TASK-DRIVEN SUPERVISORY CONTROL OF DISCRETE EVENT SYSTEMS

C.H. Golaszewski and R.P. Kurshan

AT&T Bell Laboratories, Murray Hill, NJ 07974

Abstract

The supervisory control framework formulated by Ramadge and Wonham is extended to allow the synthesis of supervisors which control a given system to perform an arbitrary ω-regular task specified by a nondeterministic Büchi automaton. To this end, the supervisory control paradigm is applied to R.P. Kurshan's L-processes which provide a convenient model for nonterminating discrete event system behaviors. Necessary and sufficient conditions for the existence of supervisors are derived and the synthesis of supervisors is discussed.

1. Introduction

One of the goals of supervisory control for discrete event systems is the synthesis of supervisors which guarantee that the closed-loop (or supervised) system performs certain prescribed tasks. In this paper a task is assumed to be specified by an ω-regular language defined by a Büchi automaton. The given discrete event system is not constrained to be completely observable, i.e., it may be required to synthesize a supervisor for the system which cannot observe all system events and which may not possess complete state information about the system.

To motivate our study, consider a communication system consisting of a sender A, a receiver B and an unreliable channel C which nondeterministically loses packets. A standard problem is to design a supervisor (communication protocol) that ensures that a packet sent by A is eventually received by B, under the assumption that the channel eventually delivers some message which was transmitted by A. The channel can delay packets arbitrarily long, the loss of a message is unobservable and all components operate asynchronously. Thus, based on the past observations of the system trajectory it cannot be decided whether a packet which was not received was actually lost or will be received at some point in the future. Consequently, a supervisor which is to guarantee packet delivery must at some point cause retransmission of that message, irrespective of the event of packet loss. The timing of this retransmission is not critical, and for efficiency may be adjusted with consideration of system component delay. Therefore, it is natural to model the supervisory event which causes retransmission as a nondeterministic control action. However, in order for the supervisor to fulfill its required task, it will often be required to satisfy certain obligations eventually. These considerations lead in a natural way to the definition of the supervisor as a nondeterministic automaton with acceptance conditions for infinite sequences.

The major new contribution of this paper is a synthesis procedure for Büchi supervisors. These can be understood as supervisors that for certain states of the system may choose nondeterministically between several control actions. The choice is not forever

arbitrary, however, as we require that eventually some particular sequence of control actions is selected; this ultimate occurrence of sequences of control actions is characterized by automata acceptance conditions. Allowing nondeterminism in the supervisor permits to defer certain timing and sequencing decisions to a lower level of system design, while at the same time providing "place holders" during the high-level supervisor synthesis. It was already illustrated with the previous example that such an extension is needed for the synthesis of communication protocols. Lacking this feature, the RW-Model cannot be applied directly to the synthesis of, say, communication protocols as they are conventionally modeled. For example, Cieslak et al. had to include the structure of the alternating bit protocol in the model of the open-loop system in order to derive their solution in [2]. Clearly, a general synthesis procedure should not require much knowledge about the solution. This suggests that the basic RW-Model lacks certain features that are necessary to synthesize a supervisor for this class of systems.

The present investigation is by no means the first attempt to extend the RW-Model so that it can be applied to a larger class of supervisory control problems. For example, recently it was realized by Thistle and Wonham, and Ramadge that one important aspect absent from the basic model is the notion of periodic and asymptotic behaviors of discrete event systems. This led to proposals for extensions of the RW-Model to include infinite sequences ([8,10]). These approaches are very similar in spirit in that they both extend the notion of string controllability of the basic model to the prefix set of (infinite) sequences. The extended model was used to pose and solve several interesting supervisory control problems ([3,8]). However, these extensions do not address the problem of achieving a desired sequential closed-loop behavior which is neither realizable ([10]) nor topologically closed ([8]). Simply stated, this framework applies only to the case of transition structures which are structurally incapable of generating infinite sequences incompatible with the given task. We show in this paper that this is too restrictive in the context of communication protocols. Instead, by defining the supervisor as a nondeterministic L-process, we may place constraints on the eventuality of the closed-loop behavior to exclude arbitrarily long but finite periods of behavior, which, if continued forever, would constitute a failure of the system.

Some of our research overlaps with work done by Brave et al. in [1] and Özveren et al. in [6]; our independently derived results differ from the work of both of these groups, though. Their research is focused on the synthesis of deterministic supervisors which guarantee that the closed-loop system visits a given subset of states infinitely often. In contrast, we use a formal framework for the infinite behavior of discrete event systems, which includes Büchi supervisors and L-processes, to pose and solve supervisory synthesis problems for ω-regular tasks.

The organization of the paper is as follows. In Section 2 the discrete event system model is reviewed. Section 3 introduces the control-theoretic framework used in the paper. Section 4 discusses the existence and synthesis of supervisors for fully observable systems and in Section 5 these results are extended to the general case.

2. L-Processes

In this paper, the logical behavior of discrete event systems is modeled by *L-processes*. The reason is that L-processes are a class of automata with a simple (polynomial)

product and the property that the language of the product system is the intersection of the languages of the subsystems. We will review only the basic definitions; for a detailed presentation we refer to [4].

Let $L = \langle \mathbf{L}, +, *, \sim, 0, 1 \rangle$ be a Boolean algebra over the set L. A Boolean algebra L admits a partial order $\leq$ defined by $x \leq y$ if and only if $x * y = x$. An *atom* of L is a minimal element with respect to this order.

Let $L_1, \ldots, L_k$ be subalgebras of L. Define their product

$$\prod_{i=1}^{k} L_i = \left\{ \sum_{j \in J} x_{1j} * \ldots * x_{kj} | x_{ij} \in L_i \text{ for } i = 1, \ldots, k, j \in J \text{ and } J \text{ is finite} \right\}$$

Clearly, $\prod L_i$ is a subalgebra of L. $L_1, \ldots, L_k$ are *independent* if

$$0 \neq x_i \in L_i \text{ for } i = 1, \ldots, k \Rightarrow x_1 * \ldots * x_k \neq 0.$$

Subalgebras will be used later to describe the concurrent operation of several discrete event processes.

Let V be a non-empty set. The map $M : V \times V \to L$ is an *L-matrix* with state space $V(M)$; the elements of $V(M)$ are the *states* of M. It is *deterministic* if for all $u, v, w \in V(M)$, $v \neq w \Rightarrow M(u, v) * M(u, w) = 0$, and *complete* if for all $v \in V(M)$

$$\sum_{w \in V(M)} M(v, w) = 1.$$

Let M and N be L-matrices over the same Boolean algebra L. Their *tensor product* $M \otimes N$ is defined by

$$(M \otimes N)\big((v, w), (v', w')\big) = M(v, v') * N(w, w').$$

Given an L-matrix M, a sequence $s \in L^\omega$ and a sequence of states $w \in V(M)^\omega$, we call w a *state trajectory* of s starting at q if $w_o = q$ and $s_i * M(w_i, w_{i+1}) \neq 0$ for all i.

An *L-automaton* is a quadruple $\Gamma = (M_\Gamma, I(\Gamma), R(\Gamma), Z(\Gamma))$, where M_Γ, the *state transition matrix*, is a complete L-matrix, $\emptyset \neq I(\Gamma) \subseteq V(M_\Gamma)$ are the *initial states*, $U(\Gamma) \subseteq V(M_\Gamma)$ are the *recurring states* (Büchi states), and $Z(\Gamma) \subseteq 2^{V(M_\Gamma)}$ is the *cycle set.*

Denote the set of all sequences of atoms of L by $S(L)^\omega$. A sequence $s \in S(L)^\omega$ is Γ-*cyclic* if for some $N > 0$ and some $C \in Z(\Gamma)$, the corresponding state trajectory w satisfies $w_i \in C$ for all $i > N$; it is Γ-*recurring* if $w_i \in U(\Gamma)$ for infinitely many i. A sequence $s \in S(L)^\omega$ is a *tape* of Γ if it is Γ-cyclic or Γ-recurring. The *behavior* (or language) $\mathcal{L}(\Gamma)$ accepted by Γ is the set of tapes of Γ.

An L-process G is a quintuple $G = (M_G, S_G, I(G), U(G), Z(G))$, where M_G, $I(G)$, $U(G)$, and $Z(G)$ are as in the definition of an L-automaton in the previous section. However, M_G is not required to be complete. Finally, S_G is the nondeterministic *selection function* $S_G : V(G) \to 2^L$. We restrict attention to the case where $S_G(q)$ is an atom for every $q \in V(G)$.

A selection $x \in S_G(q)$ enables the state transition $q \to p$ if $x * M_G(q, p) \neq 0$. We

also require that

$$\sum_{p \in V(G)} M_G(q,p) \leq \sum_{x \in S_G(q)} x. \tag{2.1}$$

If equality holds in (2.1) for all $q \in V(G)$, and if $0 \notin S_G(q)$, then G is said to be *lockup-free.* (This is similar to the notion of nonblocking in [8].)

In contrast to the definition of a tape for an L-automaton, an element s of $S(L)^\omega$ is a tape for G if and only if it is not G-cyclic and not G-recurring. If $s \in S(L)^\omega$ is not a tape for G because it is G-cyclic or G-recurring, then s is said to be *excepted* from the language of G. The *behavior* (or language) $\mathcal{L}(G)$ generated by an L-process G is the set of all tapes of G.

The concurrent operation of several discrete event processes is modeled as follows: each individual component system is represented by an L_i-process G_i over a Boolean algebra L_i. The algebras L_i can be interpreted as subalgebras of a Boolean algebra L which is given by the product $L = \prod_i L_i$. This procedure associates with each L_i-process a corresponding L-process.

Given L-processes $G_1, \ldots, G_k$ define their *synchronous product* to be

$$G = \bigotimes_{i=1}^{k} G_i = (M_G, S_G, I(G), U(G), Z(G)) =$$

$$= \left(\bigotimes_{i=1}^{k} M_{G_i}, \prod_{i=1}^{k} S_{G_i}, I(G_1) \times \ldots \times I(G_k), \bigcup_{i=1}^{k} \pi_{G_i}^{-1} U(G_i), \bigcup_{i=1}^{k} \pi_{G_i}^{-1} Z(G_i) \right),$$

where π_G denotes the projection $G \times H \to G$ and

$$\left(\prod_{i=1}^{k} S_{G_i} \right) (q_1, \ldots, q_k) = \{x_1 * \ldots * x_k | x_i \in S_{G_i}(q_i), i = 1, \ldots, k\}.$$

The model described so far applies to the concurrent operation of synchronous discrete event processes, i.e., state transitions occur simultaneously in the component processes. To model partially and completely asynchronous selections we introduce two additional selections called *wait* and *pause*, respectively. A process can select *pause* independently of the selections of the other processes and, for example, use *pause* to self-loop for an undetermined amount of "time." The *wait* selection is used to model the case where the possible state transitions of a process depend on the selections of other processes. Until this process can proceed it remains at the current state (in a self-loop) by selecting *wait*.

Consider two alphabets Σ_1 and Σ_2, and let $\phi: \Sigma_1 \to \Sigma_2$ be an arbitrary map. We can extend ϕ to a *language homomorphism* $\Sigma_1^\omega \to \Sigma_2^\omega$ in the usual fashion by setting $\phi(s) = \phi(s_1)\phi(s_2)\ldots$ for all $s \in \Sigma_1^\omega$.

Let G_1 be an L_1-process over the alphabet Σ_1 and G_2 be an L_2-process over the alphabet Σ_2. The pair $\Phi = (\varphi, \phi)$, with $\varphi: V(G_1) \to V(G_2)$ and ϕ a language

homomorphism, is said to be a *process homomorphism* if

$$\sigma \in M_1(v,w) \Rightarrow \phi(\sigma) \in M_2(\varphi(v),\varphi(w)). \tag{2.2}$$

Now let G be an L-process, not necessarily deterministic. Define a binary relation Ψ on $V(G) \times V(G)$ according to

(1) for all $q \in V(G)$, $(q,q) \in \Psi$, and
(2) $(p,q) \in \Psi$ if there exists $(u,v) \in \Psi$ such that $\sigma \in M(u,p)$ and $\sigma \in M(v,q)$ for some $\sigma \in \Sigma$.

A pair of states $(p,q) \in \Psi$ is said to be *indistinguishable*. Note that in general Ψ is not an equivalence relation.

For two relations Ψ_1 and Ψ_2, a partial order is defined in the usual fashion:

$$\Psi_1 \leq \Psi_2 \quad \Leftrightarrow \quad \big((p,q) \in \Psi_1 \Rightarrow (p,q) \in \Psi_2\big).$$

Partial or incomplete observations of the tapes of G are modeled by letting the language homomorphism ϕ be a projection onto a subset of the alphabet. In this case, $\ker(\phi) = \{\sigma \in \Sigma : \phi(\sigma) = 1\}$ induces a binary relation on $V(G) \times V(G)$ via (2.2).

3. Tasks and Supervisors

For a given L-process G with behavior $\mathcal{L}(G)$ we can specify its admissible behavior $\mathcal{L}$. In this paper $\mathcal{L}$ is assumed to be an ω-regular language. Thus a task can be represented by a deterministic L-automaton T. A standard verification problem is to decide if $\mathcal{L}(G) \subseteq \mathcal{L}$. Suppose this condition fails to hold, but there exists some L-process C such that $\mathcal{L}(G \otimes C) \subseteq \mathcal{L}$. In situations where G is independent from its environment this cannot happen. However, it is often the case that some (but not necessarily all) of the state transitions of G depend on shared or controllable variables which can be accessed by other processes. Thus, the language $L(G)$ can be restricted to a subset by external *control selections*.

We call the L-process C in (3.1), if it exists, a *Büchi supervisor* (or supervisor) for G with respect to the *task* $\mathcal{L}$. Note that we permit a supervisor to be an L-process with nondeterministic selection function. Hence, certain control selections (or sequences of control selections) can be required to occur eventually. For example, in the model of a communication protocol an eventual control action may represent the fact that the protocol will retransmit a window after a time-out has occurred.

Let $h: V(G) \to 2^{V(C)}$ be the map induced by $G \otimes C$, which for each tape in $\mathcal{L}(G \otimes C)$ assigns to the present state q of G the corresponding state(s) $h(q)$ of C. A state transition $M(q,p)$ of G is *disabled*, i.e., prevented from occurring, if

$$S_G(q) * S_C(h(q)) * M(q,p) = 0;$$

otherwise $M(q,p)$ is *enabled* by C.

Define an equivalence relation Θ on $V(G) \times V(G)$ according to

$$\Theta = \{(q,p): \quad S_C(h(q)) = S_C(h(p))\}$$

Let $B \subseteq V(G)$ be an arbitrary subset. Say that B is *controlled-invariant* if there

exists a supervisor C such that every state-trajectory of the process $G \otimes C$, once it enters B, remains indefinitely inside B. A state $q \in V(G)$ is *B-steerable*, if there exists a supervisor C such that for all $p \in V(G)$, $S_C(h(q)) * M_G(q,p) = 1$, implies that p is either B-steerable or an element of B.

Now consider two tasks T_1 and T_2 and suppose C_1 and C_2 are two supervisors with

$$\mathcal{L}(G \otimes C_1) = \mathcal{L}_1 \subseteq \mathcal{L}(T_1) \quad \text{and} \quad \mathcal{L}(G \otimes C_2) = \mathcal{L}_2 \subseteq \mathcal{L}(T_2).$$

It is natural to require that supervisors which control the same system do not share control variables. From Proposition 4.9 in [6] it follows that if G, C_1 and C_2 are independent L-processes, i.e., there exist independent subalgebras L_G, L_1 and L_2 such that

$$S_G = \bigcup_{q \in V(G)} S_G(q) \subseteq L_G, \quad S_{C_1} = \bigcup_{q \in V(C_1)} S_{C_1}(q) \subseteq L_1$$

and

$$S_{C_2} = \bigcup_{q \in V(C_2)} S_{C_2}(q) \subseteq L_2,$$

then

$$\mathcal{L}(G \otimes C_1 \otimes C_2) = \mathcal{L}_1 \cap \mathcal{L}_2,$$

and the resulting product system is lockup-free.

Now suppose we are given an L-process G, a task T and a supervisor C satisfying $\mathcal{L}(G \otimes C) \subseteq \mathcal{L}(T)$, i.e., G is controllable relative to T. Furthermore suppose that for some reason (e.g. distributed implementation) we desire to partition T into subtasks $T_1, \ldots, T_n$ with $\mathcal{L}(T) = \mathcal{L}(T_1) \cap \ldots \cap \mathcal{L}(T_n)$. Since there is a supervisor for T there also exists a supervisor C_i which implements task T_i. However, if there are two tasks T_i and T_j such that any pair of supervisors C_i and C_j, which implement the tasks, is dependent, then it is impossible to realize T through the given subtasks $T_1, \ldots, T_n$.

In certain cases it is possible to eliminate the dependency between supervisors by introducing new variables. For example, let $x \in S_{C_1} \cap S_{C_2}$ be a "shared" control variable and introduce two variables $x_1 \notin S_G \cup S_{C_2}$ and $x_2 \notin S_G \cup S_{C_1}$ (possibly by extending $\mathbf{L}$) and set $x = f(x_1, x_2)$ where f is an appropriately chosen Boolean function. At the present time it is not clear when such an f exists; this question is currently under investigation.

4. Existence of Supervisors: Deterministic Case

Consider an L-process G with deterministic resolution and a task T. Our goal is to derive necessary and sufficient conditions for the existence of a lockup-free supervisor C such that $\mathcal{L}(G \otimes C) \subseteq \mathcal{L}(T)$. Intuitively, such a supervisor C exists if all state trajectories which originate in $I(G)$ and reach a void cycle of uncontrollable events, can be disabled by appropriately selected control actions.

Recall that any tape of $\mathcal{L}(G)$ has to be T-cyclic or T-recurring to be a tape for $\mathcal{L}(T)$. Since the two acceptance conditions are independent of each other, they are investigated separately in the next two sections.

4.1 Largest Invariant Recurrent Set

The case of recurrent sets is similar to the problem considered in [1] and [6] in the context of the RW-Model for supervisory control. The following algorithm computes the largest invariant recurrent set via a fixpoint iteration:

Algorithm 1:

$K_1 := \pi_G(V(G \otimes T))$;
$H_1 := \pi_G(U(G \otimes T))$;
REPEAT
  $K_{i+1} := \{q \in K_i : q \text{ is } H_i - \text{steerable } \}$;
  $H_{i+1} := H_i \cap K_{i+1}$;
UNTIL $H_{i+1} = H_i$;

Denote the (unique) fixpoint of the algorithm by $H(G) = \lim_{i \to \infty} K_i$. Since we consider only finite state systems it is clear that the algorithm converges in a finite number of steps.

Proposition 4.1. *The set $H(G)$ is controlled invariant.*

Proof. Every state in $H(G)$ is steerable to some other state in $H(G)$. ◊

Proposition 4.2. *Suppose $I(G) \subseteq H(G)$. Then there exists a lockup-free supervisor C such that $\mathcal{L}(G \otimes C) \subseteq \mathcal{L}(T)$.*

Proof. Follows from control invariance and the fact that the states of every cycle which does not intersect $U(G \otimes T)$ are $U(G \otimes T) \cap H(G)$-steerable. ◊

Proposition 4.3. *Suppose $Z(G \otimes T) = \emptyset$. Then there exists a lockup-free supervisor C such that $\mathcal{L}(G \otimes C) \subseteq \mathcal{L}(T)$ if and only if $I(G) \subseteq H(G)$.*

Proof. (if.) This is the statement of Proposition 4.2.

(only if.) Suppose there is a supervisor such that $\mathcal{L}(G \otimes C) \subseteq \mathcal{L}(T)$. Since $Z(G \otimes T) = \emptyset$ this implies that C can control G in such a way that every state trajectory intersects the set $U(G \otimes T)$ infinitely often, irrespective of the initial state. But now it follows from the construction of the set $H(G)$ that $I(G) \subseteq H(G)$. ◊

The last proposition implies that, in the absence of cycle sets, Algorithm 1 produces the largest set of initial states for which a lockup-free supervisor with the desired properties exists.

4.2 Cycle Sets

We proceed in a similar fashion to derive conditions on the set of initial states with respect to the cycle set.

Algorithm 2:

FOR ALL $K \in Z(G \otimes T)$ DO
  $A(K) :=$ largest controlled invariant set contained in $\pi_G(K)$;
  $H(K) := \{q \in V(G) : q \text{ is } A(K) - \text{steerable } \}$;

The computation of the sets $A(K)$ can be done with a variant of Algorithm 1:

$V_1 := \pi_G(K)$;
$H_1 := \pi_G(K)$;
REPEAT
  $V_{i+1} := \{q \in V_i : q \text{ is } H_i - \text{steerable } \}$;
  $H_{i+1} := H_i \cap V_{i+1}$;
UNTIL $H_{i+1} = H_i$;

For each $K \in Z(G \otimes T)$ set $A(K) = \lim_{i \to \infty} V_i$, i.e., $A(K)$ is the fixpoint of the above procedure.

The following two propositions summarize the properties of the sets $H(K)$:

Proposition 4.4. *Suppose $I(G) \subseteq \bigcup_{K \in Z(G \otimes T)} H(K)$. Then there exists a lockup-free supervisor C such that $\mathcal{L}(G \otimes C) \subseteq \mathcal{L}(T)$.*

Proof. We can assume without loss that there is only one set $H(K)$. Consider the set $D = H(K) - A(K)$. Since each state in D is $A(K)$-steerable it follows that the states on every cycle in D not contained in $Z(G \otimes T)$ are $A(K)$-steerable. This implies that the required supervisor C exists. ◇

Proposition 4.5. *Suppose $U(G \otimes T) = \emptyset$. Then there exists a lockup-free supervisor C such that $\mathcal{L}(G \otimes C) \subseteq \mathcal{L}(T)$ if and only if $I(G) \subseteq \bigcup_{K \in Z(G \otimes T)} H(K)$.*

Proof. (if.) This is the statement of Proposition 4.4.

(only if.) The argument is identical to the one used for the proof of Proposition 4.3. ◇

4.3 Necessary and Sufficient Conditions

The sufficient conditions of the two previous sections can now be combined to yield necessary and sufficient conditions for the existence of a lockup-free supervisor. Let G be an L-process and T be a task. With the notation from above define

$$B = \bigcup_{K \in Z(G \otimes T)} H(K) \cup H(G).$$

Theorem 4.1. *There exists a lockup-free supervisor C such that $\mathcal{L}(G \otimes C) \subseteq \mathcal{L}(T)$ if and only if $I(G) \subseteq B$.*

Proof. (if). Follows from the previous propositions.

(only if). Suppose $I(G) \not\subseteq B$. Then there exists a sequence $s \in S(G)^\omega$ starting at an initial state in $I(G) - B$ which, independent of any control actions chosen, eventually reaches a cycle which does not intersect $U(G \otimes T)$ and is not contained in $Z(G \otimes T)$. Hence, there cannot exist a supervisor C with the desired property. $\Diamond$

4.4 Supervisory Synthesis

Based on the above theorem the synthesis of a supervisor C for the task T is straightforward. We already presented algorithms (Algorithm 1, Algorithm 2) which compute the sets $H(K)$ for $K \in Z(G \otimes T)$, and $H(G)$. There is a standard procedure in supervisory control to synthesize a lockup-free controller for a controlled invariant set (see [8]). So, for each of the above sets we can compute a supervisor, say C_i. The supervisor $C' = C_1 \otimes C_2 \otimes \ldots$, forces G to remain inside the set of "good" states B, provided $I(G) \subseteq B$. Now extend C' to a supervisor C by excepting all states which correspond to void cycles, i.e., C eventually "interrupts" every cycle in $H(G)$ and $H(K)$, $K \in Z(G \otimes T)$, which does not intersect $U(G \otimes T)$ and is not contained in $Z(G \otimes T)$. Then the language of $G \otimes C$ satisfies $\mathcal{L}(G \otimes C) \subseteq \mathcal{L}(T)$.

5. Existence of Supervisors: Nondeterministic Case

We now consider the case where G is permitted to have nondeterministic resolution; everything else is like in the previous section. In the present setting, L-processes with nondeterministic resolution arise from ones with deterministic resolution and partial observations. As discussed previously, partial observations are modeled by a language homomorphism and give rise to a relation Ψ on the state space of G.

It is clear that Algorithms 1 and 2 can still be used to compute the sets $H(G)$ and $H(K)$. However, the existence of supervisors does no longer follow immediately. For the moment assume that there is a supervisor C with $\mathcal{L}(G \otimes C) \subseteq \mathcal{L}(T)$, which by some unspecified means is capable of exact state observations, independent of Ψ. In this case a necessary and sufficient condition for the existence of an "ordinary" supervisor C' with $\mathcal{L}(G \otimes C') = \mathcal{L}(G \otimes C)$, whose resolution is limited to $\ker(\Psi)$, is that

$$\Psi \leq \Theta. \tag{5.1}$$

This means that whenever two states are indistinguishable the respective control actions must be identical.

We contend that the requirement (5.1) is artificial and too severe for many applications; for example, it is rarely satisfied in communication systems or distributed systems. On the other hand, if (5.1) fails to hold, then clearly a mechanism is needed that allows the supervisor to glean the necessary state information from G. Mathematically, we can always change the system through *cylindrification* and satisfy (5.1). Roughly, this amounts to refining Ψ by adding new selections to transition conditions, i.e., making G_Φ more deterministic. In communication systems cylindrification is achieved through the introduction of sequence numbers or time stamps; more generallly, cylindrification can

be interpreted as introducing tags for events. However, to give a generally applicable practical solution appears difficult and remains an open problem for future research.

6. Conclusion

A modified supervisory control framework based on the models proposed by Ramadge and Wonham ([7]), and by Kurshan ([4]) was presented. We introduced the notion of a Büchi supervisor to implement arbitrary ω-regular behaviors by a feedback control. In contrast to work along similar lines by other researchers, these behaviors are allowed to violate the conditions of realizability in [10] and topological closure in [8]. We briefly discussed the decomposition of complex tasks into subtasks, a topic which will receive more attention in a future paper. Conditions for the existence of supervisors that lead to feedback implementations of prespecified ω-regular tasks were derived for the case of L-processes with deterministic state transition matrix.

Future research will address the questions of distributed control, task decomposition and computational aspects.

References

[1] I. Brave and M. Heymann, "On Stabilization of Discrete-Event Processes", *Proceedings of the 28th Conference on Decision and Control*, Tampa, Florida, 1989.

[2] R. Cieslak, C. Desclaux, A. Fawaz, and P. Varaiya, "Supervisory Control of Discrete Event Processes with Partial Observations", *IEEE Transactions on Automatic Control*, Vol. 33(3), pp. 249-260, March 1988.

[3] C.H. Golaszewski and P.J. Ramadge, "Mutual Exclusion Problems for Discrete Event Systems with Shared Events", *Proceedings of the 27th Conference on Decision and Control*, pp. 234-239, Austin, Texas, 1988.

[4] R.P. Kurshan, "Analysis of Discrete Event Coordination", Lecture Notes on Computer Science 430, pp. 414-453, Springer Verlag, 1990.

[5] R.P. Kurshan and C.H. Golaszewski, "An Automaton-Based Approach to the Synthesis of Communication Protocols", in preparation.

[6] C.M. Özveren and A.S. Willsky, "Output Stabilizability of Discrete Event Dynamic Systems", Discrete-Event Dynamical Systems", *Proceedings of the 28th Conference on Decision and Control*, Tampa, Florida, 1989.

[7] P.J. Ramadge and W.M. Wonham, "Supervisory Control of a Class of Discrete-Event Processes", *SIAM J. Control and Optimization*, Vol. 25(1), pp. 206-230, January 1987.

[8] P.J. Ramadge, "Some Tractable Supervisory Control Problems for Discrete Event Systems Modeled by Büchi Automata", *IEEE Transactions on Automatic Control*, Vol. 34(1), pp. 10-19, January 1989.

[9] P.J. Ramadge, "Observability of Discrete Event Systems", *Proceedings of the 25th Conference on Decision and Control*, Athens, Greece, 1986.

[10] J.G. Thistle and W.M. Wonham, "On the Synthesis of Supervisors Subject to ω-Language Specifications", *22nd Annual Conference on Information Sciences and Systems*, Princeton NJ, pp.440-444, March 1988.