On Supervisory Control of Sequential Behaviors *

Ratnesh Kumar
Department of Electrical Engineering
University of Kentucky
Lexington, KY 40506-0046
Email: Kumar@engr.uky.edu

Vijay K. Garg
Department of Electrical and Computer Engineering
University of Texas at Austin
Austin, TX 78712-1084

Steven I. Marcus
Department of Electrical Engineering and
Institute of Systems Research
University of Maryland at College Park, College Park, MD 20712

Abstract

We address the supervisory synthesis problem for controlling the sequential behaviors of Discrete Event Dynamical Systems (DEDS's) under complete as well as partial information through the use of synchronous composition of the plants and the supervisors. We present the notion of *complete* languages, discuss some of its algebraic properties and show its close relation to ω-languages. We prove that the supremal (closed,) complete and controllable sublanguage of a given language exists and present an algorithm to compute it. We present a closed form expression for the supremal ω-controllable sublanguage of a given ω-language in terms of the supremal (closed,) complete and controllable sublanguage. This closed form expression suggests that certain operations on a given ω-language can alternatively be achieved by performing certain other but similar operations on its prefix (which is a finite language) and then taking the limit (to obtain the desired ω-language). A necessary and sufficient condition for the existence of a supervisor in case of partial observation is presented in terms of *ω-observability*. Notion of *ω-normality* is also introduced and a closed form expression for the supremal ω-normal sublanguage in terms of the supremal closed, complete and normal sublanguage is presented.

1 Introduction

In this paper we study the problem of synthesizing supervisors for controlling the *sequential* behavior of a plant under both complete and partial observation. The supervisory synthesis problem for controlling the sequential or infinite string behavior was first studied by Ramadge [14] and Thistle and Wonham [17]. Further related work by the authors is reported in [12]. We follow the

*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.

framework of [14] for modeling the sequential behavior of a plant and the closed loop system, and use the notion of the synchronous composition of two machines [9, 11, 10] for describing the control achieved by a supervisor over the sequential behavior of a plant.

The necessary and sufficient condition that the desired ω-language must satisfy for the existence of a supervisor is called *ω-controllability* [14]. It is further shown in [14] that ω-controllability is preserved under union, hence the supremal ω-controllable sublanguage of a given ω-language exists and is unique. One of the main results of this paper is to present a closed form expression for the supremal ω-controllable sublanguage. Using this closed form representation, we also present an algorithm for computing the supremal ω-controllable sublanguage.

We introduce the notion of *complete* languages and show that these languages are closely related to the ω-languages. We discuss some of the algebraic properties of complete languages and show that they are algebraically well behaved. Completeness of languages is preserved under union, hence the supremal complete, (closed and controllable) sublanguage of a given language exists and is unique. The closed form expression for the supremal ω-controllable sublanguage is obtained in terms of the supremal (closed,) complete and controllable sublanguage of a given language. This suggests that certain operations on the ω-languages can alternatively be performed by performing certain other but similar operations on their prefixes (which are complete languages) and then taking the *limits* (to obtain the desired ω-language). As the operations of taking the prefix of an ω-language and taking the limit of a (finite string) language can be easily performed, the problem of performing certain computations in the space of ω-languages can be reduced to performing certain other similar computations in the space of (finite string) languages. An algorithm for constructing the supremal complete, closed and controllable sublanguage of a given language is presented. The computational complexity of the algorithm is linear in product of the number of states in the finite state machine (FSM) realization of the given language and the number of states in the FSM realization of the plant.

In case the sequential behavior of the plant is not fully observed, we show that the supervisor for achieving the desired closed loop sequential behavior exists if and only if the desired behavior is *ω-observable* (a concept defined in this paper). ω-observable languages are not closed under union; however, a stronger notion, called *ω-normality* that we define next is preserved under union, so that the supremal ω-normal sublanguages exist and the construction of minimally restrictive supervisors is possible. We present a closed form expression for computing the supremal ω-normal sublanguages. This closed form expression is similar to the one obtained for the supremal ω-controllable sublanguages.

2 Notation and Terminology

The DEDS to be controlled, called the plant, is represented as a deterministic trim [16] state machine (SM). Letting P denote the plant, it is represented as a 5-tuple [7]: $P \stackrel{\text{def}}{=} (X, \Sigma, \alpha, x_0, X_m)$, where X denotes the state set 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 set of *marked* states of the plant.

The finite string behavior of the plant is described by the set of finite 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: $L(P) = \{s \in \Sigma^\star \mid \alpha(s, x_0)!\}$, and $L_m(P) = \{s \in L(P) \mid \alpha(s, x_0) \in X_m\}$, where $\Sigma^\star$ denotes the set of all the finite strings of events belonging to Σ; 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 [7].

A supervisor for controlling a given plant is another DEDS modeled as a deterministic trim SM (for a more general definition see [16]). 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 state set 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 set of marked states of the supervisor.

The supervisor executes *synchronously* with the plant and thereby controls the plant behavior. The synchronous composition [6, 5, 11] of the plant and the supervisor is represented by another SM $P \Box S$, where "$\Box$" represents the synchronous composition operator. 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 $P \Box S$, 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, $P \Box S$ is another deterministic SM, represented by the 5-tuple: $P \Box S \stackrel{\text{def}}{=} (Z, \Sigma, \gamma, z_0, Z_m)$, where $Z = X \times Y$ is the state set of the coupled SM, $P \Box S$; Σ denotes the finite event set as before; $\gamma : \Sigma \times Z \to Z$ denotes the partial state transition map for $P \Box S$. 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 [6, 5].

Example 2.1 Consider the plant P shown in Figure 1 ("$\bigcirc$" denotes a state; "$\to$" entering a state denotes the initial state; a directed arc, labeled with an event, between two states denotes a state transition; and "$\circledcirc$" denotes a marked state). The event set Σ of P equals the set $\{a, b, c, u\}$. Then $L(P) = \overline{(a+b)^\star ca^\star(b+u)a^\star}$ and $L_m(P) = (a+b)^\star ca^\star(b+u)a^\star$.

Let the state machine S shown in Figure 1 be a supervisor for the plant P. Then $L(S) = \overline{(ca + (ab)^\star acu)a^\star}$ and $L_m(S) = (ca + (ab)^\star acu)a^\star$. We note that $L(S) \subseteq L(P)$ and $L_m(S) \subseteq L_m(P)$. It is easily shown that the state machine $P \Box S$ has the same structure as that of S so that $L(P \Box S) = L(S)$ and $L_m(P \Box S) = L_m(S)$.

Lemma 2.2 [9, 11] 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 $L_m(P \Box S) = L_m(P) \cap L_m(S)$.

The event set Σ is partitioned into $\Sigma_u \cup (\Sigma - \Sigma_u)$, the sets of *uncontrollable* and *controllable* events. It is shown in [16, 9, 11] that given any plant P, there exists a supervisor which when run synchronously with the plant can restrict its behavior to any language K if and only if K is *controllable*.

Definition 2.3 $K \subseteq \Sigma^\star$ is said to be *controllable* with respect to P, if $\overline{K}\Sigma_u \cap L(P) \subseteq \overline{K}$, where $\overline{K}$ denotes the prefix closure [7] of K.

Thus K is controllable if and only if $\overline{K}$ is controllable. In case K is not controllable, a *minimally restrictive* supervisor which restricts the plant's behavior to the *supremal controllable* sublanguage $K^\uparrow \subseteq K$ can be constructed [16, 15]. A closed form expression for $K^\uparrow$ and an optimal algorithm for computing $K^\uparrow$ assuming K to be closed is presented in [9, 1, 11].

If the plant behavior is incompletely observed, then a supervisor can be constructed to restrict the plant's behavior to the desired behavior K if and only if K is controllable and *observable* [13, 3]. Let $M : \Sigma \to \Delta \cup \epsilon$ denote an observation map, called a *mask*, from Σ to the observation space $\Delta \cup \epsilon$. Let M be extended to the space $\Sigma^\star$ in the obvious manner.

Definition 2.4 $K \subseteq \Sigma^\star$ is said to be *observable* with respect to P and M, if for $s, t \in L(P) \cap \overline{K}$ such that $M(s) = M(t)$, the following hold:

1. $\sigma \in \Sigma, s\sigma \in L(P) \cap \overline{K}, t\sigma \in L(P)$ implies $t\sigma \in \overline{K}$, and
2. $s \in K, t \in L_m(P)$ implies $t \in K$.

In case K is not observable, a minimally restrictive supervisor cannot be constructed, for a supremal observable sublanguage of K does not exist [13, 3]. However, *maximal* observable sublanguages of K exist [2], and algorithms for constructing them are given in [2]. A stronger notion of observability, called *normality* [13], is often considered for supervisory control under partial observation.

Definition 2.5 $K \subseteq \Sigma^\star$ is said to be *normal* with respect to P and M if $M^{-1}(M(\overline{K})) \cap L(P) = \overline{K}$.

Normality of languages is preserved under union [13]. Hence, if K is not observable, a supervisor which restricts the plant's behavior to the supremal normal sublanguage of K can be constructed. Algorithms and closed form expressions for computing the supremal normal sublanguage are given in [9, 1, 11].

3 Formula for the Supremal ω-Controllable Sublanguage

The synthesis of the supervisors for controlling the infinite or *sequential* behavior of DEDS's can also be easily studied in the above framework. We extend the above notations (following the framework of Ramadge [14]) to describe the infinite behavior of a given plant, and present a closed form expression for computing the *supremal ω-controllable* sublanguage of a given *ω-language*.

Let Σ^ω denote the set of all infinite strings of events belonging to Σ. An *infinite* string or *ω-language* is a sublanguage of Σ^ω. Let $e^n \in \Sigma^\star$ denote the prefix of size n of the infinite string $e \in \Sigma^\omega$. Given two infinite strings $e_1, e_2 \in \Sigma^\omega$, the distance $d(e_1, e_2)$ between the two infinite strings is defined to be [4]:

$$d(e_1, e_2) \stackrel{\text{def}}{=} \begin{cases} 1/(n+1) & \text{if } e_1^n = e_2^n \text{ and } e_1^{n+1} \neq e_2^{n+1} \ (n \in \mathcal{N}) \\ 0 & \text{if } e_1 = e_2 \end{cases}$$

Given a language $L \subseteq \Sigma^\star$, its *limit*, denoted as L^∞, is the ω-language defined as:

$$L^\infty \stackrel{\text{def}}{=} \{e \in \Sigma^\omega \mid e^n \in L \text{ for infinitely many } n \in \mathcal{N}\}$$

We will use $t \leq s$ to denote that $t \in \Sigma^\star$ is a prefix of $s \in \Sigma^\star \cup \Sigma^\omega$. If t is a proper prefix of s, then it is written as $t < s$. Given an infinite sequence of strings $s_1 < s_2 < \ldots < s_n < \ldots$ with $s_n \in \Sigma^\star$ for each n, there exists a unique infinite string $e \in \Sigma^\omega$ such that $s_n < e$ for each n. In this case, the infinite string e is also written as $e = \lim_{n\to\infty} s_n$. Given an ω-language $\mathcal{L} \subseteq \Sigma^\omega$, its *prefix*, denoted by $pr\mathcal{L}$, is the language:

$$pr\mathcal{L} \stackrel{\text{def}}{=} \{s \in \Sigma^\star \mid \exists e \in \mathcal{L} \text{ s.t. } s < e\}$$

Note that $pr\mathcal{L} = pr\overline{\mathcal{L}}$, where $\overline{\mathcal{L}}$ denotes the topological closure[1] of $\mathcal{L}$ in the metric space (Σ^ω, d) [4]. It is readily verified that for a ω-language $\mathcal{L} \subseteq \Sigma^\omega$, $(pr\mathcal{L})^\infty = \overline{\mathcal{L}}$. Thus $(pr\mathcal{L})^\infty = \mathcal{L}$ if and only if $\mathcal{L}$ is topologically closed, i.e. $\overline{\mathcal{L}} = \mathcal{L}$. We will show that for any language $L \subseteq \Sigma^\star$, $pr(L^\infty) = L$ if and only if L is *complete* (see Definition 3.3) and prefix closed, which, as we will see, is a useful result.

With these preliminary notions we can study the problem of controlling the infinite behavior of a given DEDS. Let $P \stackrel{\text{def}}{=} (X, \Sigma, \alpha, x_0, X_m)$ denote the plant. Then as defined above, $L_m(P), L(P) \subseteq \Sigma^\star$ denote its (finite string) marked, generated languages respectively. The ω-language generated by P, denoted by $\mathcal{L}(P)$, is defined to be:

$$\mathcal{L}(P) \stackrel{\text{def}}{=} \{e \in L(P)^\infty \mid \exists \text{ infinitely many } n \in \mathcal{N} \text{ s.t. } \alpha(e^n, x_\circ) \in X_m\} = (L_m(P))^\infty$$

Note that the ω-language $\mathcal{L}(P)$ generated by P as defined above is also the ω-language generated by P viewed as a Büchi automaton [4]. P is said to *nonblocking* if $pr\mathcal{L}(P) = L(P)$. We have shown in [8] that P is nonblocking if and only if it is *live* (see Definition 3.4 and Lemma 4.8). Let $S \stackrel{\text{def}}{=} (Y, \Sigma, \beta, y_0, Y_m)$ denote the supervisor that controls P by operating in synchrony with it as described above. Then the ω-language generated by the closed loop system $P\Box S$ is defined to be:

$$\mathcal{L}(P\Box S) \stackrel{\text{def}}{=} (L(P\Box S))^\infty \cap \mathcal{L}(P)$$

Example 3.1 (continued) Consider the plant P and the supervisor S described in Example 2.1. Then $\mathcal{L}(P) = (a+b)^\star ca^\star(b+u)a^\omega$ and $\mathcal{L}(P\Box S) = (L(P\Box S))^\infty \cap \mathcal{L}(P) = (L(S))^\infty \cap \mathcal{L}(P) = (ab)^\omega + (ca + (ab)^\star ac(u+ba))a^\omega \cap (a+b)^\star ca^\star(b+u)a^\omega = (ca + (ab)^\star ac(u+ba))a^\omega$.

Let $\mathcal{K} \subseteq \mathcal{L}(P)$ be the desired ω-language. It is shown in [14] that a complete, nonblocking supervisor exists for achieving the desired behavior if and only if it is *ω-controllable* with respect to P.

Definition 3.2 $\mathcal{K} \subseteq \Sigma^\omega$ is said to be *ω-controllable* with respect to a plant P if $pr\mathcal{K}$ is controllable with respect to P, and $\mathcal{K}$ is topologically closed with respect to $\mathcal{L}(P)$.

It is further shown in [14] that if $\mathcal{K}$ is not ω-controllable, but is topologically closed with respect to $\mathcal{L}(P)$, then the *supremal ω-controllable* sublanguage, denoted by $\mathcal{K}^\uparrow$, of $\mathcal{K}$ exists[2]. Thus the construction of a *minimally restrictive supervisor* is possible.

In the next theorem we present a closed form expression for the supremal ω-controllable sublanguage. This is one of the main results presented in this paper. First we define the notion of *complete* languages which will be useful in studying the control of infinite behaviors.

Definition 3.3 Consider a language $L \subseteq \Sigma^\star$. A string $s \in L$ is said to have an *extension* in L if there exists $t \in L$ such that $s < t$. L is said to be *complete* if for every string $s \in L$, there exists an extension in L.

The notion of complete languages is closely related to the notion of *live* SM's that we define next:

Definition 3.4 Consider a SM, $V \stackrel{\text{def}}{=} (Q, \Sigma, \delta, q_0, Q_m)$. A state $q \in Q$ is said to be *live* if there exists a $\sigma \in \Sigma$ such that $\delta(\sigma, q)!$. V is said to be *live* if all the states in Q are live.

[1] The notation $\overline{\mathcal{L}}$ is used to denote topological closure whenever $\mathcal{L} \subseteq \Sigma^\omega$, and the notation $\overline{L}$ is used to denote the prefix closure whenever $L \subseteq \Sigma^\star$.

[2] The notation $\mathcal{K}^\uparrow$ is used to denote the supremal ω-controllable sublanguage of $\mathcal{K} \subseteq \Sigma^\omega$, and the notation $K^\uparrow$ is used to denote the supremal controllable sublanguage of $K \subseteq \Sigma^\star$.

A state that is not live will be referred to as a *dead* state.

Lemma 3.5 [8] A language $L \subseteq \Sigma^\star$ is complete if and only if any deterministic trim SM recognizing it is live.

Next we study some of the properties of a complete language.

Lemma 3.6 [8] $K \subseteq \Sigma^\star$ is complete if and only if $\overline{K}$ is complete.

Example 3.7 (continued) Consider the SM's P and S described in Example 2.1. Since both P and S are live (refer to Figure 1), it follows from Lemma 3.5 that both $L_m(P)$ and $L_m(S)$ are complete languages. Also, since $L(P) = \overline{L_m(P)}$ and $L(S) = \overline{L_m(S)}$, it follows from Lemma 3.6 that $L(P)$ and $L(S)$ are complete languages as well.

Proposition 3.8 Consider $K \subseteq \Sigma^\star$. Then $pr(K^\infty) = K$ if and only if K is prefix closed and complete.

Proof: Assume that K is prefix closed and complete. We first show that $pr(K^\infty) \subseteq K$. Pick $s \in pr(K^\infty)$; then there exists $e \in K^\infty$ such that $s < e$. Since $e \in K^\infty$, there exists an infinite sequence of strings $s_1 < s_2 < \ldots < s_n \ldots$, $s_n \in K$ for each n, such that $\lim_{n\to\infty} s_n = e$. Pick a string s_m from the above infinite sequence of strings such that $s \leq s_m$. Since $s_m \in K$, and K is prefix closed, $s \in K$.

Next we show that $K \subseteq pr(K^\infty)$. Pick $s \in K$; then since K is complete, there exists an infinite sequence of strings $t_1 < t_2 < \ldots < t_n \ldots$,such that $t_n \in K$ for each n, and $s < t_1$. Let $e \stackrel{\text{def}}{=} \lim_{n\to\infty} t_n$; then $e \in K^\infty$. Since $s < e$, $s \in pr(K^\infty)$.

Next assume that $pr(K^\infty) = K$. Since $pr(K^\infty)$ is prefix closed by definition, it follows that K is prefix closed. It remains to show that K is complete. Pick $s \in K$; then $s \in pr(K^\infty)$, i.e. there exists $e \in K^\infty$ such that $s < e$. Since $e \in K^\infty$, there exist infinitely many $n \in \mathcal{N}$ such that $e^n \in K$. Then there exists $m \in \mathcal{N}$ such that $s < e^m$. Since $e^m \in K$ and $s < e^m$, we obtain that K is complete. $\Box$

Lemma 3.9 [8] The set of complete languages is closed under union.

Corollary 3.10 [8] Consider $K \subseteq \Sigma^\star$ and a plant P. Then the supremal (closed,) complete and controllable sublanguage of K with respect to P exists, and is unique.

The notation $K^\Uparrow$ will be used to denote the supremal complete and controllable sublanguage, and the notation $K^{\overline{\Uparrow}}$ will be used to denote the supremal complete, closed and controllable sublanguage of $K \subseteq \Sigma^\star$.

Corollary 3.11 Assume that $K \subseteq \Sigma^\star$ is closed, then $K^\Uparrow = \overline{K^\Uparrow} = K^{\overline{\Uparrow}}$.

Proof: Since $K^\Uparrow \subseteq K$ and K is prefix closed, $\overline{K^\Uparrow} \subseteq K$. Also, $\overline{K^\Uparrow}$ is complete and controllable (follows from Lemma 3.6 and the definition of controllability). Since $K^\Uparrow$ is the supremal complete and controllable sublanguage of K, it follows that $K^\Uparrow = \overline{K^\Uparrow}$. Thus $K^\Uparrow$ is closed. Since $K^{\overline{\Uparrow}}$ is the supremal complete, closed and controllable sublanguage of K, it follows that $K^\Uparrow = K^{\overline{\Uparrow}}$. $\Box$

With the above remarks on complete languages and some of their algebraic properties, we can present a closed form expression for the supremal ω-controllable sublanguage of a given ω-language.

Lemma 3.12 If $\mathcal{K} \subseteq \mathcal{L}(P)$ is topologically closed, then it is topologically closed with respect to $\mathcal{L}(P)$.

Proof: We have $\overline{\mathcal{K}} \cap \mathcal{L}(P) = \mathcal{K} \cap \mathcal{L}(P) = \mathcal{K}$, where the first equality follows from the fact that $\mathcal{K}$ is closed, and the last equality follows from the fact that $\mathcal{K} \subseteq \mathcal{L}(P)$. $\square$

Thus it follows from Lemma 3.12 that if $\mathcal{K} \subseteq \mathcal{L}(P)$ is topologically closed, then $\mathcal{K}^{\uparrow}$ exists.

Theorem 3.13 Let $\mathcal{K} \subseteq \mathcal{L}(P)$ be topologically closed. Then $\mathcal{K}^{\uparrow} = ((pr\mathcal{K})^{\overline{\Uparrow}})^{\infty}$.

Note that in Theorem 3.13, $((pr\mathcal{K})^{\overline{\Uparrow}})^{\infty} = ((pr\mathcal{K})^{\Uparrow})^{\infty}$ (follows from Corollary 3.11, for $pr\mathcal{K}$ is a closed language).

Remark 3.14 We have obtained a closed form expression for $\mathcal{K}^{\uparrow}$ in a more general setting, where we do not assume $\mathcal{K}$ to be topologically closed. We have proved that if $\mathcal{K}$ can be written as the limit of a complete language, i.e. if there exists a complete $K \subseteq \Sigma^{\star}$ such that $\mathcal{K} = K^{\infty}$, then $\mathcal{K}^{\uparrow} = ((pr(K)^{\Uparrow})^{\infty}$. This result is reported in [8]. However, assuming that $\mathcal{K}$ is topologically closed results in a much simpler proof.

We first prove some lemmas before proving Theorem 3.13.

Lemma 3.15 Let $\mathcal{H} \stackrel{\text{def}}{=} ((pr\mathcal{K})^{\overline{\Uparrow}})^{\infty}$. Then the following hold:

1. $\mathcal{H} \subseteq \mathcal{K}$
2. $pr\mathcal{H} = (pr\mathcal{K})^{\overline{\Uparrow}}$
3. $\mathcal{H}$ is topologically closed.

Proof: 1. Since $(pr\mathcal{K})^{\overline{\Uparrow}} \subseteq pr\mathcal{K}$, we have $\mathcal{H} = ((pr\mathcal{K})^{\overline{\Uparrow}})^{\infty} \subseteq (pr\mathcal{K})^{\infty} = \overline{\mathcal{K}} = \mathcal{K}$, where the last equality follows from the fact that $\mathcal{K}$ is closed.

2. Follows from Proposition 3.8, for $(pr\mathcal{K})^{\overline{\Uparrow}}$ is by definition closed and complete.

3. We have $\mathcal{H} = ((pr\mathcal{K})^{\overline{\Uparrow}})^{\infty} = (pr\mathcal{H})^{\infty} = \overline{\mathcal{H}}$, where the second equality follows from part 2 above. Thus $\mathcal{H} = \overline{\mathcal{H}}$. $\square$

Corollary 3.16 $\mathcal{H}$ is topologically closed with respect to $\mathcal{L}(P)$.

Proof: From part 1 of Lemma 3.15, we have $\mathcal{H} \subseteq \mathcal{K}$. Hence $\mathcal{H} \subseteq \mathcal{L}(P)$. Since $\mathcal{H}$ is also closed (part 3 of Lemma 3.15), it follows from Lemma 3.12 that $\mathcal{H}$ is closed with respect to $\mathcal{L}(P)$. $\square$

Lemma 3.17 $pr(\mathcal{K}^{\uparrow}) = (pr\mathcal{K})^{\overline{\Uparrow}}$

Proof: Since $\mathcal{K}^{\uparrow} \subseteq \mathcal{K}$, $pr(\mathcal{K}^{\uparrow}) \subseteq pr\mathcal{K}$. Thus $pr(\mathcal{K}^{\uparrow})$ is a complete, closed and controllable sublanguage of $pr\mathcal{K}$. Hence $pr(\mathcal{K}^{\uparrow}) \subseteq (pr\mathcal{K})^{\overline{\Uparrow}}$, for $(pr\mathcal{K})^{\overline{\Uparrow}}$ is the supremal complete, closed and controllable sublanguage of $pr\mathcal{K}$.

It remains to show that $(pr\mathcal{K})^{\overline{\Uparrow}} \subseteq pr(\mathcal{K}^{\uparrow})$. Using part 2 of Lemma 3.15 we obtain $pr\mathcal{H} = (pr\mathcal{K})^{\overline{\Uparrow}}$, showing that $pr\mathcal{H}$ is controllable. Since $\mathcal{H}$ is also topologically closed with respect to $\mathcal{L}(P)$ (Corollary 3.16), $\mathcal{H}$ is an ω-controllable sublanguage of $\mathcal{K}$. Hence $\mathcal{H} \subseteq \mathcal{K}^{\uparrow}$ (since $\mathcal{K}^{\uparrow}$ is the supremal one), showing that $pr\mathcal{H} \subseteq pr(\mathcal{K}^{\uparrow})$, i. e. $(pr\mathcal{K})^{\overline{\Uparrow}} \subseteq pr(\mathcal{K}^{\uparrow})$. $\square$

Proof (of Theorem 3.13): We have $\mathcal{K}^{\uparrow} = \overline{\mathcal{K}^{\uparrow}} \cap \mathcal{L}(P) = (pr(\mathcal{K}^{\uparrow}))^{\infty} \cap \mathcal{L}(P) = ((pr\mathcal{K})^{\overline{\Uparrow}})^{\infty} \cap \mathcal{L}(P) = \mathcal{H} \cap \mathcal{L}(P) = \mathcal{H}$, where the first equality follows from the fact that $\mathcal{K}^{\uparrow}$ is topologically closed with respect to $\mathcal{L}(P)$ ($\mathcal{K}^{\uparrow}$ is ω-controllable), the third equality follows from Lemma 3.17, and the last equality follows from the fact that $\mathcal{H} \subseteq \mathcal{K} \subseteq \mathcal{L}(P)$. $\square$

Corollary 3.18 If $\mathcal{K} \subseteq \mathcal{L}(P)$ is closed, then so is $\mathcal{K}^{\uparrow}$.

Proof: Follows from the fact that $\mathcal{K}^{\uparrow} = \mathcal{H}$, and $\mathcal{H}$ is closed. $\Box$

Remark 3.19 Let π be a property of languages that is preserved under union. An ω-language $\mathcal{K} \subseteq \mathcal{L}(P)$ is said to be ω-π if it is topologically closed with respect to $\mathcal{L}(P)$ and $pr\mathcal{K}$ satisfies π. Then following the proof of Proposition 3.3 [14], it is easily shown that the supremal ω-π sublanguage of $\mathcal{K}$ exists and is unique. Letting $\mathcal{K}^{\pi}$ denote the supremal ω-π sublanguage of $\mathcal{K}$ and assuming that $\mathcal{K}$ is closed, it can be proved, in view of Theorem 3.13, that $\mathcal{K}^{\pi}$ is same as the ω-language obtained by taking the limit of the supremal complete and closed sublanguage of $pr\mathcal{K}$ satisfying the property π. The interested reader is referred to [8] for a more general result, where $\mathcal{K}$ is not assumed to be topologically closed.

4 Computation of $K^{\overline{\Uparrow}}$

In the previous section we showed that the supremal complete, closed and controllable sublanguage, $K^{\overline{\Uparrow}}$, of a given language $K \subseteq \Sigma^{\star}$ with respect to a plant P exists and is unique (Corollary 3.10). In this section we present an algorithm for computing $K^{\overline{\Uparrow}}$ assuming that K is closed, in which case $K^{\overline{\Uparrow}}$ is the same as $K^{\Uparrow}$ (Corollary 3.11). This algorithm can then be used for computing the supremal ω- controllable sublanguages by using the formula in Theorem 3.13.

Consider $P \stackrel{\text{def}}{=} (X, \Sigma, \alpha, x_0, X_m)$ and let $S \stackrel{\text{def}}{=} (Y, \Sigma, \beta, y_0, Y_m)$ be such that $L(S) = K$ (K is assumed to be closed). Since K represents the desired plant behavior, we can assume without loss of generality that $K \subseteq L(P)$. Construct $P \Box S \stackrel{\text{def}}{=} (Z, \Sigma, \gamma, z_0, Z_m)$; then from Lemma 2.2, $L(P \Box S) = K$. Since K is closed, $\overline{K} = K$. Thus K is controllable with respect to P if and only if $K\Sigma_u \cap L(P) \subseteq K$. We define $s \in K$ to be an *uncontrollable string* if there exists $\sigma_u \in \Sigma_u$, such that $s\sigma_u \in L(P) - K$. Let $z = (x_z, y_z) \in X \times Y$ be the state reached in $P \Box S$ by accepting an uncontrollable string $s \in K$. Then the state reached in P by accepting s is x_z. Letting $\Sigma_u(P)(x_z), \Sigma_u(P \Box S)(z)$ denote the sets of uncontrollable events defined at states x_z, z respectively, i.e.

$$\Sigma_u(P)(x_z) = \{\sigma_u \in \Sigma_u | \alpha(\sigma_u, x_z)!\},\ \Sigma_u(P \Box S)(z) = \{\sigma_u \in \Sigma_u | \gamma(\sigma_u, z)!\},$$

it follows from the definition of uncontrollable string that $\Sigma_u(P)(x_z) \not\subseteq \Sigma_u(P \Box S)(z)$. A string $s \in K$ is said to be a *bad* string if either it is an uncontrollable string, or it has no extension in K. Letting $z = (x_z, y_z) \in X \times Y$ denote the state reached in $P \Box S$ by accepting a bad string $s \in K$, it follows from the definition of bad strings that either $\Sigma_u(P)(x_z) \not\subseteq \Sigma_u(P \Box S)(z)$ or z is a dead state (Lemma 3.5).

Given a deterministic SM $V \stackrel{\text{def}}{=} (Q, \Sigma, \delta, q_0, Q_m)$, there is a natural equivalence relation R_V [7, 2] induced by V on $\Sigma^{\star}$, which is defined by $s \cong t(R_V) \Leftrightarrow \delta(s, q_0) = \delta(t, q_0)$[3], where $s, t \in \Sigma^{\star}$. We use $[s](R_V)$ to denote the equivalence class under the equivalence relation R_V, containing the string s.

Lemma 4.1 $s \in K$ is a bad string if and only if all the strings $[s](R_{P \Box S})$ are bad.

Proof: Assume that all the strings in $[s](R_{P \Box S})$ are bad; then clearly s is a bad string, for $s \in [s](R_{P \Box S})$.

Assume next that $s \in K$ is a bad string. Let $z \in Z$ be the state reached by accepting s in $P \Box S$, i. e. $z = \gamma(s, z_0)$. Let $z = (x_z, y_z) \in X \times Y$. Since s is a bad string, either $\Sigma_u(P)(x_z) \not\subseteq \Sigma_u(P \Box S)(z)$ or z is a dead state. Thus if $t \in [s](R_{P \Box S})$, then the state reached in $P \Box S$ by accepting the string

[3]This is meant to include the condition that $\delta(s, q_0)$ is undefined $\Leftrightarrow$ $\delta(t, q_0)$ is undefined.

t is again z. Thus either t is an uncontrollable string (if s is uncontrollable) or t has no extension in K (if s has none). Hence t is a bad string. $\Box$

The implication of Lemma 4.1 is that if a string $s \in K$ is bad, then the state $z \in Z$ reached by accepting s must be removed from $P \Box S$ in order to construct the generator for $K^{\overline{\Uparrow}}$. Let $Z_b \subseteq Z$ be the set of all the states that need to removed from $P \Box S$. Then an algorithm for computing Z_b is given below:

Algorithm 4.2 We need to compute Z_b. Assume that P is a FSM and the closed language K is regular, so that $P \Box S$ is a FSM. This is needed for the algorithm to terminate in a finite number of iterations.

1. **Initiation step:**
   Set $n = 0$, $Z^{-1} = \emptyset$, and
   $Z^0 = \{z \in Z |$ either $\Sigma_u(P)(x_z) \not\subseteq \Sigma_u(P \Box S)(z)$ or z is a dead state$\}$

2. **Iteration step:**

   (a) Let $\hat{Z}_n \subseteq Z$ be the set of states from which $Z^n - Z^{n-1}$ can be reached in a single transition, i.e.

   $$\hat{Z}_n = \{z \in Z | \exists \sigma \in \Sigma \text{ s.t. } \gamma(\sigma, z) \in Z^n - Z^{n-1}\}$$

   The set $\hat{Z}_n$ is constructed by considering the SM obtained by reversing all the transitions in $P \Box S$.

   (b) Consider $z \in \hat{Z}_n$. If there exists an uncontrollable transition from z to Z^n or if all the transitions from z lead to Z^n, then $Z^{n+1} = Z^n \cup \{z\}$. Repeat this for every $z \in \hat{Z}_n$, i. e.

   $$Z^{n+1} = Z^n \cup \{z \in \hat{Z}_n | \text{ either } \exists \sigma_u \in \Sigma_u \text{ s.t. } \gamma(\sigma_u, z) \in Z^n; \text{ or } \forall \sigma \in \Sigma, \gamma(\sigma, z) \in Z^n\}$$

3. **Termination step:** If $Z^{n+1} = Z^n$, then stop; else set $n = n + 1$ and go to step 2.

Theorem 4.3 Let $n_0 \in \mathcal{N}$ be the number of the last iteration step in Algorithm 4.2. Then $Z_b = Z^{n_0}$.

Proof: We use induction to prove Theorem 4.3. We show that at every iteration step we remove only those states that are reached by accepting only the bad strings, and eventually we remove all such states. In other words, $Z^n \subseteq Z_b$ for each iteration $n \leq n_0$, and $Z_b \subseteq Z^{n_0}$.

Let $n \geq 0$ denote the number of iteration step. Then the above statement holds true for $n = 0$ (by definition, the set Z^0 contains states reached by only the bad strings). Let the statement be true for the nth iteration, i.e. assume that $Z^n \subseteq Z_b$. Consider $z \in \hat{Z}_n$ such that $\gamma(\sigma_u, z) \in Z^n$ for some $\sigma_u \in \Sigma_u$; then clearly $z \in Z^{n+1}$, for all the strings that lead to state z are uncontrollable and hence bad. If $z \in \hat{Z}_n$ is such that $\gamma(\sigma, z) \in Z^n$ for all $\sigma \in \Sigma$, then again $z \in Z^{n+1}$, for all the strings that lead to state z have no extension in the reduced SM obtained at the end of the nth iteration, and hence are bad. Since these are the only states added to Z^n at the nth iteration (for computing Z^{n+1}), $Z^{n+1} \subseteq Z_b$.

As n_0 is the number of the last iteration step, $Z^{n_0} = Z^{n_0+1}$. Hence none of the states in the set $Z - Z^{n_0}$ are reached by the bad strings, showing that the strings that reach the states in $Z - Z^{n_0}$ belong to the generator of $K^{\overline{\Uparrow}}$. Thus $Z_b = Z^{n_0}$. $\Box$

Corollary 4.4 Let $(P \Box S) |_{(Z - Z_b)} \stackrel{\text{def}}{=} ((Z - Z_b), \Sigma, \gamma |_{\Sigma \times (Z - Z_b)}, z_0, (Z - Z_b) \cap Z_m)$. Then $K^{\overline{\Uparrow}} = L((P \Box S) |_{(Z - Z_b)})$.

Proof: By definition, $Z_b \subseteq Z$ is the set of all the states that need to be removed from $P \square S$ in order to construct the generator for $K^{\overline{\Uparrow}}$, hence the result. $\square$

In view of Corollary 4.4, the generator for $K^{\overline{\Uparrow}}$ can thus be constructed using Algorithm 4.2.

Theorem 4.5 The computational complexity of the Algorithm 4.2 is of order $O(mn)$, where mn is the number of states in $P \square S$ (the number of states in SM's P, S are m, n respectively).

Proof: Assume that at the end of nth iteration the number of transitions leading into the set $Z^{n+1} - Z^n$ from $Z - Z^{n+1}$ is E_n. Then step 2 of the algorithm can be computed in $O(E_n)$ time. This follows, since (a) the states in the set $\hat{Z}_n$ can be computed by considering E_n transitions, and (b) there are at most E_n states in $\hat{Z}_n$, so the states to be added to Z^{n+1} (for computing Z^{n+2}) can be determined in $O(E_n)$ time.

Since the sets $Z^{n+1} - Z^n$ for each n are all disjoint, the transitions leading into them from $Z - Z^{n+1}$ are also all disjoint. Hence the states in the set Z_b can be computed in order $O(\sum_n E_n \leq E)$, where E denotes the number of transitions in $P \square S$. Since the SM's are deterministic, $E \leq |\Sigma| \cdot (mn)$; hence the theorem follows. $\square$

Remark 4.6 The computational complexity of Algorithm 4.2 for computing $K^{\overline{\Uparrow}}$ is of the same order as that of the optimal algorithm that computes $K^{\uparrow}$ presented in [9, 11].

4.1 Computation of $\mathcal{K}^{\uparrow}$

The computation of $\mathcal{K}^{\uparrow}$ using the formula of Theorem 3.13 involves computations of the operators $pr(\cdot)$, $(\cdot)^{\overline{\Uparrow}}$ and $(\cdot)^{\infty}$. We discussed above computation of the operator $(\cdot)^{\overline{\Uparrow}}$. Next we discuss computations of the other two operators.

Lemma 4.7 [4, Proposition 1.1, Chapter 14] Consider a trim SM $V \stackrel{\text{def}}{=} (Q, \Sigma, \delta, q_0, Q_m)$. Then $(L_m(V))^{\infty} = \mathcal{L}(V)$.

Thus if V is live so that $L_m(V)$ is complete (Lemma 3.5), $\mathcal{L}(V)$ can be written as the limit of a complete language.

Lemma 4.8 [8] Consider a deterministic trim SM $V \stackrel{\text{def}}{=} (Q, \Sigma, \delta, q_0, Q_m)$. Then V is nonblocking, i.e. $pr\mathcal{L}(V) = L(V)$, if and only if it is live.

Corollary 4.9 [8] Let V be as in Lemma 4.8. Then $(L(V))^{\infty} = \overline{\mathcal{L}(V)}$.

Remark 4.10 The implication of Lemma 4.7 is that given a trim SM that recognizes a language $L \subseteq \Sigma^{\star}$, L^{∞} is the ω-language generated by the same SM, and the implication of Lemma 4.8 is that given a *live* trim SM that generates the ω-language $\mathcal{L}$, $pr\mathcal{L}$ is the finite language generated by the same SM. Thus the supremal ω-language $\mathcal{K}^{\uparrow}$ of a given topologically closed ω-language $\mathcal{K} \subseteq \Sigma^{\omega}$, generated by a deterministic FSM (i.e. $\mathcal{K} \in DRAT$ [4]) with respect to plant P can easily be computed in view of Theorem 3.13, Algorithm 4.2, Lemma 4.7 and Lemma 4.8. The complexity of the computation is of order $O(mn)$, where m, n are the number of states in the SM's representing P, generator of $pr\mathcal{K}$ respectively (Theorem 4.5).

Example 4.11 (continued) Consider the plant P and the supervisor S shown in Figure 1. Then as mentioned in Example 2.1, $P \square S$ has the same structure as that of the machine S. Let the uncontrollable event set $\Sigma_u \subseteq \Sigma$ be the singleton containing the event u. We make an additional assumption in this example that all the states in P as well as in S are marked. Hence $L_m(P) =$

$L(P) = \overline{(a+b)^\star ca^\star(b+u)a^\star}$, $L_m(S) = L(S) = \overline{(ca+(ab)^\star ac(u+ba))a^\star}$, $\mathcal{L}(P) = (a+b)^\omega + (a+b)^\star ca^\omega + (a+b)^\star ca^\star(u+b)a^\omega$, and $\mathcal{L}(S) = (ab)^\omega + (ca+(ab)^\star ac(u+ba))a^\omega$. We note that $\mathcal{L}(P) = (L(P))^\infty$ and $\mathcal{L}(S) = (L(S))^\infty$, which also follows from Lemma 4.7.

Let the desired sequential behavior $\mathcal{K} \subseteq \mathcal{L}(P)$ be given by $\mathcal{K} = \mathcal{L}(S)$. Since S is a live trim machine, it follows from Lemma 4.8 that $pr\mathcal{K} = L(S)$. Thus $\overline{\mathcal{K}} = (pr\mathcal{K})^\infty = (L(S))^\infty = \mathcal{L}(S) = \mathcal{K}$, showing that $\mathcal{K}$ is topologically closed. Hence it follows from Lemma 3.12 that $\mathcal{K}$ is topologically closed with respect to $\mathcal{L}(P)$. Since the string $c \in pr\mathcal{K}$ and the string $cu \in L(P) - pr\mathcal{K}$, it follows that $pr\mathcal{K}$ is not controllable with respect to P; thus $\mathcal{K}$ is not ω-controllable with respect to P. Hence we cannot construct a supervisor so that the closed loop sequential behavior is $\mathcal{K}$ [14]. However, since $\mathcal{K}$ is topologically closed, $\mathcal{K}^\uparrow$, the supremal ω-controllable sublanguage of $\mathcal{K}$, exists (Lemma 3.12 and Proposition 3.2 in [14]), so that the construction of the minimally restrictive supervisor is possible. Thus the existence of the minimally restrictive supervisor is guaranteed; we use the closed form expression in Theorem 3.13 to compute $\mathcal{K}^\uparrow$.

First we need to compute $\overline{(pr\mathcal{K})^\Uparrow}$. From the above discussions, $pr\mathcal{K} = L(S)$. We employ Algorithm 4.2 for computing $\overline{(pr\mathcal{K})^\Uparrow}$. Since the machine $P\Box S$ is the same as the machine S, no further construction is need for obtaining the machine $P\Box S$. Also, since the string $c \in pr\mathcal{K} = L(S)$ is the only bad string in $pr\mathcal{K}$, $Z^0 = \{[c](R_{P\Box S})\} = \{[c](R_S)\}$, where Z^0 is defined in Algorithm 4.2. It then follows from Figure 1 that $\hat{Z}_0 = \{[\epsilon](R_S), [ac](R_S)\}$, where $\hat{Z}_0$ is the set of states in $P\Box S$ (=S), from which Z^0 can be reached in a single transition. Since not all the transitions from the states in the set $\hat{Z}_0$ lead to Z^0, and since no uncontrollable transition from the states in the set $\hat{Z}_0$ lead to Z^0, it follows that $Z^1 = Z^0 = Z_b$, where Z^1 and Z_b are defined in Algorithm 4.2. Hence the generator for $\overline{(pr\mathcal{K})^\Uparrow}$ is given by the SM shown in Figure 2, which is obtained by removing the state in the set Z_b from S. Thus $\overline{(pr\mathcal{K})^\Uparrow} = \overline{(ab)^\star acua^\star}$. It then follows from Lemma 4.7 that $(\overline{(pr\mathcal{K})^\Uparrow})^\infty$ is the ω-language generated by the SM in Figure 2 and is given by $(\overline{(pr\mathcal{K})^\Uparrow})^\infty = (ab)^\omega + (ab)^\star acua^\omega$. Hence from Theorem 3.13, $\mathcal{K}^\uparrow = (\overline{(pr\mathcal{K})^\Uparrow})^\infty = (ab)^\omega + (ab)^\star acua^\omega$.

If $b \in \Sigma$ is also given to be an uncontrollable event, then it is clear from Figure 1 that there exists an uncontrollable transition, namely b, from the state $[ac](R_S) \in \hat{Z}_0$ leading to Z^0. Hence $Z^1 = Z^0 \cup \{[ac](R_S)\} = \{[c](R_S), [ac](R_S)\}$. It can be argued as above to conclude that in this case, $Z^2 = Z^1 = Z_b$. Hence, in this case, the generator for $\overline{(pr\mathcal{K})^\Uparrow}$ is given by the SM in Figure 1 which is obtained by removing the states in the set Z^2 from S. It then follows from Theorem 3.13 that in this case, $\mathcal{K}^\uparrow = (ab)^\omega$.

5 Partially Observed Sequential Behavior

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 (refer to the example in [3, p. 259]). In this section we consider the supervisory synthesis problem for a plant whose sequential behavior is partially observed.

The definition of the mask M can be easily extended to the space Σ^ω. First we prove the following result:

Lemma 5.1 Let $\{s_n\}_{n\in\mathcal{N}}, \{t_m\}_{m\in\mathcal{N}}$ be any two sequences of strings such that $s_n \in \Sigma^\star, s_n < s_{n+1}$ for each n, $t_m \in \Sigma^\star, t_m < t_{m+1}$ for each m, and $\lim_{n\to\infty} s_n = \lim_{m\to\infty} t_m$. Then $\lim_{n\to\infty} M(t_n) = \lim_{m\to\infty} M(t_m)$.

Proof: Since $\lim_{n\to\infty} s_n = \lim_{m\to\infty} t_m$, we have for each m there exist $n_m, n_{m+1} \in \mathcal{N}$ such that $s_{n_m} < t_m < s_{n_{m+1}}$. Since M distributes over concatenation, we have $M(s_{n_m}) < M(t_m) < M(s_{n_{m+1}})$ for each m. Similarly, it can be shown that for each n there exist $m_n, m_{n+1} \in \mathcal{N}$ such that $M(t_{m_n}) < M(s_n) < M(t_{m_{n+1}})$. Hence $\lim_{n\to\infty} M(s_n) = \lim_{m\to\infty} M(t_m)$. $\Box$

Using the result of Lemma 5.1, we can extend the definition of M to the space Σ^ω as follows:

Definition 5.2 Consider a infinite string $e \in \Sigma^\omega$; let $\{s_n\}_{n\in\mathcal{N}}$ be a sequence of strings such that $s_n \in \Sigma^\star, s_n < s_{n+1}$ for each n and $\lim_{n\to\infty} s_n = e$. Then $M(e)$ is defined to be, $M(e) \stackrel{\text{def}}{=} \lim_{n\to\infty} M(s_n)$.

Thus if $\mathcal{L}(P)$ denotes the infinite string behavior of the plant P, then the behavior observed under the mask M is given by $M(\mathcal{L}(P))$. Let $\mathcal{K} \subseteq \Sigma^\omega$ denote the desired closed loop sequential behavior. We introduce the notion of *ω-observability* of ω-languages and state a necessary and sufficient condition for the existence of a supervisor for achieving the desired closed loop sequential behavior.

Definition 5.3 $\mathcal{K} \subseteq \Sigma^\omega$ is said to be *ω-observable* with respect to P and M if $\mathcal{K}$ is topologically closed with respect to $\mathcal{L}(P)$ and $pr\mathcal{K}$ is observable with respect to P and M.

Assume that all the events are controllable.

Theorem 5.4 Let $\mathcal{K} \subseteq \mathcal{L}(P)$ be nonempty. Then there exists a nonblocking supervisor S such that $\mathcal{L}(P\Box S) = \mathcal{K}$ if and only if $\mathcal{K}$ is ω-observable with respect to P and M.

Proof: Assume that $\mathcal{K}$ is ω-observable. Since $pr\mathcal{K}$ is observable and nonempty with respect to P and M, there exists a supervisor S such that $L(P\Box S) = pr\mathcal{K}$ [13, 3]. Hence $\mathcal{L}(P\Box S) = (L(P\Box S))^\infty \cap \mathcal{L}(P) = (pr\mathcal{K})^\infty \cap \mathcal{L}(P) = \overline{\mathcal{K}} \cap \mathcal{L}(P) = \mathcal{K}$, where the last equality follows from the fact the $\mathcal{K}$ is topologically closed with respect to $\mathcal{L}(P)$. Also $pr(\mathcal{L}(P\Box S)) = pr\mathcal{K} = L(P\Box S)$; thus S is nonblocking.

Next assume that there exists a nonblocking supervisor S such that $\mathcal{L}(P\Box S) = \mathcal{K}$. Then $L(P\Box S)$ is observable [13, 3]], and $pr(\mathcal{L}(P\Box S)) = L(P\Box S)$ (since S is nonblocking). Thus $pr\mathcal{K} = pr(\mathcal{L}(P\Box S)) = L(P\Box S)$, showing that $pr\mathcal{K}$ is observable. Also $\overline{\mathcal{K}} \cap \mathcal{L}(P) = (pr\mathcal{K})^\infty \cap \mathcal{L}(P) = (L(P\Box S))^\infty \cap \mathcal{L}(P) = \mathcal{L}(P\Box S) = \mathcal{K}$; hence, $\mathcal{K}$ is closed with respect to $\mathcal{L}(P)$. $\Box$

Remark 5.5 Since the observability of languages is not preserved under union, the *supremal ω-observable* sublanguage of a given ω-language does not exist. Also, since the prefix of ω-languages is not preserved under intersection, the *infimal ω-observable* superlanguage of a given ω-language does not exist either. We introduce the notion of *ω-normality* that is stronger notion than that of ω-observability, and is preserved under union.

Definition 5.6 An ω-language $\mathcal{K} \subseteq \mathcal{L}(P)$ is said to be *ω-normal* with respect to the plant P and the mask M if $\mathcal{K}$ is topologically closed with respect to $\mathcal{L}(P)$, and $pr\mathcal{K}$ is normal with respect to P and M, i. e.

1. $\overline{\mathcal{K}} \cap \mathcal{L}(P) = \mathcal{K}$, and
2. $M^{-1}M(pr\mathcal{K}) \cap L(P) = pr\mathcal{K}$

Remark 5.7 Since normality of languages is preserved under union, it follows from Remark 3.19 that the supremal ω-normal sublanguage of a given ω-language exists. Thus if the desired behavior is not ω-observable, a supervisor can be constructed so that the closed loop system generates the supremal ω-normal sublanguage of the desired behavior. Letting $\mathcal{K}^\circ$ denote the supremal ω-normal sublanguage of a given topologically closed ω-language $\mathcal{K} \subseteq \Sigma^\omega$, it follows from Remark 3.19 that it is given by the limit of the supremal complete, closed and normal sublanguage of $pr\mathcal{K}$.

6 Conclusion

In this paper, we have used the notion of *synchronous composition* for describing the control of sequential behavior of DEDS's.

We have introduced the notion of complete languages, and shown that these languages are closely related to ω-languages. The supremal complete, closed and controllable sublanguage of a language exists, and an algorithm for constructing it has been presented. A closed form expression for the supremal ω-controllable sublanguage of a given ω-language has been derived using the supremal complete, closed and controllable sublanguage. This result indicates that certain operations on ω-languages can alternatively be performed by first performing certain other similar operations on their prefixes (which are finite string languages) and then taking the limits (to obtain the desired ω-language). Thus a computational problem on the space of infinite languages can be reduced to another similar computational problem on the space of finite languages, for the operations of computing the prefix of an ω-langauge and the limit of a (finite string) language can be easily performed.

The problem of supervisory synthesis is then extended to deal with controlling the sequential behavior under partial observation. We introduce the notion of ω-observability and show that it is a necessary and sufficient condition for the existence of the supervisor. ω-observability is not preserved under union or intersection. Hence a stronger notion called ω-normality that is algebraically better behaved has been presented. We prove that the supremal ω-normal sublanguage exists and is unique, and describe a method for computing it.

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. Supremal and maximal sublanguages arising in supervisor synthesis problems with partial observations. *Mathematical Systems Theory*, 22:177–211, 1989.

[3] 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.

[4] S. Eilenberg. *Automata, Languages, and Machines: Volume A.* Academic Press, New York, NY, 1974.

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

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

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

[8] R. Kumar. *Supervisory Synthesis Techniques for Discrete Event Dynamical Systems: Transition Model Based Approach.* PhD thesis, Department of Electrical and Computer Engineering, University of Texas at Austin, 1991.

[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] R. Kumar, V. K. Garg, and S. I. Marcus. Language stability of deds. In *Proceedings of 1990 International Conference on Mathematical Theory of Control*, Indian Institute of Technology, Bombay, India, December 1990.

[11] R. Kumar, V. K. Garg, and S. I. Marcus. On controllability and normality of discrete event dynamical systems. *Systems and Control Letters*, 17(3):157–168, 1991.

[12] R. Kumar, V. K. Garg, and S. I. Marcus. On ω-controllability and ω-normality of deds. In *Proceedings of 1991 ACC*, pages 2905–2910, Boston, MA, June 1991.

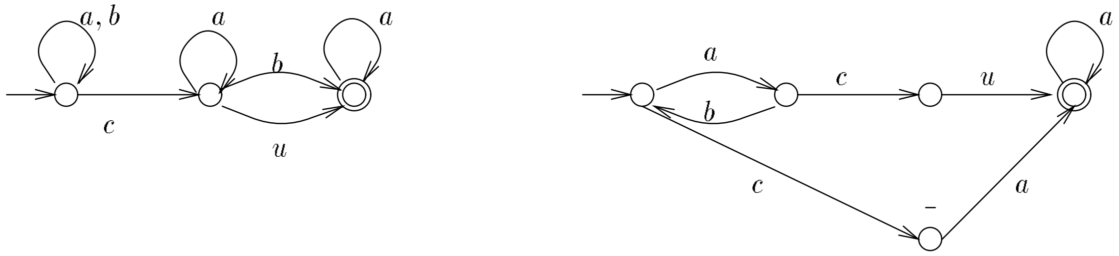
[13] F. Lin and W. M. Wonham. On observability of discrete-event systems. *Information Sciences*, 44(3):173–198, 1988.

[14] P. J. Ramadge. Some tractable supervisory control problems for discrete event systems modeled by buchi automata. *IEEE Transactions on Automatic Control*, 34(1):10–19, 1989.

[15] 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.

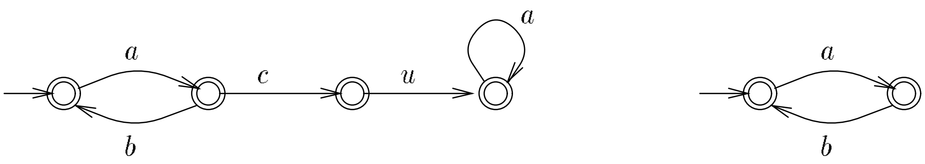
[16] P. J. Ramadge and W. M. Wonham. Supervisory control of a class of discrete event processes. *SIAM Journal of Control and Optimization*, 25(1):206–230, 1987.

[17] J. G. Thistle and W. M. Wonham. On the synthesis of supervisors subject to ω-language specifications. In *Proceedings of 22nd Annual Conference on Information Sciences and Systems*, pages 440–444, Princeton, NJ, 1988.



P **S**

Figure 1: Example to illustrate synchronous composition



Generator for $(pr\mathcal{K})^{\overline{\Uparrow}}$ $(\Sigma_u = \{u\})$ **Generator for** $(pr\mathcal{K})^{\overline{\Uparrow}}$ $(\Sigma_u = \{b, u\})$

Figure 2: Example to illustrate computation of $\mathcal{K}^{\uparrow}$