Copyright © 2002 IFAC
15th Triennial World Congress, Barcelona, Spain

OVERLAPPING DECOMPOSITIONS OF LARGE–SCALE DISCRETE–EVENT SYSTEMS

Aydın AYBAR and **Altuğ İFTAR**

Department of Electrical and Electronics Engineering
Anadolu University
26470 Eskişehir, Turkey

aaybar@anadolu.edu.tr *aiftar@anadolu.edu.tr*

Abstract: Overlapping decompositions and expansions of discrete–event systems (DESs) modeled by automata or formal languages are considered. Inclusion principle for such systems is defined. A decentralized supervisory controller design approach is then introduced. To apply the proposed approach, the automaton of the given DES is first decomposed overlappingly and expanded to obtain disjoint subautomata. A controller is then designed for each disjoint subautomaton. These controllers are then combined to obtain a controller for the expanded DES. In the final phase, a controller for the original DES is obtained from the controller determined for the expanded DES. It is shown that, for large–scale DESs, the proposed approach requires very little computation to design a controller compared to a centralized design approach. *Copyright © 2002 IFAC*

Keyword: Large–scale systems, discrete–event systems, decentralized control, supervisory control, automata, formal languages

1. INTRODUCTION

Many systems, such as manufacturing systems, communication systems, and transportation systems, may be described by occurrence of certain events. Two most common modeling approaches for these systems, which are commonly named as discrete-event systems (DESs), are *automata* and *formal languages* (Ho, 1992; Cassandras and Lafortune, 1999). Ramadge and Wonham (1989) initiated supervisory control theory for such systems. Decentralized supervisory control is investigated in (Ramadge and Wonham, 1989) by using partial observations and in (Lin and Wonham, 1990) and in (Rudie and Wonham, 1992) by using local controllers for global constraints. Mixed centralized/decentralized approaches have also been considered (Cho and Lim, 1999).

In this work, we consider the problem of designing a supervisory controller for a DES, modeled by an automaton or equivalently by a language, to avoid *deadlock*. A deadlock is said to occur in a DES if the system reaches to a state such that no event can occur. This problem is a special case of designing a controller to avoid certain states. The dual problem of designing a controller to lead the DES to certain states could also be considered (Ramadge and Wonham, 1989). Deadlock avoidance for flexible manufacturing systems is studied by Banaszak and Krogh (1990). Deadlock problem is also dealt for DESs modeled by Petri nets (e.g., see Aybar and İftar, 2001, and references there in).

Here, we introduce an alternative approach to design decentralized supervisory controllers. Our approach is based on *overlapping decompositions*. The overlapping decompositions approach was first introduced by Ikeda and Šiljak (1980) for the case of continuous-state systems (systems described by differential or difference equations with continuous state variables). This approach was then used for DESs by İftar and Özgüner (1998), who considered a state vector modeling, and by Aybar and İftar (2002), who considered Petri net modeling. To our best knowledge, this approach is used here for the first time for DESs modeled by automata or formal languages.

In the automata approach, a DES is modeled by an automaton, which is represented as a tuple $\mathcal{A} = (Q, \Sigma, \gamma, q_0)$. Here, Q is the set of states, Σ is the set of events, $\gamma : \Sigma \times Q \to Q$ is a partial function, called the *transition function*, defined for some $q \in Q$ and $t \in \Sigma$, and $q' = \gamma(t, q)$ denotes the next state when event t occurs at state q ($\gamma(t, q)$ is defined if and only if event t may occur at state q). Finally, $q_0 \in Q$ denotes the initial state of the system. The same DES can equivalently be represented by a language $L = \{s \in \Sigma^* \mid \gamma(s, q_0) \text{ is defined}\}$ where Σ^* is the set of all finite strings of elements of Σ plus the empty event ϵ, and $\gamma : \Sigma^* \times Q \to Q$ is an extension of $\gamma : \Sigma \times Q \to Q$ defined above.

The elements of Σ can be decomposed into two partitions as controllable and uncontrollable events, $\Sigma = \Sigma_c \cup \Sigma_{uc}$. Here, Σ_c is the set of controllable events and Σ_{uc} is the set of uncontrollable events. Elements of Σ_c can be disabled at any time whereas elements of Σ_{uc} can not be disabled.

2. INCLUSION PRINCIPLE

Inclusion principle provides the theoretical framework for controller design using overlapping decompositions (Ikeda and Šiljak, 1986). In this section, we will extend this principle to DESs modeled by automata or formal languages.

Let us consider two systems $\mathcal{S}$ and $\tilde{\mathcal{S}}$, where $\mathcal{S}$ is defined by the automaton $\mathcal{A} = (Q, \Sigma, \gamma, q_0)$, or equivalently by the language $L = \{s \in \Sigma^* \mid \gamma(s, q_0) \text{ is defined}\}$, and $\tilde{\mathcal{S}}$ is defined by the automaton $\tilde{\mathcal{A}} = (\tilde{Q}, \tilde{\Sigma}, \tilde{\gamma}, \tilde{q}_0)$, or equivalently by the language $\tilde{L} = \{\tilde{s} \in \tilde{\Sigma}^* \mid \tilde{\gamma}(\tilde{s}, \tilde{q}_0) \text{ is defined}\}$.

Definition 1: $\tilde{\mathcal{S}}$ *includes* $\mathcal{S}$, if there exist transformations $\Theta : \tilde{Q} \to Q$ and $H : Q \to \tilde{Q}$ which satisfy the following conditions.

(i) $\Theta(H(q)) = q, \ \forall q \in Q$
(ii) $\tilde{q}_0 = H(q_0)$
(iii) for any $\tilde{q} = \tilde{\gamma}(\tilde{s}, \tilde{q}_0)$, $\tilde{s} \in \tilde{\Sigma}^*$, if $\tilde{q}' = \tilde{\gamma}(\tilde{s}', \tilde{q})$, $\tilde{s}' \in \tilde{\Sigma}^*$, then there exists an $s \in \Sigma^*$ such that $q' = \gamma(s, q)$, where $q = \Theta(\tilde{q})$ and $q' = \Theta(\tilde{q}')$.
(iv) for any $q = \gamma(s, q_0)$, $s \in \Sigma^*$, there exists $\tilde{s} \in \tilde{\Sigma}^*$ such that $\tilde{q} = \tilde{\gamma}(\tilde{s}, \tilde{q}_0)$, where $q = \Theta(\tilde{q})$.

We have the following two results.

Theorem 1: Suppose that $\tilde{\mathcal{S}}$ includes $\mathcal{S}$ and that $\tilde{\mathcal{S}}$ avoids the states $\tilde{q} \in \tilde{Q}^m \subset \tilde{Q}$. Then $\mathcal{S}$ avoids the states $q \in Q^m \subset Q$, where $Q^m = Q \setminus \Theta(\tilde{Q} \setminus \tilde{Q}^m)$.

Proof: Suppose there exists $s \in \Sigma^*$ such that $q = \gamma(s, q_0) \in Q^m$. Then, by (iv) of Definition 1, there exists $\tilde{s} \in \tilde{\Sigma}^*$ such that $\tilde{q} = \tilde{\gamma}(\tilde{s}, \tilde{q}_0)$. Since $\tilde{\mathcal{S}}$ avoids the states in $\tilde{Q}^m$, we must have $\tilde{q} \in \tilde{Q} \setminus \tilde{Q}^m$. Then, since $q = \Theta(\tilde{q})$, $q \in \Theta(\tilde{Q} \setminus \tilde{Q}^m) = Q \setminus Q^m$, which is a contradiction. □

Theorem 2: Suppose that $\tilde{\mathcal{S}}$ includes $\mathcal{S}$ and that for any $\tilde{s} \in \tilde{L}$, there exists $\tilde{s}' \in \tilde{\Sigma}^*$ such that $\tilde{\gamma}(\tilde{s}', \tilde{\gamma}(\tilde{s}, \tilde{q}_0)) \in \tilde{Q}^m \subset \tilde{Q}$. Then for any $s \in L$, there exists $s' \in \Sigma^*$ such that $\gamma(s', \gamma(s, q_0)) \in Q^m \subset Q$, where $Q^m = \Theta(\tilde{Q}^m)$.

Proof: Immediately follows from (iii) of Definition 1. □

The first result, Theorem 1, can be used in designing controllers to avoid *marked states*, $q \in Q^m$, where the second result, Theorem 2, can be used in designing controllers to lead a DES to marked states.

3. OVERLAPPING DECOMPOSITIONS AND EXPANSIONS

In this section we consider overlapping decompositions and expansions of DESs for the purpose of decentralized supervisory controller design. Here, we propose obtaining an overlapping decomposition of a DES using the topological structure of its automaton. In our approach, *overlapping subautomata* of an automaton are first identified by examining the topological structure of the given automaton. These subautomata are identified such that the only interconnection between the subautomata are through the overlapping part, i.e., no event should connect two states in different subautomata, unless one of these states is in the overlapping part of the two subautomata. As an example, the automaton shown in Figure 1a can be decomposed into two overlapping subautomata as shown in Figure 1b.

Once an overlapping decompositon of the original automaton is obtained, in order to obtain disjoint subautomata, we expand the overlappingly decomposed automaton as follows:

i) A state or an event in the overlapping part of n subautomata is repeated n times and each repeated state/event is assigned to a different subautomaton. All repeated events corresponding to a controllable (uncontrollable) event are taken to be controllable (uncontrollable).
ii) Two uncontrollable events are added between any two repeated states, such that when such an event occurs the state changes from one repeated state to the other.
iii) If the inital state is in the overlapping part of the original automaton, then the initial state of the expanded automaton can be chosen as any one of the repeated states of the original intial state. Otherwise, the inital state of the original automaton is chosen as the inital state of the expanded automaton.

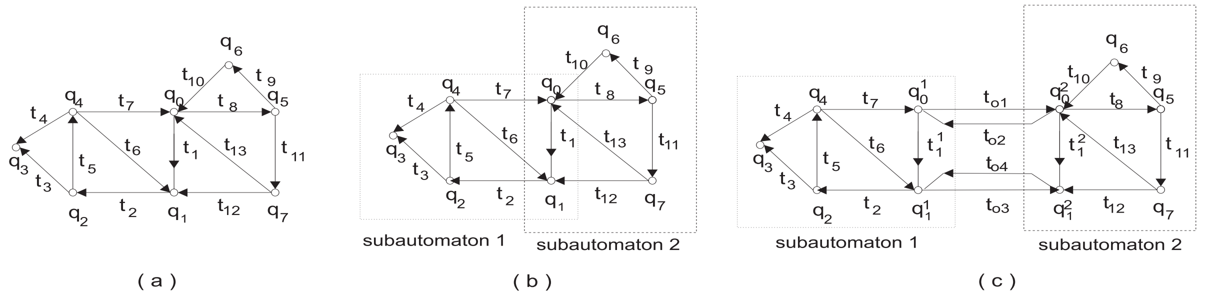

Fig. 1. a) Original automaton b) Overlappingly decomposed automaton c) Expanded automaton

As a result of this procedure, an expanded automaton, which consists of K disjoint subautomata, is obtained from an original automaton which was decomposed into K overlapping subautomata. As an example, the overlappingly decomposed automaton shown in Figure 1b is expanded to the automaton shown in Figure 1c, where the states q_0 and q_1 and the event t_1 are repeated and new events, t_{o1}, t_{o2}, t_{o3}, and t_{o4}, are introduced. The initial state $\tilde{q}_0$ of the expanded automaton may be chosen either as q_0^1 or q_0^2.

To identify the sets of states and events of the expanded automaton, we note that each state/event of a subautomaton corresponds to a state/event of the original automaton. The set of states of the expanded automaton is given by $\tilde{Q} \overset{\triangle}{=} \cup_{i=1}^{K} Q_i$, where Q_i is the set of states of the i^{th} subautomaton. For the example shown in Figure 1c, $Q_1 = \{q_0^1, q_1^1, q_2, q_3, q_4\}$ and $Q_2 = \{q_0^2, q_1^2, q_5, q_6, q_7\}$. The set of events of the expanded automaton, on the other hand, is given by $\tilde{\Sigma} \overset{\triangle}{=} \hat{\Sigma} \cup \bar{\Sigma}$. Here, $\hat{\Sigma} = \cup_{i=1}^{K} \Sigma_i$, where Σ_i is the set of events of the i^{th} subautomaton and $\bar{\Sigma}$ is the set of additional events introduced between the repeated states. For the example shown in Figure 1c, $\Sigma_1 = \{t_1^1, t_2, t_3, t_4, t_5, t_6, t_7\}$, $\Sigma_2 = \{t_1^2, t_8, t_9, t_{10}, t_{11}, t_{12}, t_{13}\}$, and $\bar{\Sigma} = \{t_{o1}, t_{o2}, t_{o3}, t_{o4}\}$. For the original automaton shown in Figure 1a, we assume that t_6 and t_{12} are uncontrollable events. Then $\Sigma_{1_{uc}} = \{t_6\}$ and $\Sigma_{2_{uc}} = \{t_{12}\}$ are obtained ($\Sigma_{1_c} = \Sigma_1 \setminus \Sigma_{1_{uc}}$ and $\Sigma_{2_c} = \Sigma_2 \setminus \Sigma_{2_{uc}}$). Therefore, $\tilde{\Sigma}_{uc} = \{t_6, t_{12}\} \cup \bar{\Sigma}$ is obtained for the expanded automaton shown in Figure 1c.

To describe the transition function for the expanded automaton, let $\mathcal{E}(q)$ denote the set of states in the expanded automaton which correspond to the state q in the original automaton and $\mathcal{F}(t)$ denote the set of events in the expanded automaton which correspond to the event t in the original automaton. Then, define the transformations $\mathcal{F}^{-1} : \hat{\Sigma} \to \Sigma$ and $\mathcal{E}^{-1} : \tilde{Q} \to Q$ such that $t = \mathcal{F}^{-1}(\tilde{t}) \iff \tilde{t} \in \mathcal{F}(t)$ and $q = \mathcal{E}^{-1}(\tilde{q}) \iff \tilde{q} \in \mathcal{E}(q)$. Now, the transition function of the expanded automaton, $\tilde{\gamma}$, is defined as follows:

- If $\tilde{t} \in \hat{\Sigma}$,

$$\tilde{\gamma}(\tilde{t}, \tilde{q}) = \begin{cases} \mathcal{E}(\gamma(\mathcal{F}^{-1}(\tilde{t}), \mathcal{E}^{-1}(\tilde{q}))) \cap \mathcal{X}(\tilde{q}), \\ \quad \text{if } \tilde{q} \text{ and } \tilde{t} \text{ belong to the same} \\ \quad \text{subautomaton} \\ \text{not defined, otherwise} \end{cases}$$

  Here, $\mathcal{X}(\tilde{q})$ is the set of states which belong to the same subautomaton as $\tilde{q}$.
- If $\tilde{t} \in \bar{\Sigma}$,

$$\tilde{\gamma}(\tilde{t}, \tilde{q}) = \begin{cases} \tilde{q}', & \text{if } \tilde{t} \text{ is introduced from } \tilde{q} \text{ to } \tilde{q}' \\ \text{not defined,} & \text{otherwise} \end{cases}$$

The *disjoint subautomata* of the expanded automaton are described by tuples $\mathcal{A}_k = (Q_k, \Sigma_k, \gamma_k, q_0^k)$, for some $q_0^k \in Q_k$, where $\gamma_k : \Sigma_k \times Q_k \to Q_k$ is defined as follows:

$$\gamma_k(t, q) = \begin{cases} \tilde{\gamma}(t, q), & \text{if } t \in \Sigma_k \text{ and } q \in Q_k \\ \text{not defined,} & \text{otherwise} \end{cases}$$

Now, let $\mathcal{S}$ denote a DES, to be called the *original discrete-event system (ODES),* defined by the automaton $\mathcal{A} = (Q, \Sigma, \gamma, q_0)$, or equivalently by the language $L = \{s \in \Sigma^* \mid \gamma(s, q_0) \text{ is defined}\}$. Also let $\tilde{\mathcal{S}}$, to be called the *expanded discrete-event system (EDES),* be defined by the automaton $\tilde{\mathcal{A}} = (\tilde{Q}, \tilde{\Sigma}, \tilde{\gamma}, \tilde{q}_0)$, or equivalently by the language $\tilde{L} = \{\tilde{s} \in \tilde{\Sigma}^* \mid \tilde{\gamma}(\tilde{s}, \tilde{q}_0) \text{ is defined}\}$, where $\tilde{\mathcal{A}}$ is obtained by expanding $\mathcal{A}$ as explained above. To show that $\tilde{\mathcal{S}}$ includes $\mathcal{S}$, we first introduce the following two results.

Lemma 1: Let $\tilde{q} = \tilde{\gamma}(\tilde{s}, \tilde{q}_0) \in \tilde{Q}$, for some $\tilde{s} \in \tilde{\Sigma}^*$, and $\tilde{q}' = \tilde{\gamma}(\tilde{s}', \tilde{q}) \in \tilde{Q}$, for some $\tilde{s}' \in \tilde{\Sigma}^*$. Then, there exists $s \in \Sigma^*$ such that $q' = \gamma(s, q)$, where $q = \mathcal{E}^{-1}(\tilde{q})$ and $q' = \mathcal{E}^{-1}(\tilde{q}')$.

Proof: If $\tilde{s}' = \epsilon$, then $s = \epsilon$ satisfies the required relation. Thus, suppose that $\tilde{s}' \neq \epsilon$ and let $\tilde{s}' = \tilde{t}_1 \tilde{t}_2 \ldots \tilde{t}_n$. Initialize $s = \epsilon$ and do the following for $i = 1, 2, \ldots, n$:

- If $\tilde{t}_i \in \bar{\Sigma}$, skip to next i.
- If $\tilde{t}_i \in \hat{\Sigma}$, suffix $\mathcal{F}^{-1}(\tilde{t}_i)$ to s.

It can now inductively be shown that $\mathcal{E}^{-1}(\tilde{\gamma}(\tilde{s}', \tilde{q})) = \gamma(s, \mathcal{E}^{-1}(\tilde{q}))$. □

Lemma 2: Let $q = \gamma(s, q_0) \in Q$, for some $s \in \Sigma^*$. Then there exists $\tilde{s} \in \tilde{\Sigma}^*$ such that $\tilde{q} = \tilde{\gamma}(\tilde{s}, \tilde{q}_0) \in \mathcal{E}(q)$.

Proof: If $s = \epsilon$, then $\tilde{s} = \epsilon$ satisfies the required relation. Thus, suppose that $s \neq \epsilon$ and let $s = t_1 t_2 \ldots t_n$.

Also define $s_i \overset{\triangle}{=} t_1 t_2 \ldots t_i, i = 1, 2, \ldots, n$. Initialize $\tilde{s} = \epsilon$ and do the following for $i = 1, 2, \ldots, n$:

- If $\tilde{\gamma}(\tilde{s}, \tilde{q}_0)$ and $\gamma(s_i, q_0)$ belong to the same subautomaton, say to subautomaton k, then suffix $\tilde{t} \in \mathcal{F}(t_i) \cap \Sigma_k$ to $\tilde{s}$ (note that $\tilde{t}$ is uniquely defined).
- If $\tilde{\gamma}(\tilde{s}, \tilde{q}_0)$ belongs to subautomaton k, but $\gamma(s_i, q_0)$ belongs to the subautomaton $l \neq k$, then there exists $\tilde{t} \in \bar{\Sigma}$ which connects $\tilde{\gamma}(\tilde{s}, \tilde{q}_0)$ to $\tilde{q} \in \mathcal{E}(\mathcal{E}^{-1}(\tilde{\gamma}(\tilde{s}, \tilde{q}_0))) \cap Q_l$ (note that $\tilde{q}$ is uniquely defined). Suffix $\tilde{t}$ to $\tilde{s}$. Now, $\tilde{\gamma}(\tilde{s}, \tilde{q}_0)$ and $\gamma(s_i, q_0)$ both belong to subautomaton l. Thus, suffix $\tilde{t} \in \mathcal{F}(t_i) \cap \Sigma_l$ to $\tilde{s}$.

It can now inductively be shown that $\tilde{\gamma}(\tilde{s}, \tilde{q}_0) \in \mathcal{E}(q)$. □

We can now prove the following result.

Theorem 3: $\tilde{\mathcal{S}}$ includes $\mathcal{S}$.

Proof: Let us define $\Theta : \tilde{Q} \to Q$ as $\Theta(\tilde{q}) = \mathcal{E}^{-1}(\tilde{q})$, for all $\tilde{q} \in \tilde{Q}$, and $H : Q \to \tilde{Q}$ as

$$H(q) = \begin{cases} \tilde{q}_0 \ , & \text{if } q = q_0 \\ \tilde{q} \ , \text{for some } \tilde{q} \in \mathcal{E}(q) \ , & \text{otherwise} \end{cases}$$

It is easy to check that conditions (i) and (ii) of Definition 1 are satisfied by the above choice of Θ and H. Furthermore, condition (iii) is satisfied by Lemma 1 and condition (iv) is satisfied by Lemma 2. □

Now, using Theorems 1 and 2, we have the following results.

Corollary 1: Suppose that $\tilde{\mathcal{S}}$ avoids the states $\tilde{q} \in \tilde{Q}^m \subset \tilde{Q}$. Then $\mathcal{S}$ avoids the states $q \in Q^m \subset Q$, where $Q^m = Q \setminus \mathcal{E}^{-1}(\tilde{Q} \setminus \tilde{Q}^m)$.

Corollary 2: Suppose that for any $\tilde{s} \in \tilde{L}$, there exists $\tilde{s}' \in \tilde{\Sigma}^*$ such that $\tilde{\gamma}(\tilde{s}', \tilde{\gamma}(\tilde{s}, \tilde{q}_0)) \in \tilde{Q}^m \subset \tilde{Q}$. Then for any $s \in L$, there exists $s' \in \Sigma^*$ such that $\gamma(s', \gamma(s, q_0)) \in Q^m \subset Q$, where $Q^m = \mathcal{E}^{-1}(\tilde{Q}^m)$.

4. DECENTRALIZED CONTROL DESIGN

For a DES described by an automaton $\mathcal{A} = (Q, \Sigma, \gamma, q_0)$, or equivalently by the language $L = \{s \in \Sigma^* \mid \gamma(s, q_0) \text{ is defined}\}$, to design a controller to avoid states $q \in Q^m \subset Q$, we first define the following sets: $\lambda(q) \overset{\triangle}{=} \{t \in \Sigma \mid \gamma(t, q) \text{ is defined}\}$, $Q^{d_0} \overset{\triangle}{=} Q^m$, $Q^{d_i} \overset{\triangle}{=} \{q \in Q \mid \gamma(t, q) \in Q^{d_{i-1}}, \ \forall t \in \lambda(q) \text{ or } \exists t \in \Sigma_{uc} \text{ such that } \gamma(t, q) \in Q^{d_{i-1}}\}$, for $i = 1, 2...$, $Q^d = \cup_{i=0}^{\infty} Q^{d_i}$, $Q^b \overset{\triangle}{=} \{q \in Q \mid \exists t \in \Sigma_c \text{ such that } \gamma(t, q) \in Q^d\}$, and

$$D(q) \overset{\triangle}{=} \begin{cases} \{t \in \Sigma_c \mid \gamma(t, q) \in Q^d\} \ , & \text{if } q \in Q^b \\ \emptyset \ , & \text{otherwise} \end{cases}$$

Then, a controller which avoids states $q \in Q^m$ is obtained by disabling events $t \in D(q)$ at state $q \in Q$. Here, our aim is to avoid deadlock. Thus, we take Q^m as the set of states at which no event can occur. For non-triviality, we assume that $q_0 \notin Q^d$.

Now, we will introduce a controller design methodology, where a controller is first designed for each disjoint subautomata of the EDES. A controller for the EDES is then designed by combining these controllers. Using the controller designed for the EDES, we will then describe a controller for the ODES and show that this controller avoids deadlock in the ODES.

4.1 *Controller design for a subautomaton*

We first design a controller for each disjoint subautomaton k, described by $\mathcal{A}_k = (Q_k, \Sigma_k, \gamma_k, q_0^k)$, for each possible initial state $q_0^k \in Q_k$. A state $q \in Q_k$ is a possible initial state for $\mathcal{A}_k$, if q can be the first state in Q_k for some event sequence $\tilde{s} \in \tilde{L}$. With some possible conservatism, we will take any state in the overlapping part as a possible initial state. If the initial state q_0 of the ODES is an element of Q_k, then q_0 is also taken as a possible initial state for $\mathcal{A}_k$. Thus, we first identify $Q_k^m \subset Q_k$, the set of states at which no event can occur in $\mathcal{A}_k$. Then, Q_k^d, Q_k^b, and $D_k(q)$ (for $q \in Q_k$) are defined similiar to Q^d, Q^b and $D(q)$. Then the events $t \in D_k(q)$ at state $q \in Q_k$ are disabled.

For the example shown in Figure 1c, $Q_1^m = \{q_3\}$, $Q_1^d = \{q_3\}$, $Q_1^b = \{q_2, q_4\}$, $D_1(q_2) = \{t_3\}$, $D_1(q_4) = \{t_4\}$ ($D_1(q) = \emptyset$, for $q \notin \{q_2, q_4\}$), $Q_2^m = \{q_1^2\}$, $Q_2^d = \{q_1^2, q_7\}$, $Q_2^b = \{q_0^2, q_5\}$, $D_2(q_5) = \{t_{11}\}$, and $D_2(q_0^2) = \{t_1^2\}$ ($D_2(q) = \emptyset$, for $q \notin \{q_0^2, q_5\}$).

4.2 *Controller design for the EDES*

Once a controller is designed for each subautomaton as described above, a controller is obtained for the expanded automaton (equivalently for the EDES) by disabling any event $\tilde{t} \in \tilde{D}(\tilde{q})$ at $\tilde{q} \in \tilde{Q}$, where the set $\tilde{D}(\tilde{q})$ is obtained simply as $\tilde{D}(\tilde{q}) = D_k(\tilde{q})$, where k indicates the subautomaton which the state $\tilde{q}$ belongs.

For the example shown in Figure 1c, $\tilde{D}(q_0^2) = D_2(q_0^2) = \{t_1^2\}$, $\tilde{D}(q_2) = D_1(q_2) = \{t_3\}$, $\tilde{D}(q_4) = D_1(q_4) = \{t_4\}$, $\tilde{D}(q_5) = D_2(q_5) = \{t_{11}\}$, and $\tilde{D}(\tilde{q}) = \emptyset$, for $\tilde{q} \notin \{q_0^2, q_2, q_4, q_5\}$.

4.3 *Controller design for the ODES*

Once a controller is designed for the EDES, a controller is obtained for the ODES by disabling any event $t \in D(q)$ at $q \in Q$, where the set $D(q)$ is obtained as

$$D(q) = \mathcal{H}\left(\bigcup_{\tilde{q} \in \mathcal{E}(q)} \tilde{D}(\tilde{q})\right) \ , \qquad q \in Q$$

where, for $\vartheta \subset \tilde{\Sigma}_c$, $\mathcal{H}(\vartheta) \overset{\triangle}{=} \{t \in \Sigma_c \mid \mathcal{F}(t) \subset \vartheta\}$.

For the example shown in Figure 1a, $D(q_0) = \mathcal{H}(\tilde{D}(q_0^1) \cup \tilde{D}(q_0^2)) = \mathcal{H}(\emptyset \cup \{t_1^2\}) = \emptyset$, $D(q_2) = \mathcal{H}(\tilde{D}(q_2)) = \mathcal{H}(\{t_3\}) = \{t_3\}$, $D(q_4) = \mathcal{H}(\tilde{D}(q_4)) = \mathcal{H}(\{t_4\}) = \{t_4\}$, $D(q_5) = \mathcal{H}(\tilde{D}(q_5)) = \mathcal{H}(\{t_{11}\}) = \{t_{11}\}$, and $D(q) = \emptyset$, for all other $q \in Q$.

Now, we present the following result.

Theorem 4: The controlled EDES includes the controlled ODES.

Proof: The proof is similar to the proof of Theorem 3, with the same Θ and H. □

We can now present the following result.

Theorem 5: The controller designed as above for the ODES avoids deadlock in the ODES.

Proof: Let $\kappa(q)$ denote the set of indices of the subautomata to which $q \in Q$ belongs. Suppose that no event can occur in the ODES (i.e., deadlock occurs) at state $q \in Q$. Then, by the expansion procedure indroduced above, no event can occur in $\mathcal{A}_k$ at state $\mathcal{E}(q) \cap Q_k$, for all $k \in \kappa(q)$. This implies that, $\mathcal{E}(q) \subset \tilde{Q}^m$, where $\tilde{Q}^m \subset \tilde{Q}$ denotes the set of states avoided by the controller obtained for the EDES. Therefore, $q \notin \mathcal{E}^{-1}(\tilde{Q} \setminus \tilde{Q}^m)$, which means that $q \in Q \setminus \mathcal{E}^{-1}(\tilde{Q} \setminus \tilde{Q}^m)$. The desired result now follows from Theorems 4 and 1. □

Before concluding this section, we note that the controlled ODES may be described by the language $L_c = \{s \in L \mid e_i(s) \notin D(\gamma(\Lambda_{i-1}(s), q_0)),\ \forall i \in \{1, 2, ..., E(s)\}\}$, where $e_i(s)$ denotes the i^{th} event in s, $E(s)$ denotes the number of events in s, and $\Lambda_{i-1}(s)$ is the prefix of $e_i(s)$ in s (i.e., $\Lambda_i(s) = \Lambda_{i-1}(s)e_i(s)$, $i = 1, 2, \ldots, E(s)$, with $\Lambda_0(s) = \epsilon$).

5. COMPUTATIONAL COMPLEXITY

In this section, we will compare the computational complexity of the proposed decentralized controller design approach to the computational complexity of the centralized design. To design a controller for a DES, described by an automaton $\mathcal{A} = (Q, \Sigma, \gamma, q_0)$, or equivalently by the language $L = \{s \in \Sigma^* \mid \gamma(s, q_0)$ is defined$\}$, one first needs to identify the sets Q^m, Q^d and Q^b. To identify these sets, all non-repeating event sequences starting from the initial state must be followed. Therefore, computational complexity of controller design for L is proportional to the number of non-repeating sequences in L.

To determine the order of the number of non-repeating event sequences in a large-scale DES, let us consider the systems described by ladder automata shown in Figure 2. Let $n = |Q|$ denote the number of states. Note that, for the ladder automata considered, the number of events, $|\Sigma|$, increase proportionally with n. Table 1 shows the number of non-repeating event sequences, C_n, of the n-state ladder automaton, for upto $n = 20$. It is seen that C_n, thus the computational complexity of controller design, increases exponentially (with base $\alpha \approx 1.6$) with n.

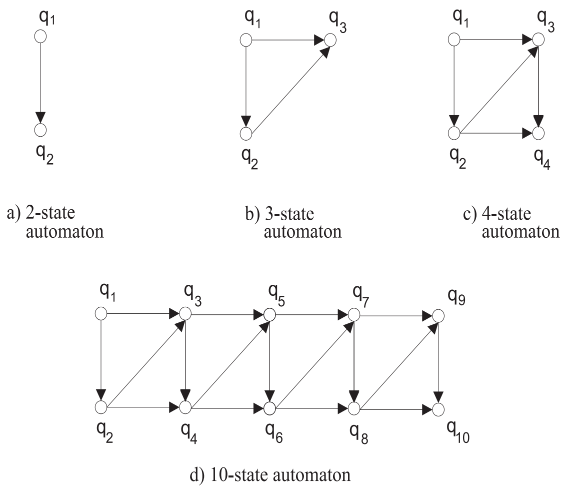

Fig. 2. Ladder automata.

Table 1: Number of non-repeating event sequences for the ladder automata

n	$\lvert\Sigma\rvert$	C_n	C_n/C_{n-1}
2	1	1	–
3	3	3	3
4	5	6	2
5	7	11	1.83
6	9	19	1.78
7	11	32	1.68
8	13	53	1.66
9	15	87	1.64
10	17	142	1.63
11	19	231	1.63
12	21	375	1.62
13	23	608	1.62
14	25	985	1.62
15	27	1595	1.62
16	29	2582	1.61
17	31	4179	1.61
18	33	6763	1.61
19	35	10944	1.61
20	37	17709	1.61

Therefore, to compare the computational complexity of the proposed design approach to the computational complexity of the centralized design, let us consider a DES with $n = 100$ states. Let us assume that the automaton of this DES is overlappingly decomposed into 12 subautomata, where each subautomaton has 10 states. We also assume that, for each subautomaton, there exists 5 states which were in the overlapping part before the expansion. Therefore, for each subautomaton, we take five possible initial states. We assume that the number of events in each subautomaton and in the original automaton are proportional to the number of states, n, of that automaton, and that the number of non-repeating event sequences in each automaton increases exponentially (with base α) with n. Therefore, the computational complexity of the controller design for each subautomaton is proportional to $5 \times \alpha^{10}$ and the computational complexity of controller design using the proposed approach is proportional to $12 \times 5 \times \alpha^{10} = 60\alpha^{10}$. On the other hand, since the

ODES has 100 states, the computational complexity of designing a centralized controller is proportional to α^{100}. Therefore, it is $\alpha^{100}/60\alpha^{10} = \alpha^{90}/60$ times easier and faster to use the proposed controller design approach, compared to a centralized design. For $\alpha = 1.6$, this ratio is equal to 3.91×10^{16}.

6. CONCLUSIONS

We have introduced the inclusion principle for DESs described by automata or formal languages. Using this principle, we have proposed a decentralized supervisory controller design approach.

As discussed in Section 5, for DESs which involve many states and events, the proposed approach requires very little computation to design a controller compared to a centralized design approach. However, we should note that, controllers designed by this approach may be conservative compared to centrally designed controllers. For example, the controller designed using the proposed approach for the system shown in Figure 1a disables t_3, t_4 and t_{11}. However, if we use a centralized design approach, we can find that disabling only t_3 and t_4 is sufficient to avoid deadlock. This conservatism (or suboptimality) is, however, the usual price to pay for the ease obtained in the design stage. This trade off is also well known in the case of continuous-state systems (Ikeda, *et al.*, 1981; İftar, 1993).

Here we presented only controller design to avoid deadlock. The proposed approach, however, can directly be extended to the more general case of designing controllers to avoid certain marked states. Alternatively, our approach may also be used to design controllers to lead the given DES to marked states. Where, in the latter case, one should obtain the controller for the EDES and the ODES so that Theorem 2 can be used instead of Theorem 1 in proving that the controller for the ODES leads the ODES to the desired states.

7. REFERENCES

Aybar, A. and A. İftar (2001). Decentralized control design for interconnected discrete–event systems. In: *Preprints of the 9th IFAC Symposium on Large Scale Systems*. Bucharest, Romania. pp. 415–418.

Aybar, A. and A. İftar (2002). Overlapping decompositions and expansions of Petri nets. *IEEE Transactions on Automatic Control*, **47**, 511–515.

Banaszak, Z. A. and B. H. Krogh (1990). Deadlock avoidance in flexible manufacturing systems with concurrently competing process flows. *IEEE Transactions on Robotics and Automation*, **6**, 724–734.

Cassandras, C. G. and S. Lafortune (1999). *Introduction to Discrete Event Systems*. Kluwer Academics. Norwell, MA.

Cho, K. H. and J. T. Lim (1999). Mixed centralized/decentralized supervisory control of discrete-event dynamic systems. *Automatica*, **35**, 121–128.

Ho, Y. (Ed.) (1992). *Discrete Event Dynamic Systems: Analyzing Complexity and Performance in the Modern World*. A Selected Reprint Volume, The Institute of Electrical and Electronics Engineers, Inc. New York.

İftar, A. (1993). Overlapping decentralized dynamic optimal control. *International Journal of Control*, **58**, 187–209.

İftar, A. and Ü. Özgüner (1998). Overlapping decompositions, expansions, contractions, and stability of hybrid systems. *IEEE Transactions on Automatic Control*, **43**, 1040–1055.

Ikeda, M. and D. D. Šiljak (1980). Overlapping decompositions, expansions, and contractions of dynamic systems. *Large Scale Systems*, **1**, 29–38.

Ikeda, M. and D. D. Šiljak (1986). Overlapping decentralized control with input, state, and output inclusion. *Control Theory and Advanced Technology*, **2**, 155–172.

Ikeda, M., D. D. Šiljak and D. E. White (1981). Decentralized control with overlapping information sets. *J. Optimization Theory and Applications*, **34**, 279–310.

Lin, F. and W. M. Wonham (1990). Decentralized control and coordination of discrete-event systems with partial observation. *IEEE Transactions on Automatic Control*, **35**, 1330–1337.

Ramadge, P. J. G. and W. M. Wonham (1989). The control of discrete event systems. *Proceedings of the IEEE*, **77**, 81–98.

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