such a scenario occurs, then Algorithm MAX-RANGE may get stuck before it correctly returns T^*. However, the following result reveals that $\Xi(y)$ is always non-empty for any Y-state y encountered in Algorithm MAX-RANGE, i.e., the control decision Act in line 5 of Algorithm MAX-RANGE is always well-defined.

Proposition 1: For any Y-state y reached in procedure DoDFS, if $y|_{\mathbf{R}} \neq \emptyset$, then $\Xi(y) \neq \emptyset$.

VII. Correctness of the Algorithm

In this section, we establish that Algorithm MAX-RANGE is correct, 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 s a safe supervisor.

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

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

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

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

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

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

Theorem 3: 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.

VIII. Conclusion

We have solved a generalized supervisor synthesis problem, called the range control problem, for partially-observed DES. We considered both an upper bound specification that describes the legal behavior and a lower bound specification that describes the required behavior. We provided an effective algorithm to solve this problem based on the notions of AIC and CSR. This results in a "meaningful" maximally permissive safe supervisor that contains a given behavior. Extending the results of this paper to include nonblockingness (i.e., non-prefix-closed specifications) remains an open problem.

References

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

[2] R. Brandt, V. Garg, R. Kumar, F. Lin, S.I. Marcus, and W.M. Wonham. Formulas for calculating supremal controllable and normal sublanguages. *Syst. & Contr. Lett.*, 15(2):111–117, 1990.

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

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

[5] H. Cho and S.I. Marcus. On supremal languages of classes of sublanguages that arise in supervisor synthesis problems with partial observation. *Math. Contr. Sig. Syst.*, 2(1):47–69, 1989.

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

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

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

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

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

[11] J.G. Thistle and H.M. Lamouchi. Effective control synthesis for partially observed discrete-event systems. *SIAM J. Control and Optim.*, 48(3):1858–1887, 2009.

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

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

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

[15] X. Yin and S. Lafortune. Synthesis of maximally permissive supervisors for partially observed discrete event systems. *IEEE Trans. Autom. Contr.*, 61(5):1239–1254, 2016.

[16] X. Yin and S. Lafortune. A uniform approach for synthesizing property-enforcing supervisors for partially-observed discrete-event systems. *IEEE Trans. Autom. Contr.*, 61(8):2140–2154, 2016.

[17] X. Yin and S. Lafortune. Synthesis of maximally-permissive supervisors for the range control problem. *University of Michigan, Tech. Rep.*, May, 2016.

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