Language Convergence in Controlled Discrete-Event Systems

Yosef M. Willner and Michael Heymann

Abstract—**Discrete-event systems modeled as state machines in the framework of Ramadge and Wonham are considered. In this paper, convergence of the language generated or marked by the system to a specified legal language is investigated. The convergence property is studied both with respect to open-loop (uncontrolled) systems and with respect to controlled systems. In the latter case, both questions of existence and synthesis of supervisors that force convergence are examined. Algorithms for verification of convergence and for synthesis of stabilizing supervisors are provided. Finally, the concept of asymptotic behavior of systems is defined, and questions related to this concept are investigated.**

I. Introduction

In the main paradigm of supervisory control of discrete-event systems (DES) [1]-[3], the system is modeled as a finite automaton that executes state transitions in response to a stream of events that occur nondeterministically and asynchronously. The supervisory control problem consists of synthesizing a supervisor whose task is to disable events in an orderly fashion, from a specified subset of controlled events, so as to confine the behavior of the supervised system to within a specified legal language. It is generally further required that the synthesized supervisor be minimally restrictive in the sense that, to achieve legal behavior, the smallest number of events is disabled.

There are situations, however, when legal supervision is either impossible or impractical. For example, when a system failure occurs, it may be driven to a state at which it is impossible to prevent the occurrence of illegal event sequences. Furthermore, even when the behavior of the system can be confined to within legal bounds, such a constraint may lead to the design of a supervisor that is overly restrictive. In such situations the question of convergence of the system to the legal behavior is of great interest. Intuitively, we say that the system converges to the legal behavior, if, after a finite and bounded number of transitions, it begins to behave legally. The supervisory control task is then to force such convergence to take place.

In [4] and [5] the problem of stabilization of discrete-event systems has been introduced, and the stabilization, or convergence, concept was presented in terms of legal and illegal states of the system. In [4] legal behavior was specified by a set T of legal states. A system was then defined to be stable if, for any arbitrary initial state, it reaches a state in T after a finite and bounded number of state transitions and thereafter remains in T indefinitely. A system was called stabilizable if there exists a supervisor under which the supervised system is stable. Algorithms for synthesis of stabilizing supervisors were also presented. Optimal stabilizing supervisors were presented in [8].

Manuscript received April 20, 1992; revised May 31, 1994. Recommended by Past Associate Editor, P. J. Ramadge. This work was supported in part by the Technion Fund for Promotion of Research.

Y. Willner is with the Department of Electrical Engineering, Technion—Israel Institute of Technology, Haifa 32000, Israel.

M. Heymann is with the Department of Computer Science, Technion—Israel Institute of Technology, Haifa 32000, Israel.

IEEE Log Number 9408278.

It is quite clear that the legal behavior of a DES cannot always be specified by a set of (legal) states, which constitutes a static specification, but rather by a legal language which constitutes a dynamic specification. In the latter case the system is said to converge to legal behavior if, after a bounded number of illegal state transitions, the system produces only legal behavior, that is, strings of the specification language.

The concept of language convergence was first introduced by Kumar *et al.* (who called it language-stability) in [9], more recently in [16] and [18], and independently by Willner and Heymann in [15]. A similar concept was also studied in a somewhat different setting by Ozveren and Willsky in [19]. In [15] the convergence concept was formulated in terms of w-languages (i.e., languages that consist of infinite strings). In the present paper we formulate the problem in terms of languages over Σ^* and extend the discussion far beyond that of [15]. We provide algorithms for determining the convergence of a regular language to another regular language and for synthesizing supervisors that guarantee that convergence takes place whenever such supervisors exist.

We show that for suffix-closed legal languages, i.e., languages in which every suffix of a string is also a string of the language, the convergence problem can be resolved with algorithms of polynomial complexity. In fact, for suffix-closed languages the complexity is essentially of the same order as the static stabilization problem of [4]. When the specification language E is not suffix closed, the convergence problem is resolved by an algorithm of complexity that is polynomial in the size of the recognizer of the system language but exponential in the size of the recognizer of the specification language.

A further concept introduced in the present paper is the asymptotic behavior of a DES. The asymptotic behavior is the language that the system executes after performing an arbitrarily large number of state transitions. We discuss the problem of synthesizing a supervisor that guarantees the confinement of the asymptotic behavior of the supervised system within a given legal language. In the case of a suffix-closed specification, the inclusion of the asymptotic behavior in the specification language is equivalent to the convergence of the supervised system to the legal language. In the general case, inclusion of the asymptotic behavior in the specification language ensures the convergence of the behavior of the supervised system to the legal language. The inverse property, that language convergence implies confinement of the asymptotic behavior, is not in general true.

The paper is organized as follows. In Section II, a brief review of the basic elements of the framework of [1] is given, and the concept of language convergence is introduced. Algorithms for verifying language convergence for regular languages are presented in Section III. In Section IV we discuss the controlled convergence problem, i.e., the problem of existence and synthesis of supervisors that guarantee convergence. Problems associated with asymptotic behavior are considered in Section V. At various points of the paper our results are compared with the results obtained in [9] and [18].

II. Preliminaries

We adopt the Ramadge–Wonham (RW) [1] discrete-event control formalism, that is, the DES under consideration is modeled as a deterministic automaton

$$P = (Q, \Sigma, \delta, q_0, Q_m) \tag{2.1}$$

where Q is a finite set of states, Σ is a finite set of event symbols, partitioned into two disjoint subsets Σ_c of controlled events and Σ_u of uncontrolled events. The map $\delta : \Sigma \times Q \to Q$ is a partial function called the state transition function, q_0 is the initial state, and $Q_m (\subset Q)$ is a subset of final (or marked) states.

Let Σ^* denote the set of all finite strings over Σ including the empty string ϵ. A subset $L \subset \Sigma^*$ is called a language over Σ.

Letting δ be extended in the standard way (see, e.g., [6]) to a function $\Sigma^* \times Q \to Q$, the language generated by P is defined as

$$\mathcal{L}(P) := \{t \in \Sigma^* | \delta(t, q_0)!\} \tag{2.2}$$

where the notation "!" means "is defined." The language marked by P is defined as

$$\mathcal{L}_m(P) := \{t \in \mathcal{L}(P) | \delta(t, q_0) \in Q_m\}. \tag{2.3}$$

A sequence of states $q_1, \cdots, q_n \in R \subset Q$ is called a path of P contained in R (starting at q_1 and ending at q_n) if there exists a sequence of event symbols $\sigma_1, \cdots, \sigma_{n-1}$ such that $q_{i+1} = \delta(\sigma_i, q_i)$ for each $i = 1, \cdots, n-1$. When $q_n = q_1$ we call the path a cycle (in R). A subset $R \subset Q$ is called acyclic if P has no cycles in R. If P has no cycles in Q we also say that P is acyclic. We shall say that a state q' is accessible from a state q if there is a path of P starting at q and ending at q'. We shall denote by $\mathcal{A}(P, q)$ the set of all states of P that are accessible from q.

For a subset $\hat{Q} \subset Q$, we shall denote by $P_{\hat{Q}}$ the automaton obtained from P by replacing the initial state by the set $\hat{Q}$, that is

$$P_{\hat{Q}} := (Q, \Sigma, \delta, \hat{Q}, Q_m). \tag{2.4}$$

The language generated by $P_{\hat{Q}}$ is then defined as

$$\mathcal{L}(P_{\hat{Q}}) := \{t \in \Sigma^* | \delta(t, \hat{q})! \text{ for some } \hat{q} \in \hat{Q}\}. \tag{2.5}$$

The language marked by $P_{\hat{Q}}$ is then given by (2.3) upon replacing P by $P_{\hat{Q}}$. When $\hat{Q}$ is a singleton $\{q\}$, we shall write P_q instead of $P_{\{q\}}$ and $P_{q_0} = P$.

If $P_1 = (Q_1, \Sigma, \delta_1, q_{10}, Q_{1m})$ and $P_2 = (Q_2, \Sigma, \delta_2, q_{20}, Q_{2m})$ are two automata over the event set Σ, we say that P_2 is a subautomaton of P_1, denoted $P_2 \subset P_1$, if

a) $Q_2 \subset Q_1, \quad Q_{m2} \subset Q_{m1}, \quad q_{10} = q_{20}$

b) $\delta_2(\sigma, q) = \delta_1(\sigma, q) \; \forall \sigma \in \Sigma, q \in Q_2$ s.t. $\delta_2(\sigma, q)!$.

Thus, a subautomaton is obtained from a given automaton by deleting from it transitions and/or states along with all the transitions incident on the deleted states. For a subset $\hat{Q} \subset Q$ we define the restriction of P to $Q - \hat{Q}$, denoted $P|\hat{Q}$, as the subautomaton of P obtained by deleting from it all the states of $\hat{Q}$ and the transitions incident on these states.

For given deterministic automata $P = (Q, \Sigma, \delta, q_0, Q_m)$ and $A = (X, \Sigma, \xi, x_0, X_m)$, the strict synchronous composition $P||A$ of P and A is defined as

$$P||A = (Q \times X, \Sigma, \alpha, (q_0, x_0), Q_m \times X_m)$$

where $\alpha : \Sigma \times Q \times X \to Q \times X$ is defined by

$$\alpha(\sigma, (q, x)) := \begin{cases} (\delta(\sigma, q), \xi(\sigma, x)) & \text{if } \delta(\sigma, q)! \text{ and } \xi(\sigma, x)! \\ \text{undefined} & \text{otherwise.} \end{cases}$$

It is well known that $\mathcal{L}(P||A) = \mathcal{L}(P) \cap \mathcal{L}(A)$ and $\mathcal{L}_m(P||A) = \mathcal{L}_m(P) \cap \mathcal{L}_m(A)$.

A supervisor for a DES P is a map$S : \mathcal{L}(P) \to 2^{\Sigma_c}$ such that for each $t \in \mathcal{L}(P), S(t) (\subset \Sigma_c)$ is the set of controlled events that must be disabled next. The concurrent operation of the system P and the supervisor S, denoted by S/P, is called the closed-loop system. That is, a transition in S/P can take place whenever it can take place in P and is not disabled by S. The language $\mathcal{L}(S/P)$ generated by the closed-loop system is thus given recursively by

$$\epsilon \in \mathcal{L}(S/P)$$
$$(\forall s \in \mathcal{L}(S/P)) s\sigma \in \mathcal{L}(S/P) \Leftrightarrow s\sigma \in \mathcal{L}(P) \wedge \sigma \notin S(s).$$

The language marked by S/P will be defined by

$$\mathcal{L}_m(S/P) = \mathcal{L}(S/P) \cap \mathcal{L}_m(P) \tag{2.6}$$

and consists of all the strings marked by P that are not disabled by S.

We shall assume that the system P under consideration is trim, i.e., every state $q \in Q$ is accessible from q_0 and $\mathcal{L}(P) = \overline{\mathcal{L}_m(P)}$. A supervisor S is then called nonblocking if $\mathcal{L}(S/P) = \overline{\mathcal{L}_m(S/P)}$.

For strings $s, t \in \Sigma^*$ we let $s \cdot t$ denote their concatenation. A string s is a prefix of t, denoted $s \leq t$, if $t = sw$ for some $w \in \Sigma^*$. If $w \neq \epsilon$, s is said to be a proper prefix of t, denoted $s < t$. The prefix closure $\bar{L}$ of a language $L \subset \Sigma^*$ is the set of all prefixes of strings in L and L is prefix closed if $\bar{L} = L$. The length of a string t is denoted $|t|$ and the prefix of t of length i is denoted $pr_i(t)$ $(pr_0(t) = \epsilon)$.

For a string $t \in \Sigma^*$ we denote by $\mathrm{suf}_i(t)$ the string obtained by deleting from t its first i symbols ($\mathrm{suf}_0(t) = t$ and $\mathrm{suf}_{|t|}(t) = \epsilon$). For a language $L \subset \Sigma^*$, the suffix closure of L is the subset of all suffixes of strings in L, that is

$$\mathrm{suf}(L) = \{\mathrm{suf}_i(t) \mid \forall t \in L,\ \forall i \le |t|\}. \tag{2.7}$$

A language L is suffix closed if $\mathrm{suf}(L) = L$. The following properties of the suffix operator are readily verified:

- The suffix operator is monotone, i.e.,

$$L_1 \subset L_2 \Rightarrow \mathrm{suf}\ (L_1) \subset \mathrm{suf}\ (L_2). \tag{2.8}$$

- For a family of languages $\{L_\alpha \subset \Sigma^*\}$

$$\mathrm{suf}(\bigcup_\alpha L_\alpha) = \bigcup_\alpha \mathrm{suf}\ (L_\alpha), \tag{2.9}$$

$$\mathrm{suf}(\bigcup_\alpha L_\alpha) \subset \bigcup_\alpha \mathrm{suf}\ (L_\alpha) \tag{2.10}$$

with equality in (2.10) if for each α, L_α is suffix closed.

Let $L \subset \Sigma^*$ be a regular language and let $P = (Q, \Sigma, \delta, q_0, Q_m)$ be a (trim) recognizer for L, i.e., $\mathcal{L}_m(P) = L$ and $\mathcal{L}(P) = \overline{\mathcal{L}_m(P)}$. A recognizer for $\mathrm{suf}(L)$ is then given by $P_Q = (Q, \Sigma, \delta, Q, Q_m)$, that is, $\mathcal{L}_m(P_Q) = \mathrm{suf}\ (L)$ (with $\mathcal{L}_m(P_Q)$ as defined following (2.5)). From (2.7), $L \subset \mathrm{suf}(L)$, so that L is suffix closed if and only if $\mathcal{L}_m(P_Q) \subset \mathcal{L}_m(P)$. This inclusion is equivalent to the condition that $\mathcal{L}_m(P_Q) \cap (\Sigma^* - \mathcal{L}_m(P)) = \phi$, which can be verified by an algorithm with complexity of $O(|Q|^2)$ (see e.g., [10, Proposition 2.3]).

We conclude this section with the definition of language convergence that will be needed in the sequel. Let $L, M \subset \Sigma^*$ be two languages such that $M \neq \phi$.

Definition 2.1: The language L is said to converge asymptotically to M, denoted $M \leftarrow L$, if for each $t \in L$ there exists an integer $i \ge 0$ such that $\mathrm{suf}_i(t) \in M$.

Remark 2.1: It is worth noting that the asymptotic convergence $M \leftarrow L$ is equivalent to $L \subset \Sigma^* \cdot M$ (where $\Sigma^* \cdot M$ denotes the language concatenation, i.e., $s \in \Sigma^* \cdot M$ if and only if $s = r \cdot t$ where $r \in \Sigma^*$ and $t \in M$). Thus, the asymptotic convergence problem can be investigated as a special case of the language confinement problem that was investigated in detail by RW in [1].

Definition 2.2: The language L is said to converge finitely (or, simply, to converge) to M, denoted $M \Leftarrow L$, if there exists a finite integer $k \ge 0$ such that for each $t \in L$ there exists i, $i \le k$, for which $\mathrm{suf}_i(t) \in M$. In that case the least k for which the above holds is called the convergence time of L to M and is denoted $e_M(L)$.

While finite convergence obviously implies asymptotic convergence, the converse is not generally true as illustrated by the simple example where $L = \alpha^*\beta$ and $M = \{\beta\}$.

It is easily verified that the class of languages that converge asymptotically to a given language is closed under arbitrary intersections and arbitrary unions while the class of languages that converge finitely to a given language is closed under arbitrary intersections but only under finite unions. Finally, if $L_1, L_2, M \subset \Sigma^*$ are languages such that $L_1 \subset L_2$ and $M \neq \phi$, then

$$(M \leftarrow L_2) \Rightarrow (M \leftarrow L_1) \tag{2.11}$$

and

$$(M \Leftarrow L_2) \Rightarrow (M \Leftarrow L_1). \tag{2.12}$$

III. Finite Language Convergence

In the present section we shall develop necessary and sufficient conditions for finite language convergence. Furthermore, we develop algorithms for testing whether $M \Leftarrow L$ for regular languages $L, M \subset \Sigma^*, M \neq \phi$. For a variety of interesting special cases, algorithms of polynomial time complexity are constructed. In the most general case, however, our algorithm is of polynomial complexity in the size of the recognizer for L but exponential in the size of the recognizer for M.

Let $L, M \subset \Sigma^*, M \neq \phi$ be two regular languages, let $P = (Q, \Sigma, \delta, q_0, Q_m)$ and $A = (X, \Sigma, \xi, x_0, X_m)$ be deterministic trim recognizers for L and M, respectively, so that $\mathcal{L}_m(P) = L, \mathcal{L}_m(A) = M, \overline{\mathcal{L}_m(P)} = \mathcal{L}(P)$ and $\overline{\mathcal{L}_m(A)} = \mathcal{L}(A)$. Let T_s be the subset of states in Q given by

$$T_s = \{q \in Q | \mathcal{L}_m(P_q) \subset \ \mathrm{suf}\ (M)\} \tag{3.1}$$

(where, as defined earlier, P_q is the automaton P initialized at q).

The following is a necessary condition for (finite-) convergence of L to M.

Proposition 3.1: Let $L, M \subset \Sigma^*$ be the languages recognized by the automata P and A, respectively. Then $M \Leftarrow L$ only if the set $Q - T_s$ is acyclic (in P).

Proof: Suppose that P has a cycle C in $Q - T_s$. Since P is trim, every state of P is accessible from q_0 and there is a sequence of strings $\{t_i\}$, $t_i \in \mathcal{L}(P) = \overline{L}$, $t_1 < t_2 < t_3 \cdots$ such that for each i, $\delta(t_i, q_0) \in C$. Assume that $M \Leftarrow L$ with convergence time $k (= e_M(L))$. Let t_j be a string of the above sequence such that $n := |t_j| > k$. Then $q = \delta(t_j, q_0) \in C$ and there exists $s \in \mathcal{L}_m(P_q)$ such that $s \notin \ \mathrm{suf}\ (M)$ (since the states of C do not belong to T_s). But $t_j \cdot s \in \mathcal{L}_m(P) = L$ and by the convergence of L to M, there exists $l \le k$ such that $\mathrm{suf}_l(t_j \cdot s) \in M$. Since $n > l$, it follows that $s = \mathrm{suf}_n(t_j \cdot s) = \mathrm{suf}_{n-l}\ \mathrm{suf}_l(t_j \cdot s) \in \mathrm{suf}(M)$. This is a contradiction, and the proof is complete. □

Next, we present a sufficient condition for convergence of$L$ to M. Let T be the set of all states in Q defined by

$$T := \{q \in Q | \mathcal{L}_m(P_q) \subset M\} \tag{3.2}$$

and assume that $q_0 \notin T$, for otherwise $L \subset M$ and convergence is trivial. Next, let R be defined as the set of all states of $Q - T$ that are accessible from q_0 through a path that does not intersect T, that is

$$\begin{aligned} R := \{q \in Q | \exists t \in \overline{L}, \delta(t, q_0) = q \\ \text{and } (\forall i \le |t|) \delta(pr_i(t), q_0) \in Q - T\}. \end{aligned} \tag{3.3}$$

The set R can conveniently be computed as follows. Consider the automaton $P|T$ (obtained from P by deleting from it all the states of T and the transitions incident on these states).

Then R is the set of all states accessible in $P|T$ from the initial state q_0, that is

$$R = \mathcal{A}(P|T, q_0) \tag{3.4}$$

(of course, if $q_0 \in T$ then $L \subset M$ and $R = \phi$).

Proposition 3.2: If R is acyclic and $R \cap Q_m = \phi$, then $M \Leftarrow L$.

Proof: Assume first that $T = \phi$. Then (since P is trim) $R = Q$ so that $R \cap Q_m = Q_m$. If $R \cap Q_m = \phi$, then $Q_m = \phi$ and hence $L = \mathcal{L}_m(P) = \phi$ and $M \Leftarrow L$ holds trivially. Assume now that $T \neq \phi$. If $R \cap Q_m = \phi$, then for each $t \in L = \mathcal{L}_m(P)$ there must exist $i, i \leq |t|$, such that $q = \delta(pr_i(t), q_0) \in T$. Now the acyclicity of R implies that this i must satisfy the condition that $i \leq |R|$. Hence we conclude that for each $t \in L$ there exists $i \leq |R|$ such that $q = \delta(pr_i(t), q_0) \in T$, implying that $\text{suf}_i(t) \in \mathcal{L}_m(P_q) \subset M$. It follows that $M \Leftarrow L$ and $e_M(L) \leq |R|$. □

Proposition 3.2 suggests the following algorithm for determining the convergence of L to M.

Algorithm 3.1:

1) Compute the set T of states defined by (3.2). If $q_0 \in T$ then $L \subset M$. Convergence holds trivially. Halt.
2) Compute the set R as defined by (3.4).
3) Check (e.g., by the algorithm of [7, p. 64]) whether P has cycles in R. If the answer is yes, the algorithm is inconclusive. Halt.
4) If $R \cap Q_m = \phi$, then $M \Leftarrow L$. Halt.

The computation of R (which is a standard accessibility computation) and the algorithm for testing acyclicity of R as given in [7] are both of complexity[1] $O(|Q|)$. As regards the computation of T we proceed as follows.

A state $q \in Q$ is in T if and only if $\mathcal{L}_m(P_q) \subset \mathcal{L}_m(A)$ where $A = (X, \Sigma, \xi, x_0, X_m)$ is a deterministic recognizer for $M(\mathcal{L}_m(A) = M)$. Equivalently, $q \in T$ if and only if $\mathcal{L}_m(P_q) \cap (\Sigma^* - \mathcal{L}_m(A)) = \phi$. A deterministic recognizer for $\Sigma^* - \mathcal{L}_m(A)$ is given by the automaton $\hat{A} = (\hat{X}, \Sigma, \hat{\xi}, x_0, \hat{X}_m)$, where $\hat{X} = X \cup \{d\}$, d being an additional "dump" state, $\hat{X}_m = \{d\} \cup (X - X_m)$, and $\hat{\xi} : \Sigma \times \hat{X} \to \hat{X}$ is given by

$$\hat{\xi}(\sigma, \hat{x}) = \begin{cases} \xi(\sigma, \hat{x}) \text{ if } \hat{x} \neq d & \text{and } \xi(\sigma, \hat{x})! \\ d & \text{otherwise.} \end{cases}$$

The construction of the automaton $\hat{A}$ is of complexity $O(|X|)$.

For a state $q \in Q$, a recognizer for $\mathcal{L}_m(P_q) \cap (\Sigma^* - \mathcal{L}_m(A))$ is given by the automaton $P_q||\hat{A}$ (see Section II) and we have that $\mathcal{L}_m(P_q) \subset \mathcal{L}_m(A)$, or equivalently $q \in T$, if and only if $\mathcal{L}_m(P_q||\hat{A}) = \phi$. Thus, to compute T, we construct the automaton

$$P_Q||\hat{A} = (Q \times \hat{X}, \Sigma, \alpha, \{q, x_0)|q \in Q\}, Q_m \times \hat{X}_m)$$

where $\alpha(\sigma, (q, \hat{x})) = (\delta(\sigma, q), \hat{\xi}(\sigma, \hat{x}))$ is defined if and only if $\delta(\sigma, q)!$ and $\hat{\xi}(\sigma, \hat{x})!$. Next, we let $C(Q_m \times \hat{X}_m)$ be the set of all states $(q, \hat{x}) \in Q \times \hat{X}$ from which the set $Q_m \times \hat{X}_m$ is accessible in $P_Q||\hat{A}$, that is

$$C(Q_m \times \hat{X}_m) := \{(q, \hat{x}) \in Q \times \hat{X} | \exists t \in \Sigma^*, \alpha(t, (q, \hat{x})) \in Q_m \times \hat{X}_m\}.$$

A state q is thus in T if and only if $(q, x_0) \notin C(Q_m \times \hat{X}_m)$, whence T is given by

$$T = Q - \{q \in Q | (q, x_0) \in C(Q_m \times \hat{X}_m)\}.$$

The computation of $C(Q_m \times \hat{X}_m)$ is of complexity $O(|Q||X|)$ and hence the overall complexity of Algorithm 3.1 is $O(|Q||X|)$.

An interesting case of the convergence problem is when the language M includes the empty string ϵ. This implies that all deadlocking behavior of L is regarded as convergent. Specifically, since for each string $t \in L$, $\text{suf}_{|t|}(t) = \epsilon \in M$, it follows that if all strings of L are of bounded length, then L converges to any language M satisfying $\epsilon \in M$. A further observation is the following. When $\epsilon \in M$, then the acyclicity of R (as defined in (3.3) or (3.4)) implies convergence of L to M. Indeed, for $t \in L$ such that $|t| \leq |R|$, $\text{suf}_{|t|}(t) = \epsilon \in M$, and for $t \in L$ such that $|t| > |R|$, the acyclicity of R implies that there exists $i \leq |R|$ such that $\text{suf}_i(t) \in M$. Thus we have the following.

Proposition 3.3: Let $L, M \subset \Sigma^*$ be regular languages recognized by automata P and A, respectively, and let R be the set defined by (3.4). If $\epsilon \in M$ and R is acyclic, then $M \Leftarrow L$.

A special case of languages M that include the empty string are suffix-closed languages. Such language specifications correspond to cases when one is interested in the eventual correctness of the logical behavior of the process P but does not insist on specific initialization. Thus, one is satisfied with the fact that the behavior of P "merges" with that of a given specification language rather than performing complete strings of the specification and one wishes to find conditions for $\text{suf}(M) \Leftarrow \mathcal{L}_m(P) = L$.

Let M be suffix closed. Then $\epsilon \in M$, and it can be further noted that the sets T_s as defined in (3.1) and T as defined in (3.2) coincide. Proposition 3.1 then states that if $M \Leftarrow L$ then $Q - T$ is acyclic, and since $R \subset Q - T$, so is also R. Combining this fact with Proposition 3.3 gives the following interesting theorem.

Theorem 3.1: Let $L, M \subset \Sigma^*$ be (regular) languages recognized by automata P and A, respectively. Assume that M is suffix closed. Then $M \Leftarrow L$ if and only if $Q - T$ is acyclic.

Theorem 3.1 provides the theoretical foundation for the following algorithm that determines the convergence of L to M in case L and M are regular languages and M is suffix closed.

Algorithm 3.2:

1) Compute the set T of states defined by (3.2). If $q_0 \in T$ then $L \subset M$. Convergence holds trivially. Halt.
2) Check (e.g., by the algorithm of [7, p. 64]) whether P has cycles in $Q - T$.
3) L converges to M if and only if the answer to 2) is negative. Halt.

[1] In the present paper we shall assume $|\Sigma| = O(1)$ and we give the complexity bounds only in terms of the number of states.

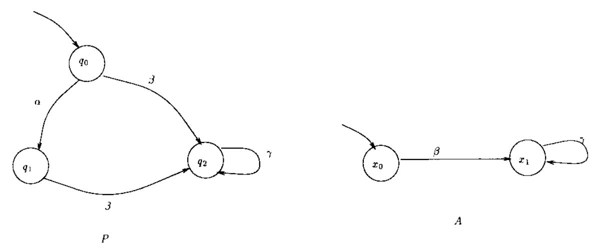


Fig. 1.

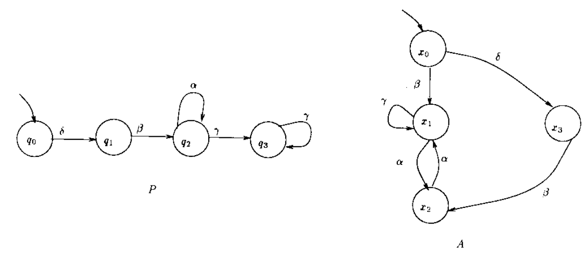


Fig. 2.

The complexity of Algorithm 3.2 is $O(|Q||X|)$ (see the analysis of complexity of Algorithm 3.1 for details).

We turn next to the problem of language convergence in the case when M is a general (not necessarily suffix-closed) regular language. The conditions of Proposition 3.2, and in the case when $\epsilon \in M$ of Proposition 3.3, are sufficient for convergence and, in view of their relative efficiency, should be tested first. When the corresponding algorithms (Algorithms 3.1 and 3.2, respectively) are inconclusive, one must resort to a more elaborate examination. As we shall see later, the algorithms in the general case are no longer of polynomial complexity.

Let us first examine two examples that will prove helpful in obtaining some intuitive insight. The first example shows that while the acyclicity of R is sufficient for convergence of L to M in case $\epsilon \in M$ (Proposition 3.3), it is not necessary when M is not suffix closed.

Example 3.1: Consider the automata P and A as given in Fig. 1, where all the states of both P and A are final states.

Thus, $\mathcal{L}_m(P) = \mathcal{L}(P)$ and$\mathcal{L}_m(A) = \mathcal{L}(A)$ and, obviously, $\mathcal{L}(A) \Leftarrow \mathcal{L}(P)$. (Here and in subsequent figures the entrance arrows indicate the initial states.) Observe that $T = \{q_1\}$, i.e., q_1 is the only state $q \in Q$ such that $\mathcal{L}(P_q) \subset \mathcal{L}(A)$ and the set $R = \{q_0, q_2\}$ is not acyclic. □

Example 3.2: Here P and A are given by the automata of Fig. 2 and again all states (of P and A) are final states.

To verify that $\mathcal{L}(A) \Leftarrow \mathcal{L}(P)$ note that

$$\mathcal{L}(P) = \overline{\delta\beta\alpha^*\gamma^*} = \overline{\delta\beta(\alpha\alpha)^*\gamma^*} + \overline{\delta\beta\alpha(\alpha\alpha)^*\gamma^*}$$

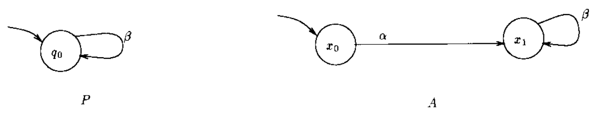


Fig. 3.

and that both $\overline{\beta(\alpha\alpha)^*\gamma^*}$ and $\overline{\delta\beta\alpha(\alpha\alpha)^*\gamma^*}$ are sublanguages of $\mathcal{L}(A)$.

Next note that

$$\mathcal{L}(P_{q_2}) = \alpha^*\gamma^* = (\alpha\alpha)^*\gamma^* + \alpha(\alpha\alpha)^*\gamma^*$$

and that $(\alpha\alpha)^*\gamma^* \subset \mathcal{L}(A_{x_1})$ and $\alpha(\alpha\alpha)^*\gamma^* \subset \mathcal{L}(A_{x_2})$.

Thus

$$\mathcal{L}(P_{q_2}) \subset \mathcal{L}(A_{x_1}) \cup \mathcal{L}(A_{x_2}) = \mathcal{L}(A_{\{x_1,x_2\}}).$$

Note further that if $\delta(t, q_0) = q_2$ for $t \in \Sigma^*$, then $\xi(t, x_0) \in \{x_1, x_2\}$. □

In Example 3.2 the language $L = \mathcal{L}(P)$ (that converges to $M = \mathcal{L}(A)$) satisfies the condition that the system P, starting at q_0, reaches a state q such that

$$\mathcal{L}(P_q) \subset \mathcal{L}(A_\chi) \tag{3.5}$$

for some subset $\chi \subset X$. We shall see below that this is actually a necessary condition for convergence of L to M. Condition (3.5) is of course not sufficient for convergence as can be seen from the example in Fig. 3.

Here $\mathcal{L}(P)$ does not converge to$\mathcal{L}(A)$ although $\mathcal{L}(P_{q_0}) \subset \mathcal{L}(A_{x_1})$.

Next we present an algorithm that tests the convergence of L to M for general regular languages L and M. The algorithm is based on the construction of a new recognizer for the language L, in which the necessary condition that the automaton reaches a state that satisfies condition (3.5) is also sufficient. The complexity of the proposed algorithm is exponential in $|X|$, the dimension of the state set of A, but it is polynomial in $|Q|$, the dimension of the state set of P.

Theorem 3.2: Let $P = (Q, \Sigma, \delta, q_0, Q_m)$ and $A = (X, \Sigma, \xi, x_0, X_m)$ be given automata. The following algorithm verifies the convergence $\mathcal{L}_m(A) \Leftarrow \mathcal{L}_m(P)$.

Algorithm 3.3:

1) Construct a deterministic automaton $B = (V, \Sigma, \alpha, v_0, V_m)$, where $V = Q \times 2^X$, $v_0 = (q_0, \{x_0\})$, $V_m = Q_m \times X$, and where $\alpha : \Sigma \times V \to V$ is defined as follows. Let $\chi \in 2^X$ be any element. Then see (3.6) at the bottom of the page. We assume that B is trim. Otherwise construct the maximal trim subautomaton of B.
2) Test whether there exists in B a state $v = (q, \chi) \in V$ such that $q \in Q_m$ and $\chi \cap X_m = \phi$. If such a state exists, $\mathcal{L}_m(A) \not\Leftarrow \mathcal{L}_m(P)$. Halt.
3) Compute the set $\tilde{T} \subset V$ defined by

$$\tilde{T} = \{v = (q, \chi) \in V | \mathcal{L}_m(P_q) \subset \mathcal{L}_m(A_\chi)\}. \tag{3.7}$$

$$\alpha(\sigma, (q, \chi)) = \begin{cases} (\delta(\sigma, q), \chi') & \text{where } \chi' = \{x_0\} \cup \{x' \in X | (\exists x \in \chi)\xi(\sigma, x) = x'\} \quad \text{if } \delta(\sigma, q)! \\ \text{undefined} & \text{otherwise} \end{cases} \tag{3.6}$$

4) Test whether $V - \tilde{T}$ is acyclic (in B) (use, e.g., the algorithm of [7]).
5) If the answer to 4) is affirmative, then $\mathcal{L}_m(A) \Leftarrow \mathcal{L}_m(P)$. Otherwise not. Halt.

Proof: First observe from the definition of$B$ that a string $t \in \Sigma^*$ is in $\mathcal{L}(B)$ if and only if $\alpha(t, (q_0, \{x_0\})) = (q, \chi)$, where

$$q = \delta(t, q_0) \tag{3.8}$$

and

$$\chi = \{x \in X | (\exists i \leq |t|)\ x = \xi(\ \mathrm{suf}_i(t), x_0)\}. \tag{3.9}$$

Indeed, (3.8) holds since for each state $(q, \chi') \in V$ the transition $\alpha(\sigma, (q, \chi'))!$ if and only if $\delta(\sigma, q)!$. To see that also (3.9) holds, we proceed by induction on the length of strings. For $t = \epsilon$, (3.9) clearly holds with $\chi = \{x_0\}$. Suppose (3.9) holds for an arbitrary string t, so that

$$\chi_t = \{x \in X | (\exists i \leq |t|)\ x = \xi(\ \mathrm{suf}_i(t), x_0)\}$$

and consider the string $t\sigma$ for an arbitrary $\sigma \in \Sigma$. Employing (3.6) and substituting the above expression for χ_t yields

$$\begin{aligned}\chi = \chi_{t\sigma} &= \{x_0\} \cup \{x' \in X \mid (\exists x \in \chi_t)\xi(\sigma, x) = x'\}\\ &= \{x_0\} \cup \{x' \in X \mid (\exists i \leq |t|)\xi(\sigma, \xi(\ \mathrm{suf}_i(t), x_0)) = x'\}\\ &= \{x_0\} \cup \{x' \in X \mid (\exists i \leq |t\sigma|)\xi(\ \mathrm{suf}_i(t\sigma), x_0) = x'\}\\ &= \{x' \in X \mid (\exists i \leq |t\sigma|)\xi(suf_i(t\sigma), x_0) = x'\}.\end{aligned}$$

An immediate consequence of (3.8) and (3.9) is that

$$\mathcal{L}(B) = \mathcal{L}(P) \tag{3.10}$$

and since $V_m = Q_m \times X$

$$\mathcal{L}_m(B) = \mathcal{L}_m(P). \tag{3.11}$$

Suppose now that there exists a state $v = (q, x) \in V$ such that $q \in Q_m$ and $\chi \cap X_m = \phi$. Let $t \in \Sigma^*$ be any string satisfying $\alpha(t, v_0) = v$. By (3.8) it follows that $q = \delta(t, q_0) \in Q_m$, i.e., $t \in \mathcal{L}_m(P)$. By (3.9) it follows that there is no $i \leq |t|$ such that $\xi(\mathrm{suf}_i(t), x_0) \in X_m$. Thus, t has no suffix that belongs to $\mathcal{L}_m(A)$. It follows that $\mathcal{L}_m(P)$ does not converge to $\mathcal{L}_m(A)$. This proves the correctness of step 2).

Suppose next that every state $v = (q, \chi) \in V$ for which $q \in Q_m$ satisfies $\chi \cap X_m \neq \phi$. Then, in view of (3.8) and (3.9), every string $t \in \mathcal{L}_m(P)$ has a suffix that belongs to $\mathcal{L}_m(A)$. Clearly, if $|t| \leq |V - \tilde{T}|$, then the above implies that there exists $i \leq |V - \tilde{T}|$ such that $\mathrm{suf}_i(t) \in \mathcal{L}_m(A)$. We shall show next that when $|V - \tilde{T}|$ is acyclic, there exists $i \leq |V - \tilde{T}|$ such that $\mathrm{suf}_i(t) \in \mathcal{L}_m(A)$ also if $|t| > |V - \tilde{T}|$. If $\tilde{T} = \phi$ this is obvious because then V is acyclic and $|t| \leq |V| = |V - \tilde{T}|$ for all $t \in \mathcal{L}(B)$. When $\tilde{T} \neq \phi$, the acyclicity of $|V - \tilde{T}|$ implies that if $|t| > |V - \tilde{T}|$ then there exists $i < |V - \tilde{T}|$ such that $\alpha(pr_i(t), v_0) = v = (q, \chi) \in \tilde{T}$. Then, in view of (3.7)

$$\mathrm{suf}_i(t) \in \mathcal{L}_m(P_q) \subset \mathcal{L}_m(A_\chi)$$

or, alternately

$$\mathrm{suf}_i(t) \in \mathcal{L}_m(A_x) \tag{3.12}$$

for some $x \in \chi$. By (3.9) it then follows that there exists $j < i$ such that

$$x = \xi(\mathrm{suf}_j\ (pr_i(t)), x_0). \tag{3.13}$$

Upon combining (3.12) and (3.13), we obtain that

$$\begin{aligned}&\xi(\mathrm{suf}_i(t), x) =\\ &\xi(\mathrm{suf}_i(t), \xi(\mathrm{suf}_j(pr_i(t)), x_0) =\\ &\xi(\mathrm{suf}_j(pr_i(t)) \cdot\ \mathrm{suf}_i(t), x_0) =\\ &\xi(\mathrm{suf}_j(t), x_0)\end{aligned}$$

which implies that

$$\mathrm{suf}_j(t) \in \mathcal{L}_m(A). \tag{3.14}$$

Thus, the convergence time of $\mathcal{L}_m(P)$ to$\mathcal{L}_m(A)$ is bounded by $|V - \tilde{T}|$ and hence $\mathcal{L}_m(A) \Leftarrow \mathcal{L}_m(P)$.

To see that the acyclicity of $(V - \tilde{T})$ is necessary for convergence of $\mathcal{L}_m(P)$ to $\mathcal{L}_m(A)$, suppose that there exists a cycle C in $(V - \tilde{T})$. Since B is trim, every state of B is accessible from v_0. Thus there exists a sequence $\{t_i\}$ of strings, $t_1 < t_2 < t_3 \cdots$ such that for each $i, \alpha(t_i, v_0) = v_i = (q_i, \chi_i) \in C$. Suppose now that $\mathcal{L}_m(A) \Leftarrow \mathcal{L}_m(P)$ and let k be the convergence time. Choose a string t_l of the above sequence such that $n := |t_l| > k$. Since no state of C belongs to $\tilde{T}$, there exists $s \in \mathcal{L}_m(P_{q_l})$ such that $s \notin \mathcal{L}_m(A_x)$ for every $x \in \chi_l$. Now, $t_l s \in \mathcal{L}_m(P)$ and by the convergence assumption of $\mathcal{L}_m(P)$ to $\mathcal{L}_m(A)$, there exists $j \leq k$ such that

$$\mathrm{suf}_j(t_l s) =\ \mathrm{suf}_j(t_l) \cdot s \in \mathcal{L}_m(A).$$

Letting $x := \xi(\mathrm{suf}_j(t_l), x_0)$, it follows that $s \in \mathcal{L}_m(A_x)$. But by (3.9) $x \in \chi_l$, a contradiction. □

The complexity of steps 1), 2), and 4) of Algorithm 3.3 is $O(|Q|2^{|X|})$. The test whether $\mathcal{L}_m(P_q) \subset \mathcal{L}_m(A_\chi)$ can be performed by constructing a deterministic recognizer A^χ for $\mathcal{L}_m(A_\chi)$ (an algorithm of complexity $O(2^{|X|})$) and checking whether $\mathcal{L}_m(P_q) \subset \mathcal{L}_m(A^\chi)$ which can be done by a computation of complexity $O(|Q|2^{|X|})$. Thus, the complexity of step 3) as well as that of the complete algorithm is $O((|Q|2^{|X|})^2)$.

Remark 3.1: An alternate approach to verifying the convergence of $\mathcal{L}_m(P)$ to $\mathcal{L}_m(A)$ is based on the reversal of the automata P and A so as to obtain the reflections $\mathcal{L}_m(P)^R$ and $\mathcal{L}_m(A)^R$ of the languages $\mathcal{L}_m(P)$ and $\mathcal{L}_m(A)$. Then $\mathcal{L}_m(A) \Leftarrow \mathcal{L}_m(P)$ if and only if there exists a bounded language L (i.e., a language all of whose strings are of bounded length) such that $\mathcal{L}_m(P)^R \subset \mathcal{L}_m(A)^R \cdot L$. (Here $\mathcal{L}_m(A)^R \cdot L$ denotes the concatenation of the languages $\mathcal{L}_m(A)^R$ and L.) The reversed automata P^R and A^R are nondeterministic in general, and the algorithm for testing the convergence involves computing deterministic equivalents to P^R and A^R. Such an algorithm, whose complexity is $O((2^{|X|+|Q|})^2)$, has recently been described in [9] and [18]. Algorithm 3.3 above, whose complexity is $O((|Q| \cdot 2^{|X|})^2)$, is superior to the algorithm in [9] and [18] since it is only of polynomial complexity in the size of Q (although it is also of second-degree exponential complexity in the size of X). Thus, Algorithm 3.3 can be regarded as relatively efficient or at least tractable when the specification can be described by a small automaton as opposed to the algorithm of [9] and [18], which is generally impractical.

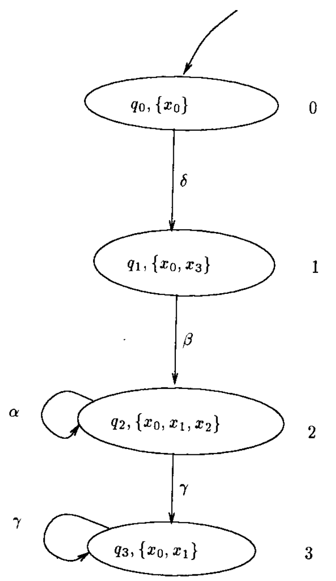


Fig. 4.

Next we illustrate Algorithm 3.3 by employing it to verify the convergence problem of Example 3.2.

Example 3.3 (Verification of Convergence Problem of Example 3.2 —Algorithm 3.3):

Step 1: The automaton B is given in Fig. 4.

To illustrate property (3.9) let us examine state 2). Every string $t \in \Sigma^*$ that satisfies $\alpha(t, v_0) = (q_2, \{x_0, x_1, x_2\})$ is of the form $\delta\beta\alpha^i$ for some $i \geq 0$. If i is even, then the suffixes $\epsilon, \beta\alpha^i, \delta\beta\alpha^i$ satisfy the property $\xi(\epsilon, x_0) = x_0, \xi(\beta\alpha^i, x_0) = x_1$, and $\xi(\delta\beta\alpha^i, x_0) = x_2$. If i is odd, then these suffixes satisfy $\xi(\epsilon, x_0) = x_0, \xi(\beta\alpha^i, x_0) = x_2$, and $\xi(\delta\beta\alpha^i, x_0) = x_1$.

Step 2: Since in the automaton A of Fig. 2 $X_m = X$, we have $\{x\} \cap X_m = \{x\} \cap X = \{x\}$ for each state $x \in X$. Hence $\chi \cap X_m \neq \phi$ for each $(q, \chi) \in V$.

Step 3: It is easy to verify that

$$\tilde{T} = \{1, 2, 3\} \neq \phi.$$

Step 4: There does not exist a cycle (of$B$) in $V - \tilde{T} = \{0\}$.

Step 5: $\mathcal{L}_m(A) \Leftarrow \mathcal{L}_m(P)$. □

IV. Language Convergence in Controlled DES

In the previous section we examined the problem of verification whether a language L (finitely) converges to a given language M. In the present section we shall deal with the problem of convergence of the language generated by a DES $P = (Q, \Sigma, \delta, q_0, Q_m)$ to a given specification of logical behavior given by a language $E \subset \Sigma^*$. Specifically, we shall examine the problem of existence of a nonblocking supervisor S such that $E \Leftarrow \mathcal{L}_m(S/P)$. Thus we formally define the following.

Controlled Convergence Problem (CCP): Synthesize a nonblocking supervisor S for P such that

$$E \Leftarrow \mathcal{L}_m(S/P). \quad (4.1)$$

We recall from [1] that a language $M, \phi \neq M \subset \mathcal{L}_m(P)$ can be realized by a nonblocking supervisor S, i.e., $\mathcal{L}_m(S/P) = M$, if and only if M satisfies the following two conditions:

- M is $\mathcal{L}_m(P)$-closed, i.e.,

$$M = \bar{M} \cap \mathcal{L}_m(P). \quad (4.2)$$

- M is controllable with respect to $L(P)$, i.e.,

$$\bar{M}\Sigma_u \cap \mathcal{L}(P) \subset \bar{M}. \quad (4.3)$$

We further recall that, for a language N, the class $CL(\mathcal{L}_m(P) \cap N)$ of controllable and $\mathcal{L}_m(P)$-closed sublanguages of $\mathcal{L}_m(P) \cap N$ has a unique supremal element $\hat{N}_p$. This implies that a nonblocking supervisor S exists such that $\mathcal{L}_m(S/P) \subset N$ if and only if $\hat{N}_p \neq \phi$. In view of the above observations we have the following immediate result.

Proposition 4.1: CCP is solvable if and only if there exists a controllable and $\mathcal{L}_m(P)$-closed sublanguage $\phi \neq N \subset \mathcal{L}_m(P)$ such that $E \Leftarrow N$.

A special case of interest occurs when the process $P = (Q, \Sigma, \delta, q_0, Q_m)$ satisfies $Q_m = Q$, in which case $\mathcal{L}_m(P) = \mathcal{L}(P)$ and $\mathcal{L}_m(S/P) = \mathcal{L}(S/P)$. Condition (4.2) then becomes that M is prefix closed and, since the class of closed and controllable sublanguages of $\mathcal{L}(P)$ is closed under arbitrary intersections (see e.g., [1]), it contains a unique infimal element. Thus, in view of (2.12) when $Q_m = Q$, the solvability test of CCP simplifies to the following.

Let P^u be the automaton obtained from P upon deleting from it all controlled transitions, i.e., $P^u = (Q, \Sigma, \delta^u, q_0, Q_m)$, where

$$\delta^u(\sigma, q) = \begin{cases} \delta(\sigma, q) & \text{if } \sigma \in \Sigma_u \wedge \delta(\sigma, u)! \\ \text{undefined} & \text{otherwise.} \end{cases} \quad (4.4)$$

Clearly $\mathcal{L}(P^u)$ is controllable and closed and for any supervisor S

$$\mathcal{L}(P^u) \subset \mathcal{L}(S/P).$$

Proposition 4.1 can now be replaced (for the case $Q_m = Q$) by the following proposition.

Proposition 4.2: Let $P = (Q, \Sigma, \delta, q_0, Q_m)$ with $Q_m = Q$ and let $E \subset \Sigma^*$ be given. Then CCP is solvable if and only if $E \Leftarrow \mathcal{L}(P^u)$.

The above proposition yields a simple procedure for solving CCP in the special case when $Q_m = Q$. First, one constructs P^u and checks whether $E \Leftarrow \mathcal{L}(P^u)$. This verification can be done with Algorithm 3.2 (which is of polynomial complexity) when E is suffix closed, and with Algorithm 3.3 otherwise. When $E \Leftarrow \mathcal{L}(P^u)$, then the most restrictive supervisor, i.e., the supervisor that disables all controlled events, is a solution of CCP because this supervisor satisfies

$$E \Leftarrow \mathcal{L}(S/P) = \mathcal{L}(P^u).$$

The above solution is, of course, not a very efficient one and in general better solutions might be sought. We shall return to this issue later.

Remark 4.1: In [9] and [18], a problem similar to CCP was discussed. A language L was called l-stabilizable w.r.t. E if there exists a sublanguage $H \subset L$, controllable w.r.t. L, such that $E \Leftarrow H$. The authors argued there that for a system $P = (Q, \Sigma, \delta, q_0, Q_m), \mathcal{L}_m(P)$ is l-stabilizable w.r.t. a given language E if and only if $E \Leftarrow \mathcal{L}_m(P^u)$. The justification for this conclusion was the claim that $\mathcal{L}_m(P^u)$ is the infimal controllable sublanguage of $\mathcal{L}_m(P)$. When $Q_m = Q$, this claim is indeed true. It is false, however, in the general case when $Q_m \neq Q$, since in that case $\mathcal{L}_m(P^u)$ is not necessarily a controllable language. To solve the l-stabilizability problem as well as CCP in the general case, a different approach must be taken as discussed in detail below.

We turn next to the problem of solving CCP in the general case when $Q_m \neq Q$ and E is a general regular language. To this end we shall make use of two known algorithms that are briefly discussed below.

The first algorithm computes the supremal controllable and $\mathcal{L}_m(P)$-closed sublanguages of a given language. Specifically, let $P = (Q, \Sigma, \delta, q_0, Q_m)$ be a DES and $M \subset \mathcal{L}_m(P)$ be a given regular language. Let $A = (X, \Sigma, \xi, x_0, X_m)$ be a deterministic recognizer for M ($\mathcal{L}_m(A) = M$). The construction of a deterministic recognizer for the supremal controllable (with respect to $\mathcal{L}(P)$) and $\mathcal{L}_m(P)$-closed sublanguage of M is accomplished by an algorithm of complexity of $O(|Q||X|)$ as follows. First construct the synchronous composition $C = P||A = (Q \times X, \Sigma, \alpha, (q_0, x_0), Q_m \times X_m)$ as described in Section II. Then $\mathcal{L}_m(C) = \mathcal{L}_m(P) \cap \mathcal{L}_m(A) = \mathcal{L}_m(P) \cap M = M$. The supremal $\mathcal{L}_m(P)$-closed sublanguage of M is then given by $\mathcal{L}_m(C')$ (see [14, Appendix B]), where C' is given by the restriction

$$C' = C|Q_m \times (X - X_m).$$

The complexity of constructing C' is $O(|Q||X|)$. By [2, Proposition, 6.1], the supremal controllable sublanguage of $\mathcal{L}_m(C')$ equals the supremal controllable and $\mathcal{L}_m(P)$-closed sublanguage of M. A deterministic recognizer for the supremal controllable sublanguage of $\mathcal{L}_m(C')$ can be constructed by the algorithm of [11] (see also [12]), whose complexity is $O(|Q|^2 \cdot |X|^2)$. An algorithm of complexity $O(|Q||X|)$ has recently been given in [13]. Thus, the construction of the supremal controllable and $\mathcal{L}_m(P)$-closed sublanguage of M can be accomplished with overall complexity $O(|Q||X|)$.

The second algorithm that we shall employ below appeared in [4] (see also [5], [9], [18]) and is briefly described next. Let $P = (Q, \Sigma, \delta, \cdot, Q_m)$ be a DES with unspecified initial state. Fix a subset $\hat{Q} \subset Q$ (which we shall regard as the set of legal states). The set $\hat{Q}$ is called a strong attractor for a state $q \in Q$ if, when initialized at q, the system P always reaches a state in $\hat{Q}$ after a bounded number of transitions. Formally, $\hat{Q}$ is a strong attractor for $q \in Q$ if $\mathcal{A}(P|\hat{Q}, q)$ is acyclic and no state of $\mathcal{A}(P|\hat{Q}, q)$ is a deadlock state of P (i.e., a state at which no transition is defined). A set $\hat{Q}$ is called a weak attractor for $q \in Q$ if there exists a supervisor S such that $\hat{Q}$ is a strong attractor for q in (S/P). Such a supervisor, when it exists, can be chosen to be static, i.e., for $s, t \in \mathcal{L}(P), S(t) = S(s)$ whenever $\delta(s, q_0) = \delta(t, q_0) (= q)$. Note also that a static supervisor can be written as a map $S: X \to 2^\Sigma$. The set of all states $q \in Q$ for which $\hat{Q}$ is a weak attractor is called the region of weak attraction and is denoted by $\Omega_P(\hat{Q})$. If $\hat{Q} = \phi$, then $\Omega_P(\hat{Q}) := \phi$. In [4] and [5] algorithms are given for computation of $\Omega_P(\hat{Q})$ with complexity $O(|Q|^2)$. An algorithm of complexity $O(|Q|)$ has recently been presented in [9] and [18].

We return to our main discussion. To test the solvability of CCP, we first construct the automaton $B = (V, \Sigma, \alpha, v_0, V_m)$ as defined in Step 1) of Algorithm 3.3. In view of the fact that $\mathcal{L}(B) = \mathcal{L}(P)$ and $\mathcal{L}_m(B) = \mathcal{L}_m(P)$, [(3.10), (3.11)], it is clear that B can be regarded as a new model of our process. Thus, CCP is solvable with respect to P if and only if it is solvable with respect to B.

The automaton B offers an obvious advantage over P for testing the solvability of CCP in that its structure provides a simple way of identifying all strings of $\mathcal{L}_m(B)$ that do not have a suffix in E. Indeed, let $t \in \mathcal{L}_m(B)$ and $v = (q, \chi) \in V_m$ satisfy $\alpha(t, (q_0, \{x_0\})) = v$. Then, since $V_m = Q_m \times 2^X$, it follows that $q \in Q_m$ and, see (3.9), t has a suffix in E if and only if $\chi \cap X_m \neq \phi$. Let F be the set defined by

$$F = \{(q, \chi) \in V_m | q \in Q_m \text{ and } \chi \cap X_m \neq \phi\}. \quad (4.5)$$

Define B' as the automaton

$$B' = (V, \Sigma, \alpha, v_0, F)$$

which is obtained from B by restricting its set of final states to F. Suppose now that S is a solution to CCP with respect to B, i.e., $E \Leftarrow \mathcal{L}_m(S/B)$, and let $t \in \mathcal{L}_m(B) - \mathcal{L}_m(B')$. Since t has no suffix in E, it follows that $t \notin \mathcal{L}_m(S/B)$ and it follows further that $\mathcal{L}_m(S/B) \subset \mathcal{L}_m(B')$. Since $\mathcal{L}_m(S/B)$ is controllable with respect to $\mathcal{L}(B)$ and is $\mathcal{L}_m(B)$-closed, we conclude that $\mathcal{L}_m(S/B) \subset \sup CL(\mathcal{L}_m(B'))$, where $\sup CL(\mathcal{L}_m(B'))$ is the supremal controllable (w.r.t. $\mathcal{L}(B)$) and $\mathcal{L}_m(B)$-closed sublanguage of $\mathcal{L}_m(B')$. Define now the automaton

$$\hat{B} = (\hat{V}, \Sigma, \hat{\alpha}, \hat{v}_0, \hat{V}_m) \quad (4.6)$$

as a recognizer for $\sup CL(\mathcal{L}_m(B'))$. Such an automaton can be constructed by an algorithm of complexity $O(|Q| \cdot 2^{|X|})$ as was described earlier. From the algorithm it is also clear that $\hat{B}$ is a subautomaton of B. Now, the inclusion $\mathcal{L}_m(S/B) \subset \mathcal{L}_m(\hat{B}) \subset \mathcal{L}_m(B)$ implies that $\mathcal{L}_m(S/B) = \mathcal{L}_m(S/\hat{B})$. Hence, CCP is solvable with respect to B only if it is solvable with respect to $\hat{B}$. To see the reverse implication, i.e., that solvability of CCP with respect to $\hat{B}$ implies the solvability of CCP with respect to B, observe that $\mathcal{L}_m(\hat{B})$ is a controllable (w.r.t. $\mathcal{L}(B)$) and $\mathcal{L}_m(B)$-closed sublanguage of $\mathcal{L}_m(B)$. Thus, there exists a supervisor $\hat{S}$ such that $\mathcal{L}_m(\hat{S}/B) = \mathcal{L}_m(\hat{B})$. If there exists a supervisor S such that $E \Leftarrow \mathcal{L}_m(S/\hat{B})$, then $E \Leftarrow \mathcal{L}_m(S/(\hat{S}/B)) = \mathcal{L}_m(S \times \hat{S}/B)$, where $S \times \hat{S}$ is the supervisor that consists of the conjunction of S and $\hat{S}$ (see e.g., [2]). We have just proved the following.

Lemma 4.1: For a DES P and a regular language E, let $\hat{B}$ be the automaton defined by (4.6). Then CCP is solvable with respect to P if and only if CCP is solvable with respect to $\hat{B}$.

The following corollary is immediate.

Corollary 4.1: If $\mathcal{L}_m(\hat{B}) = \phi$ then CCP is unsolvable.

In Theorem 3.2 (see Algorithm 3.3) we have shown that the convergence $\mathcal{L}_m(A) \Leftarrow \mathcal{L}_m(P)$ is equivalent to the acyclicity of $V - \hat{T}$ where $\hat{T}$ [see (3.7)] consists of all states $v = (q, \chi)$ such that $\mathcal{L}_m(P_q) \subset \mathcal{L}_m(A_\chi)$. Similarly, we construct now a subset $\hat{T}$ of states of $\hat{B}$ such that for each $v = (q, \chi) \in \hat{T}$ there exists a supervisor S with the property that $\mathcal{L}_m(S/P_q) \subset \mathcal{L}_m(A_\chi)$. We will then show that the solvability of CCP is equivalent to the possibility of preventing the system $\hat{B}$ from executing cycles in $\hat{V} - \hat{T}$.

For a state $v = (q, \chi)$ of $\hat{B}$ it follows from (4.2) and (4.3) that there exists a nonblocking supervisor S such that $\mathcal{L}_m(S/P_q) \subset \mathcal{L}_m(A_\chi)$ if and only if the class of controllable (with respect to $\mathcal{L}(P_q)$) and $\mathcal{L}_m(P_q)$-closed sublanguages of $\mathcal{L}_m(P_q) \cap \mathcal{L}_m(A_\chi)$ does not consist of the empty language. To check this condition we proceed as follows. First note that A_χ is not a deterministic automaton since it has in general more than one initial state. Hence we begin by constructing a deterministic recognizer $A^\chi = (W, \Sigma, \beta, w_0, W_m)$ for $\mathcal{L}_m(A_\chi)$, i.e., $\mathcal{L}_m(A^\chi) = \mathcal{L}_m(A_\chi)$. Next, by the algorithm described earlier we construct a deterministic automaton D^v such that

$$\mathcal{L}_m(D^v) = \sup CL(\mathcal{L}_m(P_q) \cap \mathcal{L}_m(A^\chi)). \tag{4.7}$$

The required supervisor S such that $\mathcal{L}_m(S/P_q) \subset \mathcal{L}_m(A_\chi)$ exists if and only if $\mathcal{L}_m(D^v) \neq \phi$. The construction of D^v requires $O(|Q| \cdot 2^{|X|})$ computations since the number of states of A^χ is bounded by $2^{|X|}$ and the number of states of P_q is $|Q|$.

Now, let $\hat{T}$ be the set of states defined by

$$\hat{T} = \{v \in \hat{V} | \mathcal{L}_m(D^v) \neq \phi\}. \tag{4.8}$$

Since the number of states of $\hat{B}$ is bounded by $|Q| \cdot 2^{|X|}$ (the number of states of B) the construction of $\hat{T}$ requires $O((|Q| \cdot 2^{|X|})^2)$ computations.

The final step in our computation is the employment of the algorithm of [9] and [18] to compute the set $\Omega_{\hat{B}}(\hat{T})$. The complexity of this algorithm is $O(|Q| \cdot 2^{|X|})$.

We now have the following necessary and sufficient condition for the solvability of CCP.

Theorem 4.1: Let P be a DES, and let $E \subset \Sigma^*$ be a regular language. Let $\hat{B}$ and $\hat{T}$ be the automaton and the set of states as defined in (4.6) and (4.8), respectively. Then CCP is solvable if and only if $v_0 \in \Omega_{\hat{B}}(\hat{T})$.

Proof: Only If. By Lemma 4.1 CCP is solvable with respect to P if and only if it is solvable with respect to $\hat{B}$. Suppose that CCP is solvable so that there exists a nonblocking supervisor S such that $E \Leftarrow \mathcal{L}_m(S/\hat{B})$. We shall show that the assumption that $v_0 \notin \Omega_{\hat{B}}(\hat{T})$ leads to a contradiction. First we shall show that, under S, there are no deadlock states in $\hat{V} - \hat{T}$. Indeed, let $\hat{v} = (q, \chi) \in \hat{V} - \hat{T}$ and $t \in \mathcal{L}(S/\hat{B})$ be such that $\hat{\alpha}(t, \hat{v}_0) = \hat{v}$. Suppose that $\hat{v}$ is a deadlock state, i.e., $ts \in \mathcal{L}(S/\hat{B}) \Rightarrow s = \epsilon$. If $q \in Q - Q_m$, then $(q, \chi) \notin \hat{V}_m$, implying that $t \notin \mathcal{L}_m(S/\hat{B})$. Thus $\mathcal{L}(S/\hat{B}) \neq \mathcal{L}_m(S/\hat{B})$, contradicting the nonblocking property of S. If $q \in Q_m$ then $\epsilon \in \mathcal{L}_m(P_q)$. By definition of $\hat{B}$, $\chi \cap X_m \neq \phi$ so that $\epsilon \in \mathcal{L}_m(A_\chi)$, implying that $\epsilon \in \mathcal{L}_m(P_q) \cap \mathcal{L}_m(A_\chi)$. Since the language $\{\epsilon\}$ is trivially $\mathcal{L}_m(P_q)$-closed and controllable with respect to $\mathcal{L}(P_q)$, it follows that $\{\epsilon\} \in CL(\mathcal{L}_m(P_q) \cap \mathcal{L}_m(A_\chi))$, so that $\mathcal{L}_m(D^{\hat{v}}) \neq \phi$. This implies that $\hat{v} \in \hat{T}$, a contradiction.

Since $(S/\hat{B})$ has no deadlock states in $\hat{V} - \hat{T}$, the assumption that $\hat{v}_0 \notin \Omega_{\hat{B}}(\hat{T})$ implies that $(S/\hat{B})$ has a cycle in $\hat{V} - \hat{T}$. With the employment of the same arguments as in the proof of Theorem 3.2, it is not difficult to show that the existence of a cycle in $\hat{V} - \hat{T}$ contradicts the assumption that $E \Leftarrow \mathcal{L}_m(S/\hat{B})$.

If. The proof of the "if" part of the theorem consists of the following algorithm for construction of the supervisor S that satisfies the condition that $E \Leftarrow \mathcal{L}_m(S/\hat{B})$ whenever $\hat{v}_0 \in \Omega_{\hat{B}}(\hat{T})$.

Algorithm 4.1:

Step 1) Define the supervisor $S_1 : \hat{V} \to 2^{\Sigma_c}$ as a static supervisor such that $\hat{T}$ is a strong attractor for $\Omega_{\hat{B}}(\hat{T})$ with respect to $S_1/\hat{B}$. The existence of this supervisor follows from the definition of $\Omega_{\hat{B}}(\hat{T})$.

Step 2) For each $\hat{v} \in \hat{V}$, let $D^{\hat{v}}$ be the automaton defined earlier (see (4.7)).
The required supervisor is given by $S : L(\hat{B}) \to 2^{\Sigma_c}$ as found in (4.9) at the bottom of the page.

The supervisor S is obviously well defined because for each $t \in \mathcal{L}(\hat{B})$ it defines a unique set of controlled events to be disabled. Next, note that under the supervision of S, the system $\hat{B}$ reaches a state $\hat{v} \in \hat{T}$ within a number of transitions bounded by $|\hat{V} - \hat{T}|$ because for states in $\Omega_{\hat{B}}(\hat{T}) - \hat{T}$ S acts as S_1. When a state $\hat{v} = (q, \chi) \in \hat{T}$ is reached, then it is guaranteed by S that all marked strings generated by $\hat{B}$ (starting at $\hat{v}$) belong to

$$\mathcal{L}_m(D^{\hat{v}}) = \sup CL(\mathcal{L}_m(P_q) \cap \mathcal{L}_m(A^\chi)) \subset \mathcal{L}_m(A_\chi).$$

It thus follows that for $t \in \mathcal{L}_m(S/\hat{B}), |t| \geq |\hat{V} - \hat{T}|$, there exists $i \leq |\hat{V} - \hat{T}|$ such that $\alpha(pr_i(t), \hat{v}_0) = \hat{v} = (q, \chi) \in \hat{T}$ and $\text{suf}_i(t) \in \mathcal{L}_m(A_\chi)$. By (3.12)–(3.14) it follows that there exists $j \leq i$ such that $\text{suf}_j(t) \in E$. If $|t| < |\hat{V} - \hat{T}|$, the existence of i $(\leq |\hat{V} - \hat{T}|)$ such that $\text{suf}_i(t) \in E$ follows from the definition of $\hat{B}$. This concludes the proof. □

The complexities of computing the automaton $\hat{B}$ and the set $\hat{T}$ are $O(|Q| \cdot 2^{|X|})$ and $O((|Q| \cdot 2^{|X|})^2)$, respectively. The computation of $\Omega_{\hat{B}}(\hat{T})$ requires $O(|Q| \cdot 2^{|X|})$ computations. Thus, the complexity of verifying the solvability of CCP (i.e., checking the condition $\hat{v}_0 \in \Omega_{\hat{B}}(\hat{T})$) is $O((|Q| \cdot 2^{|X|})^2)$. In case CCP is solvable, the supervisor defined by (4.9) is a solution of CCP.

$$S(t) = \begin{cases} S_1(\hat{v}) \quad \text{where } \hat{v} = \hat{\alpha}(t, \hat{v}_0) & \text{if } \{\forall_i\} \hat{\alpha}(pr_i(t), \hat{v}_0) \notin \hat{T} \\ \{\sigma \in \Sigma_c | \ \text{suf}_{i_0}(t)\sigma \notin \mathcal{L}(D^{\hat{v}})\}, & \text{if there exists } i \text{ such that} \\ \quad \text{where } i_0 \text{ is the least integer } i & \hat{\alpha}(pr_i(t), \hat{v}_0) \in \hat{T} \\ \quad \text{that satisfies } \hat{\alpha}(pr_i(t), \hat{v}_0) \in \hat{T} & \end{cases} \tag{4.9}$$

When E is suffix closed, the solvability of CCP can be verified by an algorithm of linear complexity (i.e., of complexity $O(|Q||X|)$) as described below. The correctness of the following algorithm follows from Theorems 3.1 and 4.1.

Theorem 4.2: Let $P = (Q, \Sigma, \delta, q_0, Q_m)$ be a given DES, and let E be a suffix-closed regular language. Let $A = (X, \Sigma, \xi, x_0, X_m)$ be a deterministic automaton such that $\mathcal{L}_m(A) = E$. Then the following algorithm verifies the solvability of CCP.

Algorithm 4.2:

Step 1) For each $q \in Q$ construct a deterministic automaton D^q such that $\mathcal{L}_m(D^q)$ is the supremal controllable (w.r.t $\mathcal{L}(P_q)$) and $\mathcal{L}_m(P_q)$-closed sublanguage of $\mathcal{L}_m(P_q) \cap \mathcal{L}_m(A)$.

Step 2) Compute the set Y of states defined by

$$Y := \{q \in Q | \mathcal{L}_m(D^q) \neq \phi\}.$$

Step 3) By the algorithm of [9], [18] compute the set $\Omega_P(Y)$.

Step 4) CCP is solvable if and only if $q_0 \in \Omega_P(Y)$. □

The set Y can be constructed with the algorithm of [13] using the same method as was described in Section III for the computation of the set T. The complexity of constructing the set Y as well as the overall complexity of Algorithm 4.2 is $O(|Q||X|)$.

Algorithms 4.1 and 4.2, which constitute tests for solvability of CCP, also provide a construction of nonblocking supervisors that guarantee convergence of the closed-loop language to the specification E whenever CCP is solvable. The synthesis is, however, neither optimal nor unique since, in general, there does not exist a least restrictive supervisor that guarantees convergence. Specifically, there does not, in general, exist a supervisor $\hat{S}$ such that $E \Leftarrow \mathcal{L}_m(\hat{S}/P)$ and such that if S is any supervisor satisfying $E \Leftarrow \mathcal{L}_m(S/P)$, then $\mathcal{L}_m(S/P) \subset \mathcal{L}_m(\hat{S}/P)$. To see this, consider the following simple example. Let $\mathcal{L}_m(P) = \alpha^*\beta$ and let $E = \beta$. Assume further that $\Sigma_c = \{\alpha\}$ and $\Sigma_u = \{\beta\}$. The solvability of CCP is obvious, and for any $k \geq 0$ there is a supervisor S_k such that $\mathcal{L}_m(S_k/P) = \overline{\alpha^k\beta}$ (which clearly satisfies $\beta \Leftarrow \overline{\alpha^k\beta}$). But for $k_1 \leq k_2$, $\overline{\alpha^{k_1}\beta} \subset \overline{\alpha^{k_2}\beta}$ and k can be arbitrarily large!

The issue of optimal supervisors (in contexts other than "minimally restrictive") is discussed in [17].

V. Asymptotic Behavior Of DES

In nonterminating systems, i.e., systems that operate indefinitely, one can distinguish between their transient and permanent (or asymptotic) behaviors. Intuitively, the asymptotic behavior of a system consists of all strings that the system can execute after having already performed an arbitrarily large number of transitions.

For a language $L \subset \Sigma^*$ we define its asymptotic suffix, denoted L_∞ as

$$L_\infty := \{s \in \Sigma^* | \forall i \geq 0, \exists t_i \in \Sigma^*, |t_i| \geq i \wedge t_i s \in L\}. \quad (5.1)$$

Clearly, L_∞ is an empty language whenever L is a bounded language (see Remark 3.1). It is also easy to see that L_∞ is suffix closed for every language $L \subset \Sigma^*$.

Let $P = (Q, \Sigma, \delta, q_0, Q_m)$ be a DES. The asymptotic behavior of P is defined as $\mathcal{L}_m(P)_\infty$. A recognizer for $\mathcal{L}_m(P)_\infty$ can be constructed as follows. Let $cy(P)$ be the set of all states of P that are accessible from q_0 and belong to a cycle of P, that is

$$cy(P) := \{q \in Q | (\exists s, t \in \Sigma^*) q = \delta(t, q_0), s \neq \epsilon \wedge \delta(s, q) = q\}. \quad (5.2)$$

Let $Q_\infty := \mathcal{A}(P, cy(P))$ and let $Q_{m\infty} := Q_m \cap Q_\infty$. Define

$$P_\infty := (Q_\infty, \Sigma, \delta, Q_\infty, Q_{m\infty}). \quad (5.3)$$

The following proposition states that P_∞ is a recognizer for $\mathcal{L}_m(P)_\infty$.

Proposition 5.1: $\mathcal{L}_m(P)_\infty = \mathcal{L}_m(P_\infty)$.

Proof: $\underline{\mathcal{L}_m(P)_\infty \subset \mathcal{L}_m(P_\infty)}$. Let $t \in \mathcal{L}_m(P)_\infty$ be any string. By (5.1) there exists a string $w, |w| > |Q|$, such that $wt \in \mathcal{L}_m(P)$. Let $q_0, q_1, \cdots, q_l$, $l = |w|$, be the path associated with w. Since $l > |Q|$, there exists a state $q \in Q$ that occurs along this path at least twice, i.e., there exist $i_1, i_2, 0 \leq i_1 < i_2 \leq l$ such that $q = q_{i_1} = q_{i_2}$. Thus, $q \in cy(P)$ and $q' := \delta(w, q_0) \in \mathcal{A}(P, cy(P)) = Q_\infty$. Since $wt \in \mathcal{L}_m(P)$ and P is deterministic, it follows that

$$t \in \mathcal{L}_m(Q_\infty, \Sigma, \delta, q', Q_{m\infty}) \subset \mathcal{L}_m(Q_\infty, \Sigma, \delta, Q_\infty, Q_{m\infty}) = \mathcal{L}_m(P_\infty).$$

$\underline{\mathcal{L}_m(P)_\infty \supset \mathcal{L}_m(P_\infty)}$. Let $w \in \mathcal{L}_m(P_\infty)$. Then $w \in \mathcal{L}_m(Q_\infty, \Sigma, \delta, q, Q_{m\infty})$ for some $q \in Q_\infty$. Thus see the equation at the bottom of the page.

By (5.1) this implies that $w \in \mathcal{L}_m(P)_\infty$ concluding the proof. □

The construction of the set $cy(P)$ can be accomplished by the well-known algorithm for computing strongly connected components of P (see [7, p. 64]) whose complexity (under our assumption that $|\Sigma| = \mathcal{O}(1)$) is $O(|Q|)$. The construction of $\mathcal{A}(P, cy(P))$ requires $O(|Q|)$ computations. Thus, the complexity of the construction of P_∞ is linear in the number of states of P.

Our next goal will be to show that for a given system$P$, the convergence of $\mathcal{L}_m(P)$ to a given language E can be determined by testing the asymptotic behavior of P. The following proposition gives a necessary condition for the convergence $E \Leftarrow \mathcal{L}_m(P)$ in terms of the asymptotic behavior of P.

Proposition 5.2: Let P be a given DES, and let $E \subset \Sigma^*$ be a given language. If $E \Leftarrow \mathcal{L}_m(P)$, then $\mathcal{L}_m(P)_\infty \subset \text{suf}(E)$.

$$\begin{array}{lll} & \exists \quad u \in \Sigma^*, \exists q' \in cy(P), \delta(u, q') = q & \text{(definition of } Q_\infty) \\ \Rightarrow & \exists \quad (t, s \in \Sigma^*) q' = \delta(t, q_0), s \neq \epsilon \text{ and } q' = \delta(s, q') & \text{(follows by (5.2))} \\ \Rightarrow & \quad ts^*uw \in \mathcal{L}_m(P). & \end{array}$$

Proof: Assume that $E \Leftarrow \mathcal{L}_m(P)$, let the convergence time be k and let $s \in \mathcal{L}_m(P)_\infty$ be any string. By (5.1), for each $i \geq 0$ there exists $t_i \in \Sigma^*, |t_i| \geq i$, such that $t_i s \in \mathcal{L}_m(P)$. Choose $t_i > k$. Since there exists $j \leq k$ such that $\text{suf}_j(t_i s) \in E$, it follows that

$$s = \text{suf}_{|t_i|}(t_i s) = \text{suf}_{|t_i|-j}\,(\text{suf}_j(t_i s)) \in \text{suf}(E)$$

concluding the proof. □

The following lemma states that every regular language $L \subset \Sigma^*$ converges to L_∞.

Lemma 5.1: For a DES $P = (Q, \Sigma, \delta, q_0, Q_m)$, $\mathcal{L}_m(P)_\infty \Leftarrow \mathcal{L}_m(P)$.

Proof: Since the language $\mathcal{L}_m(P)_\infty$ is suffix closed it follows that $\epsilon \in \mathcal{L}_m(P)_\infty$. Thus for each $t \in \mathcal{L}_m(P)$, $\text{suf}_{|t|}(t) = \epsilon \in \mathcal{L}_m(P)_\infty$ implying that if $|t| < |Q|$ there exists $i \leq |Q|$ such that $\text{suf}_i(t) \in \mathcal{L}_m(P)_\infty$.

Let $t \in \mathcal{L}_m(P), |t| > |Q|$. Since the number of states of P is $|Q|$, there exists $i, i \leq |Q|$, such that $\delta(pr_i(t), q_0) = q$ where $q \in cy(P) \subset Q_\infty$. Since by Proposition 5.1 $\mathcal{L}_m(P)_\infty = \mathcal{L}_m(P_\infty)$, it then follows that $\text{suf}_i(t) \in \mathcal{L}_m(P_\infty) = \mathcal{L}_m(P)_\infty$. Hence the convergence time of $\mathcal{L}_m(P)$ to $\mathcal{L}_m(P)_\infty$ is bounded by $|Q|$, concluding the proof. □

It should be noted that if L is not a regular language, then the convergence $L_\infty \Leftarrow L$ does not necessarily hold. Consider, for example, the language $\{\alpha^k \beta^k \gamma^* | \forall k \geq 0\}$.

Proposition 5.2 and Lemma 5.1 yield the following.

Corollary 5.1: Let P be a DES. Then $\mathcal{L}_m(P)_\infty$ is the infimal (in the sense of language inclusion) suffix-closed language that $\mathcal{L}_m(P)$ converges to, i.e.,

$$\mathcal{L}_m(P)_\infty = \cap\{M \subset \Sigma^* |\ \text{suf}(M) = M \text{ and } M \Leftarrow \mathcal{L}_m(P)\}.$$

A sufficient condition for the convergence $E \Leftarrow \mathcal{L}_m(P)$ is given be the following proposition.

Proposition 5.3: For a DES P and a language $E \subset \Sigma^*$, if $\mathcal{L}_m(P)_\infty \subset E$, then $E \Leftarrow \mathcal{L}_m(P)$.

Proof: By Lemma 5.1, $\mathcal{L}_m(P)_\infty \Leftarrow \mathcal{L}_m(P)$. Thus, if $\mathcal{L}_m(P)_\infty \subset E$, it follows directly by (2.12) that also $E \Leftarrow \mathcal{L}_m(P)$. □

By combining Propositions 5.2 and 5.3 for the special case of suffix-closed languages we obtain the following necessary and sufficient condition for $E \Leftarrow \mathcal{L}_m(P)$.

Theorem 5.1: Let P be a DES and let $E \subset \Sigma^*$ be a suffix-closed language. Then E $\mathcal{L}_m(P)$ if and only if

$$\mathcal{L}_m(P)_\infty \subset E. \tag{5.4}$$

Next we turn to the problem of supervisory control. In the previous section we considered the problem of language convergence, that is, the problem of synthesizing a supervisor that guarantees convergence of the supervised language to a specified legal language. We shall now be interested in the less restrictive control problem wherein only the asymptotic behavior of the supervised process is required to lie in a specified legal language. Thus, we shall consider the following.

Asymptotic Control Problem (ACP): Let P be a given DES and let $E \subset \Sigma^*$ be a given language. Synthesize a nonblocking supervisor S such that

$$\mathcal{L}_m(S/P)_\infty \subset E. \tag{5.5}$$

In the case when E is a suffix-closed language there is an obvious equivalence between ACP and CCP. This equivalence is stated in the following reinterpretation of Theorem 5.1.

Proposition 5.4: Let E be suffix closed. Then ACP is solvable if and only if CCP is solvable.

Let us now examine the solvability of ACP in the case when$E$ is not suffix closed. Since L_∞ is suffix closed for any language L, ACP is solvable (i.e., there exists a nonblocking supervisor that satisfies (5.5)) if and only if there exists a suffix-closed sublanguage $M \subset E$ and a nonblocking supervisor S such that

$$\mathcal{L}_m(S/P)_\infty \subset M. \tag{5.6}$$

By Theorem 5.1, a supervisor S satisfies (5.6) if and only if

$$M \Leftarrow \mathcal{L}_m(S/P). \tag{5.7}$$

Thus, to determine the solvability of ACP, it is sufficient to check whether CCP is solvable with respect to the supremal suffix-closed sublanguage of E. The existence of the supremal suffix-closed sublanguage of E follows from the closeness of the set of suffix-closed languages under arbitrary unions.

Proposition 5.5: For a language E, ACP is solvable if and only if CCP is solvable with respect to the supremal suffix-closed sublanguage of E.

We conclude this section with an algorithm for computing the supremal suffix-closed sublanguage of a given regular language E.

Algorithm 5.1:

Input) A deterministic automaton

$$A = (X, \Sigma, \xi, x_0, X_m)$$

such that (without loss of generality) $\xi(\sigma, x) \neq x_0$ for all $\sigma \in \Sigma$, $x \in X$.

Output) A deterministic recognizer for the supremal suffix-closed sublanguage of $\mathcal{L}_m(A)$.

Construct a deterministic automaton

$$C = (U, \Sigma, \beta, u_0, U_m)$$

where $U = 2^X, u_0 = \{x_0\}, U_m = \{u \in U | (u - \{x_0\}) \subset X_m$and $\beta : \Sigma \times U \to U$ is defined by

$$\beta(\sigma, u) = \begin{cases} \{x_0\} \cup \{\xi(\sigma, x)\ | x \in u\} & \text{if } \xi(\sigma, x)! \text{ for all } x \in u \\ \text{undefined} & \text{otherwise.} \end{cases} \tag{5.8}$$

Proposition 5.6: $\mathcal{L}_m(C)$ is the supremal suffix-closed sublanguage of $\mathcal{L}_m(A)$.

Proof: First observe that for any two states $u, w \in U$ such that $u \subset w$ (as elements of 2^X), $\beta(\sigma, w)!$ only if $\beta(\sigma, u)!$, and if $\beta(\sigma, w)!$ then $\beta(\sigma, u) \subset \beta(\sigma, w)$. Moreover, if $\beta(\sigma, u)!$ and $x \in u$, then $\xi(\sigma, x)!$

Let $s = \sigma_1 \cdots \sigma_n \in \mathcal{L}_m(C)$ be any string, and let $u_0, \cdots, u_n$ be the associated path in C, i.e., $u_i = \beta(\sigma_i, u_{i-1}), i = 1, \cdots, n$ and $u_n \in U_m$. Then since $u_0 = \{x_0\}$, it follows from the above observation that there exists a path $x_0, \cdots, x_n$ in A such that $x_i = \xi(\sigma_i, x_{i-1})$ and

$x_i \in u_i - \{x_0\}$ for all $i = 1, \cdots, n$. From the definition of U_m, $x_n \in u_n - \{x_0\} \subset X_m$, so that $s \in \mathcal{L}_m(A)$, implying that $\mathcal{L}_m(C) \subset \mathcal{L}_m(A)$.

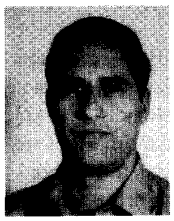
To see that $\mathcal{L}_m(C)$ is suffix closed, consider $\text{suf}_j(s) = \sigma_{j+1} \cdots \sigma_n$ for some $0 \le j \le n$. We must show that $\text{suf}_j(s) \in \mathcal{L}_m(C)$. Note that $u_0 \subset u_j$, whence since $u_{j+1} = \beta(\sigma_{j+1}, u_j)$ is defined, so is $\hat{u}_1 := \beta(\sigma_{j+1}, u_0)$ and $\hat{u}_1 \subset u_{j+1}$. Proceeding inductively, we obtain a sequence of states $u_0, \hat{u}_1, \cdots, \hat{u}_{n-j}$ with $\hat{u}_1 = \beta(\sigma_{j+1}, u_0), \hat{u}_{i+1} = \beta(\sigma_{i+j+1}, \hat{u}_i)$ and $\hat{u}_i \subset u_{i+j}$. Since $\hat{u}_{n-j} \subset u_n \in U_m$, we conclude that $\hat{u}_{n-j} \in U_m$, whence $\text{suf}_j(s) \in \mathcal{L}_m(C)$ as claimed.

Finally, to see that $\mathcal{L}_m(C)$ is the supremal suffix-closed sublanguage of $\mathcal{L}_m(A)$, we must show that if $s = \sigma_1 \cdots \sigma_n \in \mathcal{L}_m(A)$ is a string such that $\text{suf}_j(s) \in \mathcal{L}_m(A)$ for all $j = 0, \cdots, n$, then $s \in \mathcal{L}_m(C)$. Suppose that $s \notin \mathcal{L}_m(C)$. Let $l, 1 \le l \le n$, be the smallest integer such that $u_j = \beta(\sigma_j, u_{j-1})$ is defined for all $j = 1, \cdots, l-1$ but $\beta(\sigma_l, u_{l-1})$ is undefined. This implies [see (5.8)] that there exists $k, 0 \le k \le l-1$, and a sequence of states $x_0, x_1, \cdots, x_{l-k-1}$ such that $x_j = \xi(\sigma_{k+j}, x_{j-1})$ for all $j = 1, \cdots, l-k-1$ and $\xi(\sigma_l, x_{l-k-1})$ is undefined. But then $\text{suf}_k(s) \notin \mathcal{L}(A)$, a contradiction.

□

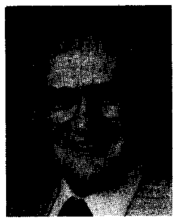
REFERENCES

[1] P. J. Ramadge and W. M. Wonham, "Supervisory control of a class of discrete-event processes," *SIAM J. Control Optim.*, vol. 25, no. 1, pp. 206-230, Jan. 1987.
[2] ——, "Modular supervisory control of discrete-event systems," *Math. Contr. Signals Syst.*, vol. 1, pp. 13-30, 1988.
[3] ——, "On the supremal controllable sublanguage of a given language," *SIAM J. Contr. Optim.*, vol. 25, no. 3, pp. 637-659, May 1987.
[4] Y. Brave and M. Heymann, "On stabilization of discrete-event processes," *Int. J. Contr.*, vol. 51, no. 5, pp. 1101-1117, 1990.
[5] C. M. Özveren, A. S. Willsky, and P. J. Antaklis, "Stability and stabilizability of discrete-event dynamic systems," *J. ACM*, vol. 38, no. 3, pp. 730-752, 1991.
[6] J. E. Hopcroft and J. D. Ulman, *Introduction to Automata Theory, Languages and Computations*. Reading, MA: Addison-Wesley, 1979.
[7] S. Even, *Graph Algorithms*. Maryland: Computer Science Press, 1979.
[8] Y. Brave and M. Heymann, "On optimal attraction in discrete-event processes," *Inform. Sci.*,vol. 67, pp. 245-267, 1993.
[9] R. Kumar, V. Garg, and S. I. Marcus, "Stability of DES behavior," in *Proc. 1991 IFAC Intl. Symp. Distributed Intelligence Syst.*, Arlington, VA., 1991, pp. 13-18.
[10] S. Eilenberg, *Automata, Languages, and Machines*, vol. A. New York: Academic, 1974.
[11] S. Lafortune and E. Chen, "The infimal closed controllable superlanguage and its application in supervisory control," *IEEE Trans. Automat. Contr.*, vol. 35, pp. 398-405, 1990.
[12] F. Lin and W. M. Wonham, "On the computation of supremal controllable sublanguages," in *Proc. 23rd Annual Allerton Conf. Communication, Contr., Computing*, Urbana, IL, 1985, pp. 942-950.
[13] M. Heymann, "Some algorithmic questions in discrete-event control," to appear.
[14] H. Cho and S. I. Marcus, "On supremal languages of class of sublanguages that arise in synthesis problem with partial observations," *Math. Contr. Sig. Syst.*, vol. 2, pp. 47-69, 1989.
[15] Y. Willner and M. Heymann, "On language convergence in discrete-event systems," in *Proc. 17th Conv. IEEE Israel*, Mar. 1991.
[16] R. Kumar, V. Garg, and S. I. Marcus, "On language stability of DEDS," in *Proc. Int. Conf. Mathematical Theory Contr.*, I.I.T. Bombay, Bombay, India, 1990.
[17] Y. Willner and M. Heymann, "Optimal language convergence in discrete-event control," to appear.
[18] R. Kumar, V. Garg, and S. I. Marcus, "Language stability and stabilizability of discrete-event dynamical systems," *SIAM J. Contr. Optim.*, vol. 31, no. 5, pp. 1294-1320, 1993.
[19] C. M. Özveren and A. S. Willsky, "Tracking and restrictability in discrete-event dynamical systems," *SIAM J. Contr. Optim.*, vol. 30, no. 6, pp. 1423-1446, 1992.



Yosef M. Willner received the B.Sc., the M.Sc., and the D.Sc. degrees in electrical engineering from Technion—Israel Institute of Technology, Haifa, Israel, in 1984, 1987, and 1992, respectively.

Since 1992, Dr. Willner has been a Research Associate in the Department of Education in Technology and Science, Technion. From 1993–1994, he was on leave at NASA—Ames Research Center, Moffet Field, CA, as an NRC Research Associate. His research interests include multivariable systems, adaptive control, discrete-event systems, hybrid systems, and graph algorithms.



Michael Heymann received the B.Sc. and the M.Sc. degrees from Technion—Israel Institute of Technology, Haifa, Israel, in 1960 and 1962, respectively, and the Ph.D. degree from the University of Oklahoma, Norman, in 1965, all in chemical engineering.

From 1965–1966, he was on the faculty of the University of Oklahoma. From 1966–1968, he was with Mobil Research and Development Corp., researching control and systems theory. From 1968–1970, he was with the Ben-Gurion University of the Negrev, Beer-Sheva, where he established and headed the Department of Chemical Engineering department. Since 1970, he has been with the Technion, where he is currently a Professor in the Department of Computer Science, holding the Carl Fecheimer Chair. He has previously been with the Department of Electrical Engineering and Chairman of the Department of Applied Mathematics. He held visiting positions at the University of Toronto, the University of Florida, the University of Eindhoven, Concordia University. CSIR, Yale University, the University of Bremen, and the University of Newcastle. From 1983–1984, 1988–1989, and during several summers he was an NRC-Senior Research Associate with NASA—Ames Research Center. His current research interests include discrete-event systems, hybrid systems, and the theory of concurrent processes.

Dr. Heymann is on the editorial board of *SIAM Journal of Control and Optimization*