Nonblocking Supervisory Control of Nondeterministic Systems *

Michael Heymann[1] and Feng Lin[2]

Abstract

In this paper we extend the theory of supervisory control of nondeterministic discrete-event systems, subject to nondeterministic specification, developed in [9]. We focus our attention on nonblocking and liveness considerations and develop algorithms for nonblocking-supervisor synthesis.

1 2

1 A personal perspective (of the first author)

The recognition in the late 1950s that linear filtering can be accomplished recursively, inspired the development of the Kalman filter and its dual, the linear quadratic optimal controller. The concepts of controllability and observability, the cornerstones of Algebraic Systems Theory, were not far behind. Early research on Algebraic Systems Theory focused on these concepts and issues related to canonical forms, canonical (minimal) realizations, system structural equivalences and invariants. These and related questions occupied much of the Algebraic Systems Theory research agenda of the 1960s. In the second half of that decade, the discovery of the pole shifting theorem provided the first insights into the connection between controllability and state-feedback. This later led to the emergence of the "geometric" theory of linear systems where the connection between pole shifting and controllability (and their obsevational duals) played major roles. At around the same time the fundamental feedback invariants of linear systems were first discovered.

I became familiar with Paul Fuhrmann's work on systems theory when I first came accross his classical paper "Algebraic System Theory - an Analyst's Point of View". At the time, I was working on the paper "Linear Feedback - An Algebraic

*This research is supported in part by the National Science Foundation under grant ECS-9315344 and in part by the Technion Fund for Promotion of Research.

[1]Department of Computer Science, Technion, Israel Institute of Technology, Haifa 32000, Israel, e-mail: heymann@cs.technion.ac.il. The work by this author was completed while he was a Senior NRC Research Associate at NASA Ames Research Center, Moffett Field, CA 94035.

[2]Department of Electrical and Computer Engineering, Wayne State University, Detroit, MI 48202, e-mail: flin@ece.eng.wayne.edu.

Approach" which dealt with the algebraic stucture of linear state-feedback. (The two papers have many points of contact). This first exposure and many interactions thereafter led to my continuing friendship with Paul for which I am very grateful.

In the early 1980s I reached the (personal) conclusion that a major obstacle to the practical "implementability" of continuous (especially linear) control theory, is the fact that many practical systems possess discontinuities, discrete interactions with their environment, parametric and structural uncertainty, nondeterminism and the like. A major component in the puzzle of how to control complex systems appeared to be missing!

This led to my embarkation on a new path - exploring discrete event control systems - where attention is focused exclusively on the logical (or discrete) aspects of system dynamics. With the recent maturing of the discrete event control theory and with many of the discrete aspects becoming clarified, it is progressively becoming feasible to begin an assault on "hybrid" systems in which the continuous and discrete aspects of system dynamics fully interact and coexist. It is not unlikely that in the not too distant future algebraic aspects of hybrid systems will become important, and an "Algebraic " theory of hybrid systems will emerge.

The present paper deals with the control of nondeterministic discrete event systems which, hopefully, sometime in the not too distant future will find its logical connection to my "algebraic" origins.

2 Introduction

Most of the published research on control of discrete-event systems (DES) has focused on systems that are modeled as deterministic finite state machines. For such systems, an extensive theory has been developed [23]. A great deal of attention was also given to the control of partially observed discrete-event systems [15], in which only a subset of the system's events are available for external observation. For such systems, necessary and sufficient conditions for existence of supervisors [15] [22] [23], algorithms for supervisor synthesis [2] [15] [16], for off-line as well as on-line implementation [2] [7], have been obtained, and a wide variety of related questions have been investigated.

Partially observed systems frequently exhibit nondeterministic behavior. There are, however, situations in which the system's model is nondeterministic not because of partial observation but, rather, because either the system is inherently nondeterministic, or because only a partial model of the system is available and some or all of its internal activities are unmodeled.

In contrast to deterministic discrete-event systems, whose behaviors are fully specified by their generated language, nondeterministic systems exhibit behaviors whose description requires much more refinement and detail. Further, while in the deterministic case, legal behavior of a system can be adequately expressed in terms of a language specification, this is clearly not always true when the system is nondeterministic. Indeed, to formally capture and specify legal behavior of the controlled system, it may be necessary to state, in addition to the permitted language, also the degree of nondeterminism that the controlled system is allowed

to retain. Various semantic formalisms have been introduced over the years for modeling and specification of nondeterministic behaviors. These differ from each other, among other things, in the degree of nondeterministic detail that they capture and distinguish. These formalisms include CSP [11] and the associated *failures* semantics, bisimulation semantics [19] and labeled transition systems [4]. In [5] and [6] the *trajectory model* formalism was introduced as a semantic framework for modeling and specification of nondeterministic behaviors with specific focus on discrete event control. It was shown there that this semantic is a *language congruence* that adequately captures nondeterministic behaviors that one might wish to discriminate and distinguish by discrete-event control. Thus, for control purposes, nondeterministic discrete-event systems can be modeled either as nondeterministic automata (with ϵ-transitions) or as trajectory models.

In recent years, there has been increasing interest in supervisory control of nondeterministic systems as reported, e.g., in [3] [12] [20] [21] [25]. However, while some existence conditions for control of nondeterministic systems have been derived, only limited progress on development of algorithms for supervisor synthesis has been reported (see e.g. [20] where a synthesis algorithm based on failures semantics is presented). Indeed, the direct supervisor synthesis for nondeterministic systems seems to be quite a difficult task (and, as will be shown below, unnecessary).

Motivated by this observation, we began an investigation, [8] [9], of the connection between the supervisory control problem for general nondeterministic systems and the corresponding problem for partially observed deterministic systems. Our work led us to develop an approach to synthesis of supervisors for nondeterministic systems wherein direct advantage is taken of the existing theory for control under partial observation.

In [9] we considered the supervisory control problem of nondeterministic discrete-event systems subject to trajectory-model specifications. Our approach to the supervisor synthesis was based on the following basic idea: We first synthesized from the given system, by adding to it hypothetical transitions and hypothetical uncontrollable and unobservable events, a deterministic system whose partially observed image is the original nondeterministic system (in the sense that the hypothetical events are obviously not observed). We called this procedure *lifting*. Before performing the lifting, the legal (trajectory model) specification was embedded in the original nondeterministic system model so that it can readily be dealt with in the corresponding lifted deterministic system. The next step of the synthesis was to construct a supervisor for the lifted system subject to the (obvious) condition that the artificially added events are neither observable nor controllable. Such a supervisor can easily be constructed using the well known theory and algorithms for supervisory control of partially observed systems. It is self evident, and we showed it formally, that a supervisor synthesized in this way is applicable for the original nondeterministic system and satisfies the specifications. Moreover, we showed that if the supervisor designed using this approach is optimal for the lifted system, it is also the optimal supervisor for the original system. Thus, since control under partial observation is well understood, we only had to, ultimately focus on the

auxiliary steps of model lifting and specification embedding.

The present paper is a continuation of this research. In [8] [9] we focused our attention only on safety specifications, without consideration of liveness issues. We did not worry about questions related to task completion, nor about the problem of possible blocking. We extend here the results of [9] to include nonblocking issues and liveness considerations. This generalization which, in spirit, is very similar to the parallel situation in the deterministic case, introduces several additional complexities to the theory, that have to be examined in detail. We develop the theory and the associated synthesis algorithms for nonblocking supervisory control by examining the so called, *static case*, where a subset of target (or marked) states and a subset of forbidden states of the system are specified. The control objective is then to disable the smallest subset of transitions such that, in the controlled system, no path leads to a forbidden state and every path can be extended to a target state. It can be shown that the more general *dynamic case*, where the specification is given by a trajectory model (or as a nondeterministic automaton), is transformed into the simpler static setting, in which the supervisor is then synthesized. Detailed algorithms for optimal supervisor synthesis are provided. We also briefly address the problem of control under partial observation (where some of the actual events in the modeled system are unobservable) and the problem of decentralized control.

Due to space limitation, some details are omitted, which can be found in [10].

3 Nonblocking supervisors

We model a nondeterministic discrete-event system by the trajectory model introduced in [6], whose notations are adopted in this paper. For the purpose of specification, we often represent a trajectory model by a nondeterministic automaton. Similar to the language model and deterministic automaton used in modeling deterministic systems, the trajectory model representation and the automaton representation of nondeterministic systems are interchangeable: For each nondeterministic automaton, there exists a unique trajectory generated by the automaton; and for each trajectory model, we can construct an automaton generating the trajectory model. In particular, for each path p in the automaton, we can find its corresponding trajectory t_p. To simplify the notation, we will use the same symbol to denote both the nondeterministic automaton and its associated trajectory model.

Since we are interested in the blocking and liveness issues in this paper, in addition to the usual elements of an automaton, we specify a set of marked states Q_m that represent, for example, task completions. To specify the desired behavior of the controlled system, we can use either a "static" specification or a "dynamic" specification. Similar to the algorithm developed in [9], we can always transform a dynamic specification into a static specification, where a subset $Q_b \subseteq Q$ of forbidden states that the system is not allowed to visit is specified. Thus, our system model can be written as

$$\mathcal{P} = (\Sigma \cup \{\epsilon\}, Q, \delta, q_0, Q_m, Q_b).$$

As in the deterministic case, we assume that $\mathcal{P}$ is trim (i.e., both accessible and co-accessible).

We define the set of marked trajectories of $\mathcal{P}$ as

$$\mathcal{P}_m = \{t_p : p \text{ ends in a marked state}\}.$$

The supervisory control problem is to synthesize a supervisor γ, (defined as a function $\gamma : L(\mathcal{P}) \to 2^{\Sigma_c}$ that after each observed string $s \in L(\mathcal{P})$ of executed transitions, disables a subset $\gamma(s) \subseteq \Sigma_c$ of controllable events,) such that the supervised system satisfies the state restrictions in that each path of the supervised system is a *legal* path; that is, each path ends at a target state (in Q_m) without ever entering a forbidden state (in Q_b). When such a supervisor exists, we would like to find, among all possible solutions, a least restrictive one; that is, a solution that disables as few as possible transitions.

For a supervisor γ, the language generated by the supervised system $\gamma/\mathcal{P}$ is given inductively as [9]

1. $\epsilon \in L(\gamma/\mathcal{P})$; and
2. $(\forall s \in L(\gamma/\mathcal{P}))(\forall \sigma \in \Sigma) s\sigma \in L(\gamma/\mathcal{P}) \Leftrightarrow s\sigma \in L(\mathcal{P}) \wedge \sigma \notin \gamma(s)$.

The supervised system is then given by

$$\gamma/\mathcal{P} = \mathcal{P} || det(L(\gamma/\mathcal{P}))$$

where $||$ denotes the strict synchronous (parallel) composition and $det(L(\gamma/\mathcal{P}))$ the deterministic process generating language $L(\gamma/\mathcal{P})$ (as defined in [9]).

In principle, our goal is to design a supervisor γ such that

$$\gamma/\mathcal{P} = \mathcal{P}_s$$

where $\mathcal{P}_s$ is (the trajectory model of) the (largest) trim subautomaton of

$$\overline{\mathcal{P}_s} = (\Sigma \cup \{\epsilon\}, Q_s, \delta_s, q_0, Q_{sm}).$$

where $Q_s = Q - Q_b$, $\delta_s = \delta|_{Q_s}$ ($\delta|_{Q_s}$ being the restriction of δ to Q_s), and $Q_{sm} = Q_s \cap Q_m$. Without loss of generality we shall assume that $\mathcal{P}_s = \overline{\mathcal{P}_s}$.

Such a supervisor is nonblocking in the sense that every trajectory enabled by the supervisor is a prefix of a trajectory that ends at a marked state.

As we shall see, such a supervisor does not always exist, and when it does not, we shall seek its best nonblocking approximation, as will be discussed below.

To obtain the desired supervisor, we proceed, just as in [9], by first transforming $\mathcal{P}$ to a deterministic automaton

$$\tilde{\mathcal{P}} = (\Sigma \cup \Sigma', \tilde{Q}, \tilde{\delta}, \tilde{q}_0, \tilde{Q}_m, \tilde{Q}_b)$$

using the procedure "Extend" given below.

Procedure Extend

Input: $\mathcal{P} = (\Sigma \cup \{\epsilon\}, Q, \delta, q_0, Q_m, Q_b)$.

Output: $\tilde{\mathcal{P}} = (\Sigma \cup \Sigma', \tilde{Q}, \tilde{\delta}, \tilde{q}_0, \tilde{Q}_m, \tilde{Q}_b)$.

1. $\tilde{Q} := Q$;

2. For each $q \in \tilde{Q}$ and $\sigma \in \Sigma$

   If $|\delta(q,\sigma)| > 1$, add one more state, q'

   and add $\epsilon-$transitions as follows:

   $\tilde{Q} := \tilde{Q} \cup \{q'\}$;
   $\tilde{\delta}(q,\sigma) := \{q'\}$;
   $\tilde{\delta}(q',\epsilon) := \delta(q,\sigma)$;

   else set

   $\tilde{\delta}(q,\sigma) := \delta(q,\sigma)$;

3. For each $q \in \tilde{Q}$

   replace the $\epsilon-$transitions by transitions labeled $\tau_1, \tau_2, ...$ as follows:
   If $\tilde{\delta}(q,\epsilon) = \{q_1, ..., q_n\}$, then set

   $\tilde{\delta}(q,\tau_1) := \{q_1\}$;
   ...
   $\tilde{\delta}(q,\tau_n) := \{q_n\}$;

4. Set

   $\Sigma' := \{\tau_1, \tau_2, ...\}$,
   $\tilde{Q}_m := Q_m$,
   $\tilde{Q}_b := Q_b \cup \{\tilde{q} \in \tilde{Q} - Q : \delta(\tilde{q},\epsilon) \subseteq Q_b\}$.

5. End of algorithm

■

We now define the following languages:

$$L(\tilde{\mathcal{P}}) := \{s \in \Sigma^* : \tilde{\delta}(\tilde{q}_0, s) \quad \text{is defined}\},$$

$$L_m(\tilde{\mathcal{P}}) := \{s \in L(\tilde{\mathcal{P}}) : \tilde{\delta}(\tilde{q}_0, s) \in \tilde{Q}_m\},$$

$$E := \{s \in L_m(\tilde{\mathcal{P}}) : (\forall s' \leq s)\tilde{\delta}(\tilde{q}_0, s') \in \tilde{Q} - \tilde{Q}_b\}.$$

From the definition of E it is clear that

$$E = L_m(\tilde{\mathcal{P}}) \cap \overline{E},$$

that is, E is $L_m(\tilde{\mathcal{P}})$-closed [22]. From Proposition 7 of [9] it follows that the projection of $\tilde{\mathcal{P}}$ on Σ is $\mathcal{P}$, that is,

$$\tilde{\mathcal{P}}\backslash_{\Sigma'} = \mathcal{P}.$$

We call a path marked if it ends in a marked state of $Q_m = \tilde{Q}_m$. We can prove the following

Proposition 1 A marked path p of the system $\mathcal{P}$ is legal (that is, is a path in $\mathcal{P}_s$) if and only if it is the projection of a path associated with a string $s \in E$ in $\tilde{\mathcal{P}}$.

Proof

Consider a marked path $p = (q_0, ..., \sigma_i, q_i, ..., \sigma_k, q_k)$ of $\mathcal{P}$ that visits a state $q_b \in Q_b$. The corresponding path in $\tilde{\mathcal{P}}$ has the same form with possible insertions of pairs (executions) σ', q', where $\sigma' \in \Sigma'$ and $q' \in \tilde{Q} - Q$. Hence the corresponding path $\tilde{p}$ in $\tilde{\mathcal{P}}$ also visits $q_b \in Q_b \subseteq \tilde{Q}_b$. Conversely, let $\tilde{p} = (q_0, ..., \sigma_i, q_i, ..., \sigma_k, q_k)$ be a marked path in $\tilde{\mathcal{P}}$ and assumes it visits a state $q_b \in \tilde{Q}_b$. If $q_b \in Q_b$, then the projected path in $\mathcal{P}$ also visits the state q_b. If $q_b \in \tilde{Q}_b - Q_b$, then $q_b \neq q_k$ and by the definition of $\tilde{Q}_b$, the next state visited by the path in $\tilde{\mathcal{P}}$ must be in Q_b. This bad state will be visited also by the projected path in $\mathcal{P}$. Thus, a marked path in $\tilde{\mathcal{P}}$ visits only states in $\tilde{Q} - \tilde{Q}_b$ if and only if the corresponding marked path in $\mathcal{P}$ visits only states in $Q - Q_b$.

■

We can now state the main result of this section that summarizes the conditions for existence of the desired supervisor.

Theorem 1 There exists a nonblocking supervisor γ such that $\gamma/\mathcal{P} = \mathcal{P}_s$ if and only if E is controllable and observable with respect to $L(\tilde{\mathcal{P}})$.

Proof

By the results of [15], there exists a nonblocking supervisor $\gamma : PL(\tilde{\mathcal{P}}) = L(\mathcal{P}) \rightarrow 2^{\Sigma_c}$ such that $L_m(\gamma/\tilde{\mathcal{P}}) = E$ if and only if E is controllable, observable, and $L_m(\tilde{\mathcal{P}})$-closed.

Since E is $L_m(\tilde{\mathcal{P}})$-closed by definition, the result follows from Proposition 1.

■

If E is not controllable and observable, we will synthesize an optimal supervisor γ (under partial observation) for $\tilde{\mathcal{P}}$ such that $L_m(\gamma/\tilde{\mathcal{P}}) = sup\mathcal{CN}(E)$, the supremal controllable and normal sublanguage of E. The reason that we can replace here the requirement of observability by normality, is due to the fact that in the lifted system all unobservable events Σ' are also uncontrollable, in which case a language is controllable and observable if and only if it is controllable and normal [18].

The synthesis is discussed in the next section.

4 Supervisor synthesis

Our objective is to design a nonblocking supervisor γ for $\tilde{\mathcal{P}}$ such that

$$L_m(\gamma/\tilde{\mathcal{P}}) = sup\mathcal{CN}(E).$$

This supervisor tracks only the events of Σ, and hence can be applied directly to $\mathcal{P}$. It will be least restrictive in the sense that it allows the system $\mathcal{P}$ to visit as many states in Q_s as possible (see [9]).

Such a supervisor can be designed with or without the lifting procedure, as outlined in the two ensuing algorithms.

Algorithm 1 *(Synthesis by lifting)*

1. Lift $\mathcal{P}$ to $\tilde{\mathcal{P}}$ using Procedure Extend:

$$\tilde{\mathcal{P}} = (\Sigma \cup \Sigma', \tilde{Q}, \tilde{\delta}, \tilde{q}_0, \tilde{Q}_m, \tilde{Q}_b).$$

2. Compute the sublanguage $sup\mathcal{CN}(E)$, that is, the supremal controllable and normal sublanguage of E.

3. Compute the projection $P(sup\mathcal{CN}(E))$ of the language $sup\mathcal{CN}(E)$ on Σ and let the supervisor γ be defined by

$$(\forall s \in \overline{Psup\mathcal{CN}(E)})\gamma(s) := \{\sigma \in \Sigma : s\sigma \notin \overline{Psup\mathcal{CN}(E)}\}.$$

■

In the above algorithm, Step 1 is described in Section 3. Steps 2 and 3 are standard elements in the design of supervisors under partial observation [15].

The correctness of the above algorithm is obvious (see also [9]) and is stated in the following

Theorem 2 The supervisor synthesized using Algorithm 1 is nonblocking and satisfies

$$L_m(\gamma/\tilde{\mathcal{P}}) = sup\mathcal{CN}(E).$$

Proof

Elementary.

■

An alternate procedure for supervisor synthesis, that does not require the lifting of $\mathcal{P}$, is described in the next algorithm, where $Acc(.)$ denotes the accessible part of an automaton.

Algorithm 2 *(Synthesis without lifting)*

1. Ignore the set Q_m of marked states and convert the automaton $\mathcal{P} = (\Sigma \cup \{\epsilon\}, Q, \delta, q_0, Q_s)$ to a deterministic automaton $\hat{\mathcal{P}} = Acc(\Sigma, \hat{Q}, \hat{\delta}, \hat{q}_0, \hat{Q}_s)$, where

$$\begin{aligned}
&\hat{Q} := 2^Q; \\
&\hat{\delta}(\hat{q}, \sigma) := \{q' \in Q : (\exists q \in \hat{q}) q' \in \epsilon^*(\delta(q, \sigma))\}; \\
&\hat{q}_0 := \{q' \in Q : q' \in \epsilon^*(q_0)\}; \\
&\hat{Q}_s := \{\hat{q} \in \hat{Q} : (\forall u \in \Sigma_{uc}^*)\hat{\delta}(\hat{q}, u) \subseteq Q_s\};
\end{aligned}$$

2. If $\hat{Q}_s = \hat{Q}$, go to 7.

3. Set

$$\begin{aligned}
&\hat{\delta} := \hat{\delta}|_{\hat{Q}_s}; \\
&\hat{Q} := \{\hat{q} \in \hat{Q}_s : (\exists s \in \Sigma^*)\hat{q} = \hat{\delta}(\hat{q}_o, s)\},
\end{aligned}$$

and form the product automaton

$$\mathcal{P}' := (\Sigma \cup \{\epsilon\}, Q \times \hat{Q}, \delta', (q_0, \hat{q}_0), Q_m \times \hat{Q}),$$

where

$$\delta'((q, \hat{q}), \sigma) := \begin{cases} (\delta(q, \sigma), \hat{\delta}(\hat{q}, \sigma)) & \text{if both } \delta(q, \sigma) \text{ and } \hat{\delta}(\hat{q}, \sigma) \text{ are defined} \\ \text{undefined} & \text{otherwise.} \end{cases}$$

4. By trimming $\mathcal{P}'$ compute the set:

$$\begin{aligned}
Q_t := \{&q \in Q : (\exists \hat{q} \in \hat{Q})(q, \hat{q}) \text{ is accessible in } \mathcal{P}' \text{ from } (q_o, \hat{q}_o) \\
&\text{and co} - \text{accessible in } \mathcal{P}' \text{ to } Q_m \times \hat{Q}\};
\end{aligned}$$

5. If $Q_t = Q_s$, go to 7. Otherwise, set

$$\begin{aligned}
&Q_s := Q_t; \\
&\hat{Q}_s := \{\hat{q} \in \hat{Q} : (\forall u \in \Sigma_{uc}^*)\hat{\delta}(\hat{q}, u) \subseteq Q_s\};
\end{aligned}$$

6. Go to 3

7. Define the supervisor γ:

$$\gamma(s) = \{\sigma \in \Sigma_c : \hat{\delta}(\hat{q}_0, s\sigma) \text{ is not defined}\}.$$

■

To prove that Algorithm 2 designs the correct supervisor in a finite number of steps (for finite automata), we first define, for languages B and M with $B \subseteq M = \overline{M} \subseteq (\Sigma \cup \Sigma')^*$,

$$sup\mathcal{N}(B) = \overline{B} - P^{-1}P(M - \overline{B})(\Sigma \cup \Sigma')^*$$
$$\Omega(B, M) = M \cap P^{-1}(P(sup\mathcal{N}(B)) - ((P(M) - P(sup\mathcal{N}(B)))/\Sigma_{uc}^*)\Sigma^*)$$
$$\Delta(B, M) = B \cap \Omega(B, M).$$

where $L/\Sigma_{uc}^* = \{s \in \Sigma^* : (\exists u \in \Sigma_{uc}^*)su \in L\}$. In the above, the operator $sup\mathcal{N}(B)$ calculates the supremal normal sublanguage of $\overline{B}$ (with respect to M) [1], the operator $\Omega(B, M)$ generates the supremal controllable and normal sublanguage of $\overline{B}$ (with respect to M) [1] and the operator $\Delta(B, M)$ intersects $\Omega(B, M)$ with B.

Suppose we apply these operators repeatedly with respect to the lifted automaton $\tilde{\mathcal{P}}$ and the corresponding legal language E as follows.

$$M_0 = L(\tilde{\mathcal{P}}), \qquad B_0 = E$$
$$M_{i+1} = \Omega(B_i, M_i), \quad B_{i+1} = \Delta(B_i, M_i), \quad i = 0, 1, 2, ...$$

Then we can show that B_i converges to $sup\mathcal{CN}(E)$ in the following

Lemma 1 If there exists a positive integer N such that $B_{N+1} = B_N$, then

$$B_N = sup\mathcal{CN}(E).$$

Proof

Omitted.

■

Using the above lemma, we can prove the following theorem, which states the correctness of Algorithm 2.

Theorem 3 The supervisor synthesized using Algorithm 2 is nonblocking and satisfies

$$L_m(\gamma/\tilde{\mathcal{P}}) = sup\mathcal{CN}(E).$$

Outline of Proof

We only give an outline of the proof because its details are tedious and provide no additional insight.

It is clear that in Algorithm 2, the first part of Step 1 is equivalent to calculating $sup\mathcal{N}(B_0)$ (without explicitly introducing Σ') and the first part of Step 5 calculates $sup\mathcal{N}(B_i)$. The second parts (where $\hat{Q}_s$ is calculated) of Steps 1 and 5 calculate

$$P\Omega(B_i, M_i) = P(sup\mathcal{N}(B_i)) - ((P(M_i) - P(sup\mathcal{N}(B_i)))/\Sigma_{uc}^*)\Sigma^*$$

Steps 3 and 4 are equivalent to calculating

$$B_i \cap P^{-1}P(\Omega(B_i, M_i))$$
$$= B_i \cap P^{-1}P(\Omega(B_i, M_i)) \cap M_i$$
$$= B_i \cap \Omega(B_i, M_i)$$
$$= \Delta(B_i, M_i).$$

Therefore, Algorithm 2 implements the recursive computation of B_i, and calculates $sup\mathcal{CN}(E)$.

■

5 Control under partial observation

We now consider the situation when not all the events in Σ are observable and the supervisor must be based on a subset $\Sigma_o \subseteq \Sigma$ of observable events. In this case, the set of unobservable events in the lifted process, is $(\Sigma \cup \Sigma') - \Sigma_o$, and if we denote by $T : \Sigma^* \to \Sigma_o^*$ the projection operator, then the projection from $\Sigma \cup \Sigma'$ to Σ_o is obtained by the composition of T and P.

In view of Theorem 1, the existence (and synthesis) of a supervisor under partial observation for $\mathcal{P}$ is equivalent to that of the corresponding supervisor for $\tilde{\mathcal{P}}$, because Theorem 1 hold for any supervisor, and a supervisor under partial observation is a special case. Therefore, we obtain the following corollary to Theorem 2.1 in [15].

Corollary 1 There exists a nonblocking partial observation supervisor $\gamma : TPL(\tilde{\mathcal{P}}) \to 2^{\Sigma_c}$ such that $\gamma/\mathcal{P} = \mathcal{P}_s$ if and only if E is controllable (with respect to Σ_c and $L(\tilde{\mathcal{P}})$) and observable (with respect to Σ_o and $L(\tilde{\mathcal{P}})$).

The supervisor can be synthesized with respect to $\tilde{\mathcal{P}}$. However, since it is no longer true that all the controllable events are also observable, observability can no longer be replaced by normality. Consequently, since the supremal observable sublanguage may not exist, a unique optimal supervisor may not exist either. To overcome this difficulty, two approaches can be employed: (1) to synthesize a suboptimal supervisor based on the supremal controllable and normal sublanguage (with respect to Σ_o); and (2) to synthesize a maximal controllable and observable sublanguage, which may not be unique. Both approaches have been studied extensively in the literature and will not be repeated here.

If the specification is a language specification, then E is normal [9]. In such a case, as we shall show in the following lemma, E is observable with respect to Σ_o and $L(\tilde{\mathcal{P}})$ if and only if PE is observable with respect to Σ_o and $PL(\tilde{\mathcal{P}})$.

Lemma 2 Let B be normal with respect to Σ and $L(\tilde{\mathcal{P}})$. Then B is observable with respect to Σ_o and $L(\tilde{\mathcal{P}})$ if and only if PB is observable with respect to Σ_o and $PL(\tilde{\mathcal{P}}) = L(\mathcal{P})$.

Proof

Omitted.

■

Using the lemma, we can immediately obtain the following

Corollary 2 For a nondeterministic system $\mathcal{P}$ and a language specification $L(\hat{\mathcal{H}})$, there exists a nonblocking partial observation supervisor γ such that $L(\gamma/\mathcal{P}) = L(\hat{\mathcal{H}})$ if and only if $L(\hat{\mathcal{H}})$ is controllable and observable with respect to $L(\mathcal{P})$.

This result was obtained in [14], where only language specifications were considered. The results in this section show that there is no need to treat the unobservable events $\Sigma_{uo} = \Sigma - \Sigma_o$ differently from the events Σ', except that some events in Σ_{uo} may be controllable. As a consequence, the supervisor synthesis may be more complex.

6 Decentralized control

The design of decentralized supervisors for nondeterministic systems can also be dealt with by using the deterministic theory and the lifting procedure. Since the methodology is quite analogous to what we have seen, we shall only outline the approach.

Without lose of generality, we may consider the case of two (decentralized) supervisors γ_1 and γ_2. For $i = 1, 2$, γ_i can observe events in $\Sigma_{io} \subseteq \Sigma$ and control events in $\Sigma_{ic} \subseteq \Sigma$. Letting $T_i : \Sigma^* \to \Sigma_{io}^*$ denote the projections, we can can write the supervisors γ_i as maps

$$\gamma_i : T_i L(\mathcal{P}) \to 2^{\Sigma_{ic}}.$$

As in [24], an event is enabled if it is enabled by both supervisors. The following existence result is then a corollary of Theorem 4.1 of [24].

Corollary 3 There exists two nonblocking decentralized supervisors γ_1 and γ_2 such that $(\gamma_1 \wedge \gamma_2)/\mathcal{P} = \mathcal{P}_s$ if and only if E is controllable (with respect to $\Sigma_{1c} \cup \Sigma_{2c}$) and co-observable.

Therefore, we conclude that both decentralized control and control under partial observation of nondeterministic systems can be synthesized by the existing methods for deterministic systems if we lift the corresponding processes.

7 Computational complexity

Since, in general, a supervisor synthesis problem under partial observation is of exponential complexity in terms of the number of transitions in the automata, it may be expected that the complexity of supervisor synthesis for nondeterministic systems also be exponential. Denote the number of states in an automaton $\mathcal{P}$ by $|\mathcal{P}|$ and the number of events by $|\Sigma|$. We outline the complexity analysis as follows.

Algorithm 1 involves two essential steps: (1) the procedure Extend that lifts $\mathcal{P}$ to a deterministic one, and (2) controller synthesis with respect to the lifted automaton. The procedure Extend adds at most $|\mathcal{P}| \times |\Sigma|$ states and $|\mathcal{P}|$ event labels to the process. The lifted automaton has, therefore, at most $|\mathcal{P}|(|\Sigma| + 1)$ states and $|\mathcal{P}| + |\Sigma|$ event labels. The complexity of executing Extend is of order $|\mathcal{P}|(|\Sigma| + 1)(|\mathcal{P}| + |\Sigma|)$. The synthesis of the optimal controller for the lifted process cannot be executed "on-line" because of the nonblocking requirement and, therefore, Algorithm 1 is of complexity

$$O((|\Sigma| + |\mathcal{P}|)2^{|\mathcal{P}|(|\Sigma|+1)}).$$

For Algorithm 2, the complexity of executing Steps 3-6 is at most $|\Sigma||\mathcal{P}|2^{|\mathcal{P}|}$. These steps will be repeated at most $|\mathcal{P}|$ times. Therefore, Algorithm 2 is of complexity

$$O(|\Sigma||\mathcal{P}|^2 2^{|\mathcal{P}|}).$$

References

[1] R. D. Brandt, V. Garg, R. Kumar, F. Lin, S. I. Marcus and W. M. Wonham, 1990. Formulas for calculating supremal controllable and normal sublanguages. *Systems & Control Letters, 15*, pp. 111-117.

[2] S. L. Chung, S. Lafortune and F. Lin, 1992. Limited lookahead policies in supervisory control of discrete event systems. *IEEE Transactions on Automatic Control, 37(12),* pp. 1921-1935.

[3] M. Fabian and B. Lennartson, 1994. Object oriented supervisory control with a class of nondeterministic specifications, Report No CTH/RT/I-94/007, Chalmers University of Technology, Goteborg, Sweden.

[4] M. Hennesy, *Algebraic Theory of Processes*, MIT Press, 1988.

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

[6] M. Heymann and G. Meyer, 1991. An algebra of discrete event processes. *NASA Technical Memorandum 102848.*

[7] M. Heymann and F. Lin, 1994. On-line control of partially observed discrete event systems. *Discrete Event Dynamic Systems: Theory and Applications, 4(3)*, pp. 221-236.

[8] M. Heymann and F. Lin, 1995. On observability and nondeterminism in discrete event control, *Proceedings of the 33rd Allerton conference on Communication Control and Computing*, pp. 136-145.

[9] M. Heymann and F. Lin, 1996. Discrete event control of nondeterministic systems. *CIS Report 9601*, Technion, Israel.

[10] M. Heymann and F. Lin, 1996. Nonblocking supervisory control of nondeterministic systems. *CIS Report 9620*, Technion, Israel.

[11] C.A.R. Hoare, *Communicating Sequential Processes*, Prentice Hall, 1985.

[12] K. Inan, 1994. Nondeterministic supervision under partial observation. in G. Cohen and J.-P. Quadrat, Eds., *11th International Conference on Analysis and Optimization of Systems*, pp. 39-48, Springer Verlag.

[13] R. Kumar and M. A. Shayman, 1993. Non-blocking supervisory control of nondeterministic systems via prioritized synchronization. *Technical Research Report, T.R. 93-58*, Institute for Systems Research, University of Maryland.

[14] R. Kumar and M. A. Shayman, 1994. Supervisory control under partial observation of nondeterministic systems via prioritized synchronization. *Technical Report,* Department of Electrical Engineering, University of Kentucky.

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

[16] F. Lin and W. M. Wonham, 1988. Decentralized supervisory control of discrete-event systems. *Information Sciences, 44(3)*, pp. 199-224.

[17] F. Lin and W. M. Wonham, 1990. Decentralized control and coordination of discrete event systems with partial observation. *IEEE Transactions on Automatic Control, 35(12)*, pp. 1330-1337.

[18] F. Lin and W. M. Wonham, 1994. Supervisory control of timed discrete event systems under partial observation, *IEEE Transactions on Automatic Control, 40(3),* pp. 558-562.

[19] R. Milner, *A Calculus of Communicating Systems*, LNCS 94, Springer Verlag, 1980.

[20] A. Overkamp, 1994. Supervisory control for nondeterministic systems. *Proccedings of 11th International Conference on Analysis and Optimization of Systems*, pp. 59-65.

[21] A. Overkamp, 1994. Partial observation and partial specification in nondeterministic discrete-event systems, *preprint.*

[22] R. J. Ramadge and W. M. Wonham, 1987. Supervisory control of a class of discrete event processes. *SIAM J. Control and Optimization, 25(1)*, pp. 206-230.

[23] P. J. Ramadge and W. M. Wonham, 1989. The control of discrete event systems. *Proceedings of IEEE, 77(1)*, pp. 81-98.

[24] K. Rudie and W. M. Wonham, 1992. Think globally, act locally: decentralized supervisory control. *IEEE Transactions on Automatic Control, 37(11)*, pp. 1692-1708.

[25] M. Shayman and R. Kumar, 1995. Supervisory control of nondeterministic systems with driven events via prioritized synchronization and trajectory models. SIAM Journal of Control and Optimization, 33(2), pp. 469-497.

[26] J. N. Tsitsiklis, 1989. On the control of discrete-event dynamical systems. *Mathematics of Control, Signals, and Systems, 2(1)*, pp. 95-107.