On Controllability and Normality of Discrete Event Dynamical Systems *

Ratnesh Kumar Vijay Garg Steven I. Marcus

Department of Electrical and Computer Engineering,
The University of Texas at Austin,
Austin, Texas 78712-1084

Abstract

This paper studies the supervisor synthesis problem of Discrete Event Dynamical Systems (DEDS's) through the use of *synchronous composition* of the plant and the supervisor and discusses some of the simplifications achieved by using the synchronous composition to model supervisory control. An alternate definition of controllability is presented. This definition of controllability is then used to derive an algorithm that is computationally more efficient than previously existing ones for the construction of the supremal controllable sublanguage; the algorithm is also shown to be optimal. The observability and normality issues arising due to the partial observation of the system dynamics under an arbitrary mask are then investigated. Closed form representations of the supremal normal, and supremal closed and normal sublanguages are derived in the more general setting of arbitrary mask and nonclosed languages, thus extending earlier results of the authors and others.

*This research was supported in part by the Advanced Technology Program of the State of Texas under Grant 003658-093, in part by the Air Force of Scientific Research under Grant AFOSR-86-0029, in part by the National Science Foundation under Grant ECS-8617860, in part by the Air Force Office of Scientific Research (AFSC) under Contract F49620-89-C-0044, in part by a University Research Institute Grant and in part by a Bureau of Engineering Research Grant.

1 Introduction

Ramadge and Wonham in their work on supervisory control [13, 12] have modeled a DEDS, also called a *plant*, by a state machine (SM), the *event* set of which is finite and is partitioned into the set of *controllable* and *uncontrollable* events. The *language* generated by such a SM is used as a model to describe the behavior of the plant at the logical level. The control task is formulated as that of synthesis of a *controller*, also called a *supervisor*. The way a supervisor exercises *closed loop control* over the plant is by disabling some of the events in order that the plant may achieve a certain prescribed behavior. Supervisors designed for closed loop control, so that none of the uncontrollable events that can occur in the plant behavior are prevented from occurring in the closed loop system, are called *complete*.

Since under the closed-loop control of a complete supervisor, all uncontrollable events that can occur in the plant, can also occur in the closed loop system, there may not always exist a complete supervisor for a given plant such that the closed loop system has a prespecified desired behavior. The question then arises as to how to construct a complete supervisor which is *minimally restrictive*, so that the closed loop system can engage in some maximal behavior and still maintain the prescribed behavioral constraint. Another problem related to the design of such supervisors is to reduce the *computational complexity* of finding the minimally restrictive supervisor [13, 12, 11].

In [13, 12] the supervisor is modeled as another SM together with a *feedback map* that is used to disable certain controllable events so that those events can no longer occur in the controlled plant behavior. We use the notion of the *synchronous composition* [7, 5] of the two machines, the plant and the supervisor, to describe the control achieved by the supervisor over the plant behavior. In this way we avoid the notion of the feedback map. This simplifies the analysis of supervisory synthesis problems, as demonstrated by our treatment of some of the supervisory synthesis problems discussed below. Notions similar to synchronous composition called *prioritized synchronization* for composition of SM's in [6] and *weave* and *blend* operators for composition of traces in [14, 15] have been used for control of DEDS's.

The computational complexity of the algorithm presented in [12] for constructing the minimally restrictive supervisor is quadratic [11] in the product of the number of states involved in the FSM realizations of the plant and that of the supervisor. It was mentioned without proof in [17] that if the languages describing the plant as well as the desired behavior are *prefix closed*, then the computational complexity of constructing the minimally restrictive supervisor is linear; however, no algorithm for the supervisor construction was given. We present an algorithm (Algorithm 3.10 below) for determining the minimally restrictive supervisor that is an order of magnitude faster than the algorithm described in [12] and show the optimality of this algorithm (Theorems 3.11 and 3.13 below).

Next we study the case when the dynamics of the system under consideration is not completely observed. It has been shown in [10, 4] that a supervisor for a *partially* observed system exists if and only if the desired behavior is *observable*. A computationally efficient algorithm for the verification of observability of a given language is given in [16]. Since the class of observable sublanguages is not algebraically well behaved (a supremal element does not exist) [2, 3, 10, 4], the notion of *normality* that is in some sense stronger than the notion of observability and is preserved under union is used for synthesizing supervisors for partially observed systems. We provide closed form expressions for representing the supremal normal, and supremal closed and normal sublanguages in a general setting of arbitrary masks and nonclosed languages (Theorems 4.3 and 4.4 below), and discuss the computational aspects of these expressions. This generalizes our previously reported work in [9, 1].

These closed form expressions can be used for solving the Supervisory Control and Observation Problem [10] in a general setting where the mask is not necessarily the natural projection map and the languages are not necessarily closed. These closed form representations can also be be used for the computation of the supremal controllable and normal languages, as is demonstrated (Remark 4.5 below).

2 Notation and Terminology

The DEDS that is to be controlled, also called the plant, is represented as a deterministic SM. Letting P denote the plant, it is represented as a 5-tuple [8]: $P \stackrel{\text{def}}{=} (X, \Sigma, \alpha, x_0, X_m)$, where X denotes the states of the plant (X is finite, if P is a FSM); Σ denotes the *finite* event set; $\alpha : \Sigma \times X \to X$ is the *partial state transition function*; $x_0 \in X$ denotes the *initial* state of the plant; and $X_m \subseteq X$ denotes the *marked* states of the plant.

The behavior of the plant is described in terms of the set of strings of events, called the *language*, that the plant can generate. Formally, the languages *generated* and *recognized* (or *marked*) by P are denoted by L(P) and $L_m(P)$, respectively, and are defined as: $L(P) = \{s \in \Sigma^\star \mid \alpha(s, x_0)!\}$; $L_m(P) = \{s \in L(P) \mid \alpha(s, x_0) \in X_m\}$, where $\Sigma^\star$ denotes the set of all the finite strings of events belonging to Σ, and the notation "!" is used to denote "is defined". The state transition function α is extended to the domain $\Sigma^\star \times X$ in the natural way. The language L(P) is *prefix closed* by its definition. Further, if the plant is given to be a FSM, then L(P) and $L_m(P)$ both are *regular* languages [8].

A supervisor for the given plant is another DEDS modeled as a deterministic SM (for a more general definition, see [13]). Note that we do not distinguish between the structure of a plant and that of a supervisor. Letting S denote the supervisor, it is represented as a 5-tuple: $S \stackrel{\text{def}}{=} (Y, \Sigma, \beta, y_0, Y_m)$, where Y denotes the states of the supervisor; Σ denotes the finite event set, the same as that of the plant; $\beta : \Sigma \times Y \to Y$ denotes the partial state transition map; $y_0 \in Y$ denotes the initial state of the supervisor; and $Y_m \subseteq Y$ denotes the marked states of the supervisor.

The supervisor executes *synchronously* with the plant and thereby controls the plant behavior. Let the synchronous composition [7, 6] of the plant and the supervisor be denoted by the SM $P \square S$, where "$\square$" represents the synchronous composition operator. The notion of synchronous composition operator is similar to that of the *weave* operator defined in context of the trace theory in [14, 15]. The synchronous composition of the plant and the supervisor means that only those transitions which are allowed in both the plant P and the supervisor S, are allowed in the coupled SM $P \square S$, when run synchronously. Thus, if some of the transitions are not allowed in the supervisor then they also cannot occur in the plant. This is the way the supervisor controls the plant. Formally, $P \square S$ is another deterministic SM, represented by the 5-tuple: $P \square S \stackrel{\text{def}}{=} (Z, \Sigma, \gamma, z_0, Z_m)$, where $Z = X \times Y$ is the state set of the coupled SM, $P \square S$; Σ denotes the finite event set as before; $\gamma : \Sigma \times Z \to Z$ denotes the partial state transition map for $P \square S$. Let $\sigma \in \Sigma$ and $(x, y) \in X \times Y = Z$; then we define:

$$\gamma(\sigma,(x,y)) \stackrel{\text{def}}{=} \begin{cases} (\alpha(\sigma,x),\beta(\sigma,y)) & \text{if } \alpha(\sigma,x)! \text{ and } \beta(\sigma,y)! \\ \text{undefined} & \text{otherwise} \end{cases}$$

$z_0 = (x_0, y_0)$ denotes the initial state of the coupled SM $P\Box S$; and $Z_m = X_m \times Y_m$ denotes the marked states of the coupled SM. The synchronous composition of two SM's can be defined in a more general setting, in which the event sets of the two SM's are different [7, 6].

Remark 2.1 Let the map $f : L(P) \to 2^{\Sigma}$ denote a *control policy* as described in [13], i.e. for each string $s \in L(P)$ generated by the plant P, $f(s) \subseteq \Sigma$ is the set of events that are not disabled by a supervisor. Then the control exercised by the synchronous operation of a supervisor and the plant, as described above, defines the following control policy over the set of strings generated by the plant:

$$f(s) \stackrel{\text{def}}{=} \begin{cases} \{\sigma \in \Sigma | \gamma(s\sigma, z_0)!\} & \text{if } \gamma(s,z_0)! \\ \text{undefined} & \text{otherwise} \end{cases}$$

where the string $s \in L(P)$.

Lemma 2.2 Let L($P\Box S$) be the language generated and $L_m(P\Box S)$ the language marked by $P\Box S$; then $L(P\Box S) = L(P) \cap L(S)$ and $L_m(P\Box S) = L_m(P) \cap L_m(S)$.

Proof: First we show that $L(P\Box S) = L(P) \cap L(S)$. Consider $s \in \Sigma^\star$. Then $s \in L(P\Box S)$ if and only if $\gamma(s, z_0)!$, i.e. if and only if $\alpha(s, x_0)!$ and $\beta(s, y_0)!$, i.e. if and only if $s \in L(P)$ and $s \in L(S)$, i.e. if and only if $s \in L(P) \cap L(S)$.

Similarly, it follows that $L_m(P\Box S) = L_m(P) \cap L_m(S)$. □

Remark 2.3 It has been shown in [14, 15] that a property analogous to that stated in Lemma 2.2 also holds for the weave operator in trace theory. We observe that $L(P\Box S) \subseteq L(P)$, and $L_m(P\Box S) \subseteq L_m(P)$.

Corollary 2.4 Let P denote the plant and S denote the supervisor. Then $P\Box S$ is another supervisor for the plant P, and $L(P\Box(P\Box S)) = L(P\Box S)$; also $L_m(P\Box(P\Box S)) = L_m(P\Box S)$.

Proof: Consider the SM $P\Box(P\Box S)$. It follows from Lemma 2.2 and Remark 2.3 that $L(P\Box(P\Box S)) = L(P) \cap L(P\Box S) = L(P\Box S)$. Similarly, $L_m(P\Box(P\Box S)) = L_m(P\Box S)$. □

The event set Σ is partitioned into $\Sigma_u \cup \Sigma_c$, the *uncontrollable* and *controllable* event sets.

Definition 2.5 A supervisor S is said to be *complete* [13] with respect to a given plant P if and only if for $s \in \Sigma^\star$ and $\sigma_u \in \Sigma_u$, the conditions $\gamma(s, z_0)!$ and $\alpha(s\sigma_u, x_0)!$ together imply that $\gamma(s\sigma_u, z_0)!$. In other words, the conditions $s \in L(P\Box S)$ and $s\sigma_u \in L(P)$ together imply that $s\sigma_u \in L(P\Box S)$.

Thus the language generated by the coupled system $P\Box S$ under the control of a complete supervisor satisfies the following invariance property: $L(P\Box S)\Sigma_u \cap L(P) \subseteq L(P\Box S)$. Such an invariance property of languages is called *controllability*.

Definition 2.6 A language K is said to be *controllable* with respect to another language L, if $\overline{K}\Sigma_u \cap L \subseteq \overline{K}$, where the notation $\overline{K}$ is used to denote the *prefix closure* [13, 12] of the language K.

Thus K is controllable if and only if $\overline{K}$ is controllable.

Lemma 2.7 Let P denote the plant, S denote the supervisor and $P\Box S$ denote the coupled system. Then $L(P\Box S)$ is controllable with respect to L(P) if and only if the supervisor S is complete.

Proof: Assume that the supervisor S is complete; then the language generated by the coupled system satisfies the following invariance property (see Definition 2.5): $L(P\Box S)\Sigma_u \cap L(P) \subseteq L(P\Box S)$. Since the language generated by any SM is closed, we have $L(P\Box S) = \overline{L(P\Box S)}$. Thus we obtain $\overline{L(P\Box S)}\Sigma_u \cap L(P) \subseteq \overline{L(P\Box S)}$, which shows that $L(P\Box S)$ is controllable with respect to $L(P)$.

Assume now that $L(P\Box S)$ is controllable with respect to L(P); now since $L(P\Box S)$ is closed, we have the following: $L(P\Box S)\Sigma_u \cap L(P) \subseteq L(P\Box S)$, showing that S is complete.□

Proposition 2.8 Let $K_1 \subseteq K_2 \subseteq L(P)$ be two given languages such that $K_1 \subseteq L_m(P) \subseteq L(P)$, and assume that K_2 is prefix closed. Then there exists a supervisor S such that $L_m(P\Box S) = K_1$, and $L(P\Box S) = K_2$.

Proof: Let S_1 and S_2 be two supervisors for the plant P such that $L_m(S_1) = K_1, L(S_1) = \Sigma^\star$ and $L_m(S_2) = K_2, L(S_2) = K_2$; i. e. S_1 is the recognizer for K_1 and the S_2 is the generator for K_2. Since K_2 and $\Sigma^\star$ are closed, we can construct such SM's. All the states of S_2 are marked, while S_1 can be viewed as a recognizer for K_1 with an additional dump state in order to generate $\Sigma^\star$. Let $S \stackrel{\text{def}}{=} S_1\Box S_2$. Then $L_m(P\Box S) = L_m(P\Box(S_1\Box S_2)) = L_m(P) \cap L_m(S_1) \cap L_m(S_2) = L_m(P) \cap K_1 \cap K_2 = K_1$.

Similarly, $L(P\Box S) = L(P) \cap L(S_1) \cap L(S_2) = L(P) \cap \Sigma^\star \cap K_2 = K_2$. □

Remark 2.9 By Lemma 2.7, if the language K_2 is given to be controllable as well, then the supervisor S thus constructed will be complete. Combining the results of Lemma 2.7 and that of Proposition 2.8, we obtain the result stated in Proposition 5.1 in [13]. We note the simplification achieved in proving such a result due to the introduction of the notion of synchronous composition of two SM's in describing the control of DEDS's.

3 Minimally Restrictive Supervisor

In the previous section we have shown that given any plant with uncontrolled behavior L, there exists a supervisor which when run synchronously with the plant can restrict its behavior to any closed sublanguage $K \subseteq L$. We want the supervisor thus synthesized to be complete, so by Lemma 2.7, we need the language K to be controllable with respect to L.

In case the prescribed language K is not given to be controllable, we can construct a complete supervisor which when run synchronously with the plant generates the *supremal controllable sublanguage* of K. Let $K^\uparrow$ denote the supremal controllable sublanguage of K (the supremal controllable language is well defined and is unique [12]; it is further shown in

[12] that if the given languages K and L are regular then the language $K^{\uparrow}$ is also regular, i.e. can be generated by a FSM). For the case when the languages L and K are regular, an algorithm for constructing the language $K^{\uparrow}$ is given in [12].

Definition 3.1 A complete supervisor is said to be *minimally restrictive* [12] if, when run synchronously with the given plant, it generates the supremal controllable sublanguage of the prescribed language.

Next we prove a theorem that provides an alternative definition of controllability of an arbitrary language K with respect to a closed language L. This is used later for deriving an optimal algorithm for the minimally restrictive supervisor.

Theorem 3.2 Consider $K, L \subseteq \Sigma^\star$, and let L be closed. Then K is controllable with respect to L if and only if $\overline{K}\Sigma_u^\star \cap L \subseteq \overline{K}$, where $\Sigma_u^\star$ denotes the set of finite strings of uncontrollable events.

Proof: Assume that $\overline{K}\Sigma_u^\star \cap L \subseteq \overline{K}$. Then $\overline{K}\Sigma_u \cap L \subseteq \overline{K}$, since $\Sigma_u \subseteq \Sigma_u^\star$. Thus K is controllable.

Conversely, assume that K is controllable, i. e. $\overline{K}\Sigma_u \cap L \subseteq \overline{K}$. For $n \in \mathcal{N}$, let Σ_u^n denote the set of strings of size n consisting only of uncontrollable events. Then $\Sigma_u^\star = \bigcup_{n\in\mathcal{N}} \Sigma_u^n$. To prove that $\overline{K}\Sigma_u^\star \cap L \subseteq \overline{K}$, we will prove that $\overline{K}\Sigma_u^n \cap L \subseteq \overline{K}$ for each $n \in \mathcal{N}$. Since $\overline{K}\Sigma_u^0 = \overline{K}$, $\overline{K}\Sigma_u^n \cap L \subseteq \overline{K}$ is trivially true for $n = 0$. We will assume that $\overline{K}\Sigma_u^n \cap L \subseteq \overline{K}$ for some $n \in \mathcal{N}$ and show that $\overline{K}\Sigma_u^{n+1} \cap L \subseteq \overline{K}$; hence the proof will follow by induction. Pick $s \in \overline{K}$ and $u_{n+1} = u_n\sigma_u \in \Sigma_u^{n+1}$, where $u_n \in \Sigma_u^n$ and $\sigma_u \in \Sigma_u$, such that $su_{n+1} \in L$. Since L is closed, $su_n \in L$. So using the induction hypothesis we obtain $su_n \in \overline{K}$. Thus we have $su_n \in \overline{K}, su_{n+1} = (su_n)\sigma_u \in L$; hence from the controllability of K it follows that $(su_n)\sigma_u = su_{n+1} \in \overline{K}$. □

Corollary 3.3 If $K \subseteq \Sigma^\star$ is closed, then K is controllable with respect to a closed language $L \subseteq \Sigma^\star$ if and only if $K\Sigma_u^\star \cap L \subseteq K$.

Given a deterministic SM $V \stackrel{\text{def}}{=} (Q, \Sigma, \delta, q_0, Q_m)$, there is a natural equivalence relation R_V [8, 3] induced by V on $\Sigma^\star$, which for strings $s, t \in \Sigma^\star$ is defined by $s \cong t(R_V) \Leftrightarrow \delta(s, q_0) = \delta(t, q_0)$ (this is meant to include the condition that $\delta(s, q_0)$ is undefined $\Leftrightarrow \delta(t, q_0)$ is undefined). We use $[s](R_V)$ to denote the equivalence class under the equivalence relation R_V, containing the string s.

Definition 3.4 SM V_1 is said to *refine* SM V_2, if for $s, t \in \Sigma^\star, s \cong t(R_{V_1}) \Rightarrow s \cong t(R_{V_2})$.

Lemma 3.5 If $s, t \in L(P\Box S)$ is such that $s \cong t(R_{P\Box S})$; then $s \cong t(R_P)$, i.e. P□S refines P.

Proof: $s, t \in L(P\Box S)$ implies $s, t \in L(P)$. Thus, if $s \cong t(R_{P\Box S})$, then $\gamma(s, z_0) = \gamma(t, z_0)$. So it follows from the definition of the transition function $\gamma(\cdot, \cdot)$ that $\alpha(s, x_0) = \alpha(s, x_0)$, showing that $s \cong t(R_P)$. □

Let P $\stackrel{\text{def}}{=} (X, \Sigma, \alpha, x_0, X_m)$ denote the plant with $L(P) = L$, and $S \stackrel{\text{def}}{=} (Y, \Sigma, \beta, y_0, Y_m)$ denote a supervisor with $L(S) = K \subseteq L$, and $P\Box S \stackrel{\text{def}}{=} (Z, \Sigma, \gamma, z_0, Z_m)$ denote the coupled system. Then by Lemma 2.2, $L(P\Box S) = K$ is the closed language generated by the coupled system. Assume that S is not complete; then by Lemma 2.7, K is not controllable. We want to construct the minimally restrictive supervisor that would generate $K^{\uparrow}$, the supremal controllable sublanguage of K. Since K is closed, so is $K^{\uparrow}$; hence by Proposition 2.8, it is possible to construct a generator for $K^{\uparrow}$. This could be treated as the minimally restrictive supervisor.

Since K is not controllable, by Corollary 3.3 there exists $s \in K$ such that $su \in L(P)$ and $su \notin K$ for some $u \in \Sigma_u^{\star}$ (and conversely). We say that such a string $s \in K$ is an *uncontrollable* string. Let $s \in K$ be an uncontrollable string, and $z(= (x_z, y_z)) \in Z$ be such that $z = \gamma(s, z_0)$, i. e. z is the state reached by the SM, P□S, by accepting the string s. Then $x_z = \alpha(s, x_0)$ and $y_z = \beta(s, y_0)$. We now define: $\Sigma_u^{\star}(P)(x_z) \stackrel{\text{def}}{=} \{u \in \Sigma_u^{\star} \mid \alpha(u, x_z)!\}$, and $\Sigma_u^{\star}(P\Box S)(z) \stackrel{\text{def}}{=} \{u \in \Sigma_u^{\star} \mid \gamma(u, z)!\}$. It is clear from the above discussion that s is uncontrollable if and only if $\Sigma_u^{\star}(P)(x_z) \not\subseteq \Sigma_u^{\star}(P\Box S)(z)$.

Proposition 3.6 Let $s \in K$ be an uncontrollable string. Then all the strings $t \in K$ such that $t \cong s(R_{P\Box S})$ are uncontrollable.

Proof: $s \in K$ implies $s \in L(P)$. Let $z(= (x_z, y_z)) = \gamma(s, z_0)$ be the state reached by P□S, by accepting the string s; then $x_z = \alpha(s, x_0)$ is the state reached by the P by accepting the string s. Since s is uncontrollable, we have $\Sigma_u^{\star}(P)(x_z) \not\subseteq \Sigma_u^{\star}(P\Box S)(z)$.

Assume that $t \cong s(R_{P\Box S})$; then $t \cong s(R_P)$, since by Lemma 3.5, P□S refines P. This means that the states reached by P□S and P by accepting the string t are again z and x_z , respectively. But $\Sigma_u^{\star}(P)(x_z) \not\subseteq \Sigma_u^{\star}(P\Box S)(z)$, so t is uncontrollable. □

From Proposition 3.6 it is clear that s is uncontrollable if and only if all the strings in the equivalence class $[s](R_{P\Box S})$ are uncontrollable. Thus we have that all the strings in the equivalence class $[s](R_{P\Box S})$ are uncontrollable if and only if $\Sigma_u^{\star}(P)(x_z) \not\subseteq \Sigma_u^{\star}(P\Box S)(z)$. We define the set of "bad states" Z_b to be the following subset of Z: $Z_b \stackrel{\text{def}}{=} \{z \in Z \mid \Sigma_u^{\star}(P)(x_z) \not\subseteq \Sigma_u^{\star}(P\Box S)(z)\}$.

Remark 3.7 The deletion of the uncontrollable strings from K in order to obtain $K^{\uparrow}$, is equivalent to the elimination of the states in the set Z_b from the SM realization of $P\Box S$.

Define $\Sigma_u(P)(x_z) \stackrel{\text{def}}{=} \{\sigma_u \in \Sigma_u \mid \alpha(\sigma_u, x_z)!\}$, and $\Sigma_u(P\Box S)(z) \stackrel{\text{def}}{=} \{\sigma_u \in \Sigma_u \mid \gamma(\sigma_u, z)!\}$. We partition Z_b into two sets $Z_b^0 \cup Z_b^{\star}$ as follows: $Z_b^0 \stackrel{\text{def}}{=} \{z \in Z \mid \Sigma_u(P)(x_z) \not\subseteq \Sigma_u(P\Box S)(z)\}$ and $Z_b^{\star} \stackrel{\text{def}}{=} \{z \in Z - Z_b^0 \mid \gamma(u, z) \in Z_b^0 \text{ for some } u \in \Sigma_u^{\star}\}$.

Proposition 3.8 Let Z_b, Z_b^0, and $Z_b^{\star}$ be as defined above; then $Z_b = Z_b^0 \cup Z_b^{\star}$.

Proof: We first show that $Z_b^0 \cup Z_b^{\star} \subseteq Z_b$. We have $Z_b^0 \subseteq Z_b$, since $\Sigma_u \subseteq \Sigma_u^{\star}$. To show that $Z_b^{\star} \subseteq Z_b$, let $\tilde{z} \in Z_b^{\star}$; then there exists $z \in Z_b^0$ such that $\gamma(u, \tilde{z}) = z$ for some $u \in \Sigma_u^{\star}$. Since $z \in Z_b^0$, there exists $\sigma_u \in \Sigma_u$ such that $\sigma_u \in \Sigma_u(P)(x_z)$ and $\sigma_u \notin \Sigma_u(P\Box S)(z)$. Thus, $u\sigma_u \in \Sigma_u^{\star}$ (since $u \in \Sigma_u^{\star}$ and $\sigma_u \in \Sigma_u$), $u\sigma_u \in \Sigma_u^{\star}(P)(x_{\tilde{z}})$, and $u\sigma_u \notin \Sigma_u^{\star}(P\Box S)(\tilde{z})$. This implies that $\Sigma_u^{\star}(P)(x_{\tilde{z}}) \not\subseteq \Sigma_u^{\star}(P\Box S)(\tilde{z})$, so $\tilde{z} \in Z_b$. Thus $Z_b^{\star} \subseteq Z_b$. Hence we obtain $Z_b^0 \cup Z_b^{\star} \subseteq Z_b$.

Next we show that $Z_b \subseteq Z_b^0 \cup Z_b^{\star}$. Let $z \in Z_b$; then there exists $u \in \Sigma_u^{\star}$ such that $u \in \Sigma_u^{\star}(P)(x_z)$, and $u \notin \Sigma_u^{\star}(P\Box S)(z)$. Let u be the smallest such string; then either (i) $u = \sigma_u$ for some $\sigma_u \in \Sigma_u$, in which case $\Sigma_u(P)(x_z) \not\subseteq \Sigma_u(P\Box S)(z)$, i. e. $z \in Z_b^0$ or (ii) $u = v\sigma_u$ for some $v \in \Sigma_u^{\star}$ and $\sigma_u \in \Sigma_u$, in which case $\gamma(v, z) \in Z_b^0$, i. e. $z \in Z_b^{\star}$. Thus, if $z \in Z_b$, then either $z \in Z_b^0$ or $z \in Z_b^{\star}$. Thus, $Z_b \subseteq Z_b^0 \cup Z_b^{\star}$. □

Remark 3.9 Thus, in order to construct the generator for $K^{\uparrow}$, we need to eliminate the states in the set Z_b^0 and those in Z_b^* from the SM realization of $P \Box S$.

Algorithm 3.10 (for constructing the generator for $K^{\uparrow}$) We assume that all the SM's are finite state (this is needed for the algorithm terminate in finite time), i.e. $L(P)$ and K are regular languages; K is further assumed to be closed.

1. Construct the FSM $P \Box S$.
2. Determine the states in the set $Z_b^0 = \{z \in Z \mid \Sigma_u(P)(x_z) \not\subseteq \Sigma_u(P \Box S)(z)\}$. This is done by comparing the set of uncontrollable events defined at each state $z \in Z$ in the SM $P \Box S$ to the set of uncontrollable events defined at the corresponding state $x_z \in X$ in the SM P.
3. Construct the FSM $P \Box S \mid_{\Sigma_u} \stackrel{\text{def}}{=} (Z, \Sigma_u, \gamma \mid_{\Sigma_u \times Z}, z_0, Z_m)$ by deleting all the transitions corresponding to the controllable events.
4. Determine the states in the set Z_b^* by picking those states $z \in Z - Z_b^0$ which have a path of transitions leading to some state in Z_b^0 in the FSM $P \Box S \mid_{\Sigma_u}$.
5. Delete the states $Z_b^0 \cup Z_b^*$ (and all the transitions entering or leaving these states) from the FSM realization of $P \Box S$ to obtain the generator for $K^{\uparrow}$.

Theorem 3.11 Let m and n denote the number of states in the minimal FSM realizations of the languages L(P) and K respectively. Then computational complexity of Algorithm 3.10 is $O(mn)$.

Proof: The proof to Theorem 3.11 follows from the following arguments:

1. The FSM $P \Box S$ can be constructed in order $O(mn)$ time, since the FSM $P \Box S$ can have a maximum number mn of states.
2. The states in the set Z_b^0 can be identified in order $O(mn)$ time, because the maximum number of possible bad states is mn.
3. The FSM $P \Box S \mid_{\Sigma_u}$ can again be constructed in order $O(mn)$ time, because in order to delete the transitions caused by the controllable events, we need to consider the transitions in all the possible mn states.
4. The states in the set Z_b^* can also be identified in order $O(mn)$ time, because, if we consider the FSM $P \Box S \mid_{\Sigma_u}$ with all its transitions reversed, then the states in the set Z_b^* are those states that are *reachable* from the states in the set Z_b^0.

   Let the number of states in the set Z_b^0 be $p(\leq mn)$. Then the states (other than those in Z_b^0) that can be reached by a single transition from these p states in the FSM $P \Box S \mid_{\Sigma_u}$ with all its transitions reversed, can be identified in order $O(e_p)$ time, where e_p is the total number of uncontrollable transitions leading into the state set Z_b^0. Let the set of these states be denoted by Z_b^1 and let the number of such states be $q(\leq (mn - p))$. Then the states that can be reached (other than those in $Z_b^0 \cup Z_b^1$) from these q states in the FSM $P \Box S \mid_{\Sigma_u}$ with all its transitions reversed, due to a single transition, can be identified in order $O(e_q)$ time, where e_q is the total number of uncontrollable transitions leading into the above q states. Thus the states reachable from the states in the set Z_b^0 in the FSM, $P \Box S \mid_{\Sigma_u}$ with all its transitions reversed, due to two successive transitions, can be identified in order $O(e_p + e_q)(\leq O(e))$ time, where e is the total number of uncontrollable transitions present in $P \Box S$. If we continue identifying the states that are reachable from the states in the set Z_b^0 in the above fashion, then it is easy to see that all such states can be identified in order $\leq O(e)$ time. Since the total number of uncontrollable transitions present in $P \Box S$ is at most equal to $|\Sigma_u|mn, O(e) = O(mn)$. This completes the proof (while determining the states reachable from the set Z_b^0, care is taken so that none of the transitions is explored more than once).

5. Since the states in the set $Z_b^0 \cup Z_b^\star$ have been identified, these states (and the transitions entering and leaving these states) can be deleted in $O(mn)$ time, for the maximum number of such states is mn. □

Remark 3.12 In the above analysis we have neglected the dependence of the computational complexity on any parameter other than m and n. Algorithm 3.10 for constructing the generator for $K^\uparrow$ is an order of magnitude faster than that in [12]. The computational complexity of the algorithm in [12] is of order $O(m^2n^2)$ [11].

In [17], it has been mentioned without proof that if the languages $L(P)$ and K are both assumed to be closed, then the computational complexity of constructing minimally restrictive supervisor is of order $O(mn)$; however, no algorithm for the construction of the minimally restrictive supervisor is given.

We show now that Algorithm 3.10 for constructing the supremal controllable sublanguage is also an *optimal* algorithm, i. e. given the plant P that generates the language L(P) and another closed language K with m and n states in their minimal FSM realizations, respectively, the supremal controllable sublanguage of K with respect to L(P) cannot in general be computed with time complexity less than $O(mn)$.

Theorem 3.13 Algorithm 3.10 is computationally optimal.

Proof: We provide an example of languages L(P) and K with m and n states in their minimal FSM realizations such that the generator for $K^\uparrow$ has $O(mn)$ states in its minimal FSM realization. This shows that any algorithm that constructs the generator for $K^\uparrow$ must be of space complexity $O(mn)$, and hence of time complexity $O(mn)$ [8].

Let $\Sigma = \{a, b, c, d, e\}$ and $\Sigma_u = \{e\}$. Let L(P) and K be as given below:

$$L(P) = \{s \in \Sigma^\star \mid 0 \leq \#(a, s) - \#(b, s) \leq (m-1), \text{ and for all prefixes of the form } we\ (w \in \Sigma^\star) \text{ of } s,\ 0 \leq \#(a, w) - \#(b, w) \leq (m-2)\}$$

$$K = \{s \in \Sigma^\star \mid 0 \leq \#(c, s) - \#(d, s) \leq (n-1), \text{ and for all prefixes of the form } we\ (w \in \Sigma^\star) \text{ of } s,\ 0 \leq \#(c, w) - \#(d, w) \leq (n-2)\};$$

where the notation $\#(x, y)$ denotes the number of events "x" in the string "y". The language K is not controllable with respect to L(P), for the string $c^{n-1}e \in$ L(P) but $c^{n-1}e \notin$ K. If we delete all such uncontrollable strings from K we get the the following language as $K^\uparrow$:

$$K^{\uparrow} = \{s \in \Sigma^{\star} \mid 0 \le \#(a,s) - \#(b,s) \le (m-2), \text{ and } 0 \le \#(c,s) - \#(d,s) \le (n-2)\} \cup$$
$$\{s \in \{a,b,c,d\}^{\star} \mid 0 \le \#(a,s) - \#(b,s) = (m-1), \text{ and } 0 \le \#(c,s) - \#(d,s) \le (n-1)\}.$$

The generators of languages L(P), K and $K^{\uparrow}$ are shown in Figure 1.

In the above example L(P) and K have m and n states, respectively, in their minimal FSM realizations, whereas $K^{\uparrow}$ has $m(n-1)+1$ states in its minimal FSM realization, i.e. the space complexity of any algorithm that computes $K^{\uparrow}$ is $O(mn)$; thus any algorithm that computes $K^{\uparrow}$ has $O(mn)$ time complexity [8]. □

4 On Normality of Languages

In our previous discussions we have assumed that the supervisor has complete information about the plant behavior, i. e. the supervisor observes all the events that occur in the plant perfectly. However, in many situations the supervisor has only partial information about the plant behavior. Let $M : \Sigma \to \Delta \cup \{\epsilon\}$ be a map, called a *mask*, where Δ is the set of events that are observed by the supervisor. Let P denote the plant and $L = L(P)$ denote its behavior. Consider the extension of the mask M to the set $\Sigma^{\star}$ as: $M(\epsilon) = \epsilon$, and $M(s\sigma) = M(s)M(\sigma)$, where $s \in \Sigma^{\star}$ and $\sigma \in \Sigma$. Then the plant behavior observed by a supervisor is given by $M(L)$. Here we do not impose any restriction on the type of mask M; the mask M was assumed to be the *natural projection map* from the event set Σ to its subset Σ_o, the set of *observable* events in [10, 1]. The work presented here is thus a generalization of our previously reported work in [1]. An example of a mask that cannot be represented as the natural projection map is treated in [4, p. 59]. Also, a problem in which the supervisor can observe the occurrence of two events, but cannot distinguish between them is considered in [4] (c.f., the Alternating Bit Protocol, p. 259). Thus, using the results presented in this section we can solve the Supervisory Control and Observation Problem [10] in a general setting of non-closed languages and non-projection masks.

Let the closed language $K \subseteq L$ be the desired plant behavior. In order to be able to construct a supervisor so that the coupled system generates K, the language K must be *observable* [10, 4] with respect to L and M.

Definition 4.1 Consider two closed languages $K, L \subseteq \Sigma^{\star}$ and a mask $M : \Sigma^{\star} \to \Delta^{\star}$. K is said to be *observable* (also called (L, Σ, M)-controllable) [3, 2, 4, 10] with respect to L and M, if $s, t \in L \cap K, M(s) = M(t), \sigma \in \Sigma, s\sigma \in L \cap K$, and $t\sigma \in L$ implies that $t\sigma \in K$.

An algorithm of polynomial complexity for testing the observability of a closed language K with respect to another closed language L and a mask M is given in [16]. It is further shown in [16] that even though the test for observability can be conducted in polynomial time, the problem of constructing a supervisor (in partially observed case) for a desired observable behavior is NP-complete.

4.1 Supremal normal languages

Though one needs the desired behavior of the system to be observable in order to be able to construct the appropriate supervisor, the observable languages are not algebraically well-behaved – such languages are not closed under union [10, 4]. Thus if the desired behavior is not observable, construction of a *"minimally restrictive supervisor"* is not possible, for the supremal observable language does not exist. However *maximal observable languages* exist [3], and algorithms for constructing such languages are given in [3]. The notion of *normality* [10, 4], that is in some sense stronger than the notion of observability, is preserved under union, so that the existence of the *supremal normal language* is guaranteed [10, 4].

Definition 4.2 $K \subseteq \Sigma^\star$ is said to be *normal* (*(M, L) recognizable*) with respect to $L \subseteq \Sigma^\star$ and a mask M if and only if given $s \in L \cap K$, $t \in L$, M(t) = M(s) implies $t \in K$.

Alternatively, K is normal with respect to L and M if and only if $L \cap M^{-1}M(K) = K$. Assume that the FSM's $P \stackrel{\text{def}}{=} (X, \Sigma, \alpha, x_0, X_m)$ and $S \stackrel{\text{def}}{=} (Y, \Sigma, \beta, y_0, Y_m)$ recognize languages L and K, respectively, as before. Consider a deterministic FSM G that recognizes the language $M^{-1}M(K)$ constructed in the following way:

1. Construct a FSM that recognizes M(K) by replacing each transition $\sigma \in \Sigma$ in machine S with $M(\sigma) \in \Delta \cup \{\epsilon\}$. This machine will in general be non-deterministic.

2. Convert the machine obtained in step 1 to a deterministic machine [1].

3. Construct a FSM G

   (a) by replacing each transition $\delta \in \Delta$ in the machine obtained in step 2 by the transitions $\{\sigma \in \Sigma \mid M(\sigma) = \delta\}$, and

   (b) by adding transitions that correspond to looping on a single state, labeled with the events $\sigma \in \Sigma$ which have $M(\sigma) = \epsilon$, at each state of the machine obtained in step 3 (a).

Then G by construction recognizes the language $M^{-1}M(K)$. This is true because the SM obtained in step 1 (and in step 2) recognizes the language $M(K)$; and the SM obtained in step 3 recognizes the language $M^{-1}(\cdot)$, where $(\cdot)$ is the language recognized by the SM obtained in step 2. Note that the above assertions could be made because the map M distributes over concatenation.

Thus normality of K can be checked by considering the machines $G \square P$ and S that recognize languages $M^{-1}M(K) \cap L$ and K respectively. In case K is not normal, the supremal normal sublanguage of K can be computed. Next we present a closed form expression for the supremal normal sublanguage. The notation K° will be used to denote the supremal normal sublanguage of K.

Theorem 4.3 Consider $K, L \subseteq \Sigma^\star$, and a mask M. Then K°, the supremal normal sublanguage of K with respect to L and M, is given by $K - M^{-1}M(L - K)$.

Proof: Let $H \stackrel{\text{def}}{=} K - M^{-1}M(L - K)$. Then we need to show that $K^\circ = H$.

First, we show that $H \subseteq K^\circ$. Since K° is the supremal normal sublanguage of K, and $H \subseteq K$, it suffices to show that H is normal. Pick $s \in H$; then $s \in K$, and there does not exist $t \in L - K$ such that $M(t) = M(s)$. We need to show that there is no such $t \in L - H$. Assume for contradiction that there exists such a $t \in L - H$ so that $M(t) = M(s)$; then $t \in K - H$ (since $L - H = (L - K) \cup (K - H)$, and $t \notin L - K$). Since $t \in K - H$,

$t \in K \cap M^{-1}M(L-K)$. Hence there exists $t' \in L-K$ such that $M(t') = M(t)$. But $M(t) = M(s)$, showing that $M(s) = M(t')$. This contradicts the fact that $s \in H$ (since $t' \in L-K$).

Next we show that $K^\circ \subseteq H$. Since $K^\circ, H \subseteq K$, it suffices to show that $K - H \subseteq K - K^\circ$. Pick $s \in K - H$; then $s \in K \cap M^{-1}M(L-K)$. Thus $s \in K$, and there exists $t \in L-K$ such that $M(t) = M(s)$. Hence $s \notin K^\circ$, and thus $s \in K - K^\circ$. □

The supremal normal sublanguage of K may or may not be closed. A closed form expression for the supremal closed and normal sublanguage of a given closed language is presented in the theorem that follows next. Thus if K is not observable, a supervisor can be constructed so that the closed loop system generates the supremal closed and normal sublanguage of K. The notation $K^{\overline{\circ}}$ will be used to denote the supremal closed and normal sublanguage of K.

Theorem 4.4 Consider $K, L \subseteq \Sigma^\star$, and a mask M. Let K be closed. Then $K^{\overline{\circ}}$, the supremal closed and normal sublanguage of K with respect to L and M, is given by $K - (M^{-1}M(L-K))\Sigma^\star$.

Proof: Let $H \stackrel{\text{def}}{=} K - (M^{-1}M(L-K))\Sigma^\star$. Then we need to show that $K^{\overline{\circ}} = H$.

First, we show that $H \subseteq K^{\overline{\circ}}$. Since $K^{\overline{\circ}}$ is the supremal closed and normal sublanguage of K, and $H \subseteq K$ is closed (the operation $K - (\cdot)\Sigma^\star$ on K preserves the prefix closure), it suffices to show that H is normal. Pick $s \in H$; then $s \in K$, and there does not exist a $t \in L-K$ such that $M(t) = M(s)$ ($s \notin (M^{-1}M(L-K))\Sigma^\star$ implies $s \notin M^{-1}M(L-K)$). We need to show that there is no such $t \in L-H$. Assume for contradiction that there exists such a $t \in L-H$ so that $M(t) = M(s)$; then $t \in K-H$, for $L-H = (L-K) \cup (K-H)$ and $t \notin L-K$. Since $t \in K-H$, $t \in K \cap (M^{-1}M(L-K))\Sigma^\star$. Hence there exists a prefix $u \in \Sigma^\star$ of t, and $t' \in L-K$ such that $M(t') = M(u)$. But $M(t) = M(s)$, so there exists a prefix $v \in \Sigma^\star$ of s such that $M(v) = M(u) = M(t')$. Since H is closed, $v \in H$. This is a contradiction; for $v \in H$, and there exists $t' \in L-K$ with $M(t') = M(v)$.

Next we show that $K^{\overline{\circ}} \subseteq H$. Since $K^{\overline{\circ}}, H \subseteq K$, it suffices to show that $K - H \subseteq K - K^{\overline{\circ}}$. Pick $s \in K-H$; then $s \in K \cap (M^{-1}M(L-K))\Sigma^\star$. Thus $s \in K$, and there exists a prefix $v \in \Sigma^\star$ of s, and $t \in L-K$ such that $M(t) = M(v)$. Since K is closed, $v \in K$; therefore $v \notin K^{\overline{\circ}}$ (since $K^{\overline{\circ}}$ is normal, and $v \in K \cap M^{-1}M(L-K)$). But $K^{\overline{\circ}}$ is closed, so $s \notin K^{\overline{\circ}}$, showing that $s \in K - K^{\overline{\circ}}$. □

Remark 4.5 Theorems 4.3 and 4.4 generalize the result presented in [1] where the mask was assumed to be the natural projection map, and the languages were assumed to be closed. The assumptions of projection type mask and closed language are relaxed in obtaining the result of Theorem 4.3, and Theorem 4.4 is derived again without assuming the mask to be a projection.

The above closed form representations of the supremal (closed and) normal sublanguage of K can be used for constructing algorithms for computing the supremal (closed and) normal sublanguage of K with respect to L and M, the computational complexity of which would be exponential in the product of the number of states in the minimal FSM realizations of K and L, as expected [2].

5 Conclusion

In this paper, we have used the notion of *synchronous composition* of two SM's for describing the control of DEDS's. This considerably simplifies the problem of synthesizing supervisors for controlling the uncontrolled plant behavior.

We have defined the notion of a complete supervisor in the context of synchronous composition of plants and supervisors and have shown that the language generated by the synchronous composition of a plant and a complete supervisor is controllable, and conversely. An equivalent representation of controllable languages has been developed (Theorem 3.2), and has been used to construct an algorithm (Algorithm 3.10) for computing the supremal controllable sublanguage (equivalently, for constructing the minimally restrictive supervisor). The computational complexity of this algorithm is linear in the product of the number of states in the FSM realizations of the language generated by the plant and the prescribed language that is not controllable (Theorem 3.11). Computational optimality of the above algorithm has been proved (Theorem 3.13).

Next the case, when the supervisor can observe only the partial behavior of the plant, has been considered. We present closed form expressions for computing the supremal normal, and supremal closed and normal sublanguages in general setting of non closed languages and arbitrary masks (Theorem 4.3 and Theorem 4.4). These closed form expressions can be used for solving the supervisory control and observation problem [10] and also for deriving algorithms that do not require any iterative schemes to compute the supremal normal, and supremal closed and normal sublanguages.

References

[1] R. D. Brandt, V. K. Garg, R. Kumar, F. Lin, S. I. Marcus, and W. M. Wonham. Formulas for calculating supremal controllable and normal sublanguages. *Systems and Control Letters*, 15(8):111–117, 1990.

[2] H. Cho and S. I. Marcus. On supremal languages of class of sublanguages that arise in supervisor synthesis problems with partial observations. *Mathematics of Control Signals and Systems*, 2:47–69, 1989.

[3] H. Cho and S. I. Marcus. Supremal and maximal sublanguages arising in supervisor synthesis problems with partial observations. *Mathematical Systems Theory*, 22:177–211, 1989.

[4] R. Cieslak, C. Desclaux, A. Fawaz, and P. Varaiya. Supervisory control of discrete event processes with partial observation. *IEEE Transactions on Automatic Control*, 33(3):249–260, 1988.

[5] V. K. Garg. *Specification and Analysis of Distributed Systems with a large number of processes*. PhD thesis, Department of Electrical Engineering and Computer Science, University of California at Berkeley, 1988.

[6] M. Heymann. Concurrency and discrete event control. *IEEE Control Systems Magazine*, 10(4):103–112, 1990.

[7] C. A. R. Hoare. *Communicating Sequential Processes*. Prentice Hall, Inc., Englewood Cliffs, NJ, 1985.

[8] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, Reading, MA, 1979.

[9] R. Kumar, V. K. Garg, and S. I. Marcus. Supervisory control of discrete event systems: supremal controllable and observable languages. In *Proceedings of 1989 Allerton Conference*, pages 501–510, Allerton, IL, September 1989.

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

[11] P. J. Ramadge. The complexity of basic problems for discrete event systems. In M. J. Denham and A. J. Laub, editors, *Advanced Computing Concepts and Techniques in Control Engineering*, volume F47 of *NATO ASI*, pages 171–190. Springer-Verlag, 1988.

[12] P. J. Ramadge and W. M. Wonham. On the supremal controllable sublanguage of a given language. *SIAM Journal of Control and Optimization*, 25(3):637–659, 1987.

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

[14] R. Smedinga. Using trace theory to model discrete events. In P. Varaiya and A. B. Kurzhanski, editors, *Discrete Event Systems: Models and Applications*, pages 81–99. Springer-Verlag, 1987.

[15] R. Smedinga. *Control of Discrete Events*. PhD thesis, Department of Computing Science, University of Groningen, 1988.

[16] J. N. Tsitsiklis. On the control of discrete event dynamical systems. *Mathematics of Control Signals and Systems*, 2(2):95–107, 1989.

[17] W. M. Wonham and P. J. Ramadge. Modular supervisory control of discrete event systems. *Mathematics of Control Signals and Systems*, 1(1):13–30, 1988.