A New Scheme for the Dynamic Control of Discrete Event Systems

JIANG Zhiping
(Department of Automation, Shanghai Jiaotong University · Shanghai, 200030, PRC)
WANG Liya
(CIM Research Institute, Shanghai Jiaotong University · Shanghai, 200030, PRC)
WU Zhiming
(Department of Automation, Shanghai Jiaotong University · Shanghai, 200030, PRC)

Abstract: For a given discrete event system DES G and a prescribed maximally permissive legal language specification MPLLS K, the mission for supervisory control is to find a supervisor S such that the system under supervision will never breach the MPLLS, i.e., $L(S/G) \subseteq K$. The traditional method to realize this involves the computation of so-called supremal controllable sublanguage $K^{\uparrow}$ of the given MPLLS K and thus the explicit construction of S is unavoidable. This paper exhibits a new approach to the problem which can alleviate the necessity both to compute $K^{\uparrow}$ and to construct S explicitly. Instead, the supervisor will be dynamically realized with the evolution of the DES G and the prescribed MPLLS K. To a large extent, the approach is base on an on-line set inclusion test algorithm so it will be proved to be useful in real applications, which is enhanced by an example towards the end of the paper.

Key words: discrete event systems; automaton; language; supervisory control; algorithm; on-line control

1 Introduction

The previous work on the control of discrete event systems (DES) by a language model is primarily focused on the controllability of a prescribed legal sublanguage K of the given system DES G or the existence of the required supervisor S[1]. Attention is then put on the computational aspects of the so-called supremal controllable sublanguage $K^{\uparrow}$ if the given sublanguage K of $L(G)$ would be uncontrollable[2]. It has been proved that the computation of $K^{\uparrow}$ could be finished in a polynomial-time complexity when the system is fully observable[3,4]. To real applications, however, given a system DES G and a prescribed maximally permissive legal language specification MPLLS K, we will still have to compute $K^{\uparrow}$ at first and then assign controls for each of its legal sequences of the language $L(G)$ which belongs to $K^{\uparrow}$ and thus a real supervisor S has to be constructed with a large number of states and event transitions, such a construction of S will appear uneconomic.

This paper proposes a dynamic realization approach to the above supervisory problem such that there is no need to construct the supervisor explicitly while the control task is implemented by means of reference to the MPLLS K for every newly arrived state during

Manuscript received Feb. 13. 1996, revised Oct. 8, 1996.

the evolution of the DES G. The approach alleviates the potential combinatory explosion problem of states or transitions even for a large-scale automaton-modeled DES and/or its correspoding MPLLS K. We assume that our discussions is confined to the closed language of $L(G)$ but it is not difficulty to extend our result to a more general situation. However, it is not necessary to make assumption of the containment between $L(G)$ and K, which represents the case that the MPLLS K could be arbitrarily prescribed with independence of its underlying DES.

2 Some Notations

We use $G(q) = (Q, \Sigma, \delta)$ to denote a DES modelled as a state machine in which Q is its state set, Σ is its event set and is usually considered to be composed of a controllable part Σ_c and an uncontrollable part Σ_u, $\delta: \Sigma \times Q \to Q$ is its state transition function and $q \in Q$ is an initial state, Let A be a subset of Σ, we then use $G(A, q)$ to denote a partial DES in which δ is only defined on $A \times Q$.

In comparison with the traditional expression of automata (see, e. g. , [1] or [5]), the automaton defined here emphasizes the variability of its intitial state and/or its event symbol set, Thus, for a given DES $G(A, q_i)$, there may be as many as $|Q_{(A,q_i)}|$ induced automata of the form of $G(A, q)$ where $Q_{(A,q_i)}$ stands for a subset of Q induced from q_i through $A \subseteq \Sigma$ and $q \in Q_{(A,q_i)}$ is called A- reachable from q_i through transitions all labelled by event symbols in A. Let $F(A, q_i)$ be the family of automata induced by $G(A, q_i)$. Then we have a formal definition of accessibility below.

Definition 2. 1 For a given DES $G(q_i) = (Q, \Sigma, \delta)$, $G(q_i)$ is called accessible if $|F(\Sigma, q_i)| = |Q|$ and A- acessible if $|F(A, q_i)| = |Q|$.

From this definition, it is obvious that $G(q_i)$ is accessible **iff** $G(q_i)$ is Σ-accessible. It is also noted that if q_j is A -reachable from q_i then that $G(q_i)$ is A -accessible implies that $G(q_j)$ is A -accessible, but not necessarily vice versa.

When $G(q)$ is not A -accessible, then there will arise two disjoint subsets in Q , namely, $Q_{(A,q)}$ and $Q - Q_{(A,q)}$, among which $G'(q) = (Q_{(A,q)}, \Sigma, \delta)$ is A -accessible. Analogously, when $G(q)$ is not accessible $G'(q) = (Q_{(\Sigma,q)}, \Sigma, \delta)$ stands for the accessible part in $G(q)$. If $G(q)$ is not accessible, then none of the states in $Q - Q_{(\Sigma,q)}$ can be reached from q. Therefore, for a specific deterministic automata family $F(\Sigma, q_0)$, we will always assume that $G(q_0)$ is accessible.

For a given $G(A, q)$, a string $s = \sigma_1 \cdots \sigma_k$ is a sequence of events which starts at q_i and ends up at $\delta(s, q)$ such that σ_i belongs to A for $i = 1, \cdots, k$. Sometimes, we use $\delta(s, q)!$ or $\neg \delta(s, q)!$ to denote that $\delta(s, q)$ is defined or not defined. For two strings s and t, s is called a substring of t, denoted as $s \preceq t$, if either $s = t$ or there is another string $u \in L(G(A, \delta(s, q)))$ such that $su = t$. Thus, $sL(G(A, \delta(s, q)))$ is defined to be the set of all the strings such that s can be their substring, or formally,

$$sL(G(A, \delta(s, q))) = \{t : s \preceq t\},$$

where we assume that $L(G(A, q))$ is well defined according to

$$L(G(A, q)) = \{s : s \in A^* \wedge \delta(s, q)!\}.$$

Definition 2.2 A given sublanguage K of $L(G(q))$ is called controllable if the following relation is satisfied:

$$\forall\, s \in K, sL(G(\Sigma_u, \delta(s,q))) \subseteq K. \tag{1}$$

For an arbitrarily given language K, K is not necessarily controllable in a sense that (1) need not be satisfied. In that case, we are interested in getting its supremal controllable sublanguage $K^{\uparrow}$ (See [2]). The following Proposition is regarding the computation of $K^{\uparrow}$ which is stated by the terminology we defined in this paper.

Lemma 2.3 Given a DES $G(q)$ and a MPLLS K, the supremal controllable sublanguage $K^{\uparrow}$ can be calculated according to

$$K^{\uparrow} = \bigcup_{t \in K \cap L(G(q))} \{s : s \preceq t \wedge sL(G(\Sigma_u, \delta(s,q))) \subseteq K\}. \tag{2}$$

Proof We first prove that $K^{\uparrow}$ as defined in (2) is controllable, namely, $\forall\, s \in K^{\uparrow}$, $sL(G(\Sigma_u, \delta(s,q))) \subseteq K^{\uparrow}$ holds.

Suppose on the contrary that $\exists\, s \in K^{\uparrow}$, but $sL(G(\Sigma_u, \delta(s,q))) \not\subseteq K^{\uparrow}$. By definition of $K^{\uparrow}$, $sL(G(\Sigma_u, \delta(s,q))) \subseteq K$ and $K^{\uparrow} \subseteq K$. Thus, there must exists some string $s' \in \Sigma_u^*$ such that $ss' \notin K^{\uparrow}$ while $ss' \in K$ and $ss'L(G(\Sigma_u, \delta(s',q))) \subseteq K$ both hold. On the other hand, by (2) $ss' \in K$ and $ss'L(G(\Sigma_u, \delta(ss',q))) \subseteq K$ implies $ss' \in K^{\uparrow}$, which forms a contradiction.

Next we show that the $K^{\uparrow}$ so defined is a unique supremal controllable sublanguage of K. Suppose that for a given string $s \in K$ such that $sL(G(\Sigma_u, \delta(s,q))) \subseteq K$ holds. Then since $\delta(s,q)$ is defined, $s \in K \cap L(G(q))$ holds. Noticing that $s \preceq s$, we conclude that $s \in K^{\uparrow}$ holds. Q. E. D.

A supervisor S is a map to 2^{Σ_c} from every allowed string s in $L(G(q_0))$ which is restricted by the MPLLS K, namely,

$$S(s) = \{\sigma : \sigma \in \Sigma_c\} \text{ if } s\sigma L(G(\Sigma_u, \delta(s\sigma, q_0))) \not\subseteq K. \tag{3}$$

For a given DES $G(q_0)$ and a MPLLS K, if a supervisor is synthesized according to (3), then the system under supervision, called SDES and denoted as $S/G(q_0)$, wil behave by the following rule:

$\forall\, s \in L(S/G(q_0))$ and $\sigma \in \Sigma, s\sigma \in L(S/G(q_0))$ **iff** both $s\sigma \in L(G(q_0))$ and $\sigma \notin S(s)$.

A supervisor S for a DES $G(q)$ is called to be minimally restrictive w. r. t. a MPLLS K if K for any other supervisor S', $L(S'/G(q)) \subseteq L(S/G(q))$. It has been well known that for a given DES $G(q)$ and a MPLLS K, if S is its manimally restrictive supervisor, then $K^{\uparrow} = L(S/G(q))$. (See [4] and [2])

3 Toward Dynamic Supervision

From the preceding section, we know that at any moment the behaviors of a DES $G(q)$ can be characterized by means of its occurred event string s and the language $L(G(s,q))$ produced following the s. We also observe that the SDES $S/G(q)$ under a MPLLS K can be maximally implemented through computation of $K^{\uparrow}$. Now assume that the given MPLLS K could also be represented as a deterministic and accessible automaton $G_K(q_0') = (Q', \Sigma, \delta')$. Then from Lemma 2.3 we have

Proposition 3.1 Given a DES $G(q_0)$ and a MPLLS $L(G_K(q_0'))$, the supremal control-

lable sublanguage $K^{\uparrow}$ can be calculated according to

$$L(G_K(q_0'))^{\uparrow} = \bigcup_{t\in K\cap L(G(q_0))} \{s : s \preceq t \wedge L(G(\Sigma_u,\delta(s,q_0))) \subseteq L(G_K(\Sigma_u,\delta'(s,q_0')))\}. \quad (4)$$

Proof Immediate from Lemma 2.3. Q.E.D.

Thus, for a given DES $G(q_0)$ and a MPLLS $L(G_K(q_0'))$, to test if a string s is allowed to occur in $L(G_K(q_0))$ such that it is controllable and is allowed by $L(G_K(q_0))$ could be reduced to test if $L(G(\Sigma_u,\delta(s,q_0)) \subseteq L(G_K(\Sigma_u,\delta'(s,q_0'))$ holds. To be concrete, the following algorithm presents a procedure inclusion to make such inclusion test for a given pair of automata $G(q)$ and $G'(q')$.

Algorithm 3.2 Inclusion $(G(q),G'(q'))$

Input: $G(q) = (Q,\Sigma,\delta)$ and $G'(q') = (Q',\Sigma,\delta')$.

Output: 1 if $L(G(\Sigma_u,(q)) \subseteq L(G'(\Sigma_u,(q'))$, and 0 otherwise.

Comment: lable(), push() and pop() are three pre-defined procedures.

1) label (q) and label (q'); push (q,q');

2) **do** while pop (a,b);

3) **for** each $\sigma \in \Sigma_u$;

4) **if** $\delta(\sigma,a)! \wedge \neg\delta'(\sigma,b)!$ **then exit**(0);

5) **else**;

6) **if** $(\delta(\sigma,a)! \wedge \delta(\sigma,b)!)$ but not both $\delta_1(\sigma,a)$ and $\delta_2(\sigma,b)$ were labelled;

7) **then begin** label $(\delta(\sigma,a))$; label $(\delta'(\sigma,b))$; push $(\delta(\sigma,a)!,\delta'(\sigma,b))$ **end**;

8) **exit**(1).

Some remarks on the algorithm are in order. We note first that the algorithm will finish its inclusion test in the worst case with the complexity of $O(|Q \| Q' \| \Sigma_u|)$. In other words, if the $G(q)$ and $G'(q')$ are selected to be regular then the inclusion test can be accomplished in a polynomial-time bound. Secondly, from line 4 of the algorithm, we know that the algorithm indeed makes comparison between $G(q)$ and $G'(q')$ for every string in $G(q)$ except those that there are no corresponding strings in $G'(q')$, in which case the conclusion of $L(G(\Sigma_u,q)) \nsubseteq L(G'(\Sigma_u,q'))$ is obtained. It is worthnoting that once a string in $G(q)$ is found not included in $G'(q')$, no further comparison is needed. Therefore, upon completion of the algorithm, there may be three situations, i.e., $L(G(\Sigma_u,q)) \subseteq L(G'(\Sigma_u,q'))$ when exiting as "1", or $L(G(\Sigma_u,q)) \supset L(G'(\Sigma_u,q'))$ when exiting as "0" or $L(G(\Sigma_u,q)) \neq L(G'(\Sigma_u,q'))$ when exiting still as "0".

Now that we have developed an algorithm for set inclusion test, we can proceed to realize dynamic supervision for a given DES $G(q_0)$ and a MPLLS K as follows. For every controllable event at any state of the system, we investigate in advance if this event occurring will lead the system to any unrequired state. If this is the case, then the event is disabled. First we note that if $L(G(\Sigma_u,q_0)) \nsubseteq L(G_K(\Sigma_u,q_0'))$, then, no control mechanism can ensure that the system would behave definitely within the realm of MPLLS K, which can be stated as a lemma below.

Lemma 3.3 Given a DES $G(q_0)$ and a MPLLS K. There is a non-trivial supervisor S

such that $L(S/G(q_0)) \neq \varnothing$ only if $L(G(\Sigma_u, q_0)) \subseteq L(G_K(\Sigma_u, q_0'))$.

It is worthnoting that the condition for non-trivial supervisor S in Lemma 3.3 is only necessary but not sufficient. This is because we could not infer from its intial state q_0' of G_K that there exists at least subsequent state, say $\delta'(\sigma, q_0')$ with $\sigma \in \Sigma_c$, such that both $\delta(\sigma, q_0)$ is defined and $L(G(\Sigma_u, \delta(\sigma, q_0))) \subseteq L(G_K(\Sigma_u, \delta'(\sigma, q_0')))$ holds. On the other hand, there always exists a minimally restrictive supervisor S that makes $L(S/G(q_0)) = K^{\uparrow}$. Thus we have

Lemma 3.4 There exists a non-trivial supervisor S for a given DES $G(q_0)$ and a MPLLS K **iff** $K^{\uparrow} \neq \varnothing$.

Thus, the minimally restrictive supervisor could be realized dynamically with the evolotion of the system such that for every occurred event that is allowed by both G and K, we disable in advance all the controllable events that may follow the event, The following algorithm formally summarized the procedure of dynamic supervision.

Algorithm 3.5

Input: $G(q_0)$ and $G_K(q_0')$.

Output: Legal behaviors of $K^{\uparrow}$

1) $q \leftarrow q_0, q' \leftarrow q'_0$;

2) **if** inclusion $(L(G(\Sigma_u, q)), L(G_K(\Sigma_u, q'))) = 0$ **then stop**;

3) $S(q) = \begin{cases} \{\sigma : \sigma \in \Sigma_C\}, & \text{if inclusion } (L(G(\Sigma_u, q)), L(G_K(\Sigma_u, q'))) = 0, \\ \varnothing, & \text{otherwise;} \end{cases}$

4) Wait for and supervise the state transition of G from q to $\delta(\sigma, q)$ with $\sigma \notin S(q)$;

5) $q \leftarrow \delta(\sigma, q)$; $q' \leftarrow \delta(\sigma, q')$; **goto** 3).

It should be pointed out that the above algorithm would take in the worst case the time of $O(|\Sigma_C \| Q \| Q' \| \Sigma_u|)$ for every step of on-line supervision, which is still solvable in a polynomial-time complexity.

4 A Case Illustration

A simplified salt conveying system is shown in Fig. 1 where two different kinds of salt are continualy transported through two Conveyers A and B, three push-open Valves labeled V_1 through V_3 control the flow of the salts to Silo 1 and 2 and Junk is used to load the salts that can not be further poured into the Silos.

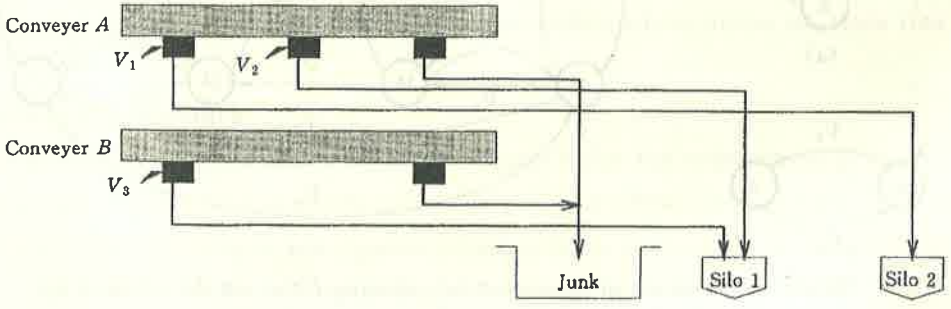

Fig. 1 A simple salt conveying system

Suppose that the Silos will be automatically closed the corresponding Valves once they are poured full and keep closed until the Silos get emptied. Henceforth, for example, the salt

in Conveyer A will automatically be transported to the Junk during Silos 1 and 2 are not empty and have the same kind of salt as on Conveyer A, which implies Valves 3 has to be prevented from opening. On the other hand, Valves 3 during the above period may be controlled open since Silo 1 might have undergone the process that empties the salt of the same kind as on Conveyer B, which as supposed would make Valves 3 controllable.

The task for the control of such a system is twofold. The first is to prevent different kinds of salt from being intermixed in the same Silo during the same period of time when the Silo is receiving one kind of salt. The second is to make full use of the two Silos so that as little salt as possible be poured into Junk where different kinds of salt are allowed to be mixed but will hence become useless.

Let V_i denote an event to open the ith Valve which is controllable whereas $\overline{V}_i$ denotes an uncontrollable event to close the ith Valve. The possible states of the Conveyer A are:

0) both Valves 1 and 2 are closed and the salt is poured into Junk;

1) Valve 1 is open and Valve 2 still closed and the salt goes into Silo 1;

2) Valve 1 is closed and Valve 2 is open and the salt goes into Silo 2;and

3) both Valves 1 and 2 are open, when salt goes into Silo 1.

Similarly, the possible states of the Conveyer B are;

a) Valve 3 is closed and the salt is poured into Junk;and

b) Valve 3 is open and the salt goes into Silo 1.

The states transitions concerning Conveyer A in response to the potential events V_i and $\overline{V}_i$ for $i = 1, 2$ are illustrated in Fig. 2(a) and the states transitions concerning Conveyer B in response to events V_3 and $\overline{V}_3$ are illustrated in Fig. 2(b). The combined behavior of the system G can then be shown in Fig. 2(c).

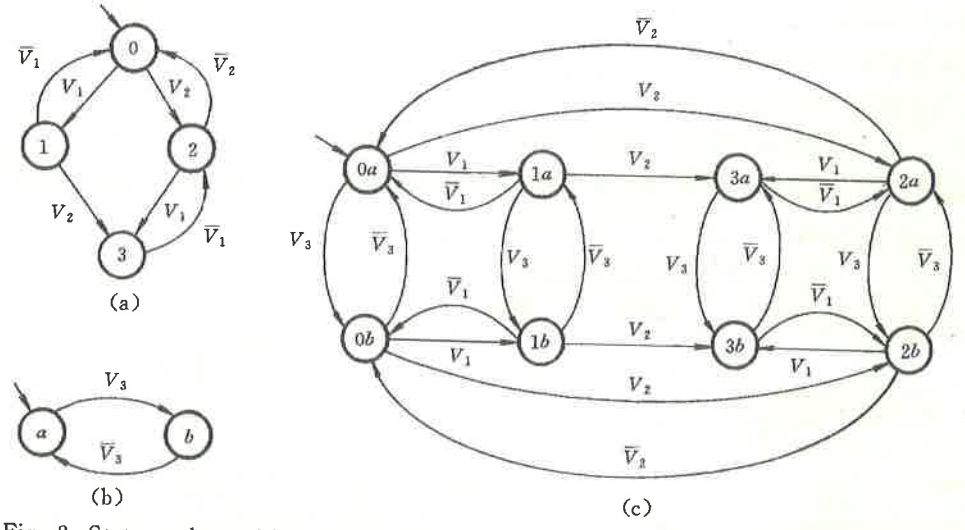

Fig. 2 States and transitions of conveyer A (a), conveyer B (b) and the system G (c)

It is noticed that once a valve is open, it can be assumed to be open again before it is uncontrollably closed. Thus one possible MPLLS K for the system may be as shown in Fig. 3 where we assume that some of the transitions are improperly specified.

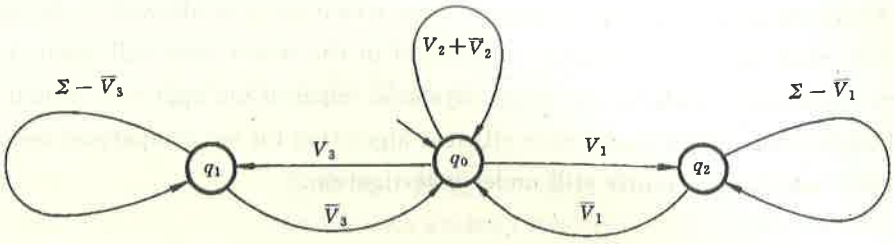

Fig. 3 the automaton of G_K for the MPPLS K which might be improperly specified.

Applying our dynamic supervison algorithm (Algorithm3. 5), we can conclude that the following inclusion relations are true:

$$L(G(\Sigma_u,0a)) = \varnothing \subseteq L(G_K(\Sigma_u,q_0)); \quad L(G(\Sigma_u,0b)) = \{\overline{V}_3\} \subseteq L(G_K(\Sigma_u,q_1));$$

$$L(G(\Sigma_u,2b)) = \{\overline{V_2V_3}\} \subseteq L(G_K(\Sigma_u,q_1)); L(G(\Sigma_u,2a)) = \{\overline{V}_2\} \subseteq L(G_K(\Sigma_u,q_0));$$

$$L(G(\Sigma_u,1a)) = \{\overline{V}_1\} \subseteq L(G_K(\Sigma_u,q_2)); \quad L(G(\Sigma_u,3a)) = \{\overline{V}_1\} \subseteq L(G_K(\Sigma_u,q_2)).$$

Analogously, we can readily verify following non-inclusion relations:

$$L(G(\Sigma_u,1b)) \nsubseteq L(G_K(\Sigma_u,q_1)); \quad L(G(\Sigma_u,1b)) \nsubseteq L(G_K(\Sigma_u,q_2));$$

$$L(G(\Sigma_u,3b)) \nsubseteq L(G_K(\Sigma_u,q_1)); \quad L(G(\Sigma_u,3b)) \nsubseteq L(G_K(\Sigma_u,q_2)).$$

Therefore, following the dynamic supervision, we can eventurally achieve the control objective which is both within the K and minimally restrictive as well. Indeed, the SDES $S/G(0a)$ for this simple example can be demenstrated in Fig. 4. To be clear, we have denoted each state of the automaton with a pair of states in G and G_K respectively such that for each pair (a,b) in the automaton, the relation $L(G(\Sigma_u,a)) \subseteq L(G_K(\Sigma_u,b))$ must be true. It should be noticed that if a real supervisor would be constructed, the state space for the supervisor will have to be as large as the number of whole possible such inclusion relations that truely hold.

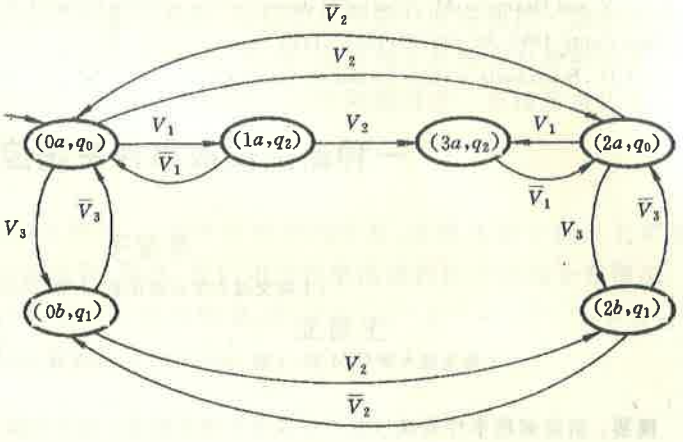

Fig. 4 the automaton for $K^\uparrow$ w. r. t. $L(G)$

5 Concluding Remark

This paper investigated a new dynamic approach for the supervision of a given DES $G(q_0)$ and a MPLLS K. The approach is attractive since it alleviates the necessity to explicitly construct a supervisor as was suggested in many other literatures. As has been seen, the dynamic supervision scheme is by and large base on a set inclusion test algorithm (Algorithm3. 2) which has been informally explained to be of polynomial-time complexity to the cardinalities of states sets of both $G(q)$ and $G_K(q')$ as well as the cardinality of event set Σ. However, when the system and/or the the MPLLS could be specified with some economic

formalism (e. g. , hierarchical state machine[6,7]) in which state is allowed to be folded expressed, direct application of Algorithm 3. 2 would in the worst case still result in a state combinatory explosion problem. Thus our dynamic supervision approach would find its widespread application only if some more efficient algorithm for set comparison test could be fully developed, which is currently still under investigation.

References

1 Ramage, P. R. and Wonham, W. M. . Supervisory control of a class of discrete event processes. SIAM J. Control and Optimization, 1987, 25(1): 206—230

2 Wonham, W. M. and Ramage, P. J. . On the supremal controllable sublanguage of a given language. SIAM J. Control and Optimization, 1987, 25(3): 637—659

3 Tsitsikils, J. N. . On the control of discrete event dynamical systems. Mathematics of Control. Signals and Systems, 1989, 2(1): 95—107

4 Ramage, P. R. and Wonham, W. M. . The control of discrete event systems. Proc. IEEE Special Issure on Discrete Event Systems, 1989, 77(1): 81—98

5 Eilenberg, S. . Automata, Languages and Machines. Vol. A. New York: Academic Press, 1974

6 Brave, Y. and Heymann, M. . Control of discrete event systems modeled as hierarchical state machines. IEEE Trans. Automat. Contr. 1993, AC-38(12): 1803—1819

7 Harel, D. . Statecharts: a visual formalism for complex systems. Science of Computer Programming, 1987, 8(3): 231—274

一种新的离散事件系统控制方案

蒋智平

(上海交通大学自动化系，上海，200030)

王丽亚 (上海交通大学 CIM 所，上海，200030)

吴智铭 (上海交通大学自动化系，上海，200030)

摘要：给定离散事件系统(DES) G 及某个规定的最大允许合法语言说明(MPLLS) K，监督控制的任务就是寻找一个监控器 S，使得系统在监控下不会突破该 MPLLS，也即满足：$L(S/G) \subseteq K$. 传统的实现方法要求给定 MPLLS K 计算其最大可控子语言 $K^{\uparrow}$，因此难免要显式地构造 S. 本文给出一种新的方案可以不必对 $K^{\uparrow}$ 和 S 作显式计算. 监控任务是随 DES G 的演化结合规定的 MPLLS K 用动态方法实现. 该方案在很大程度上依赖一个在线集合包含测试算法. 文章末尾给出的一个实例说明了该方案在实际应用中的有用性.

关键词：离散事件系统；自动机；语言；监督控制；算法；在线控制

本文作者简介

蒋智平 1959 年生. 副教授. 1992 年获上海交通大学博士学位，1994 年至 1995 年在以色列理工学院做博士后研究. 主要从事计算机科学和应用研究，离散事件系统建模与控制，以及 CIM 方面研究.

王丽亚 1960 年生. 副教授. 1993 年获上海交通大学博士学位，1993 年至 1995 年在上海交通大学做博士后研究. 主要从事计算机科学和应用研究，离散事件系统研究和 CIM 研究.

吴智铭 1939 年生. 教授. 博士生导师. 1958 年毕业于交通大学电机与电器制造专业. 主要从事离散事件系统分析及其在柔性制造系统和 CIMS 中的应用.