References

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

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

[3] 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.

[4] M. Heymann and F. Lin, 1995. On observability and nondeterminism in discrete event systems: the general case. *forthcoming.*

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

[6] 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.

[7] 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.

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

[9] 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.

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

[11] 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.

[12] M. Shayman and R. Kumar, 1994. 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.

Algorithm 2 *Off-line synthesis*

1. *Extend*($\mathcal{F} \rightarrow \tilde{\mathcal{F}}$);
2. *Sup*$\mathcal{CN}$($\tilde{\mathcal{F}} \rightarrow \tilde{\mathcal{F}}_{\mathcal{CN}}$);
3. *Design a supervisor off-line.*

In Algorithm 2, Step 1 was described earlier. Step 2 calculates the supremal controllable and normal sublanguage of E [8]. We note that since in $\tilde{\mathcal{F}}$, all controllable events are observable ($\Sigma_c \subseteq \Sigma = \Sigma_o$), controllability and observability of a language is equivalent to controllability and normality of that language. In step 3 a supervisor is designed off-line in the usual way [8]. The supervisor thus obtained is "optimal" in the sense that it generates the largest behavior possible as shown in the following theorem.

Theorem 9 *The supervisor designed using Algorithm 2 is an optimal supervisor.*

Two other approaches can be used for supervisor synthesis. The second approach is to design a supervisor on-line instead of off-line. The advantage of this approach is that the computational complexity is linear at each step of event execution.

Algorithm 3 *On-line synthesis*

1. *Extend*($\mathcal{F} \rightarrow \tilde{\mathcal{F}}$);
2. *Sup*$\mathcal{C}$($\tilde{\mathcal{F}} \rightarrow \tilde{\mathcal{F}}_{\mathcal{C}}$);
3. *Design a supervisor on-line.*

Step 1 is same as that of Algorithm 2. Step 2 calculates the supremal controllable sublanguage of the legal language E. This can be done with linear complexity for a closed language E. In Step 3, we design a supervisor on-line using the results of [3]. The resulting supervisor will generate the supremal controllable and normal sublanguage of E, and hence is also optimal. As shown in [3], the complexity at each step of event execution is linear.

In view of the way *Sup*$\mathcal{CN}$ is calculated, we can modify Algorithm 2 by directly converting $\mathcal{F}$ to a deterministic automaton $\overline{\mathcal{F}}$ without adding the unobservable events Σ'. This leads to our third approach.

Algorithm 4 *Direct synthesis*

1. *Convert*($\mathcal{F} \rightarrow \overline{\mathcal{F}}$);
2. *SupC*($\overline{\mathcal{F}} \rightarrow \overline{\mathcal{F}}_{\mathcal{C}}$);
3. *Design a supervisor off-line.*

Procedure Convert that converts the nondeterministic automaton $\mathcal{F}$ into a deterministic one $\overline{\mathcal{F}}$ is standard [5]. Each state in $\overline{\mathcal{F}}$ is now a subset of states in $\mathcal{F}$. We call such a state "bad" if it contains a bad state of $\mathcal{F}$. Step 2 computes the supremal controllable sublanguage.

We can show that the resulting supervised systems using Algorithm 2 and Algorithm 4 are the same as far as traces in Σ^* are concerned.

Theorem 10 *Let γ_1 and γ_3 be the supervisors obtained from Algorithm 2 and Algorithm 4, respectively. Then*

$$L(\gamma_1/\mathcal{R}) = L(\gamma_3/\mathcal{R}).$$

The above results and algorithms can be generalized to non-closed case and will be discussed in [4].

where || denotes the synchronous composition.

Our goal is to synthesize a supervisor γ such that, if possible,

$$\gamma/\mathcal{R} = \mathcal{H}.$$

To obtain a necessary and sufficient condition for the existence of such a supervisor, we proceed as follows. First, we extend $\mathcal{H}$ to

$$\overline{\mathcal{H}} := (\Sigma \cup \{\epsilon\}, P \cup \{b\}, \overline{\mu}, p_0, P),$$

where the state b is called the "bad" state and the transition function $\overline{\mu}$ is given by

$$\overline{\mu}(p,\sigma) := \begin{cases} \mu(p,\sigma) & \text{if } \mu(p,\sigma) \text{ is defined,} \\ b & \text{if } \mu(p,\sigma) \text{ is undefined.} \end{cases}$$

Note that $L(\overline{\mathcal{H}}) = \Sigma^*$ and $L_m(\overline{\mathcal{H}}) = L(\mathcal{H})$.

Next, we take the synchronous composition of $\mathcal{R}$ and $\overline{\mathcal{H}}$ to obtain:

$$\mathcal{F} = Acc(\mathcal{R}||\overline{\mathcal{H}}) = Acc(\Sigma \cup \{\epsilon\}, F, \delta, f_o, F_b).$$

where $Acc(.)$ denotes the accessible component. The state set $F = Q \times (P \cup \{b\})$ is partitioned into two classes: "good" states $F_g = Q \times P$ and "bad" states $F_b = Q \times \{b\}$. It is readily noted that, since $L(\overline{\mathcal{H}}) = \Sigma^*$, it follows that $L(\mathcal{F}) = L(\mathcal{R})$. Thus, $\mathcal{F}$ constitutes a model of the process, and the specification is modeled as the subprocess of $\mathcal{F}$ that consists of all trajectories whose corresponding states do not intercept F_b. In some applications, the system may be given directly as a nondeterministic process (or automaton) $\mathcal{F}$ with the specification as a sub-process. In this latter case the above procedures for obtaining $\mathcal{F}$ can be omitted.

We next use the procedure Extend, given in the previous section, to "lift" $\mathcal{F}$ to a deterministic automaton $\tilde{\mathcal{F}}$ by adding a set Σ' of unobservable events. The set of good states $\tilde{F}_g$ and bad states $\tilde{F}_b$ in $\tilde{\mathcal{F}}$ are defined accordingly. Thus, we obtain

$$\tilde{\mathcal{F}} = (\Sigma \cup \Sigma', \tilde{F}, \tilde{\delta}, f_o, \tilde{F}_b),$$

where

$$\tilde{F}_b = F_b \cup ((\tilde{F} - F) \cap \{\tilde{\delta}(q,\sigma) : |\delta(q,\sigma)| > 1 \wedge \delta(q,\sigma) \cap (F - F_b) = \emptyset.\})$$

Define the "legal" language $E \subseteq L(\tilde{\mathcal{F}})$ to be the set of all traces that visit only good states in $\tilde{\mathcal{F}}$:

$$E = \{s \in L(\tilde{\mathcal{F}}) : (\forall t \preceq s)\tilde{\delta}(x_o, t) \notin \tilde{F}_b\}.$$

Clearly, $det(E)\backslash_{\Sigma'} = \mathcal{H}$.

With these preparations, we can now state a necessary and sufficient condition for the existence of a supervisor where in the following theorem, observability is defined for $\Delta = \Sigma \cup \Sigma'$ and $\Sigma_{uo} = \Sigma'$.

Theorem 8 *For $\mathcal{R}$ and $\mathcal{H}$ given as above, there exists a supervisor γ such that $\gamma/\mathcal{R} = \mathcal{H}$ if and only if E is controllable and observable with respect to $L(\tilde{\mathcal{F}})$.*

The above theorem shows that we can translate a supervisory control problem for a nondeterministic system into a supervisory control problem for an lifted deterministic system: We can first synthesize $\tilde{\gamma}$ for the deterministic system and then (easily) obtain γ from $\tilde{\gamma}$ for the nondeterministic system by letting $\gamma(s) = \tilde{\gamma}(s) \cap \Sigma$. The significance of this result is that all the methodology developed for deterministic systems can now be applied to nondeterministic systems. In particular, if E is not controllable and observable, then the best we can do is to design a supervisor using the following

Example 1 The projection of the process $\mathcal{R} = cl(\{b\}, a, \{a\}, b, \{a, b\})$ on $\{b\}$ is $\mathcal{R}\backslash_{\{a\}} = cl(\emptyset, b, \{b\})$ which is deterministic. Note, however, that the projection of the trajectory $t = (\{b\}, a, \{a\}, b, \{a, b\})$ is $t\backslash_{\{a\}} = (\{b\}, b, \{b\})$ which is not valid.

Consider a nondeterministic automaton, possibly with $\epsilon-$transitions,

$$\mathcal{R} = (\Sigma \cup \{\epsilon\}, Q, \delta, q_o)$$

over the event set Σ. We introduce now a procedure for constructing a deterministic automaton

$$\tilde{\mathcal{R}} = (\Sigma \cup \Sigma', \tilde{Q}, \tilde{\delta}, q_o)$$

over an extended event set $\Sigma \cup \Sigma'$, such that $\mathcal{R} = \tilde{\mathcal{R}}\backslash_{\Sigma'}$. That is, $\tilde{\mathcal{R}}$ reduces to $\mathcal{R}$ upon replacing by ϵ-transitions all its transitions labeled by events in Σ'. The procedure is based on first extending $\mathcal{R}$ to a standard nondeterministic automaton with $\epsilon-$transitions, and then replacing the ϵ labels by labels from the event set $\Sigma' = \{\tau_1, \tau_2, ...\}$.

Procedure Extend$(\mathcal{R} \to \tilde{\mathcal{R}})$

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

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

5 Supervisory control: closed case

In this section, we consider the supervisory control of closed nondeterministic systems, that is, we assume every state is marked and we do not consider the issue of blocking. The general (nonclosed) case has also been investigated and will be considered in [4].

We allow both the system and the specification to be nondeterministic, that is, to be modeled by trajectory models or nondeterministic automata. We shall denote the system and specification by $\mathcal{R}$ and $\mathcal{H}$, respectively, where the nondeterministic automaton of $\mathcal{H}$ is given by

$$\mathcal{H} = (\Sigma \cup \{\epsilon\}, P, \mu, p_0).$$

Naturally, we shall require that $\emptyset \neq L(\mathcal{H}) \subseteq L(\mathcal{R})$.

Control is achieved as usual by a supervisor $\gamma : L(\mathcal{R}) \to 2^{\Sigma_c}$. The supervised system is denoted by $\gamma/\mathcal{R}$. To formally obtain $\gamma/\mathcal{R}$, we first compute the language $L(\gamma/\mathcal{R})$ as defined in Section 2. The behavior of the supervised system is then described by

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

whenever $\mathcal{Q} \subseteq \mathcal{P}$. (In particular, it is shown in [2] that $\mathcal{P}$ can then be thought of as evolving into $\mathcal{Q}$ through $\epsilon-$transitions).

For each trajectory model, we can construct a corresponding nondeterministic automaton with ϵ-transitions, using the algorithm in [2]. Similarly, for each nondeterministic automaton with ϵ-transitions, it is easy to construct a corresponding trajectory model by first identifying for each state a *maximal-refusal-set* and then associating with each $\epsilon-$stable path [4] a corresponding dominant trajectory. Therefore, we can use either the trajectory model or its associated nondeterministic automaton to model a nondeterministic system. Henceforth, in this paper, we shall use the same symbol $\mathcal{R}$ to denote both the nondeterministic automaton and the trajectory model of a process. The languages generated and marked by $\mathcal{R}$ are denoted by $L(\mathcal{R})$ and $L_m(\mathcal{R})$ respectively.

4 Nondeterminism and unobservability

Let $\tilde{\mathcal{R}}$ be a process over the event set $\Sigma \cup \Sigma'$, where Σ' is the set of unobserved internal events. In other words, an external observer (e.g., a supervisor) of $\tilde{\mathcal{R}}$ can observe the events in Σ but not events in Σ'. To obtain a suitable model for the partially observed process, we define a projection (or *internalization*) operator $\pi_{\Sigma'} : 2^{\Sigma \cup \Sigma'} \times ((\Sigma \cup \Sigma') \times 2^{\Sigma \cup \Sigma'})^* \rightarrow 2^{\Sigma} \times (\Sigma \times 2^{\Sigma})^*$ as follows.

For a trajectory

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

the projection $t\backslash_{\Sigma'} = \pi_{\Sigma'}(t)$ is obtained from t by the following procedure.

1. Delete from t all occurrences of event symbols that belong to Σ' (both as executed events and refused events). Thus, each refusal set X_i becomes $X_i - \Sigma'$.

2. Replace all consecutive refusal sets whose associated execution event symbols have been deleted, by their union. That is, if (in t) $\sigma_{i-1} \in \Sigma$ and $\sigma_i, \sigma_{i+1}, ..., \sigma_l \in \Sigma'$, then replace $X_{i-1} - \Sigma'$ by $(X_{i-1} - \Sigma') \cup (X_i - \Sigma') \cup ... \cup (X_l - \Sigma')$.

In general, the trajectory $t\backslash_{\Sigma'}$ thus obtained may not be valid. (Recall that a trajectory is valid if $\sigma_j \notin X_{j-1}$ for all j). The projection $\tilde{\mathcal{R}}\backslash_{\Sigma'}$ of a process $\tilde{\mathcal{R}}$ is a obtained as a subset of the set of valid trajectories projected from $\tilde{\mathcal{R}}$, as defined in [2] [4][5]. An alternate way to obtain the projection is to view $\mathcal{R}$ as a nondeterministic automaton, in which case $\tilde{\mathcal{R}}\backslash_{\Sigma'}$ is obtained from $\mathcal{R}$ by replacing in it all events $\sigma \in \Sigma'$ by ϵ.

Let $\tilde{\mathcal{R}}$ be a deterministic process over $\Sigma \cup \Sigma'$. We define the observability of $\tilde{\mathcal{R}}$ in terms of $L(G) = (\Sigma \cup \Sigma')^*$, $\Delta = \Sigma$ and $\Sigma_{uo} = \Sigma'$, that is, $\tilde{\mathcal{R}}$ is observable if

$$(\forall s, s' \in L(\tilde{\mathcal{R}}))Ps = Ps' \Rightarrow (\forall \sigma \in \Sigma)(s\sigma \in L(\tilde{\mathcal{R}}) \Rightarrow s'\sigma \in L(\tilde{\mathcal{R}}))$$

where P is the projection from $(\Sigma \cup \Sigma')^*$ to Σ^*.

Let $\mathcal{R}$ be the process obtained by projecting $\tilde{\mathcal{R}}$ on Σ; that is, $\mathcal{R} = \tilde{\mathcal{R}}\backslash_{\Sigma'}$. Then the observability of $\tilde{\mathcal{R}}$ is related to the determinism of $\mathcal{R}$ as shown in the following two theorems.

Theorem 6 *If $\tilde{\mathcal{R}}$ is observable, then $\mathcal{R}$ is deterministic.*

Theorem 7 *If $\tilde{\mathcal{R}}$ is unobservable and every trajectory obtained from its projection is valid, then $\mathcal{R}$ is nondeterministic.*

[5] The detailed definition of the projection of a process is omitted here because of space limitation.

Definition 4 [4] A (possibly) nondeterministic *process* $\mathcal{P}$ is a closed and saturated subset $\mathcal{P} \subseteq 2^{\Sigma} \times (\Sigma \times 2^{\Sigma})^*$ *of valid trajectories.*

The saturation condition on the set of trajectories of a process implies that if an event is impossible it will be refused. (We shall later see that while in nondeterministic processes events need not be impossible to be refused, in deterministic processes events are refused if and only if they are impossible.)

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 trajectory in $\mathcal{T}$ is called *maximal* if it is dominant and there is no trajectory $t' \in \mathcal{T}$ such that $t \preceq t'$. The subset of all maximal trajectories in $\mathcal{T}$ is called the *maximal set* of $\mathcal{T}$ and is denoted $max(\mathcal{T})$.

The following Theorem states that a process $\mathcal{P}$ is completely characterized by its dominance set.

Theorem 4 Let $\mathcal{P}$ be a process. Then $cl(dom(\mathcal{P})) = \mathcal{P}$.

We are now ready to define a *deterministic* process in the trajectory model setting:

Definition 5 A process $\mathcal{P}$ is called *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 events are refused if and only if they are impossible.

Now, let $L \subseteq \Sigma^*$ be a prefix-closed language (set of traces) and consider the class of all trajectory models that share L as their trace set. We note that this set is never empty and construct the following trajectory model, denoted $det(L)$, that belongs to this class:

Algorithm 1 *Construction of* $det(L)$.

1. $(\emptyset, \epsilon) \in det(L)$.
2. *Proceed by induction on string length: For* $t = (X_0, (\sigma_1, X_1)...(\sigma_k, X_k)) \in det(L)$ *and* $\sigma \in \Sigma$,

$$(X_0, (\sigma_1, X_1)...(\sigma_k, X_k \cup \{\sigma\})) \in det(L) \Leftrightarrow tr(t)^\frown\sigma \notin det(L)$$
$$(X_0, (\sigma_1, X_1)...(\sigma_k, X_k)(\sigma, \emptyset)) \in det(L) \Leftrightarrow tr(t)^\frown\sigma \in det(L)$$

The following theorem summarizes our preceding discussion and characterizes deterministic processes [2] [12]:

Theorem 5 Let $\mathcal{P}$ be a process and let $L(\mathcal{P})$ be its trace set. Then $\mathcal{P}$ is deterministic if and only if for every process $\mathcal{Q}$ such that $L(\mathcal{Q}) = L(\mathcal{P})$,

$$\mathcal{P} = det(L(\mathcal{P})) \subseteq \mathcal{Q}.$$

Thus a deterministic process is uniquely defined by its associated trace set and, in fact, is the smallest process associated with a given trace set.

In view of the preceding discussion it is clear that if $\mathcal{P}$ and $\mathcal{Q}$ are two processes defined over the same event set Σ, we are justified in saying that $\mathcal{P}$ is *more nondeterministic* than $\mathcal{Q}$

[4] The above definition is a simplified version of Definition 12.1 in [2] since we deal here only with termination-free nondivergent processes. The concepts of termination and divergence are discussed in detail in [2]. Intuitively, a process is divergent if it can engage in an unbounded string of unobserved transitions.

the system might have rejected (or refused), if offered, after each successful event. Thus, a trajectory is an oject 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 X_i, the ith *refusal*, denotes the set of events refused after the ith executed event. The *initial refusal* X_0 is the set of events that are refused before any event is executed. We call the integer k the *length* of t, denoted $|t|$, and the trace associated with t is defined as

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

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

Let t be a trajectory 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 trajectory r is called a *proper* prefix of t if $j < k$. We denote the prefix of length j of t by $pref_j(t)$ (so that $pref_0(t) = (X_0, \epsilon)$, where ϵ denotes the empty string, and $pref_k(t) = t$). 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 $comp(t)$.

Finally, we define the *closure* of t, denoted $cl(t)$, as

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

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

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

A set of trajectories $\mathcal{T}$ is *closed* [2] if

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

We say that a set of trajectories $\mathcal{T}$ is *saturated*[3] if the following condition holds:

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

We are now in a position to define a (nondeterministic) process through its associated set of trajectories. Intuitively, we identify a process $\mathcal{P}$ with the set of all trajectories associated with possible runs of $\mathcal{P}$. More formally, we have the following

[2] A closed set of trajectories is always nonempty since it includes the null trajectory $(\emptyset, \epsilon)$.

[3] Note that the term "saturated" as defined here is different from the way it was used in [12].

$s \in PL(G)$, the events $\sigma \in \gamma(s)$ are disabled by the supervisor. The languages generated and marked by the supervised system are denoted by $L(\gamma/G)$ and $L_m(\gamma/G) = L(\gamma/G) \cap L_m(G)$ respectively, where $L(\gamma/G)$ is given recursively by

1 $\epsilon \in L(\gamma/G)$

2 $(\forall s \in L(\gamma/G))(\forall \sigma \in \Sigma) s\sigma \in L(\gamma/G) \Leftrightarrow s\sigma \in L(G) \wedge s \in L(\gamma/G) \wedge \sigma \notin \gamma(s)$.

We say K is closed if it equals its prefix closure $K = \overline{K}$ and K is $L_m(G)$-closed if $K = L_m(G) \cap \overline{K}$. We also define controllability and observability as follows [11] [8].

Definition 1 *A language $K \subseteq L(G)$ is controllable (with respect to Σ_{uc}) if*

$$(\forall s \in \overline{K})(\forall \sigma \in \Sigma_{uc}) s\sigma \in L(G) \Rightarrow s\sigma \in \overline{K}.$$

Definition 2 *A language $K \subseteq L(G)$ is observable (with respect to Δ and Σ_{uo}) if*

$$(\forall s, s' \in \overline{K}) Ps = Ps' \Rightarrow (\forall \sigma \in \Delta)(s\sigma \in \overline{K} \wedge s'\sigma \in L(G) \Rightarrow s'\sigma \in \overline{K}).$$

The controllability and observability characterize the existence of a supervisor as proved in [8] and summarized in the following theorem.

Theorem 1 *Let $K \subseteq L(G)$ be non-empty. Then there exists a supervisor γ such that $L(\gamma/G) = K$ if and only if K is closed, controllable and observable.*

Another useful concept is normality which is defined as follows [8].

Definition 3 *A language $K \subseteq L(G)$ is normal if*

$$(\forall s \in L(G)) Ps \in P\overline{K} \Rightarrow s \in \overline{K}.$$

One important fact regarding the relation between observability and normality is the following theorem, which is basic to our development in this paper [9].

Theorem 2 *Assume $\Sigma_c \subseteq \Sigma_o$. Then a language is controllable and observable if and only if it is controllable and normal.*

A nice property of controllable (normal) languages is that they are closed under union. So let $\mathcal{C}(E)$ ($\mathcal{N}(E)$, $\mathcal{CN}(E)$, respectively) denote the set of controllable (normal, controllable and normal, respectively) sublangauges of E. then we have [8] [9].

Theorem 3 *The supremal elements $sup\mathcal{C}(E)$, $sup\mathcal{N}(E)$, and $sup\mathcal{CN}(E)$ exist and they preserve the properties of closeness and $L_m(G)$-closeness.*

3 Nondeterminism

In this section we briefly review the trajectory-model formalism of [2] (see also [12]) that has been developed as the basic tool for modeling and analysis of nondeterministic discrete event systems.

Just as the *trace* $s \in L(G) \subseteq \Sigma^*$ is a record of the string of events executed in a given run of a system G, the *trajectory* is also a record associated with a run of G. It is more detailed than the trace in that it lists, in addition to the successfully executed events, also events that

at least as powerful as the corresponding supervisor that is based only on knowledge of the projected plant and specification. This is because the nondeterministic (projected) model (and specification) can be derived from the deterministic model (and specification) while the converse is not generally true. Indeed it can be shown, that under certain circumstances, the deterministic partial observation setting is strictly more powerful than the nondeterministic counterpart. Conversely, one may be given a nondeterministic model, along with a deterministic or nondeterministic specification, as the basis for synthesis of a supervisor. One can then try to attack the problem directly in the nondeterministic setting or, alternatively, one can view the problem as a consequence of a deterministic framework that appears to be nondeterministic because of the internalization (unobservability) of some hypothetical events. In the latter case, the supervisor could have been designed by the conventional framework of control under partial observation. A natural question that arises is whether the two approaches lead to the same results. It is the focus of the present paper to investigate in detail the above raised issues.

We first briefly review the theory of supervisory control of discrete event systems under partial observation and recall the main relevant results on the existence of supervisors and some common procedures for supervisor synthesis. We also review in some detail the main concepts of nondeterministic discrete event systems and their representations, and reexamine the relation between the trajectory models and their corresponding nondeterministic automata. It is then shown that if the language describing a deterministic system is Σ^*-observable, then the projection of the system is deterministic. On the other hand, if the language is unobservable, its projection is not necessarily nondeterministic as can be shown by example. However, we prove that if all trajectories of the projected system remain "valid", then the projected system is nondeterministic.

Next, a "lifting" formalism is derived by which the nondeterministic system is translated (or lifted) to a certain deterministic system by introducing hypothetical events (that are assumed to be the internalized events whose unobservability led to nondeterminism). The lifted system is constructed so that its projection yields back the original nondeterministic system. It is then easily seen that the language obtained by the lifting of any language that is defined over the original event set to the extended event set of the lifted system, is normal with respect to the lifted system. This fact immediately implies that there is a very natural correspondence between optimal supervisors obtained for the lifted system to satisfy a normal language specification under partial observation, and supervisors for the nondeterministic system designed to satisfy deterministic (language) specifications. Specifically, a necessary and sufficient condition for satisfying a language specification by supervisory control of a nondeterministic system, is controllability of the language. This result has been previously obtained in [12] via a supervisory control theory of nondeterministic systems. While the direct derivation of this result in the nondeterministic framework is far from being trivial, it is straightforward in the proposed setting. In the present paper we examine nondeterministic systems with both deterministic and nondeterministic specifications in the closed case. The general (nonclosed) case will be discussed elsewhere [4].

2 Supervisory control under partial observation

Let us now review the basic results of supervisory control for deterministic systems under partial observation. The uncontrolled system is described by a (deterministic) automaton $G = (\Sigma, Q, \delta, q_o, Q_m)$ with its elements defined in a usual way. The languages generated and marked by G are denoted by $L(G)$ and $L_m(G)$ respectively. The event set is partitioned into controllable (observable) and uncontrollable (unobservable) event sets: $\Sigma = \Sigma_c \cup \Sigma_{uc} (= \Sigma_o \cup \Sigma_{uo})$. A supervisor is a disablement map $\gamma : PL(G) \to 2^{\Sigma_c}$ (where $P : \Sigma^* \to \Sigma_o^*$ is the projection that deletes the unobserved events) such that following an observed trace

On Observability and Nondeterminism in Discrete Event Control [1]

Michael Heymann[1] and Feng Lin[2]
[1]Department of Computer Science
Technion, Israel Institute of Technology, Haifa 32000, Israel
[2]Department of Electrical and Computer Engineering
Wayne State University, Detroit, MI 48202, USA

Abstract

The control of partially observed discrete event systems has received a great deal of attention in the literature. Necessary and sufficient conditions for existence of supervisors, algorithms for supervisor synthesis, for off-line as well as on-line implementation, have been obtained, and a wide variety of important questions have been investigated. In recent years, a related problem has been receiving increasing attention in the literature; that is, the supervisory control problem of nondeterministic discrete event systems. While it is widely understood that partial observation may lead to nondeterminism, the theory of nondeterministic supervisory control has evolved along a completely different avenue than the theory of partially observed systems. Also, while some existence conditions for control of nondeterministic systems have been derived, few explicit algorithms for supervisor synthesis have been obtained. More importantly, the precise connections between the two frameworks have never been examined in detail. In the present paper we investigate the problem of supervisory control of nondeterministic discrete event systems from the point of view of control under partial observation. We carefully examine the relation between the two frameworks and we derive algorithms for supervisor synthesis for nondeterministic systems under a wide variety of conditions. In the present paper we examine nondeterministic systems with both deterministic and nondeterministic specifications in the closed case. The general (nonclosed) case will be discussed elsewhere [4].

1 Introduction

Since the introduction of observability [8], a great deal of work has been done on the control of partially observed discrete event systems: Necessary and sufficient conditions for existence of supervisors, algorithms for supervisor synthesis, for off-line as well as on-line implementation, have been obtained, and a wide variety of have been investigated. In recent years a related problem has also been receiving increasing attention; that is, the supervisory control problem of nondeterministic discrete event systems. In this setting, a system is given either as a nondeterministic automaton (with ϵ-transitions) or as a *trajectory model* [1] [2] (see also [6] [7] [10] [12]). The specification of legal behavior of such systems can be given either in a deterministic or in a nondeterministics setting. The supervisory control problem is to synthesize a supervisor that restricts the nondeterministic plant so as to satisfy the corresponding specification.

While it is widely understood that partial observation may lead to nondeterminism, the theory of nondeterministic supervisory control has evolved along an independent and completely different avenue than the theory of partially observed systems [12] [6]. Also, while some existence conditions for control of nondeterministic systems have been derived, few explicit algorithms for supervisor synthesis have been obtained. More importantly, the precise connections between the two frameworks have never been examined in detail.

It is intuitively clear that if one were given a deterministic model, a deterministic specification of legal behavior, and a partial observation map, one could design a supervisor that is

[1]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.