On Language Convergence in Discrete-Event Systems

Yosef Willner
Department of Electrical Engineering
Technion — Israel Institute of Technology
Haifa 32000, Israel

Michael Heymann
Department of Computer Science
Technion — Israel Institute of Technology
Haifa 32000, Israel

1. Introduction

In the framework proposed by Ramadge and Wonham [1–3] a discrete-event system (DES) is modeled as a generator (finite automaton) that executes asynchronous state transitions. Normally, the control objective, in their framework, is to confine the behavior of the system to within a prescribed legal language. In some cases, however, one may want to specify only the eventual behavior of the system. That is, arbitrary initial behavior of the system is allowed, but eventually the behavior is required to converge to within a prescribed legal behavior.

In the present paper we use the formalism of ω-languages, that is, formal languages consisting of infinite strings, similar to [4], [5]. We say that the behavior of a DES P converges to a given ω-language E if after a finite number of state transitions, P executes only infinite strings that belong to E. We investigate the problem of synthesizing a supervisor under which the closed-loop system converges to E.

The paper is organized as follows. In section 2 we introduce some notations and properties of ω-languages. The property of convergence of an ω-language to another ω language is defined in section 3. Moreover, we discuss in this section the problem of synthesizing a supervisor that ensures the convergence of the closed-loop system behavior to a given ω-language. In appendix A we present some properties of realizable ω-languages.

2. Preliminaries

2.1 Notation and terminology

Let Σ^* denote the set of all finite strings, over a finite set Σ. We denote by Σ^ω the set of all infinite strings over Σ. Any subset $L \subset \Sigma^*$ is called a language over Σ and any $L \subset \Sigma^\omega$ an ω-language over Σ.

For $s \in \Sigma^*$ and $t \in \Sigma^* \cup \Sigma^\omega$, st is the concatenation of the strings s and t. For $L \subset \Sigma^* \cup \Sigma^\omega$, let $\mathrm{pr}(L)$ be the set of all prefixes of strings in L, that is

$$\mathrm{pr}(L) = \{\, t \in \Sigma^* \mid (\exists\, s \in \Sigma^* \cup \Sigma^\omega) ts \in L \,\},$$

and let $\mathrm{suf}(L)$ be the set all suffixes of strings in L, that is,

$$\mathrm{suf}(L) = \{\, t \in \Sigma^* \cup \Sigma^\omega \mid (\exists\, s \in \Sigma^*)\, st \in L \,\}.$$

Definition 2.1 An ω-language $L \subset \Sigma^\omega$ is called suffix closed if $\mathrm{suf}(L)=L$.

For a language $L \subset \Sigma^*$, the limit of L, denoted L^∞, is defined by

$$L^\infty = \{\, t \in \Sigma^\omega \mid (\exists\, s_i \in L\,, \exists\, u_i \in \Sigma^\omega) t = s_i u_i$$
$$\text{for infinitely many } i \,\}.$$

Definition 2.2 An ω-language L is called realizable if $[\mathrm{pr}(L)]^\infty = L$.

To model DES and to represent regular languages, we use finite automata of the form $A = (Q, \Sigma, \delta, Q_o)$, where Q is a finite set of states, Σ finite set of symbols, $\delta : \Sigma \times Q \to Q$ the transition function and Q_o a set of initial states. We denote by $L_f(A)$ the set of all finite strings which are accepted by A according to the standard acceptance rule where all the states of A are final states. Then we define the set of all infinite strings which are accepted by A as $L(A)=[L_f(A)]^\infty$.

2.2 Discrete event systems

We adopt the framework of Ramadge and Wonham [1–3]. Thus we assume that the discrete event system to be controlled is represented by an automaton

$$P = (Q, \Sigma, \delta, q_o).$$

The finite behavior of P is given by $L_f(P)$, and the infinite behavior of the system, namely the set of all infinite sequences of events that can be generated by P, is given by

$$L(P) = [L_f(P)]^\infty.$$

Namely, it is possible for a given infinite sequence of events to occur if and only if it is possible for every one of its initial subsequences to occur. We assume that the set Σ consists of two disjoint subsets, the set Σ_c of controlled events that can be disabled by a supervisor, and the set Σ_u of uncontrolled events that can not be disabled.

Let S be a supervisor, i.e. a device that disables at each instant a subset of the controlled events. Then S/P represents the closed-loop system. A supervisor S for P has been called nonblocking if

$$L_f(S/P) = \mathrm{pr}(L(S/P)).$$

This property ensures that every finite sequence of events generated by S/P can be extended to an infinite one.

The following result has been proved in [4] and [5].

Proposition 2.1 For a nonempty sublanguage $M \subset L(P)$, there exists a nonblocking supervisor S for P such that $L(S/P)=M$ if and only if

(a) M is controllable with respect to (w.r.t.) $L_f(P)$, i.e.: $\mathrm{pr}(M)\Sigma_u \cap L_f(P) \subset \mathrm{pr}(M)$; and

(b) M is realizable, i.e.: $[\mathrm{pr}(M)]^\infty = M$.

3. Language convergence in controlled DES

In this section we define the convergence of an ω-language to another ω-language. Then we discuss the problem of synthesizing a supervisor that ensures the convergence of the closed-loop system behavior to a given ω-language.

Let $M, L \subset \Sigma^\omega$ be given ω-languages. We shall say that L converges to M, denoted $M \Leftarrow L$, if there exists $i \in \mathbf{N}$ such that for each $t \in L$ there exist $s \in \Sigma^*$ and $u \in \Sigma^\omega$ that satisfy the following conditions.

(1) $t = su$, (2) $u \in M$, (3) $|s| \le i$,

where $|s|$ denotes the length (number of symbols) of s.

It is easily verified that the set of all ω-languages that converge to a given ω-language is closed under finite unions but it is not closed under infinite unions or under intersections.

Now we define the following stabilization problem (SP). Let $P=(Q,\Sigma,\delta,q_o)$ be a DES and let $E\subset\Sigma^\omega$ be a realizable ω-language, that we interpret as a specification for the required eventual behavior of P.

SP: Synthesize a nonblocking supervisor S such that $E \Leftarrow L(S/P)$.

Proposition 2.1, implies the following result.

Proposition 3.1 SP is solvable if and only if there exists a realizable and controllable (w.r.t. $L_f(P)$) sublanguage $M\subset L(P)$ such that $E \Leftarrow M$.

We introduce now algorithms for checking the solvability of the SP in the case that E is ω-regular. First we discuss the special case in which E is realizable and suffix closed (i.e.: $suf(E)=E$) and then we extend the result for the case that E is a general realizable ω-regular language.

From proposition A.1 (see appendix A), for the realizable ω-regular language E, there exists a deterministic automaton $A=(X,\Sigma,\xi,x_o)$ such that $L(A)=E$.

The following algorithm determines whether SP is solvable in the case that E is suffix closed. The algorithm translates the SP to the problem of pre-stabilization of P as defined in [6]. Specifically, the algorithm identifies a set T of states of P, such that the arrival of P at a state of T is equivalent to the convergence of the behavior of P to E. Then it uses the algorithm of [6] (see also [7]) to solve the new problem.

Algorithm 1

input: A *DES* $P=(Q,\Sigma,\delta,q_o)$ and a deterministic automaton $A=(X,\Sigma,\xi,x_o)$ such that $L(A)=E$, where E is suffix closed.

output: A condition for solvability of the problem SP.

(1) For each $q\in Q$, let $P_q=(Q,\Sigma,\delta,q)$ (namely, P_q is the system P initialized at q), and define

$$CR_q=\{F \mid F\subset L(P_q)\cap E \ ,F \text{ is realizable and controllable w.r.t. } L_f(P_q)\}.$$

Use algorithm A.1 (see appendix A) with input P_q and A, to construct a deterministic automaton A_q such that $L(A_q)=\sup CR_q$.

(2) compute the set T which is defined by

$$T=\{q\in Q \mid L(A_q)\neq\emptyset\}.$$

(Note that from proposition 2.1, for each $q\in T$, a supervisor can be synthesized that ensures that after P reaches q, the system generates only infinite strings belonging to E).

(3) By the algorithm of [6, proposition 3.5] check whether $q_o\in \mathrm{P}(T)$, where $\mathrm{P}(T)$ is the maximal set of states of P that are pre-stabilizable w.r.t. T. Namely the states from which the arrival of P at a state of T can be guaranteed by a supervisor.
stop.

Proposition 3.2 If E is suffix closed, then SP is solvable if and only if $q_o\in \mathrm{P}(T)$.

The complexity of constructing an automaton A_q for some $q\in Q$ is $0(n^2m^2)$, where $n=|Q|$ and $m=|X|$. The automaton A_q has at most nm states, thus the check whether $L(A_q)=\emptyset$ is $0(n^2m^2)$ computation, since it is required to determine whether there exists in A_q a cycle which is accessible from q. So the complexity of step (2) is $0(n^3m^2)$. The algorithm of [6] (see step (3)) is $0(n^2)$ computation. Thus the overall complexity of algorithm 2 is $0(n^3m^2)$.

To illustrate algorithm 1, let us examine the following example.

Example 3.1 Consider the system P and the recognizer A for E, as described in Fig. 3.1. In the system P, $\Sigma_c=\{\beta,\gamma\}$ and $\Sigma_u=\{\alpha,\delta\}$. Since $L(A)=(\alpha\beta)^\omega+(\beta\alpha)^\omega+\gamma(\alpha\beta)^\omega$ it is clear that $L(A)$ is suffix closed. We apply now algorithm 1.

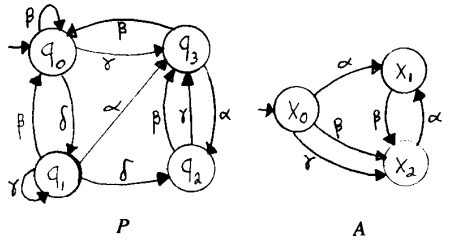


Fig. 3.1

step (1) Since in states q_o, and q_1 the uncontrolled transition δ is defined and $\delta\notin \mathrm{pr}(L(A))$,

$$\sup CR_{q_o}=\sup CR_{q_1}=\emptyset.$$

Moreover, $\sup CR_{q_2}=L(A)$, and $\sup CR_{q_3}=(\alpha\beta)^\omega$, thus

step (2) $T=\{q_2,q_3\}$.

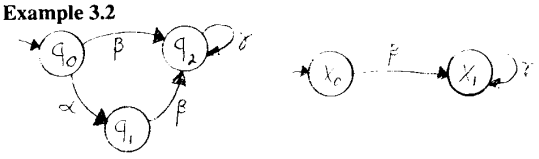
step (3) Note that the disablement of the event β at states q_o and q_1 and the event γ at q_1 ensures that the system, initialized at q_o or at q_1, reaches T. Thus $\mathrm{P}(T)=\{q_o,q_1,q_2,q_3\}$, and hence $q_o\in \mathrm{P}(T)$. So SP is solvable.

□

Remark In the case that SP is solvable, the supervisor S that ensures that $E\Leftarrow L(S/P)$ consists of a supervisor $S_1:(Q-T)\longrightarrow 2^{\Sigma_c}$ that ensures that P reaches T within a finite number of transitions and of supervisors S_q for each $q\in T$ that realizes the language $\sup CR_q$.

In case that E is not suffix closed, the problem SP is much more complex. To verify that the problem SP not can be translated to the problem of pre-stabilization of P, let us examine the following simple example.

Example 3.2



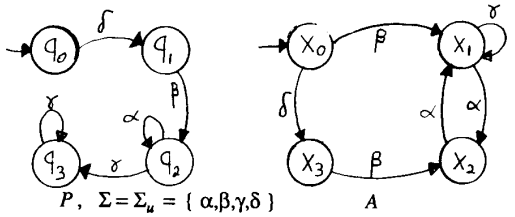
P, $\Sigma=\Sigma_u=\{\alpha,\beta,\gamma\}$ A

Clearly $E\Leftarrow L(P)$ but only the state q_1 satisfies that $L(P_{q_1})\subset E$ and P does not necessarily reach q_1.

In example 3.2, P reaches q_2 which satisfies $L(P_{q_2})\subset L(A_{x_1})$. One might conjecture that this property holds whenever $E\Leftarrow L(P)$. Namely, maybe a necessary condition for the convergence of $L(P)$ to $L(A)$ is that P reaches a state q such that

$L(P_q) \subset L(A_x)$ for some $x \in X$. If this were true then it would be sufficient to search all the pairs (q,x), where $q \in Q$ and $x \in X$, and to check the above condition. The complexity of such an algorithm would be polynomial in nm. The following example contradicts such an conjecture.

Example 3.3



P, $\Sigma = \Sigma_u = \{ \alpha,\beta,\gamma,\delta \}$ A

To verify that $L(A) \Leftarrow L(P)$, note that $L(P) = \delta\beta(\alpha^\omega + \alpha^*\gamma^\omega) = \delta\beta(\alpha^\omega + \alpha(\alpha\alpha)^*\gamma^\omega) + \delta\beta(\alpha\alpha)^*\gamma^\omega$. Since $\delta\beta(\alpha^\omega + \alpha(\alpha\alpha)^*\gamma^\omega) \subset L(A)$ and $\beta(\alpha\alpha)^*\gamma^\omega \subset L(A)$, it is clear that $L(A) \Leftarrow L(P)$.

Note that $L(P_{q_2}) = \alpha^*\gamma^\omega + \alpha^\omega$, whereas $\alpha\gamma^\omega \notin L(A_{x_1})$ and $\gamma^\omega \notin L(A_{x_2})$, thus $L(P_{q_2}) \not\subset L(A_x)$ for every $x \in X$. Similarly, it can be verified that only q_3 satisfies $L(P_{q_3}) \subset L(A_{x_1})$. But clearly, P does not necessarily reach q_3.

The following algorithm checks the convergence of P to $L(A)$ by constructing a new model (automaton) for the behavior of P, and by identifying in this model a set of states such that the arrival of P at this set is equivalent to the convergence to $L(A)$. Then it follows the lines of algorithm 1.

Algorithm 2

input: The automata $P = (Q,\Sigma,\delta,q_o)$ and $A = (X,\Sigma,\xi,x_o)$.

output: A condition for solvability of the problem SP.

(1) Construct a deterministic automaton $B = (V,\Sigma,\alpha,v_o)$ where $V = Q \times 2^X$, $v_o = (q_o,x_o)$ and $\alpha: \Sigma \times V \to V$ is defined as follows. Let χ be an element of 2^X, then

$$\alpha(\sigma,(q,\chi)) = \begin{cases} (\delta(\sigma,q),\chi') \text{ if } \delta(\sigma,q) \text{ is defined} \\ \text{undefined otherwise}, \end{cases}$$

where $\chi' = \{ x_o \} \cup \{ x \in X \mid (\exists\, x' \in \chi)\xi(\sigma,x') = x \}$.
Remark: Practically, it is sufficient to construct the accessible part of B, namely, the states of B that are accessible from v_o.

(2) For any $q \in Q$, let $P_q = (Q,\Sigma,\delta,q)$ and for any $\chi \in 2^X$, let $A_\chi = (X,\Sigma,\xi,x_o)$ (that is, the subscript denotes the initial states). For each $v = (q,x) \in V$, let $CR_v = \{ F \mid F \subset L(P_q) \cap L(A_\chi),\ F$ is realizable and controllable w.r.t. $L_f(P_q) \}$.
Use algorithm A.1 (see appendix A) to construct a deterministic automaton A_v such that $L(A_v) = \sup CR_v$.

(3) Compute the set T which is defined by

$$T = \{ v \in V \mid L(A_v) \neq \emptyset \}.$$

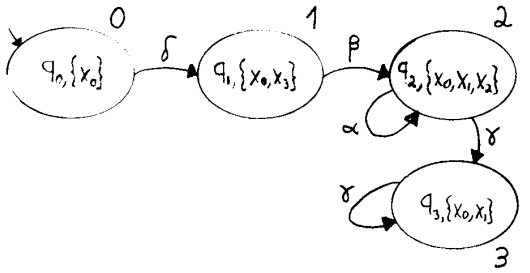
(4) By the algorithm of [6, proposition 3.5] check whether $v_o \in \mathrm{P}(T)$ and stop.

Proposition 3.3 SP is solvable if and only if $v_o \in \mathrm{P}(T)$.

The complexity of step (1) of algorithm (2) is $0(n \cdot 2^m)$. The number of states of the automaton B is bounded by $n \cdot 2^m$. For each state $v \in V$, the construction of the automaton A_v is $0((n \cdot 2^m)^2)$ computation, since it is required to construct a deterministic automaton that accepts $L(A_\chi)$ and then to use algorithm A.1. Such a deterministic automaton has at most 2^m states. The complexity of determining whether $L(A_v) = \emptyset$ is $0((n \cdot 2^m)^2)$. Thus the constructing of T is $0(n^3 2^{3m})$ computation. The algorithm of [6] in step (4) is $0((n\,2^m)^2)$ computation. Thus the overall complexity of algorithm 2 is $0(n^3 \cdot 2^{3m})$. This complexity is polynomial in the number of states of the system, but exponential in the number of states of the specification recognizer. Thus whenever the specification is described by a small automaton, this algorithm, is tractable.

We apply now algorithm 2 to example 3.2.

step (1) The automaton B



Each state $v = (q, \{ x_o, x_1, \cdots, x_k \})$ of B has the following property. Let $t \in \Sigma^*$ be a string such that $\alpha(t, v_o) = v$, then for every i, $1 \le i \le k$, there exist $t_i, s_i \in \Sigma^*$ such that $t = t_i s_i$ and $\xi(s_i, x_o) = x_i$. For example, the strings corresponding to a path ending at state 2 are $\delta\beta\alpha^i$. If i is even then the suffixes $\varepsilon, \beta\alpha^i$ and $\delta\beta\alpha^i$ satisfy that $\xi(\varepsilon, x_o) = x_o$, $\xi(\beta\alpha^i, x_o) = x_1$ and $\xi(\delta\beta\alpha^i, x_o) = x_2$, respectively. Whereas if i is odd then $\xi(\varepsilon, x_o) = x_o$, $\xi(\beta\alpha^i, x_o) = x_2$ and $\xi(\delta\beta\alpha^i, x_o) = x_1$, respectively.

Now if $v = (q,x) \in T$, then $\sup CR_v \neq \emptyset$ and hence a supervisor can ensure that all the infinite strings that P generate, after it reaches v, belong to $L(A_\chi)$. Thus such a supervisor ensures that any infinite string t generated by P, such that the infinite path generated by t entering v, belongs to $L(A)$. Namely if P reaches T then the convergence of $L(P)$ to $L(A)$ is guaranteed.

steps (2),(3)
Since in this example all the events of P are uncontrolled, $v = (q,x) \in T$ if and only if $L(P_q) \subset L(A_\chi)$. Thus it is easy to verify that

$$T = \{ 1,2,3 \}.$$

step (4) Clearly $\mathrm{P}(T) = \{ 0,1,2,3 \}$ and hence $v_o \in \mathrm{P}(T)$. Thus SP is solvable.

References

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

[2] Ramadge P.J. and Wonham W.M., "Modular feedback logic for discrete even systems," SIAM J. on Control and Optimization, 25(5), pp. 1202-1218, September 1987.

[3] Wonham W.M. and Ramadge P.J., "On the supremal controllable sublanguage of a given language," SIAM J. on Control and Optimization, 25(3), pp. 637-659, May 1987.

[4] Thistle J.G. and Wonham W.A., "On the synthesis of supervisors subject to ω-language specifications," 22nd Annual Conference on Information Sciences and Systems, Princeton, NJ, March 1988.

[5] Ramadge P.J., "Some tractable supervisory control problems for discrete event systems modeled by Büchi automata", IEEE Trans. on Automatic Control, 34(1), January 1989.

[6] Øsvern C.M., Willsky A.S. and Antsaklis P.J., "Stability and stabilizability of discrete event dynamic systems", to appear in the Journal of ACM.

[7] Brave Y. and Heymann M., "On stabilization of discrete event processes", Int. J. Control, Vol. 51, No. 5, pp. 1101-1117, 1990.

[8] Li Y. and Wonham W.A., "Deadlock issues in supervisory control of discrete event systems", 22nd Annual Conference on Information Sciences and Systems, Princeton, NJ, March 1988.

Appendix A: Realizable and controllable ω-regular languages

The first proposition characterizes the realizable ω-regular languages.

Proposition A.1 For any ω-regular language $M \subset \Sigma^w$, the following conditions are equivalent.

(a) M is realizable

(b) There exists a deterministic automaton $A = (X, \Sigma, \xi, x_o)$ such that $L(A) = M$.

□

Now we introduce an algorithm for computing the supremal realizable and controllable sublanguage of a given realizable ω-language. The importance of this sublanguage follows from proposition 2.1.

Let $P = (Q, \Sigma, \delta, q_o)$ be a DES and let $A = (X, \Sigma, \xi, x_o)$ be a deterministic automaton such that $L(A) = E$. Define $CR(E)$ to be the set of all realizable and controllable sublanguages of $E \cap L(P)$, that is

$$CR(E) = \{ F \mid F \subset E \cap L(P),$$
$$F \text{ is realizable and controllable w.r.t. } L_f(P) \} .$$

The following algorithm computes $\sup CR(E)$, this algorithm has been used in [8] for solving a different problem.

Algorithm A.1
input: The automata $P = (Q, \Sigma, \delta, q_o)$ and $A = (X, \Sigma, \xi, x_o)$ (such that $L(A) = E$).
output: A recognizer for $\sup CR(E)$.

(1) Construct the product automaton

$$P \times A = (Q \times X, \Sigma, \alpha, (q_o, x_o)) ,$$

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

$$\alpha(\sigma, (q,x)) = \begin{cases} (\delta(\sigma,q), \xi(\sigma,x)) & \text{provided both } \delta(\sigma,q) \text{ and } (\sigma,x) \text{ are defined} \\ \text{undefined} & \text{otherwise} \end{cases}$$

(2) Define $Y_o = Q \times X$,
iterate until $Y_j = Y_{j+1}$
For each $y = (q,x) \in Y_j$,
$y \in Y_{j+1}$ iff

(i) $(\exists\, \sigma \in \Sigma)\ \alpha(\sigma,y)$ is defined and $\alpha(\sigma,y) \in Y_j$; and

(ii)
$(\forall\, \sigma \in \Sigma_u)\delta(\sigma,q) = q' \Rightarrow \alpha(\sigma,(q,x)) = (q',x') \in Y_j$.

(3) Let Y be the set of states defined in step (2) at termination of step (2).
Define $B = (Y, \Sigma, \alpha, (q_o, x_o))$, and stop.

Proposition A.2 $L(B) = \sup CR(E)$.