[18] F. Lin. On controllability and observability of discrete event systems. *Ph. D. Thesis*, Department of Electrical Engineering, University of Toronto, 1987.

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

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

[21] 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.

[22] 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.

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

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

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

[26] 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.

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

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

[29] 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.

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

[5] 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.

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

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

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

[9] 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.

[10] 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.

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

[12] M. Heymann and F. Lin, 1996. Discrete event control of nondeterministic systems. Proceedings, Conference of Decision and Control, Kobe, Japan.

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

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

[15] 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.

[16] 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.

[17] 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.

the optimal controller for the lifted process cannot be executed "on-line" because of the nonblocking requirement and, therefore, is of complexity

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

Finally, we must consider also the trajectory inclusion algorithm. This algorithm involves the automata $\overline{\mathcal{P}}$ and $\hat{\mathcal{H}}$. It is not difficult to see that the numbers of states in $\overline{\mathcal{P}}$ and $\hat{\mathcal{H}}$ are, at most,

$$|\overline{\mathcal{P}}| = |\mathcal{P}|2^{|\mathcal{H}||\mathcal{P}|},$$
$$|\hat{\mathcal{H}}| = |\mathcal{H}||\mathcal{P}|.$$

In the trajectory inclusion algorithm, we must first represent both $\overline{\mathcal{P}}$ and $\hat{\mathcal{H}}$ as trajectory model automata by computing for each state its maximal refusal set. The complexity of this procedure is bounded by $|\overline{\mathcal{P}}|^2|\Sigma|$ and $|\hat{\mathcal{H}}|^2|\Sigma|$, respectively. The algorithm then requires for each state q of $\overline{\mathcal{P}}$ to find $R(q)$, which requires examining a subset of states of $\hat{\mathcal{H}}$ and for each pair of states (one in $\overline{\mathcal{P}}$ and one in $\hat{\mathcal{H}}$) to test for refusal set inclusion. This test is of complexity at most $|\Sigma|log|\Sigma|$ using standard algorithms. Thus, the complexity bound of the trajectory inclusion algorithm is

$$O(|\overline{\mathcal{P}}|^2|\Sigma| + |\hat{\mathcal{H}}|^2|\Sigma| + |\overline{\mathcal{P}}||\hat{\mathcal{H}}||\Sigma|log|\Sigma|).$$

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] H. Cho and S. I. Marcus, 1989. On supremal languages of class of sublanguages that arise in supervisor synthesis problems with partial observations. *Math. Control Signal Systems 2*, pp. 47-69.

[3] 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.

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

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 [28], an event is enabled if it is enabled by both supervisors. To find an existence condition for decentralized supervision, we need the concept of co-observability, which has been introduced in [28].

A language $K \subseteq L(\tilde{\mathcal{P}})$ is called co-observable if

$$\begin{aligned} &s, s', s'' \in \Sigma^*, T_1P(s) = T_1P(s'), T_2P(s) = T_2P(s'') \Rightarrow \\ &\quad (\forall \sigma \in \Sigma_{1c} \cap \Sigma_{2c}) s \in \overline{K} \wedge s\sigma \in L(\tilde{\mathcal{P}}) \wedge s'\sigma, s''\sigma \in \overline{K} \Rightarrow s\sigma \in \overline{K} \\ &\wedge (\forall \sigma_{1c} - \Sigma_{2c}) s \in \overline{K} \wedge s\sigma \in L(\tilde{\mathcal{P}}) \wedge s'\sigma \in \overline{K} \Rightarrow s\sigma \in \overline{K} \\ &\wedge (\forall \sigma_{2c} - \Sigma_{1c}) s \in \overline{K} \wedge s\sigma \in L(\tilde{\mathcal{P}}) \wedge s''\sigma \in \overline{K} \Rightarrow s\sigma \in \overline{K} \end{aligned}$$

From our discussions, the following is then a corollary of Theorem 4.1 of [28].

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.

8 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. We will outline the complexity analysis as follows.

Let us first consider the problem with static specifications. We denote the number of states in an automaton $\mathcal{P}$ by $|\mathcal{P}|$ and the number of events by $|\Sigma|$.

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

and

$$(\forall t, t' \in PB)Tt = Tt' \Rightarrow (\forall \sigma \in \Sigma)(t\sigma \in PB \wedge t'\sigma \in PL(\tilde{\mathcal{P}}) \Rightarrow t'\sigma \in PB),$$

and we would like to show that B is observable with respect to Σ_o and $L(\tilde{\mathcal{P}})$; that is, for all $s, s' \in (\Sigma \cup \Sigma')^*$ and $\sigma \in \Sigma \cup \Sigma'$ such that $TPs = TPs'$,

$$s\sigma \in B \wedge s' \in B \wedge s'\sigma \in L(\tilde{\mathcal{P}}) \Rightarrow s'\sigma \in B.$$

If $\sigma \in \Sigma'$, then $Ps'\sigma = Ps'$. Therefore by the hypotheses,

$$s' \in B \wedge s'\sigma \in L(\tilde{\mathcal{P}}) \Rightarrow s'\sigma \in L(\tilde{\mathcal{P}}) \cap P^{-1}PB = B.$$

If $\sigma \in \Sigma$, then let $t = Ps$ and $t' = Ps'$. We have

$$\begin{aligned} & s\sigma \in B \wedge s' \in B \wedge s'\sigma \in L(\tilde{\mathcal{P}}) \\ & \Rightarrow t\sigma \in PB \wedge t' \in PB \wedge t'\sigma \in PL(\tilde{\mathcal{P}}) \\ & \Rightarrow t'\sigma \in PB. \end{aligned}$$

Hence,

$$s'\sigma \in L(\tilde{\mathcal{P}}) \cap P^{-1}PB = B.$$

■

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 [17], 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.

7 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.

If the specification is a language specification, then E is normal [11]. 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

Note that since observability and normality is defined for the closure of a language, we can assume, without loss of generality, that B is closed.

Only if: We want to show that under the hypotheses

$$B = L(\tilde{\mathcal{P}}) \cap P^{-1}PB$$

and

$$(\forall s, s' \in B)TPs = TPs' \Rightarrow (\forall \sigma \in \Sigma \cup \Sigma')(s\sigma \in B \wedge s'\sigma \in L(\tilde{\mathcal{P}}) \Rightarrow s'\sigma \in B)$$

PB is observable with respect to Σ_o and $PL(\tilde{\mathcal{P}})$; that is, for all $t, t' \in \Sigma^*$ and $\sigma \in \Sigma$ such that $Tt = Tt'$,

$$t\sigma \in PB \wedge t' \in PB \wedge t'\sigma \in PL(\tilde{\mathcal{P}}) \Rightarrow t'\sigma \in PB.$$

Indeed,

$$\begin{aligned} &t\sigma \in PB \wedge t'\sigma \in PL(\tilde{\mathcal{P}}) \\ &\Rightarrow (\exists s, s' \in (\Sigma \cup \Sigma')^*)Ps = t \wedge Ps' = t' \wedge s\sigma \in B \wedge s'\sigma \in L(\tilde{\mathcal{P}}). \end{aligned}$$

Now, by the hypotheses,

$$t' \in PB \Rightarrow s' \in L(\tilde{\mathcal{P}}) \cap P^{-1}PB = B.$$

Again, by the hypotheses, $TPs = Tt = Tt' = TPs'$ implies

$$\begin{aligned} &s\sigma \in B \wedge s' \in B \wedge s'\sigma \in L(\tilde{\mathcal{P}}) \\ &\Rightarrow s'\sigma \in B \\ &\Rightarrow P(s'\sigma) = t'\sigma \in PB. \end{aligned}$$

If: The hypotheses are now

$$B = L(\tilde{\mathcal{P}}) \cap P^{-1}PB$$

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

Steps 3 and 4 are equivalent to calculating

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

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

■

6 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 [19].

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 sub-optimal 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.

We now proceed to prove the lemma.

($\subseteq$): Clearly, $B_N \subseteq E$. In order to show $B_N \subseteq sup\mathcal{CN}(E)$, we need to show that B_N is controllable and normal.

$$
\begin{aligned}
& B_{N+1} = B_N \\
\Rightarrow\ & B_N \cap sup\mathcal{CN}(\overline{B_N}) = B_N \\
\Rightarrow\ & B_N \subseteq sup\mathcal{CN}(\overline{B_N}) \\
\Rightarrow\ & \overline{B_N} \subseteq sup\mathcal{CN}(\overline{B_N}) \\
\Rightarrow\ & \overline{B_N} = sup\mathcal{CN}(\overline{B_N}) \\
\Rightarrow\ & \overline{B_N} \textit{ is controllable and normal} \\
\Rightarrow\ & B_N \textit{ is controllable and normal}
\end{aligned}
$$

($\supseteq$): To prove that $sup\mathcal{CN}(E) \subseteq B_N$ we proceed by induction on i.

BASE:

$$sup\mathcal{CN}(E) \subseteq E = B_0.$$

INDUCTION STEP:

$$
\begin{aligned}
& sup\mathcal{CN}(E) \subseteq B_i \\
\Rightarrow\ & sup\mathcal{CN}(E) \subseteq B_i \wedge sup\mathcal{CN}(E) \subseteq \overline{B_i} \\
\Rightarrow\ & sup\mathcal{CN}(E) \subseteq B_i \wedge sup\mathcal{CN}(sup\mathcal{CN}(E)) \subseteq sup\mathcal{CN}(\overline{B_i}) \\
\Rightarrow\ & sup\mathcal{CN}(E) \subseteq B_i \wedge sup\mathcal{CN}(E) \subseteq sup\mathcal{CN}(\overline{B_i}) \\
\Rightarrow\ & sup\mathcal{CN}(E) \subseteq B_i \cap sup\mathcal{CN}(\overline{B_i}) \\
\Rightarrow\ & sup\mathcal{CN}(E) \subseteq B_{i+1}.
\end{aligned}
$$

■

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

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

It is easy to see that a language L is controllable (normal) if and only if its closure $\overline{L}$ is controllable (normal). It is also easy to show that for languages $L_1 \subseteq L_2 \subseteq L_3$, If L_1 is controllable (normal) with respect to L_2 and L_2 is controllable (normal) with respect to L_3, then L_1 is controllable (normal) with respect to L_3.

Now, consider the languages $B_1, B_2, ...$, we have

$$\begin{aligned} B_1 &= \Delta(B_0, M_0) \\ &= B_0 \cap sup\mathcal{CN}(\overline{B_0}) \quad (wrt\ L(\tilde{\mathcal{P}})), \end{aligned}$$

and

$$\begin{aligned} B_2 &= \Delta(B_1, M_1) \\ &= B_1 \cap sup\mathcal{CN}(\overline{B_1}) \quad (wrt\ M_1) \\ &= B_1 \cap sup\mathcal{CN}(\overline{B_1}) \quad (wrt\ L(\tilde{\mathcal{P}})). \end{aligned}$$

The last equality is due to the fact that $M_1 = sup\mathcal{CN}(E)$ (wrt $L(\tilde{\mathcal{P}})$) is controllable and normal with respect to $L(\tilde{\mathcal{P}})$. Similarly, we can show

$$B_{i+1} = B_i \cap sup\mathcal{CN}(\overline{B_i}) \quad (wrt\ L(\tilde{\mathcal{P}})).$$

Algorithm 2 *(Synthesis without lifting)*

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

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

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

3. Set

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

   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:

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

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

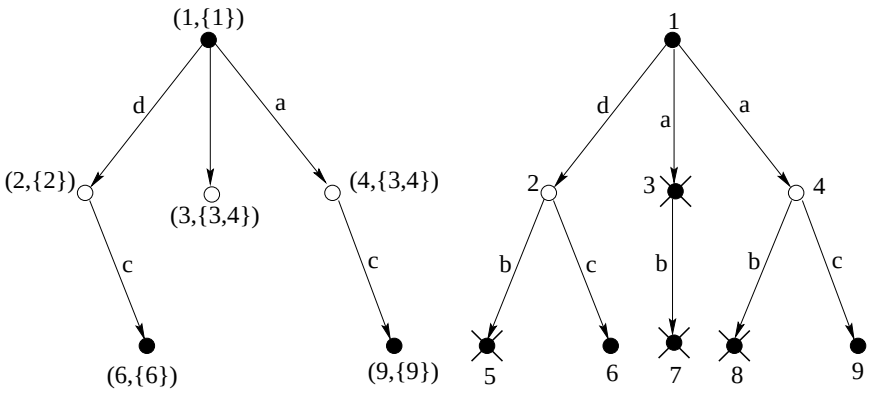
$$Q_s := Q_t;$$
$$\hat{Q}_s := \{\hat{q} \in \hat{Q} : (\forall u \in \Sigma_{uc}^*)\hat{\delta}(\hat{q}, u) \subseteq Q_s\};$$

6. Go to 3

7. Define the supervisor γ:

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

■



(a) $P \| _{new} \hat{P}$ (b) $_{new}P$

Figure 6:

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 [18] [19].

The correctness of the above algorithm is obvious (see also [11]) 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.

a complication is determining which states in $\hat{\mathcal{P}}$ are bad: the state $\{7,8\}$ contains a good state 7 and a bad state 8 (both of $\mathcal{P}$). Not to risk his job by allowing possible violation of the regulation, the administrator declares $\{7,8\}$ illegal as shown in Figure 5(a). Hence he disables both b in Figure 5(a) to obtain a new $\hat{\mathcal{P}}$ as in Figure 5(b).

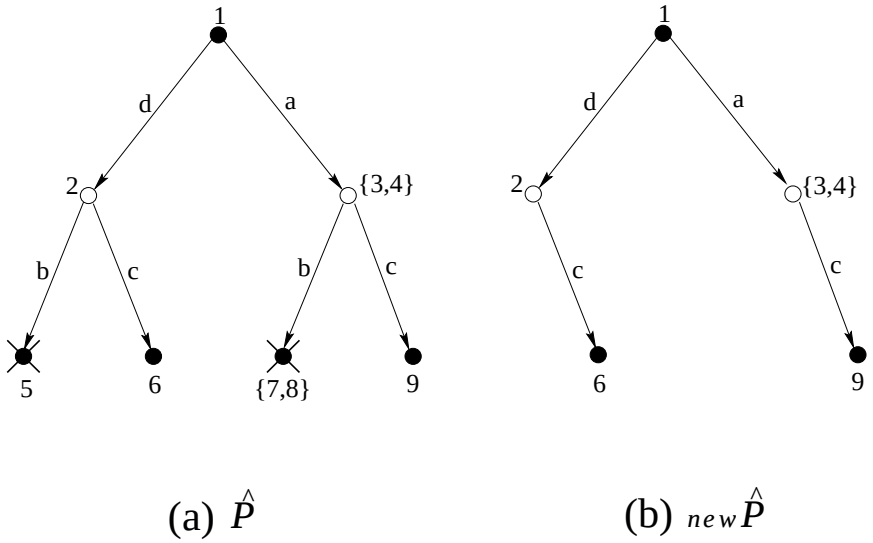


Figure 5:

The administrator soon realizes that the new design is different from the old design (based on lifting). Is his colleague wrong in saying that the design can be done without lifting? Or does he miss something important? Unable to find the answer, the administrator calls in the professor for his judgement. The professor patiently explain to him that under his new supervision, he may be stranded at the train station, as seen in Figure 6(a) that describes the supervised system $(\mathcal{P}||new\hat{\mathcal{P}})$.

The only way to avoid this mishap is to declare the states 3 and 7 in $\mathcal{P}$ also illegal as shown in Figure 6(b) ($new\mathcal{P}$). With this new set of illegal states, the administrator repeats his design without lifting, which yields the same supervisor as the one designed using lifting.

■

The following algorithm synthesizes a supervisor using lifting.

Algorithm 1 *(Synthesis by lifting)*

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

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

A travel administrator, acting as a supervisor, is asked to enforce this regulation without exception. Since he does not know if buses are available in the afternoon (and the word of the professor cannot be trusted in this case), the system is nondeterministic.

To design his strategy (that is, γ), the administrator uses design by lifting. He first lifts $\mathcal{P}$ to $\tilde{\mathcal{P}}$ as shown in Figure 4(a)

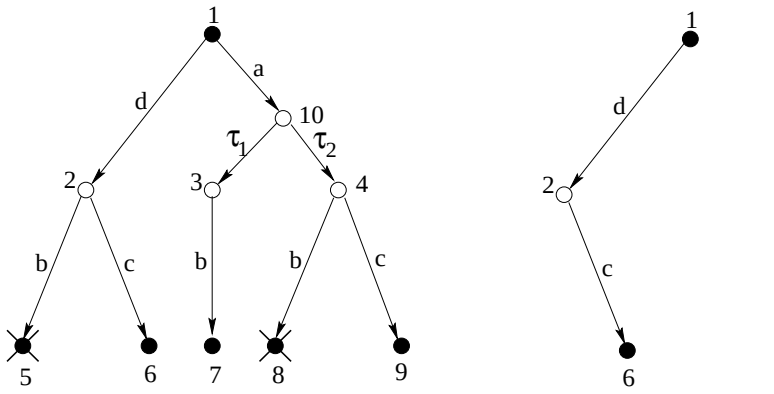


Figure 4:

Form Figure 4(a), he finds the legal language E to be

$$E = \{\epsilon, a\tau_1 b, a\tau_2 c, dc\}.$$

He then calculates the supremal controllable and normal sublanguage $sup\mathcal{CN}(E)$. He can observe the events in $\Sigma = \{a, b, c, d\}$. Also, appointed by the President of the university, he has the power to disapprove every action (except, of course, the events in $\Sigma' = \{\tau_1, \tau_2\}$). Therefore, $\Sigma_c = \Sigma$.

The resulting supremal controllable and normal sublanguage $sup\mathcal{CN}(E)$ is shown in Figure 4(b). Based on this language, the administrator designs the following supervisor γ: disable a at the beginning and disable b after d. In other words, the professor can only take the morning train and is not allowed to take a taxi at the train station.

The next day, the administrator overheard from his colleague that the same supervisor can be designed without lifting. Not knowing how to do this exactly, he decides to try the design without lifting on his own. He therefore converted $\mathcal{P}$ to a deterministic $\hat{\mathcal{P}}$. In doing so, there is no problem in marking $\hat{\mathcal{P}}$: $\hat{Q}_m = \{\{1\}, \{5\}, \{6\}, \{7, 8\}, \{9\}\}$. However, there is

uncontrollable, in which case a language is controllable and observable if and only if it is controllable and normal [22]. Thus, 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 $Q_s = Q - Q_b$ as possible (see [11]).

Such a supervisor can be designed with or without the lifting procedure, as outlined in the two ensuing algorithms. More details about synthesis aspects can be found in [11]. Before we turn to the formal presentation of the algorithms, we shall illustrate them through the following lighthearted example.

Example 2 The process $\mathcal{P}$ in Figure 3 represents the possible travel alternatives for a professor to attend a control conference in a nearby city. The professor can either take a morning train (event d) or an afternoon train (event a) to the conference city. At the train station, he can either take a bus (event c) or a taxi (event b) to the conference hotel. It is known that buses are always available in the morning but may or may not be available in the afternoon. On the other hand, taxis are available at any time. Since the professor does not want to be stranded at the train station, it is natural to assume that the marked states are $Q_m = \{1, 5, 6, 7, 8, 9\}$.

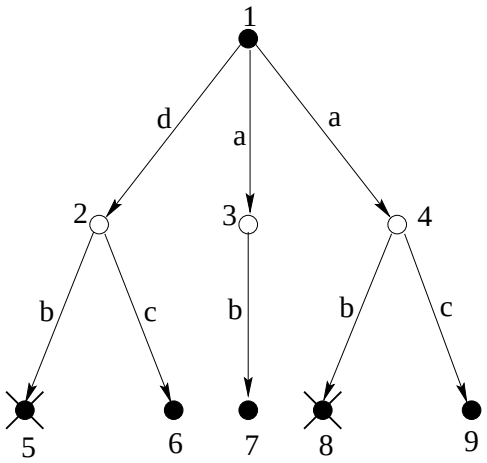


Figure 3: $\mathcal{P}$

In this time of budget cuts, the university has established a regulation stating that if buses are available, taking a taxi is not permitted. Therefore, the states 5 and 8 are illegal, that is, $Q_b = \{5, 8\}$.

Proof

The result follows from the above definition of marked states $\overline{Q}_m$.

■

By the above proposition and the fact that $\overline{\mathcal{P}}_t = \mathcal{P}$, it is clearly that the set of marked trajectories of $\overline{\mathcal{P}}_t$ is equal to those of $\mathcal{P}$.

We summarize the information in an automaton

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

that describes both the system and specification, where $\overline{Q}_b := \overline{Q} - \overline{Q}_t$, that is, the set of forbidden states. We can then use this model of our system with static specification to design the required supervisors.

Remark: The supervisor synthesis procedure with static specifications to be presented in the next section, is based directly on the nondeterministic automaton and is therefore "semantics-independent". For the case of dynamic specifications, the translation to an equivalent problem with static specifications, is crucially dependent on the semantics. Specifically, the trajectory-inclusion algorithm is obviously semantics-specific and would have to be suitably modified if another semantic formalism had been used for specification of legal behavior. However, the basic methodology employed would remain the same. The only difference would be in the embedding procedure of the specification in the plant model that would have to be modified. ■

5 Supervisor synthesis

We shall assume, without loss of generality, that we are presented with a problem, stated in terms of a static specification as in Section 3. That is, we are required to synthesize a supervisor for a system given by

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

We lift $\mathcal{P}$ to $\tilde{\mathcal{P}}$ and define the legal language E as in Section 3. If the existence condition in Section 3 is not satisfied, that is, if E is not controllable and observable, then we would like to synthesize a supervisor that achieves the largest possible sublanguage of E. This largest sublanguage is the supremal controllable and normal sublanguage of E, denoted by $sup\mathcal{CN}(E)$. 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

1. The trajectory model of $\overline{\mathcal{P}}$ is the same as $\mathcal{P}$ ([11] Proposition 9):

$$\overline{\mathcal{P}} = \mathcal{P}.$$

2. The language marked by the set of "good" states $\overline{Q}_g$ of $\overline{\mathcal{P}}$ is $L(\hat{\mathcal{H}})$:

$$L_m(\overline{\mathcal{P}}) = L(\hat{\mathcal{H}}).$$

It follows therefore, that if we can ensure that the supervised system stays within $\overline{Q}_g$, then the language specification imposed by $\hat{\mathcal{H}}$ is satisfied. Thus the automaton $\overline{\mathcal{P}}$ captures in a "static" setting the language specification imposed by $\mathcal{H}$.

In the next step, we embed in $\overline{\mathcal{P}}$ the trajectory-model constraints imposed by $\hat{\mathcal{H}}$. This is accomplished by the trajectory inclusion algorithm (Algorithm 3) of [11], which constructs from $\overline{\mathcal{P}}$ another automaton

$$\overline{\mathcal{P}}_t = (\Sigma \cup \{\epsilon\}, \overline{Q}, \overline{\delta}, \overline{q}_0, \overline{Q}_t),$$

where $\overline{Q}_t \subseteq \overline{Q}_g$ satisfies the condition that any path p in $\overline{\mathcal{P}}_t$ belongs to $\overline{Q}_t$ (i.e. visits only states in $\overline{Q}_t$) if and only if $t_p \in \hat{\mathcal{H}}$. By this property, the trajectory-model constraint imposed by $\hat{\mathcal{H}}$ is satisfied if the supervised system stays in $\overline{Q}_t$. Note also that

$$\overline{\mathcal{P}}_t = \overline{\mathcal{P}} = \mathcal{P}.$$

We turn now to the nonblocking constraint imposed by the set Q_m of marked states in $\mathcal{P}$. We must insist, in addition to what has been said above, that in the supervised system each trajectory be also a prefix of a trajectory that ends in a marked state. To this end, we need to define marking in $\overline{\mathcal{P}}_t$ in a way that correctly translates the nonblocking requirement.

This translation is quite straightforward:

$$\overline{Q}_m = Q_m \times (\hat{H}_d \cup \{h_b\}).$$

To show that this marking is indeed correct, we first note that since $\overline{Q} = Q \times (\hat{H}_d \cup \{h_b\})$ and $\overline{\mathcal{H}}_d$ is deterministic, there is a one-to-one correspondence between a path in $\mathcal{P}$ and a path in $\overline{\mathcal{P}}_t$.

Proposition 2 A path in $\mathcal{P}$ is marked by Q_m if and only if the corresponding path in $\overline{\mathcal{P}}_t$ is marked $\overline{Q}_m$.

For preferred customers, since only limited nondeterminism is allowed, the specification is given by $\mathcal{H}$ shown in Figure 2 (b).

By applying the above four step procedure, we obtain the static specification as in Figure 1 with the good states

$$\{1, 2, 3, 4, 5, 6, 7, 14, 15, 16, 17, 18\}.$$

Therefore, the orders of preferred customers will only be sent to Departments A and C.

■

We now formally outline the procedure to translate a dynamic specification into a static specification followed in [11] and emphasize especially the adjustments that need to be made to accommodate the nonblocking requirement imposed by the marked states. We begin by considering the plant as described by $\mathcal{P}$, but we temporarily ignore the set Q_m of marked states. Just as in [11], the specification is embedded in $\mathcal{P}$ prior to the translation of the problem into a "supervisory control problem under partial observation".

The first step of the procedure is to remove from $\mathcal{H}$ all trajectories whose traces do not belong to $L(\mathcal{P})$. Such trajectories, if they exist in $\mathcal{H}$, clearly cannot be required to be possible in $\mathcal{P}$ (or in any of its sub-automata) after supervision. Thus, we construct a modified specification automaton

$$\hat{\mathcal{H}} = \mathcal{H}||det(L(\mathcal{P})).$$

As was seen in [11], $\hat{\mathcal{H}}$ retains all the nondeterministic behaviors permitted by $\mathcal{H}$ and possible in $\mathcal{P}$.

The second step is to consider the language restrictions imposed by $L(\hat{\mathcal{H}})$. To this end, we first compute the automaton $det(L(\hat{\mathcal{H}}))$, the deterministic automaton whose generated language is $L(\hat{\mathcal{H}})$. We extend this automaton in standard fashion to an automaton

$$\overline{\mathcal{H}}_d = (\Sigma, \hat{H}_d \cup \{h_b\}, \overline{\psi}_d, \hat{h}_{d0}, \hat{H}_d)$$

by adding a *bad* state h_b to the state set $\hat{H}_d$ of $det(L(\hat{\mathcal{H}}))$ [11], so that $L(\overline{\mathcal{H}}_d) = \Sigma^*$ and $L_m(\overline{\mathcal{H}}_d) = L(\hat{\mathcal{H}})$. We then construct the automaton

$$\overline{\mathcal{P}} = \mathcal{P}||\overline{\mathcal{H}}_d = (\Sigma \cup \{\epsilon\}, \overline{Q}, \overline{\delta}, \overline{q}_0, \overline{Q}_g),$$

where $\overline{Q} = Q \times (\hat{H}_d \cup \{h_b\})$ and $\overline{Q}_g = Q \times \hat{H}_d$.

The automaton $\overline{\mathcal{P}}$ has the following two desired properties:

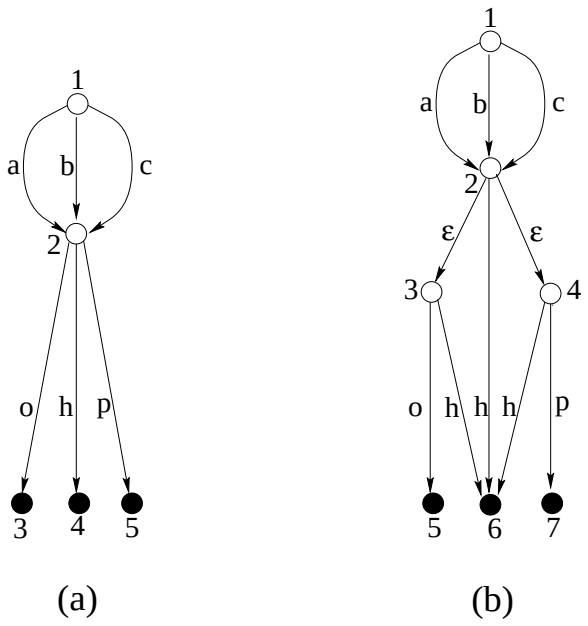


Figure 2:

In Step 3, we impose the trajectory model specification; that is, we find all trajectories in $\mathcal{P}$ that also belong to $\hat{\mathcal{H}}$ by applying the trajectory inclusion algorithm of [11]. This results the following legal states:

$$\{1, 2, 3, 4, 5, 6, 7, 17\}.$$

In other words, because the specification does not allow nondeterminism, only the deterministic subautomaton of $\mathcal{P}$ is legal.

The last step determines the marked states, which are same as $\mathcal{P}$. Therefore, after trimming, only the branches starting with a and the branch with ch are legal, which means that the orders of VIP customers will be sent to Department A, or to Department C if it is a high precision order.

For ordinary customers, since all nondeterminism is allowed, we only need a language specification such as $L(\mathcal{H}) = \Sigma^*$. This will result in declaring all states in $\mathcal{P}$ legal. This is equivalent to say that the orders of ordinary customers can be sent to any departments. The same conclusion can be obtained with the following trajectory model specification:

$$\mathcal{H} = nondet(\Sigma^*),$$

where $nondet(\Sigma^*)$ is the most nondeterministic trajectory model having Σ^* as its trace set [11].

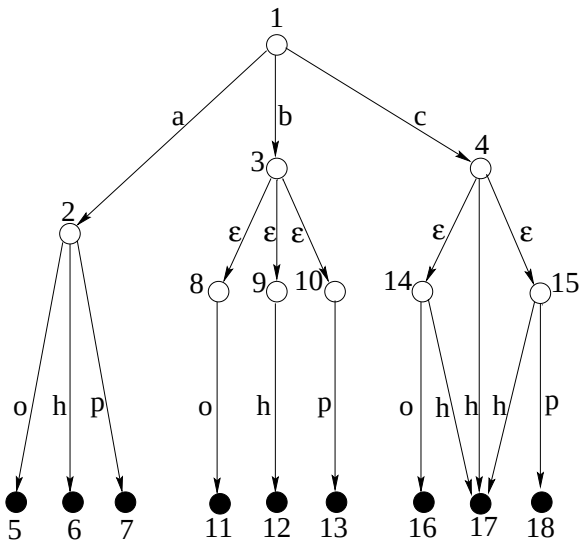


Figure 1:

1. VIP customers: their orders must be processed deterministically as requested.

2. Preferred customers: their order for high precision processing must be processed deterministically.

3. Ordinary customers: their order may be processed nondeterministically if necessary.

Let us now formally describe the specifications for each type of customers.

For VIP customers, no nondeterminism is allowed. This specification can be given by

$$\mathcal{H} = det(\Sigma^*)$$

where $\Sigma = \{a, b, c, h, p, o\}$.

The translation of this dynamic specification into an static specification can be performed in four steps. The first step is to remove from $\mathcal{H}$ all trajectories whose traces do not belong to $L(\mathcal{P})$. Therefore, we construct

$$\hat{\mathcal{H}} = \mathcal{H}||det(L(\mathcal{P})) = det(L(\mathcal{P})),$$

which is shown in Figure 2 (a).

In Step 2, we consider the language specification imposed by $\hat{\mathcal{H}}$; that is, we find all trajectories in $\mathcal{P}$ whose traces belong to $L(\hat{\mathcal{H}})$. (Since the language specification allows every trajectory possible in $\mathcal{P}$, all states in $\mathcal{P}$ are legal.)

the supervised system needs to be confined. This set of behaviors will be given as a set of trajectories or, more explicitly, as a trajectory model

$$\mathcal{H} = (\Sigma \cup \{\epsilon\}, H, \psi, h_0)$$

that specifies both the permitted set of traces and the degree of allowed nondeterminism in the supervised system. (Recall that if one trajectory model is contained in another, the first one is less nondeterministic that the second.)

Our nominal goal is to design a nonblocking supervisor γ such that (if possible)

$$\gamma/\mathcal{P} = \mathcal{H},$$

A pre-condition for the existence of such a supervisor γ is that the behavior described by $\mathcal{H}$ be physically possible in (some sub-automaton of) $\mathcal{P}$. Since the specification $\mathcal{H}$ is often obtained independently of $\mathcal{P}$, in the sense that it does not depend on pre-knowledge of $\mathcal{P}$, this pre-condition will generally not be satisfied. The automaton $\mathcal{H}$ will then have to be modified, in a way similar to that described in [11] to accommodate the behavior possible in $\mathcal{P}$, except that now marking must also be considered. After modifying $\mathcal{H}$, we will then translate the above dynamic specification into an equivalent static specification. Before formally presenting a procedure to do these, let us first illustrate the idea by the following

Example 1 Consider a plant consisting of three departments. Each department is equipped with three types of machines: high precision machines, precision machines, and ordinary machines. Denote by h, p, and o, respectively, the event that an order is processed by a high precision machine, a precision machine, and an ordinary machine.

When an order arrives, it can be sent to one of the three departments by the plant manager. This event is denoted by a, b, and c, respectively, for Departments A, B, and C.

The three departments are under different managements. Department A, under an excellent management, can process an order precisely in the sense that the selection of three types of machines is deterministic. Department B, on the other hand, is poorly managed. It processes an order in a nondeterministic fashion by randomly selecting one of the machines. Department C is somehow in between, it processes a high precision orders in a deterministic fashion, but other orders in a nondeterministic way.

The uncontrolled system is therefore described by the process $\mathcal{P}$ shown in Figure 1, where the marked states are denoted by solid dots.

To ensure the survival of the plant under this undesired nondeterminism, the plant manager has classified his customers into three categories:

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 [19], there exists a nonblocking supervisor $\gamma : PL(\tilde{\mathcal{P}}) = L(\mathcal{P}) \to 2^{\Sigma_c}$ 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 a supervisor γ (under partial observation) for $\tilde{\mathcal{P}}$ such that $L_m(\gamma/\tilde{\mathcal{P}})$ is the supremal controllable and normal sublanguage of E, which is then the optimal supervisor. (Note that the $L_m(\tilde{\mathcal{P}})$-closeness property is preserved [22].) We will discuss such a synthesis in Section 5. In the next section, we shall discuss the nonblocking problem subject to dynamic specifications.

4 Dynamic specifications

Consider again a system modeled by a nondeterministic automaton

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

Suppose now that the legal specification is not given, as before, in terms of a subset of bad states of $\mathcal{P}$ that need to be avoided but, rather, in terms of a set of behaviors to which

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 [26]. From Proposition 7 of [11] 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 $\tilde{\mathcal{P}}$.

The supervised system is then given by

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

where || denotes the strict synchronous (parallel) composition (as defined in [11]).

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

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

That is,

$$\mathcal{P}_s = trim(\overline{\mathcal{P}_s}),$$

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 [11], 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:

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

where $Q_m \subseteq Q$ is a set of marked states that represent, for example, task completions.

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

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

Note that $\mathcal{P}_m \subseteq \mathcal{M}(\mathcal{P})$. We shall say that a set of marked trajectories $\mathcal{P}_m$ is a *spanning* set of $\mathcal{P}$, if it satisfies the condition that,

$$\mathcal{P} = cl(\mathcal{P}_m).$$

In that case we shall also say that $\mathcal{P}$ is *trim*. It is easy to verify that this definition of trimness is consistent with that in deterministic automata, where an automaton is said to be trim if it is both accessible and coaccessible (see e.g. [26]). If $\mathcal{P}$ is not trim, then we can obtain $trim(\mathcal{P})$ by computing its largest accessible and coaccessible component, much in the same way as in the deterministic case[3].

We now specify a subset $Q_b \subseteq Q$ of forbidden states that the system is not allowed to visit. (Although it may be natural to assume that $Q_b \cap Q_m = \emptyset$, for technical reasons we do not make this assumption here.)

Our system model can now be written as

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

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 [11]

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

[3]We shall not elaborate on the trim operation in this paper. The reader can easily establish his own algorithm in analogy to the deterministic case.

algorithm, into a static specification in an equivalent problem, where the system is modeled as an automaton

$$\overline{\mathcal{P}} = (\Sigma \cup \{\epsilon\}, Q, \delta, q_o, Q_b)$$

in which Q_b is a set of "bad" states whose avoidance is equivalent to satisfaction of the specification as given by $\mathcal{H}$. The next step is to "lift" $\overline{\mathcal{P}}$ to a deterministic automaton

$$\tilde{\mathcal{P}} = (\Sigma \cup \Sigma', \tilde{Q}, \tilde{\delta}, q_0, \tilde{Q}_b),$$

using the procedure "Extend". Then a "legal" language $E \subseteq L(\tilde{\mathcal{P}})$ is defined with respect to $\tilde{\mathcal{P}}$ as

$$E = \{s \in L(\tilde{\mathcal{P}}) : (\forall t \leq s)\tilde{\delta}(q_o, t) \notin \tilde{Q}_b\},$$

where $t \leq s$ denotes that t is a prefix of s. It is then well known that there exists a supervisor γ (under partial observation) such that $L(\gamma/\tilde{\mathcal{P}}) = E$ if and only if E is controllable with respect to $\Sigma_c (\subseteq \Sigma)$ and observable with respect to Σ (the observable subset of $\Sigma \cup \Sigma'$). It is proved in [11] that the latter (deterministic) supervisory control problem is equivalent to the original (nondeterministic) problem and that the supervisor γ as obtained above, is precisely the supervisor that is required to achieve the solution to the nondeterministic supervisory control problem. If E is not controllable and observable, then the supremal controllable and normal sublanguage of E, denoted by $supCN(E)$, is used to synthesize a supervisor. (Since all controllable events are observable, the condition of controllability and observability is equivalent to controllability and normality and hence the solution obtained is also optimal.) Three algorithms are given in [11] for supervisor synthesis.

In [11], only the "safety" issue was considered. That is, there was no consideration of "liveness" and hence marked states were not introduced as a specification for task completion. Thus, the issue of possible blocking was not addressed. In the present paper, we shall consider the issue of supervisory control with liveness specification and, in particular, of nonblocking.

3 Marked trajectories and nonblocking supervisors

To add liveness considerations, we include, just as in the deterministic case, a set of marked states in our model. Thus, we shall consider systems modeled as nondeterministic automata of the form

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

with the automaton $\mathcal{P}$, we first associate with each state $q \in Q$ its *maximal-refusal-set* $X_q \subseteq \Sigma$, which is given by

$$X_q := \{\sigma \in \Sigma : (\forall q' \in \epsilon^*(q))\delta(q', \sigma) = \emptyset\}$$

where $\epsilon^*(q)$, the $\epsilon-$*closure* of q, is defined inductively [14] as

$$\begin{cases} q \in \epsilon^*(q); \text{ and} \\ q' \in \epsilon^*(q) \Rightarrow \delta(q', \epsilon) \subseteq \epsilon^*(q). \end{cases}$$

Then, with each path $p = (q_0, \sigma_1, q_1, ..., \sigma_k, q_k)$ of $\mathcal{P}$, (where some of the σ_i may be ϵ) we associate a trajectory t_p in the following way: First we represent p as a *formal* trajectory by replacing each state q_i in p by its maximal refusal set X_{q_i}, thus obtaining $\tilde{t}_p := (X_{q_0}, \sigma_1, X_{q_1}, ..., \sigma_k, X_{q_k})$. Then, to obtain the trajectory t_p associated with p, we delete all epsilons from $\tilde{t}_p$, and in the resulting string we replace all consecutive refusal sets by their union. Denoting by $\mathcal{M}(\mathcal{P})$ the set of trajectories associated (as described above) with all paths of $\mathcal{P}$, we obtain the trajectory-model associated with $\mathcal{P}$ as $\mathcal{P} := cl(\mathcal{M}(\mathcal{P}))$.

A state q is called $\epsilon-$*stable* if $\epsilon^*(q) = \{q\}$. A path $p = (q_0, \sigma_1, q_1, ..., \sigma_k, q_k)$ is $\epsilon-$*stable* if q_k is $\epsilon-$stable. It is readily noted that the set of trajectories t_p, associated with all $\epsilon-$stable paths in $\mathcal{P}$, is the dominance set $dom(\mathcal{P})$ of $\mathcal{P}$. Furthermore, since in a nondivergent process, there exists at least one ϵ-stable state in the ϵ-closure of every state,

$$\mathcal{M}(\mathcal{P}) := \bigcup_{t \in dom(\mathcal{P})} pref(t).$$

Conversely, for a trajectory model $\mathcal{P}$, Algorithm 1 of [11] (see also [8]) can be used to construct an automaton that generates $\mathcal{P}$, in which the state set is identified with the set $\mathcal{M}(\mathcal{P})$ of prefixes of dominant trajectories. Therefore, we can use either a trajectory model or a nondeterministic automaton to model a nondeterministic system. We shall use the same symbol to denote both the nondeterministic automaton and its associated trajectory model.

In [11], supervisory control of nondeterministic systems of the form

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

was investigated, subject to specification of legal behavior given by

$$\mathcal{H} = (\Sigma \cup \{\epsilon\}, H, \psi, h_0).$$

A supervisor synthesis procedure was presented in [11] that followed along the following steps. First, the above dynamic specification is translated, using a trajectory inclusion

We say that a set of trajectories $\mathcal{T}$ is *saturated* if the following condition holds:

$$(\forall k = 1, 2, ...)(\forall j : 0 \leq j \leq k)(\forall \sigma \in \Sigma - X_j)$$
$$(((X_0, (\sigma_1, X_1), ..., (\sigma_k, X_k)) \in \mathcal{T} \wedge (X_0, (\sigma_1, X_1), ..., (\sigma_j, X_j)(\sigma, \emptyset)) \notin \mathcal{T})$$
$$\Rightarrow (X_0, (\sigma_1, X_1), ..., (\sigma_j, X_j \cup \{\sigma\})...(\sigma_k, X_k)) \in \mathcal{T}).$$

Thus, loosely speaking, a set of trajectories is saturated if it includes trajectories in which events that are impossible appear as refusals.

With these definitions, we define a (generally) nondeterministic *process* $\mathcal{P}$ to be a closed and saturated subset of trajectories $\mathcal{P} \subseteq 2^{\Sigma} \times (\Sigma \times 2^{\Sigma})^*$. A process is *deterministic* if for every trajectory $(X_0, (\sigma_1, X_1), ..., (\sigma_k, X_k)) \in \mathcal{P}$ and any $\sigma \in \Sigma$

$$(X_0, (\sigma_1, X_1), ..., (\sigma_k, X_k), (\sigma, \emptyset)) \in \mathcal{P} \Leftrightarrow (X_0, (\sigma_1, X_1), ..., (\sigma_k, X_k \cup \{\sigma\})) \notin \mathcal{P}.$$

Thus, a process is deterministic whenever requested events are refused if and only if they are impossible.

The set of traces of all trajectories of a process $\mathcal{P}$ is called the *language* of $\mathcal{P}$ and denoted by $L(\mathcal{P})$. For a prefix-closed language $L \subseteq \Sigma^*$, the smallest[1] process $\mathcal{P}$ satisfying $L(\mathcal{P}) = L$ exist, is unique, and is denoted $det(L)$. This process is deterministic and is, in fact, the unique deterministic process whose language is L. An algorithm for construction of $det(L)$, which is based on the the above definition of determinism, can be found in [11].

Let $\mathcal{T}$ be a set of trajectories. We say that a trajectory $t \in \mathcal{T}$ is *dominant* (in $\mathcal{T}$) if there is no trajectory $t' \in \mathcal{T}$, $t' \neq t$, such that $t \sqsubseteq t'$. The set of all trajectories that are dominant in $\mathcal{T}$ is called the *dominance-set* of $\mathcal{T}$ and is denoted $dom(\mathcal{T})$. A process $\mathcal{P}$ is completely characterized by its dominance set, because $\mathcal{P} = cl(dom(\mathcal{P}))$.

An alternate way to represent a nondeterministic system is by a nondeterministic automaton (possibly with $\epsilon-$transitions),

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

over the event set Σ, with state set Q, nondeterministic transition function $\delta : Q \times (\Sigma \cup \{\epsilon\}) \rightarrow 2^Q$, and initial state q_0. We shall assume throughout the paper that the system is nondivergent, that is, that there are no unbounded $\epsilon-$paths (i.e., loops that consist entirely of $\epsilon-$transitions). [2] As before, the language generated by $\mathcal{P}$ is denoted by $L(\mathcal{P})$.

The trajectory-model representation and the automaton representation of a nondeterministic discrete-event system are related as follows. To obtain the set of trajectories associated

[1] in the sense of partial order by inclusion.

[2] Most results in this paper can be extended to divergent systems in a straightforward manner.

2 Preliminaries

In this section, we will briefly review the notations and results of [8] [11] on supervisory control of nondeterministic discrete-event systems.

A nondeterministic system can be represented by a trajectory model. A trajectory is an object in $2^{\Sigma} \times (\Sigma \times 2^{\Sigma})^*$ of the form

$$t = (X_0, \sigma_1, X_1, ..., X_{k-1}, \sigma_k, X_k),$$

where σ_i denotes the ith executed event, and where X_i, the ith *refusal*, denotes the set of events refused after the ith executed event. The trace associated with t is defined as

$$tr(t) = \sigma_1 ... \sigma_k.$$

A trajectory is called *valid* if $\sigma_i \notin X_{i-1}$ for all $i > 0$ (that is, when an event cannot be executed if it has just been refused).

Let a trajectory t be given by

$$t = (X_0, (\sigma_1, X_1), ..., (\sigma_k, X_k)).$$

A trajectory r is a *prefix* of t, denoted $r \preceq t$, if

$$r = (X_0, (\sigma_1, X_1), ..., (\sigma_j, X_j))$$

and $0 \leq j \leq k$. The set of all prefixes of t is called the *prefix-closure* of t and is denoted $pref(t)$.

A trajectory r is said to be *dominated* by t, denoted $r \sqsubseteq t$, if it is of the form

$$r = (Y_0, (\mu_1, Y_1), ..., (\mu_k, Y_k)),$$

with $\mu_i = \sigma_i$ for $1 \leq i \leq k$ and $Y_j \subseteq X_j$ for $0 \leq j \leq k$. The set of all trajectories dominated by t is called the *completion*, or *dominance-closure*, of t and denoted $comp(t)$. We also define the *closure* of t, denoted $cl(t)$, as

$$cl(t) := \bigcup_{v \in comp(t)} pref(v).$$

The closure of a set of trajectories $\mathcal{T}$, is given by

$$cl(\mathcal{T}) := \bigcup_{t \in \mathcal{T}} cl(t),$$

and a set of trajectories $\mathcal{T}$ is *closed* if

$$\mathcal{T} = cl(\mathcal{T}).$$

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 [11] and [12] 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 first 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 is then shown how 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.

While other semantic formalisms may be finer than the trajectory-model formalism, in that they may capture more nondeterministic detail, the approach developed in the present paper can easily be adapted to other formalisms as we discuss in Section 4. The trajectory model formalism was also used in [29] where discrete-event control of nondeterministic systems subject to language specifications was investigated and various existence conditions of supervisors were obtained.

The paper is organized as follows. In Section 2 we will briefly review the notations and results of [7] [8] [11] [10] on supervisory control of nondeterministic discrete-event systems. In Section 3 we introduce the concept of marked trajectories and formulate the nonblocking supervisory control problem with static specifications. In Section 4 we introduce dynamic specification and show how the problem is transformed to an equivalent one with static specifications. In Section 5 we discuss the supervisor synthesis problem and develop two synthesis algorithms: one with "lifting" and a second one that requires no lifting. In Section 6 we briefly discuss the supervisory control problem under partial observation; that is, the case in which some of the events of the nondeterministic system are unobservable. In Section 7 we briefly comment on the problem of decentralized control, and finally, in Section 8 we discuss the computational complexity problem.

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 [5] [15] [24] [25] [29]. 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. [24] 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, [10] [11] [12], 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 [11] and [12] 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 [10] [11] [12] we focused our

Abstract

In this paper we extend the theory of supervisory control of nondeterministic discrete-event systems, subject to nondeterministic specification, developed in [11]. We focus our attention on nonblocking and liveness considerations and develop algorithms for nonblocking-supervisor synthesis.

1 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 [27]. A great deal of attention was also given to the control of partially observed discrete-event systems [18] [19], 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 [18] [26] [27], algorithms for supervisor synthesis [1] [2] [3] [18] [19] [20], for off-line as well as on-line implementation [3] [9], 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 [13] and the associated *failures* semantics, bisimulation semantics [23] and labeled transition systems [6]. In [7] and [8] 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

Nonblocking Supervisory Control of Nondeterministic Systems *

Michael Heymann[1] and Feng Lin[2]

[1]Department of Computer Science
Technion, Israel Institute of Technology
Haifa 32000, Israel
e-mail: heymann@cs.technion.ac.il

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

October 17, 1996

*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.

This work by the first author was completed while he was a Senior NRC Research Associate at NASA Ames Research Center, Moffett Field, CA 94035.