Automatica, Vol. 28, No. 3, pp. 565–577, 1992
Printed in Great Britain.

0005–1098/92 $5.00 + 0.00
Pergamon Press Ltd

Aggregation and Multi-Level Control in Discrete Event Dynamic Systems*†

CÜNEYT M. ÖZVEREN‡ and ALAN S. WILLSKY§

The problem of controlling discrete-event dynamic systems so that it only produces strings of meaningful tasks is considered. This leads to the development of aggregate task-level models of such systems.

Key Words—Automata theory; control system design; discrete systems; modeling; observers; regulator theory; reliability; stability; state space methods; supervisory control.

Abstract—In this paper we consider the problem of higher-level aggregate modeling and control of discrete-event dynamic systems (DEDS) modeled as finite state automata in which some events are controllable, some are observed, and some represent events to be tracked. The higher-level models considered correspond to associating specified sequences of events in the original system to *single* macroscopic events in the higher-level model. We also consider the problem of designing a compensator that can be used to restrict microscopic behavior so that the system will only produce strings of these primitive sequences or *tasks*. With this lower level control in place we can construct higher-level models which typically have many fewer states and events than the original system. The use of these higher-level models allow us to decrease the computational complexity of problems involving complex interconnected DEDS.

1. INTRODUCTION

THE STUDY OF COMPLEX SYSTEMS has frequently prompted research on tools for aggregation and multi-level analysis. In this paper, we study such tools in the context of Discrete Event Dynamic Systems (DEDS) as introduced by Ramadge and Wonham (1987a, b). Specifically, the apparent combinatorial explosion associated with many DEDS analysis techniques would seem to place severe limits on their utility. Thus, there is ample motivation for the development of techniques for exploiting system structure and in particular for aggregating system behavior in order to mask out microscopic detail and focus on macroscopic behavior. In this paper we develop such a technique by exploiting the fact that in many applications the desired range of behavior of a DEDS is significantly smaller and more structured than its full range of possible behaviors. For example, a workstation in a flexible manufacturing system (FMS) may have considerable flexibility in the sequence of operations it performs. However, only particular sequences correspond to useful tasks, and one would like to restrict behavior so that only these are performed. This underlies the notion of a legal language introduced in Ramadge and Wonham (1987b) and generalized slightly in Özveren and Willsky (1992). It also suggests a natural way in which to perform aggregation by mapping a sequence of events, corresponding to a task, to a single macro-event at the higher-level. An important point to note is that in many applications, such as an FMS, the overall system consists of an interconnection of many subsystems, yielding extremely large composite state spaces. However, in most cases the desired coordination of the subsystems is at the task level. Thus, for example, in analyzing an FMS we can build individual aggregate models for each workstation and then consider coordination issues only at the higher level.

A key element in the concept we have described is the design of compensators that restrict behavior to the completion of desired tasks. As we have stated this is closely related to the legal language supervisory control work of others and to our work in Özveren and Willsky (1992) which not only allows us to achieve significant efficiencies by describing desired behavior in terms of primitive tasks but also, and more importantly, incorporates the notion of *eventual restrictability* through which we can directly model and accommodate the phenome-

* Received 3 October 1989; revised 7 December 1990; revised 7 January 1991; revised 30 July 1991; received in final form 20 September 1991. The original version of this paper was not presented at any IFAC meeting. This paper was recommended for publication in revised form by Associate Editor T. Başar under the direction of Editor H. Kwakernaak.

† Research supported by the Air Force Office of Scientific Research under Grant AFOSR-88-0032 and by the Army Research Office under Grant DAAL03-86-K0171.

‡ Telecommunications and Networking, Digital Equipment Corporation, 550 King Street, Littleton, MA 01460, U.S.A.

§ Laboratory for Information and Decision Systems, MIT, Cambridge, MA 02139, U.S.A.

non of set-up, i.e. the externally irrelevant transient behavior arising when one switches between tasks. In particular, the consideration of transient behavior led us to define and study a notion of stability in Özveren *et al.* (1991) that plays an important role here as well. Another important aspect of our work is the use of an intermittent observation model in which only the occurrences of certain key events are observed. We have shown that this model has significant consequences for both observability in Özveren and Willsky (1990) and output stabilization in Özveren and Willsky (1991). In particular, this model captures a critical issue in the control of complex systems, namely the coordinated timing of information and control action.

In the next section we review some of the concepts from previous work and extend our earlier work to the design of *output*-feedback compensators that achieve eventual restrictability. In Section 3 we make precise what we mean by a higher-level model based on a given set of macroscopic events each of which corresponds to strings of events at the lower level. Care must be taken in these specifications not only to ensure the faithful mapping of control and observations from microscopic to macroscopic levels but also to deal correctly with the change in the measure of logical time involved in the aggregation, since entire microscopic sequences are mapped into single macroscopic events. In Section 4 we then consider the design of task-level controllers and corresponding higher-level models for DEDS. This requires several additional concepts for the compatible control of a *set* of tasks and the construction of a system for the detection of task completions. Using these components, we construct a task-level control system which accepts task requests as input and controls the system to achieve the desired sequence. This leads to a simple higher-level model whose transitions only involve the set-up and completion of tasks. This model not only can be of value in simplifying the analysis of interconnections of DEDS but also its simple and regular form should greatly facilitate subsequent stages of DEDS analysis such as those involving timing studies or probabilistic measures of performance.

2. BACKGROUND AND PRELIMINARIES

The systems we consider (much as in Cieslak *et al.* (1988), Lin and Wonham (1988a, b), Ramadge and Wonham (1987a, b) and Vaz and Wonham (1986)), are nondeterministic finite-state automata with intermittent event observations defined over $G = (X, \Sigma, \Phi, \Gamma, \Xi)$. Here X is the state set, with $n = |X|$, Σ is the set of possible events, $\Phi \subset \Sigma$ is the set of controllable events, $\Gamma \subset \Sigma$ is the set of observable events, and $\Xi \subset \Sigma$ is the set of tracking events. Control is affected by enabling some or all of the controllable events, and the dynamics then evolve via the occurrence of events that are either enabled or uncontrollable (and thus permanently enabled). That is, the dynamics on G take the following form:

$$x[k+1] \in f(x[k], \sigma[k+1]), \tag{2.1}$$

$$\sigma[k+1] \in (d(x[k]) \cap u[k]) \cup (d(x[k]) \cap \bar{\Phi}), \tag{2.2}$$

where $\bar{\Phi}$ denotes the complement of Φ. Here, $x[k] \in X$, $\sigma[k+1] \in \Sigma$, and $u[k] \in U - 2^{\Phi}$ is the control input. The function $d: X \to 2^{\Sigma}$ specifies the set of possible events defined at each state, and the function $f: X \times \Sigma \to X$ is also set-valued (capturing nondeterminism). We assume that $\Phi \subset \Gamma$, which simplifies the presentation and computational complexity of our results.

Our model of the output process is that whenever an event in Γ occurs, we observe it. Specifically, define $h: \Sigma \to \Gamma \cup \{\epsilon\}$ by

$$h(\sigma) = \begin{cases} \sigma & \text{if } \sigma \in \Gamma \\ \epsilon & \text{otherwise,} \end{cases} \tag{2.3}$$

where ϵ is the "null transition". Then, our output equation is

$$\gamma[k+1] = h(\sigma[k+1]). \tag{2.4}$$

Note that h can be extended to a map on Σ^*, the set of all strings of finite length including the empty string ϵ, via $h(\sigma_1 \cdots \sigma_n) = h(\sigma_1) \cdots h(\sigma_n)$.

The set Ξ is the tracking alphabet, allowing us to define tracking over a selected alphabet. We use $t: \Sigma^* \to \Xi^*$, to denote the projection of strings over Σ into Ξ^*. Note that if there exists a cycle in A that consists solely of events that are *not* in Ξ, then the system may stay in this cycle indefinitely, generating no event in Ξ. It is not difficult to check for the absence of such cycles, and we assume this is the case. The quintuple $A = (G, f, d, h, t)$† representing our system can also be visualized graphically as in Fig. 1. The first symbol in each arc label denotes the event, while the symbol following "/" denotes the corresponding output. We mark the controllable events by ":u" and tracking events by "!".

There are several standard computer science concepts that we will need in our investigation.

† On occasion, we will construct auxiliary automata for which we will not be concerned with tracking. In such cases we will omit t from the specification.

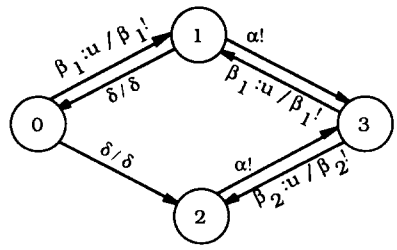

FIG. 1. A simple example.

The first is the notion of *liveness*: a DEDS is alive if $d(x)$ is nonempty for each $x \in X$. We will assume that this is the case. A second commonly used notion (Cieslak *et al.*, 1988; Golazewski and Ramadge, 1988; İnan and Varaiya, 1988), that we need is the composition $A_{12} = A_1 \| A_2$ of two automata, $A_i = (G_i, f_i, d_i, h_i)$ which share some common events. The dynamics of the composition are specified by allowing each automaton to operate as it would in isolation except that when a shared event occurs, it *must* occur in both systems.

Central to our work is the notion of stability studied in Özveren *et al.* (1991) (see also Ramadge, 1989). Let E be a given subset of X. A state $x \in X$ is *E-pre-stable* if every trajectory starting from x passes through E in a bounded number of transitions. The state $x \in X$ is *E-stable* if every state reachable from x is E-pre-stable, and the DEDS is *E-stable* if every $x \in X$ is E-stable. Note that E-stability for all of A is identical to E-pre-stability for all of A, and that this condition guarantees that all trajectories go through E infinitely often. We refer the reader to Özveren *et al.* (1991) for an $O(n^2)$ test for E-stability.

In Özveren *et al.* (1991) we also study stabilization via state feedback of the form $K: X \to U$ resulting in the closed-loop system $A_K = (G, f, d_K, h, t)$ with

$$d_K(x) = (d(x) \cap K(x)) \cup (d(x) \cap \bar{\Phi}). \quad (2.5)$$

A DEDS is E-stabilizable if we can find K so that A_K is both alive and E-stable.

In Özveren and Willsky (1992), we use compensators that use both current state and event trajectory information in order to control system behavior. Such a compensator, described by a map $C: X \times \Sigma^* \to U$, yields a closed-loop system with

$$\sigma[k+1] \in d_C(x[k], s[k]) \triangleq (d(x[k]) \cap C(x[k], s[k])) \cup (d(x) \cap \bar{\Phi}), \quad (2.6)$$

where $s[k] = \sigma[0] \cdots \sigma[k]$ with $\sigma[0] = \epsilon$. Note that this class of compensators is similar to the class of supervisors introduced in Ramadge and Wonham (1987b) although by allowing dependence on the current state we can achieve a somewhat richer class of behaviors. Also, as in our previous work, we will see here that for our purposes we can restrict attention to compensators that can be realized by finite state machines.

As in other work on DEDS, it is useful to phrase questions concerning event trajectory behavior in terms of languages (Hopcroft and Ullman, 1979). Let L be a regular language over a finite alphabet and let (A_L, x_0) be a minimal recognizer for L. Given a string $s \in L$, if $s = pqr$ for some p, q and r over Σ where p is a prefix of s and r is a suffix of s, we use s/pq to denote the suffix r, and we say that q is a substring of s. Also, for any language L, we let L^c denote the prefix closure of L, i.e.

$$L^c = \{p \in \Sigma^* \mid p \text{ is a prefix of some } s \in L\}. \quad (2.7)$$

Finally, L is a complete language if (a) every $s \in L$ is a proper prefix of some other $r \in L$ (so that all trajectories have unlimited extensions) and (b) L is prefix-closed (so that all initial segments of a trajectory are in L). Note that for a complete language all strings generated by the recognizer (A_L, x_0) are in L.

In Özveren and Willsky (1992) we study the notion of *restrictability*, i.e. the ability to force the system to generate strings in a desired regular complete language defined over Ξ. This is essentially the same as restricting behavior to a sublanguage of a specified legal language (see, e.g. Ramadge and Wonham, 1987b), with two modest differences. First, in our work we focus on restricting only the *tracking* event trajectory behavior. The second is that by focusing on regular complete languages, we can use our state-based framework. Indeed the key to the approach in Özveren and Willsky (1992) is to show how to transform the problem into one of static state feedback design to achieve controlled-invariance and stability. Furthermore in Özveren and Willsky (1992) we introduce the related notions of *eventual and stable restrictability* in which we allow an initial "set-up" transient before restricted behavior is achieved. These extensions are crucial in our present context and also are essential for the consideration of error recovery (Özveren and Willsky, 1992).

In most applications and as captured by our model, we do not have available complete knowledge of the current state nor of the full event history. In Özveren and Willsky (1990), we term a system *observable* if the current state is known perfectly at intermittent, but not

necessarily fixed, intervals of time. A necessary condition for observability is that it is not possible for our DEDS to generate arbitrarily long sequences of unobservable events. This is not difficult to check and will be assumed. Also, we will need to use some of the notation introduced in Özveren and Willsky (1990). Specifically, $R(A, x)$ denotes the set of states reachable from x, and we let Y denote the set of states that either have observable transitions defined to them or that are purely initial in that there are no transitions to them from any state. Let $q = |Y|$, and let $L(A, x)$ denote the set of all possible event trajectories of finite length that can be generated if the system is started from the state x. Also, let $L_f(A, x)$ be the set of strings in $L(A, x)$ that have an observable event as the last event, and let $\bar{L}(A) = \bigcup_{x \in X} L(A, x)$.

An observer for our system produces an "estimate" of the state corresponding to the set of possible states into which A transitioned when the last observable event occurred. Let $Z \subset 2^Y$ denote the state space of this observer and suppose that the present observer estimate is $\hat{x}[k] \in Z$ and that the next observed event is $\gamma[k+1]$. The observer must account for the possible occurrence of one or more unobservable events prior to $\gamma[k+1]$ and then the occurrence of $\gamma[k+1]$:

$$\hat{x}[k+1] = w(\hat{x}[k], \gamma[k+1]) \triangleq \bigcup_{x \in R(A \mid \Gamma, \hat{x}[k])} f(x, \gamma[k+1]), \quad (2.8)$$

$$\gamma[k+1] \in v(\hat{x}[k]) \triangleq h\left(\bigcup_{x \in R(A \mid \Gamma, \hat{x}[k])} (d(x) \cap u[k]) \cup (d(x) \cap \bar{\Phi})\right). \quad (2.9)$$

The set Z is then the reach of $\{Y\}$ using these dynamics, i.e. we start the observer in the state corresponding to a complete lack of state knowledge and let it evolve†. Our observer then is the DEDS $O = (F, w, v, i)$, where $F = (Z, \Gamma, \Phi, \Gamma)$ and i is the identity output function. For future reference we will use the symbol $\hat{\mathbf{x}}(s)$, $s \in h(\bar{L}(A))$ to denote the input–output map of O, i.e. this is the observer estimate if the observed output string is s.

When we only have available these intermittent observations we need to consider the use of an output compensator $C: \Gamma^* \to U$ (see Özveren and Willsky, 1991). In this case, however, preserving liveness is somewhat more involved since not only may we not know the current state exactly, but even if we do, the possible occurrence of unobservable transitions will lead to uncertainty at least until the occurrence of the next observable event. Suppose that we have observed the output string s, so that our observer is in $\hat{\mathbf{x}}(s)$ and our control input is $C(s)$. Then, we must make sure that any x reachable from any element of $\hat{\mathbf{x}}(s)$ by *unobservable events only* is alive under the control input $C(s)$. That is, for all $x \in R(A \mid \bar{\Gamma}, \hat{\mathbf{x}}(s))$, $d_C(x, s)$ should *not* be empty. In this case we will say that $C(s)$ is $\hat{\mathbf{x}}(\mathbf{s})$-compatible. In Özveren and Willsky (1991) we use such output compensators to solve problems of stabilization by output feedback. It is important to note that the intermittent nature of our observations introduces significant issues in this study underlining the issue of the timing of information and control and distinguishing our work from investigations such as in Cieslak *et al.* (1988). We refer the reader to Özveren and Willsky (1991) for details, including a bound of q^3 on the number of observable transitions until any trajectory enters E.

Let us now turn our attention to extending our previous results to the problem of eventual restrictability using an *output* compensator.

Definition 2.1. Given a complete regular language L over Ξ we say that A is *eventually L-restrictable by output feedback* if there exists an integer n_0 and an output compensator $C: \Gamma^* \to U$ such that A_C is alive and for all $x \in X$, $t(L(A_C, x)) \subset (\Xi \cup \{\epsilon\})^{n_0} L$. Such a C is called an *L-restrictability compensator.* □

Again, this notion represents an extension of those used, for example, in Cieslak *et al.* (1988) and Lin and Wonham (1988b), in that: (a) we focus on tracking events; (b) we allow an initial transient string outside the language L; and (c) we must deal with the intermittent nature of the observations and the accompanying fluctuation in the level of knowledge of the current state.

In order to construct a test for eventual restrictability by output feedback, we need to introduce well-known notions of dynamic invariance (see, for example, Özveren *et al.* (1991); Ramadge and Wonham (1987)). A subset Q of X is *f-invariant* if $f(Q, d) \subset Q$ where $f(Q, d) = \bigcup_{x \in Q} f(x, d(x))$ and $f(x, d(x)) = \bigcup_{\sigma \in d(x)} f(x, \sigma)$. We say that a subset Q of X is *(f, u)-invariant* if there exists a state feedback K such that Q is f-invariant in A_K. We say that a subset Q of X is a *sustainably (f, u)-invariant* set if there exists a state feedback K such that Q is alive and f-invariant in A_K. Let us also note that given any set $V \subset X$, there is a maximal

† To get Z, we consider the richest possible behavior by enabling all controllable events.

sustainably (f, u)-invariant subset W of V with a corresponding unique *minimally restrictive* feedback K. That is K disables as few events as possible in order to keep the state within W.

We now define a model to capture the desired behavior. Given L, let (A_L, x_0^L) be a minimal recognizer for L and let Z_L denote its state space. Let A'_L be an automaton which is the same as A_L except that its state space is $Z'_L = Z_L \cup \{b\}$ where b is a state used to signify that the event trajectory is no longer in L. This is the state we wish to avoid. Also, we let $d'_L(x) = \Xi$ for all $x \in Z'_L$, and

$$f'_L(x, \sigma) = \begin{cases} f_L(x, \sigma) & \text{if } x \neq b \text{ and } \sigma \in d_L(x) \\ \{b\} & \text{otherwise.} \end{cases} \tag{2.10}$$

Let O denote the observer for A, let $A(L) = A \,\|\, A'_L$, and let $O(L) = (G(L), w_L, v_L)$ denote the observer for $A(L)$; however, in this case, since we know that we will start A'_L in x_0^L, we take the state space of $O(L)$ as

$$Z(L) = R(O(L), \{\{x_0^L\} \times \hat{x} \mid \hat{x} \in Z\}). \tag{2.11}$$

Let

$$V_0 = \{\hat{z} \in Z(L) \mid \text{for all } (x_L, x_A) \in \hat{z},\ x_L \neq b\}. \tag{2.12}$$

Let $E(L)$ be the largest subset of V_0 which is sustainably (f, u)-invariant in $O(L)$ and for which the associated unique minimally restrictive feedback K^{EL} has the property that for any $\hat{z} \in Z(L)$, $K^{EL}(\hat{z})$ is $\hat{x}(\hat{z})$-compatible where

$$\hat{x}(\hat{z}) = \{x \in X \mid \exists x_L \in Z_L \text{ such that } (x_L, x) \in \hat{z}\}. \tag{2.13}$$

The construction of $E(L)$ and K^{EL} is a variation of the algorithm in Özveren *et al.* (1991) for the construction of maximal sustainably (f, u)-invariant subsets. We begin with any state $\hat{z} \in V_0$. If there are any uncontrollable events taking $\hat{z}$ outside V_0, we delete $\hat{z}$ and work with $V_1 = V_0 \backslash \{\hat{z}\}$. If not, we disable only those controllable events which take $\hat{z}$ outside V_0. If the remaining set of events defined at $\hat{z}$ is not $\hat{x}(\hat{z})$-compatible, we delete $\hat{z}$ and work with $V_1 = V_0 \backslash \{\hat{z}\}$. If not, we tentatively keep $\hat{z}$ and choose another element of V_0. In this way, we cycle through the remaining elements of V_0. The algorithm converges in a finite number of steps (at most $|V_0|^2$) to yield $E(L)$ and K^{EL} defined on $E(L)$. For $\hat{z} \in \overline{E(L)}$, we take $K^{EL}(\hat{z}) = \Sigma$.

Consider next the following subset of $E(L)$;

$$E_0(L) = \{\hat{x} \in Z \mid x_0^L \times \hat{x} \in E(L)\}. \tag{2.14}$$

Proposition 2.2. Given a complete language L over Ξ, A is eventually L-restrictable by output feedback if there exists an A-compatible state feedback $K: Z \to U$ such that the closed loop system O_K is $E_0(L)$-pre-stable.

Proof. We prove this by constructing the desired compensator $C: \Gamma^* \to U$: Given an observation sequence s, we trace it in O starting from the initial state $\{Y\}$. Let $\hat{x}$ be the current state of O given s. There are two possibilities.

(1) Suppose that the trajectory has not yet entered $E_0(L)$. Then we use O and the $E_0(L)$-pre-stabilizing feedback K to compute $C(s)$. In particular,

$$C(s) = (v(\hat{x}) \cap K(\hat{x})) \cup (v(\hat{x}) \cap \bar{\Phi}).$$

(2) When the trajectory in O enters $E_0(L)$, we switch to using the expanded observer $O(L)$ and $K^{E(L)}$. In particular, let $\hat{x}'$ be the state the trajectory in O enters when it enters $E_0(L)$ for the first time, and let s' be that prefix of s which takes $\{Y\}$ to $\hat{x}'$ in O. Then, we start O_L at the state $x_0^L \times \hat{x}' \in E(L)$, and let it evolve. Suppose that s/s' takes $x_0^L \times \hat{x}'$ to $\hat{z}$ in $O(L)$, then

$$C(s) = (v_L(\hat{z}) \cap K^{E(L)}(\hat{z})) \cup (v_L(\hat{z}) \cap \bar{\Phi}).$$

Since this feedback keeps the trajectory of O in $E(L)$ and $E(L) \subset V_0$, the behavior of A is restricted as desired. □

Proposition 2.2 is only a sufficient condition for eventual L-restrictability. Specifically, as pointed out in Kumar *et al.* (1990) and Özveren and Willsky (1992), it is possible for a state to be eventually L-restrictable without being pre-stable with respect to the set of L-restrictable states, although there are conditions under which this cannot happen. In this paper, we focus on the stronger sufficient condition of Proposition 2.2, which we refer to as *stable L-restrictability* by output feedback.

Since $E(L)$ is the *maximal* sustainably (f, u)-invariant subset of V_0 and $K^{E(L)}$ is unique, the possible behavior of an L-restrictable state x in the closed loop system constructed in the proof is the *maximal* subset of L to which the behavior of x can be restricted. Also, if $E_0 = 0$, then O cannot be $E_0(L)$-pre-stabilizable and thus A is not stably L-restrictable by output feedback. Finally, if A is stably L-restrictable by output feedback, the results of Özveren and Willsky (1991) allow us to bound the number of observable transitions until the trajectory is restricted to L.

In some situations, it is more natural to think of systems in which there are *forced* events which can be forced to occur regardless of the other events defined at the current state. It is not difficult to capture forced events in the modeling

framework described in this section (Özveren, 1989), and thus we assume that the DEDS to be controlled do not have forced events. However, at the last stage of our development we will use forced events in the description of our task standard form to provide a simple and intuitive picture of this higher-level model.

3. CHARACTERIZING HIGHER-LEVEL MODELS

In this section, we present a notion of higher-level modeling of DEDS based on a given set of primitives, each of which consists of a finite set of tracking event strings, where the occurrence of any of these strings corresponds to some macroscopic event, such as completion of a task. To illustrate our notion of modeling and to give a preview of task-level control, consider the system in Fig. 1 and suppose that we wish to use output feedback to restrict its behavior so that the "task" $\alpha\beta_1$ is continuously performed, i.e. we want to restrict behavior to $(\alpha\beta_1)^{*c}$. In Fig. 2 we illustrate an automaton that realizes such a compensator. That is, compensation is achieved by the composition of this automaton, started in the state denoted (0, 1, 2), with the DEDS of Fig. 1†. This automaton was constructed in the following manner. First, recall that the set of tracking events for the DEDS of Fig. 1 is $\{\alpha, \beta_1, \beta_2\}$, and suppose that for the moment we assume that we have perfect state knowledge at all times. Since we want to restrict tracking sequences to alternating values of α and β_1, we obviously want to disable β_2. Also, if we are in state 0, we want to disable β_1 since if we do not, a possible event sequence would be $\beta_1\delta\beta_1$ so that the resulting tracking event sequence would be $\beta_1\beta_1$. Unfortunately, we may not know the current state, and in particular we start with total ignorance of the initial state. In this case, we *might* want to disable β_1 or β_2, but we certainly cannot disable both, since this would disable *all* events from state 3. Thus, before we apply any control action we need to wait until we have *some* state information. In order to form state estimates, let us construct the observer O for our DEDS (note that $Y = \{0, 1, 2\}$). What we then do is to determine those controllable events we can disable at each observer state in order to (eventually) restrict behavior to $(\alpha\beta_1)^{*c}$ while preserving liveness, and Fig. 2, is the resulting restricted observer. First, at the initial state (0, 1, 2) we enable both β_1 and β_2. The possible first observable event is then any in the set

† Note that the actual compensator map $C: \Gamma^* \to U$ can be computed as follows: for any $s \in \Gamma^*$, determine the state of the automation of Fig. 2 starting from (0, 1, 2). Then $C(s)$ is the set of controllable events that are enabled at this state. For example, $C(\beta_1) = \{\beta_1\}$, $C(\delta) = \{\beta_2\}$, $C(\delta\beta_2) = \{\beta_1\}$.

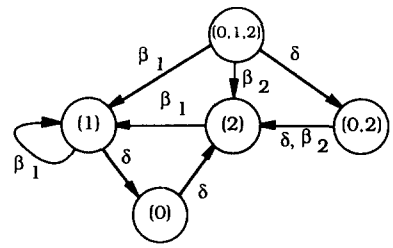

FIG. 2. Illustrating the compensator for eventual $(\alpha\beta_1)^{*c}$-restrictability by output feedback for the system of Fig. 1.

$\{\beta_1, \beta_2, \delta\}$. If, for example, we observe β_2, we know we are in state 2. At this point, we see that if we disable β_2, then the subsequent two events will necessarily be α (which is unobservable), taking us to 3, followed by β_1, taking us to 1. This is reflected in Fig. 2 where the transition β_2 takes us from (0, 1, 2) to (2) from which only β_1 is enabled. Similarly, if δ is observed first, we know that we are either in state 0 or 2. Again, we cannot disable both β_1 and β_2 since if we are in 2, the uncontrollable and unobservable event α will occur driving the system to state 3, at which all events have been disabled. However, from (0, 2) we can, as indicated in Fig. 2 disable β_1. In this case, if the DEDS is actually in 0, the next observable event will be δ while if the DEDS state is 2, the next observable event will be β_2, and in either case we will know that the DEDS state has transitioned to 2.

If we think of $\alpha\beta_1$ as a primitive, then at a higher level we might want to model only its occurrences using the simple automation of Fig. 3 where ψ_1 denotes the occurrence of this primitive. However, for this automaton, with ψ_1 observable, to truly model A_C, we should be able to use the observations in A_C to detect occurrences of $\alpha\beta_1$, perhaps with some initial uncertainty. For example, by inspection of Fig. 2, if we observe β_1, we cannot say if $\alpha\beta_1$ has occurred or not, but if we observe $\beta_1\beta_1$, we know that $\alpha\beta_1$ *must* have occurred at least once. In general, after perhaps the first occurrence of β_1, every occurrence of β_1 corresponds to an occurrence of $\alpha\beta_1$. The definition we give in this section then allows us to conclude that the automaton in Fig. 3 models the closed loop system A_C.

Let Σ' denote the macroscopic event set, where each $\sigma \in \Sigma'$ corresponds to a *set* $H_e(\sigma)$ of tracking strings in the original model where the map $H_e: \Sigma' \to 2^{\Sigma^*}$ is termed *primitive* if $H_e(\sigma)$ is a

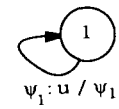

FIG. 3. Higher-level model for completions of the "task" $\alpha\beta_1$.

finite collection of strings. We allow H_e to be set-valued to capture the fact that there may be several ways to complete a desired task.

Definition 3.1. A primitive map H_e is termed *minimal* if (1) for all, distinct, σ_1, $\sigma_2 \in \Sigma'$ and for all $s \in H_e(\sigma_1)$, no suffix of s is in $H_e(\sigma_2)$, and (2) for all $\sigma \in \Sigma'$, $s \in H_e(\sigma)$, no proper suffix of s is in $H_e(\sigma)$. □

We can easily extend H_e to act on $(\Sigma')^*$: $H_e(\epsilon) = \epsilon$ and $H_e(s\sigma) = H_e(s)H_e(\sigma)$, the set of all possible concatenations of one string in $H_e(s)$ and one in $H_e(\sigma)$.

Proposition 3.2. If H_e is minimal then for all distinct r_1, r_2 such that $r_1, r_2 \neq \epsilon$, $|r_1| \leq |r_2|$, and r_1 is not a suffix or r_2, $\Xi^* H_e(r_1) \cap \Xi^* H_e(r_2) = \emptyset$.

Proof. Assume the contrary, and let $s \in \Xi^* H_e(r_1) \cap \Xi^* H_e(r_2)$. Also let σ_1 (respectively, σ_2) be the last event in r_1 (respectively, r_2). There are two cases here. First, suppose that $\sigma_1 \neq \sigma_2$. Then, there exist distinct $p_1 \in H_e(\sigma_1)$ and $p_2 \in H_e(\sigma_2)$ such that both p_1 and p_2 are suffixes of s. Assume, without loss of generality, that $|p_1| \leq |p_2|$, then p_1 is also a suffix of p_2. But then, H_e cannot be minimal. Now, suppose that $\sigma_1 = \sigma_2$. Thanks to minimality, among all elements of $H_e(\sigma_1)$, only one string, say p can be a suffix of s. Let s' be that prefix of s such that $p = s/s'$. Then, repeat the previous steps using s', and all but the last elements of r_1 and r_2. Since r_1 and r_2 are distinct, and r_1 is not a suffix of r_2, σ_1 will be different from σ_2 at some step and then we will establish a contradiction. Therefore, $\Xi^* H_e(r_1) \cap \Xi^* H_e(r_2) = \emptyset$. □

The following states that concatenation preserves minimality.

Proposition 3.3. Given minimal $H_1: \Sigma_2 \to 2^{\Sigma_1^*}$ and $H_2: \Sigma_3 \to 2^{\Sigma_2^*}$, if we define $H_3: \Sigma_3 \to 2^{\Sigma_1^*}$ so that $H_3(\sigma) = H_1(H_2(\sigma))$ for all $\sigma \in \Sigma_3$, then H_3 is a minimal primitive map. Here, since $H_2(\sigma)$ is a set of strings, $H_1(H_2(\alpha))$ is the set of strings resulting from applying H_1 to each string in $H_2(\sigma)$.

Proof. Assuming the contrary, there exists, σ_1, $\sigma_2 \in \Sigma_3$, $s \in H_3(\sigma_1)$, and a suffix r of s so that $r \in H_3(\sigma_2)$. Let $s' \in H_2(\sigma_1)$ and $r' \in H_2(\sigma_2)$ such that $s \in H_1(s')$ and $r \in H_1(r')$. Then, by minimality of H_2, r' cannot be a suffix of s' and s' cannot be a suffix of r' either. Also, since r is a suffix of s, $s \in \Sigma_1^* H_1(r')$. Then, thanks to Proposition 3.2, H_1 cannot be minimal, and we establish a contradiction. □

Our definition of higher-order modeling captures two important properties that such models must have in order to be physically meaningful. First, we want control capabilities of the macroscopic model to reflect microscopic capabilities. That is, if it is possible to construct a higher-level compensator that restricts macro-level event behavior to a particular language L, then it must be true that we can design a micro-level controller to restrict behavior to the complete language corresponding to the mapping of L down to the lower level. Secondly, by observing the output sequence at the lower level, we should be able to unambiguously determine the corresponding sequence of macro-events, except perhaps for a finite-length start-up phase until the initial state uncertainty settles out (e.g. as in the example in Fig. 2).

Definition 3.4. Given DEDS $A = (G, f, d, h, t)$ and $A' = (G', f', d', h', t')$ where $G' = (X', \Sigma', \Phi', \Gamma', \Xi')$, and a minimal primitive map $H_e: \Sigma' \to 2^{\Xi^*}$, we say that A' is an H_e-*model* of A is there exists a map $H_0: \Gamma^* \to \Sigma'^*$ and an integer n_d so that:†

(1) Compatibility. For all complete $L \subset \Sigma'^*$ such that A' is eventually L-restrictable, A is eventually $H_e(L)^c$-restrictable by output feedback.

(2) Detectability. For all $s \in L(A)$, such that $t(s) \in H_e(p)$ for some $p \in L(A')$, (a) $p \in (\Sigma' \cup \{\epsilon\})^{n_d} H_0(h(s))$, and (b) for all $r \in \Sigma^* s$, $H_0(h(r)) \in \Sigma'^* H_0(h(s))$. □

Compatibility formalizes the notion of fidelity in the modeling of control, and detectability makes precise the concept of reconstructability of macro-events. The map H_0 corresponds to this reconstruction process, while n_d corresponds to the maximum number of macro-events that might go unidentified at the start. Condition 2(b) captures the need for unambiguous reconstruction in that it requires that the reconstruction of a string of primitives should not depend upon preceding events.

The following result, which immediately

† We have chosen in our definition to look at the larger class of macroscopic languages to which A is eventually restrictable by full state feedback, rather than only with output feedback. All of our results carry over if we use this weaker notion of compatibility at the higher level. Also, we have defined the macroscopic languages over all of Σ' rather than only the tracking alphabet Ξ'. Similarly in our definition of detectability we have required the stronger condition that from lower level observations, we can reconstruct the *entire* upper-level event trajectory, not just the part in Γ'. Again, we can carry all of our development over to the weaker cases. As we will see, this stronger definition suffices for our purposes.

follows from Definition 3.4, states that the concept of modeling is invariant under compensation.

Proposition 3.5. If A' is an H_e-model of A then for any compensator $C':\Gamma'^* \to U'$ for A', there exists a compensator $C:\Gamma^* \to U$ for A such that $A'_{C'}$ is an H_e-model of A_C with the same H_0. □

In general, we may be interested in several different levels of aggregation. Thus we need the following result.

Proposition 3.6. Given the automata $A = (G, f, d, h, t)$, $A' = (G', f', d', h', t')$, and $A'' = (G'', f'', d'', h'', t'')$, and minimal primitive maps $H'_e:\Sigma' \to 2^{\Xi^*}$ and $H''_e:\Sigma'' \to 2^{\Xi'^*}$, so that A' is an H'_e-model of A with H'_0 and A'' is an H''_e-model of A' with H''_0, define $\pi:\Xi' \to 2^{\Sigma'^*}$ so that $\pi(\sigma) = \sigma(\bar{\Xi}' \cup \in \epsilon\})^{|X'|}$ for $\sigma \in \Xi'$ and define $H_e:\Sigma'' \to 2^{\Xi^*}$ as $H_e(\sigma) = H'_e(\pi(H''_e(\sigma)))$ for $\sigma \in \Sigma''$. Then

(1) H_e is a minimal primitive map.

(2) A'' is an H_e-model of A with $H_0(s) = H''_0(h'(H'_0(s)))$ for all $s \in \Gamma^*$.

Proof. Clearly, π is a minimal primitive map. By Proposition 3.3, H_e is also a minimal primitive map. Compatibility is shown as follows: if A'' is eventually L-restrictable, then A' is eventually $H''_e(L)^c$-restrictable by output feedback $\to A'$ is eventually $H''_e(L)^c$-restrictable $\to A'$, is eventually $\pi(H''_e(L)^c)$-restrictable $\to A$ is eventually $H_e(L)^c$-restrictable by output feedback. Finally, detectability is immediate. □

4. AGGREGATION

In this section, we use the concept of modeling of Section 3 to present an approach for the aggregation of DEDS. Suppose that our system is capable of performing a set of tasks. What we would like is to design a compensator that accepts as inputs requests to perform particular tasks and then controls A so that the appropriate task is performed. Assuming that the completion of this task is detected, we can construct a higher level and extremely simple model for our controlled system: tasks are requested and completed. In the first subsection we define tasks and several critical properties of sets of tasks and their compensators. In Section 4.2 we discuss the property of task observability, i.e. the ability to detect all occurrences of specified tasks. In Section 4.3 we then put these pieces together to construct a special higher-level model which we refer to as *task standard form*.

4.1. *Reachable tasks*

Our model of a task is a finite set of finite length strings, where the generation of any string in the set corresponds to the completion of the task. Let $\mathbf{T}$ be the index set of a collection of tasks, i.e. for anny $i \in \mathbf{T}$ there is a finite set L_i of finite length strings over Ξ that represent task i. We let $L_T = \bigcup_{i \in \mathbf{T}} L_i$.

Definition 4.1. Given $\mathbf{T}$, we say that $\mathbf{T}$ is an *independent* task set if for all $s \in L_T$, no substring of s, except for itself, is in L_T. □

Then when we look at a tracking sequence there is no ambiguity concerning what tasks have been completed and which substring corresponds to which task. Note that if $\mathbf{T}$ is an independent set, then the minimal recognizer (A_T, x_0) for all of L_T has a single final state x_f, i.e. all strings in L_T take x_0 to x_f, and x_f has no events defined from it (since L_T is a finite set). Furthermore, for each $i \in \mathbf{T}$, the minimal recognizer $(A_{L_i}, x_0^{L_i})$ also has a single final state $x_f^{L_i}$ which has no events defined from it.

Definition 4.2. A task $i \in \mathbf{T}$ is *reachable* if A is stably L_i^{*c}-restrictable. $\mathbf{T}$ is a *reachable set* if each $i \in \mathbf{T}$ is reachable. □

Definition 4.3. Task $i \in \mathbf{T}$ is *reachable by output feedback* if A is stably L_i^{*c}-restrictable by output feedback. $\mathbf{T}$ is *reachable by output feedback* if each $i \in \mathbf{T}$ is reachable by output feedback. □

For example, task $L_1 = \{\alpha\beta_1\}$ and $L_2 = \{\alpha\beta_2\}$ for the DEDS of Fig. 1 are both reachable by output feedback.

Given a task $i \in \mathbf{T}$ that is reachable by output feedback, let $C_i:\Gamma^* \to U$ be an L_i^{*c}-restrictability compensator. Note that states in $E_0(L_i^{*c})$, as defined in Section 2, are guaranteed to generate a sublanguage of L_i^{*c} in the closed loop system. However, for any state $\hat{x}$ outside of $E_0(L_i^{*c})$, although we cannot *guarantee* that L_i^{*c} will be generated given the particular knowledge of the current state of the system (i.e. given that the system is in some state in $\hat{x}$), it may still be possible for such a string to occur. Furthermore, in general, a string in L_j, for some other j, may be generated from a state $x \in \hat{x}$ before the trajectory in O reaches $E_0(L_i^{*c})$. If in fact this happens, then task j will have been completed while the compensator was trying to set-up the system for task i. Since this is a mismatch between what the compensator is trying to accomplish and what is actually happening in the system, we will require that it cannot happen. To

make this precise, let Z_r denote the set of *persistent* observer states where, as in Özveren and Willsky (1990), a state is persistent if it can be reached by an arbitrarily long string of events. We enforce the condition described previously only for behavior initiated from within Z_r, thereby accommodating its possible violation for the finite number of transitions until the observer reaches Z_r:

Definition 4.4. Given a reachable task $i \in \mathbf{T}$ and an L_i^{*c}-restrictability compensator C_i, C_i is *consistent* with $\mathbf{T}$ if for all $\hat{x} \in Z_r \cap \overline{E_0(L_i^{*c})}$, for all $x \in \hat{x}$, and for all $s \in L(A_{C_i}, x)$, $t(s) \notin L_T$. □

Let us consider testing the existence of and constructing consistent restrictability compensators. Note that we only need to worry about forcing the trajectory in O into $E_0(L_i^{*c})$ *without* completing any task along the way. Once that is done, restricting the behavior can be achieved by the compensator defined in Proposition 2.2. First, we need a mechanism to recognize that a task is completed. Let (A_T, x_0) be a minimal recognizer for L_T with state set X_T and final state x_f. Since not all events are defined at all states in X_T, we add a new state, g, to X_T, and for each event that is not previously defined at states in X_T we define a transition to state g. Thus if A_T enters state g, we know that the tracking event sequence generated starting from x_0 and ending in g is not the prefix of *any* task sequence. Also, to keep the automaton alive, we define self-loops for all events in Ξ at states g and x_f. Let A_T' be this new automaton. Given a string s over Ξ, if s takes x_0 to g in A_T' then no prefix of s can be in L_T. If, on the other hand, the string takes x_0 to x_f then some prefix of this string must be in L_T. Now, let $O' = (G', w', v')$ be the observer for $A \| A_T'$. We let the state space Z' of Q' be the reach of initial states

$$Z_0' = \{\hat{x} \times \{x_0\} \mid \hat{x} \subset Z_r\}, \tag{4.1}$$

i.e. $Z' = R(O', Z_0)$. Let $p: Z' \to Z_r$ be the projection of Z' into Z_r, i.e. given $\hat{z} \in Z'$, $p(\hat{z}) = \bigcup_{(x_1, x_2) \in \hat{z}} \{x_1\}$. Also, let $E_0' = \{\hat{z} \in Z' \mid p(\hat{z}) \in E_0(L_i^{*c})\}$. Our goal is to reach E_0' from the initial states Z_0' while avoiding the completion of any task. So, we remove all transitions from states in E_0' and instead create self loops in order to preserve liveness. Let $O'' = (G', w'', v'')$ represent the modified automaton. Let us now consider the set of states in which we need to keep the trajectory. These are the states that cannot correspond to a completion of any task:

$$E'' = \{\hat{z} \in Z' \mid \forall (x_1, x_2) \in \hat{z}, x_2 \neq x_f\}. \tag{4.2}$$

Let V' be the maximal (f, u)-invariant subset of E', with $K^{V'}$ the corresponding A-compatible, minimally restrictive feedback. In order for a consistent compensator to exist, Z_0' must be a subset of V'. In this case we need to steer the trajectories to E_0' while keeping them in V'. Thus, we need to find $K'': Z' \to U$ so that Z' is E_0'-pre-stable in $O''_{K^{V'}}$ and so that the combined feedback $K: Z' \to U$ with

$$K(\hat{z}) = K^{V'}(\hat{z}) \cap K''(\hat{z}), \tag{4.3}$$

for all $\hat{z} \in Z'$ is A-compatible. The construction of such a K, if it exists, proceeds much as in Section 2. Thanks to the uniqueness of $K^{V'}$, if we cannot find such a feedback, then a consistent restrictability compensator cannot exist. To continue, we assume that consistent compensators exist, i.e. that $Z_0' \subset V'$ and K exists.

Finally, let us outline how we put the various pieces together to construct C_i. Given an observation sequence s, we trace it in O starting from the initial state $\{Y\}$. Let $\hat{x}$ be the current state of O given s. There are three possibilities:

(1) If $\hat{x} \notin Z_r$ and the trajectory has not yet entered $E_0(L_i^{*c})$, we use O and an $E_0(L_i^{*c})$-pre-stabilizing feedback to construct $C_i(s)$ as in Proposition 2.2.

(2) If $\hat{x} \in Z_r$ and the trajectory has not yet entered $E_0(L_i^{*c})$, we use the observer O'' and feedback K defined above. In particular, let $\hat{x}'$ be the state in the observer O into which the trajectory moves when it enters Z_r for the first time, and let s' be that prefix of s which takes $\{Y\}$ to $\hat{x}'$ in O. Then, we start O'' at state $\hat{x}' \times x_0$. Then if s/s' takes $\hat{x}' \times x_0$ to $\hat{z}$ in O'',

$$C_i(s) = (v''(\hat{z}) \cap K(\hat{z})) \cup (v''(\hat{z}) \cap \bar{\Phi}). \tag{4.4}$$

(3) When the trajectory enters $E_0(L_i^{*c})$, we switch to using $O(L_i^{*c})$ and the (f, u)-invariance feedback $K^{L_i^{*c}}$. $C_i(s)$ can then be constructed as in Proposition 2.2.

We now describe explicitly an overall compensator which responds to requests to perform particular tasks by enabling the appropriate compensator C_i. Given a set of p tasks $\mathbf{T}$, reachable by output feedback, and a task $i \in \mathbf{T}$, let $C_i: \Gamma^* \to U$ denote the compensator corresponding to task i. The compensator C that we construct admits events corresponding to requests for tasks as inputs and, depending on the inputs, C switches in an appropriate fashion between C_i. In order to model this, we use an automaton illustrated in Fig. 4, which has p-states, where state i corresponds to using the compensator C_i to control A. The set of events τ_i^F are *forced* events, as introduced in Section 2,

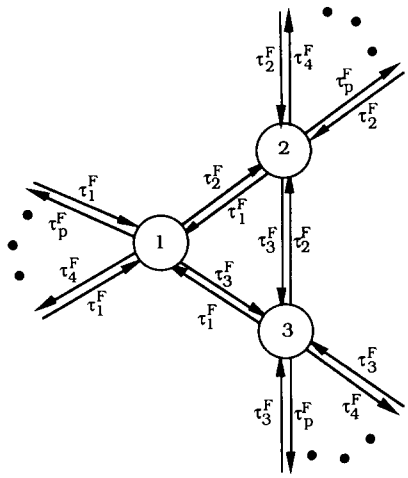

FIG. 4. An automaton to construct C.

corresponding to switching to C_i. In this case, when τ_i^F is forced, C_i is used as the compensator. Let $\Phi_T = \{\tau_1^E, \ldots, \tau_p^F\}$ and $U_T = 2^{\Phi_T}$. The input to C is a subset of Φ_T, representing the set of tasks which are requested. The compensator responds to this input as follows: if C is set up to perform task i. There are three possibilities: (1) if the input is the empty set, then C disables all events in A, awaiting future task requests; (2) if the input contains τ_i^F, then C will not force any event but continue performing task i (thereby avoiding an unnecessary set-up transient); (3) finally, if the input is not empty but it does not contain τ_i^F, then C will force one of the events in this set. At this level of modeling, we do not care which event C decides to force. If the action of C corresponds to a switch from one task to another, the activated task compensator C_i is initialized using the approach described previously. Specifically, suppose that the observer is in state $\hat{x}$ right before τ_i^F is forced. Consider the three cases described previously for C_i: if $\hat{x} \notin Z_r$ and $\hat{x} \notin E_0(L_i^{*c})$, then we use O starting from the initial state $\hat{x}$ and an $E_0(L_i^{*c})$-prestabilizing feedback. If $\hat{x} \in Z_r$ and $\hat{x} \notin E_0(L_i^{*c})$, then we start O'' at state $\hat{x} \times x_0$ and use the compensator described previously to drive the system to the desired set of states. Finally, if $\hat{x} \in E_0(L_i^{*c})$, then we start $O(L_i^{*c})$ at state $x_0^{L_i^{*c}} \times \hat{x}$, where $x_0^{L_i^{*c}}$ is the initial state of the minimal recognizer for L_i^{*c}, and we use the (f, u)-invariant feedback $K^{L_i^{*c}}$. A block diagram for A_C is illustrated in Fig. 5.

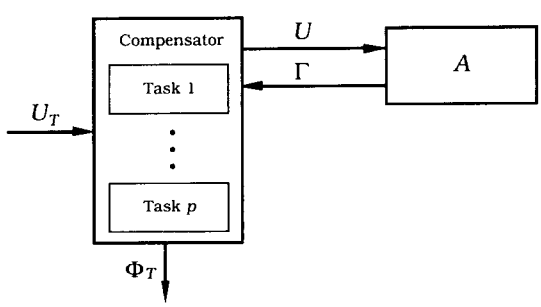

FIG. 5. Block diagram for A_C.

4.2. *Observable tasks*

In this section, we define a notion of observability for tasks. Consistent with our definition of detectability, we focus on detecting occurrences of tasks from that point in time at which the observer enters a persistent state. This can be viewed either as allowing a short start-up period or as specifying the level of initial state knowledge required in order for task detection to begin immediately.

Definition 4.5. A task $i \in \mathbf{T}$ is *observable* if there exists a function $\mathcal{J}: Z_r \times L(O, Z_r) \to \{\epsilon, \psi_i^F\}$ so that for all $\hat{x} \in Z_r$ and for all $x \in \hat{x}$, $\mathcal{J}$ satisfies:

(1) $\mathcal{J}(\hat{x}, h(s)) = \psi_i^F$ for all $s \in L(A, x)$ such that $s = p_1 p_2 p_3$ for some $p_1, p_2, p_3 \in \Sigma^*$ for which $t(p_2) \in L_i$, and

(2) $\mathcal{J}(\hat{x}, h(s)) = \epsilon$ for all other $s \in L(A, x)$.

A set of tasks $\mathbf{T}$ is *observable* if each $i \in \mathbf{T}$ is observable. □

Since we use task observability only with task control, we construct a test for the observability of task i assuming that it is reachable and that we are given a consistent L_i^{*c}-restrictability compensator C_i. Thanks to consistency, we only need to construct $\mathcal{J}$ for $\hat{x} \in E_0(L_i^{*c})$ and for strings s such that $t(s) \in L_i^{*c}$. First, we let $A'_{L_i} = (G'_{L_i}, f'_{L_i}, d'_{L_i})$ be the same as the recognizer A_{L_i} but with a self-loop at the final state $x_f^{L_i}$ for each $\sigma \in \Xi$. Now, let $Q = (G_Q, f_Q, d_Q)$, with state space X_Q, denote the live part of $A'_{L_i} \| A$, i.e. X_Q is the set of states x in $X'_{L_i} \times X$ so that there exists an arbitrarily long string in $L(A'_{L_i} \| A, x)$. In fact, note that for each $x \in X$ such that $(x_0^{L_i}, x) \in X_Q$, there exists $s \in L(A, x)$ so that $t(s) \in L_i$. Finally, let $O_Q = (F_Q, w_Q, v_Q)$ be the observer for Q with the state space Z_Q that is the reach of

$$Z_{Q0} = \bigcup_{\hat{x} \in E_0(L_i^{*c})} (\{x_0^{L_i}\} \times \hat{x}) \cap X_Q, \qquad (4.5)$$

in Q_Q, i.e. $Z_Q = R(O_Q, Z_{Q0})$. Note that if i is observable, then the last event of each string in L_i must be an observable event. Assuming that this is the case, let

$$E_Q = \{\hat{z} \in Z_Q \mid \exists (x, y) \in \hat{z} \text{ such that } x = x_f^{L_i}\}. \qquad (4.6)$$

Given the observations on A_{C_i} let us first trace the trajectory in the observer O. At some point in time, O enters some $\hat{x} \in E_0(L_i^{*c})$. When this happens we know that the system starts tracking

task i. At this point, let us start tracing the future observations in O_Q starting from the state $(\{x_0^{L_i}\} \times \hat{x}) \cap X_Q$. This trajectory will enter some $\hat{z} \in E_Q$, and at this point, we know that task i *may* have been completed. However, for task observability, we need to be *certain* that task i is completed. Thus, for an observable task, it must be true that for all $\hat{z} \in E_Q$ and for all $(x, y) \in \hat{z}$, $x = X_f^{L_i}$, In this case we can define $\mathcal{S}$ to be ϵ until the trajectory in O_Q enters E_Q and ψ_i^F from that point on. Thus, we have shown the following:

Proposition 4.6. Given a reachable task $i \in \mathbf{T}$ and a consistent L_i^{*c}-restrictability compensator C_i, if (1) the last event of each string in L_i is observable; and (2) for all $\hat{z} \in E_Q$ and for all $(x, y) \in \hat{z}$, $x = x_f^{L_i}$, then task i is observable in A_{C_i}. □

The procedure explained above allows us to detect the *first* completion of task i. Detecting other completions of task i is straightforward. Suppose that O enters the state $\hat{y}$ when O_Q enters E_Q. Note that $\hat{y} \in E_0(L_i^{*c})$. At this point we detect the first occurrence of task i, and we immediately re-start O_Q at state $x_0^{L_i} \times \hat{y} \cap X_Q$. The procedure continues with each entrance into E_Q signaling task completion and a re-start of O_Q. Note that the observer O runs continuously throughout the evolution of the system. Let $D_i^*: \Gamma^* \to \{\epsilon, \psi_i^F\}$ denote the complete task detector system. We can think of D_i^* as a combination of three automata: the observer O, the system O_Q which is re-started when a task is detected, and a single one-state automaton which has a self-transition loop, with event ψ_i^F, which occurs whenever a task is detected. This event is the only observable event for this system. Note that both the O_Q re-start and the ψ_i^F transition can be implemented as forced transitions.

Finally, in the same way in which we constructed C from the C_i, we can also define a task detector D, illustrated in Fig. 6, from the set of individual task detectors D_i. Specifically, if C is set at C_i initially, D is set at D_i. Using the output Φ_T of C, D switches between D_i. For

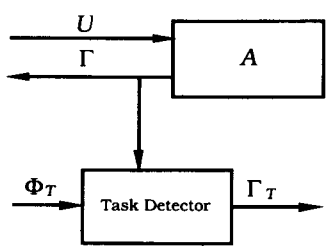

FIG. 6. Task detector block diagram.

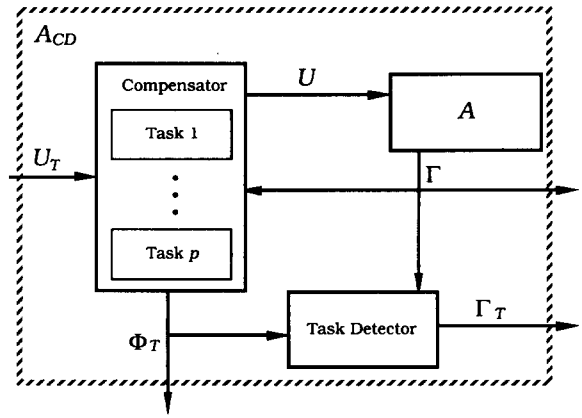

FIG. 7. The task-level closed-loop system.

example, if D is set at D_i and τ_j^F is forced by C, then D switches to D_j. The output of D takes values in $\Gamma_T = \{\psi_1^F, \ldots, \psi_p^F\}$.

4.3. *Task-level closed loop systems and task standard form*

Using the pieces developed in the preceding subsections we can now construct a task-level closed-loop system as pictured in Fig. 7. The overall system is $A_{CD} = (G_{CD}, f_{CD}, d_{CD}, t_{CD}, h_{CD})$ where

$$G_{CD} = (X_{CD}, \Sigma \cup \Phi_T \cup \Gamma_T, \Phi \cup \Phi_T, \Gamma \cup \Phi_T \cup \Gamma_T, \Xi \cup \Phi_T). \quad (4.7)$$

Note that Φ_T and Γ_T are both observable and Φ_T is controllable. Also, we include Φ_T in the tracking events to mark the fact that the system has switched compensators. This is important since following the switch, we will allow a finite length set-up. Also, since it does not make much sense in practice to force a switch to another compensator while the system is in the middle of completing a task, we impose the restriction that events in Φ_T can only be forced right after a task is completed. Since we require that all the tasks are observable (see Proposition 4.7), we can easily implement this restriction. Then, A_{CD} can only generate strings s such that

$$t(s) \in (\Xi \cup \{\epsilon\})^{n_t}(L_1^* \cup \cdots \cup L_p^*) \times (H_e(\tau_1)L_1^* \cup \cdots \cup H_e(\tau_p)L_p^*)^*, \quad (4.8)$$

where n_t is the maximum number of tracking transitions needed until O enters the set of persistent states in $E_0(L_i^{*c})$ for each $i \in \mathbf{T}$.

The higher-level operation of this system consists of the task initiation commands, Φ_T and the task completion acknowledgements, Γ_T. The input U_T indicating what subset of tasks can be enabled can be thought of as an external command containing the choices of subsets of Φ_T to be enabled. The use and control of this

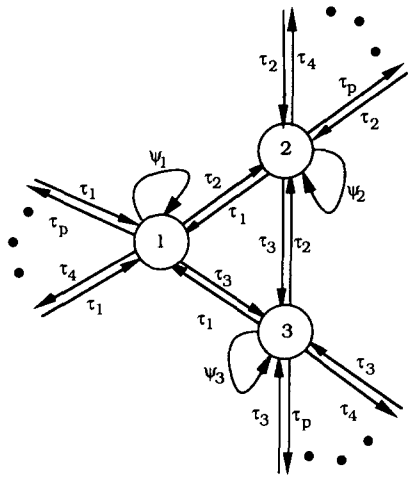

FIG. 8. Task standard form: all events are controllable and observable.

command involves higher-level modeling or scheduling issues beyond the purely task-level concept. What we show in this section is that the task-level behavior of A_{CD} can in fact be modeled, in the precise sense introduced in Section 3, by a much simpler automaton $A_{TSF} = (G_{TSF}, f_{TSF}, d_{TSF})$ illustrated in Fig. 8 where all the events are controllable and observable. We term A_{TSF} the *task standard form.*

Let us first define H_e. We first define $H_e(\epsilon) = \epsilon$ and $H_e(\psi_i) = L_i$. Note that, thanks to the independence of **T**, for any $i, j \in \mathbf{T}$, no suffix of string in $H_e(\psi_i)$ can be in $H_e(\psi_j)$. Defining $H_e(\tau_i)$ is more tricky. There are two issues.

(1) We need to take into account the fact that the closed loop system does *not* generate strings in L_i immediately after C switches to C_i. In particular, if we assume that O is in a persistent state when C switches to C_i and if we let n_e denote the maximum number of tracking transitions that can occur in A for any trajectory in O that starts from a persistent state of O up to and including the transition that takes the trajectory to a state in $E_0(L_i^{*c})$, then at most n_e tracking transitions can occur after C switches to C_i and before the behavior of the closed loop system is restricted to L_i^{*c}. Thus, we must choose H_e so that $H_e(\tau_i) \subseteq \tau_i^F(\Xi \cup \{\epsilon\})^{n_e}$.

(2) We also need to ensure the minimality of H_e. Specifically, we now know that $H_e(\tau_i) \subseteq \tau_i^F(\Xi \cup \{\epsilon\})^{n_e}$. Suppose that we let $H_e(\tau_i) = \tau_i^F(\Xi \cup \{\epsilon\})^{n_e}$. Then, no suffix of a string in $H_e(\psi_i)$ can be in $H_e(\tau_i)$ since all strings in $H_e(\tau_i)$ start with τ_i^F. Also, no suffix of a string in $H_e(\tau_i)$ can be in $H_e(\tau_j)$ even if $i = j$. However, a suffix of a string in $\tau_i^F(\Xi \cup \{\epsilon\})^{n_e}$ *may* be in $H_e(\psi_j)$ for some j. Thus, we let $H_e(\tau_i) = (\Xi \cup \{\epsilon\})^{n_e} \cap \overline{(\Xi \cup \{\epsilon\})^{n_e} L_T}$. Note that thanks to consistency, the strings in L_T cannot occur in a set-up of a task. Therefore, eliminating strings that end with a string in L_T will not cause any problems in restrictability.

Proposition 4.7. Given a set of tasks **T** that is reachable by output feedback and observable, A_{TSF} is an H_e-model of A_{CD}.

Proof. We first verify the detectability condition. Before defining H_0, let us define $t' : \Phi_T \cup \Gamma_T \to \Sigma_{TSF}$ as $t'(\tau_i^F) = \tau_i$ for all i and $t'(\psi_i^F) = \psi_i$ for all i. We then pick H_0 as t' of the projection of the observation sequence over $\Gamma \cup \Phi_T \cup \Gamma_T$ to $\Phi_T \cup \Gamma_T$, i.e. $H_0(s) = t'(s \downarrow (\Phi_T \cup \Gamma_T))$, where, $s \downarrow \Pi$, in general, denotes that part of the string s over the alphabet $\Pi \subset \Sigma$. Finally, let n_d be n_t divided by the length of the shortest string in L_T. Then, thanks to observability, the first detectability condition is satisfied. Also, using minimality it is straightforward to verify the second detectability condition.

To verify the compatibility conditon, note that A_{TSF} is eventually restrictable to any infinite length string s in $L(A_{TSF})$. Thus, if we show that A_{CD} is eventually $H_e(s)^c$-restrictable by output feedback, then the compatibility condition is verified. Let us now proceed with showing this. If the first event of s is some τ_i, then we simply force τ_i^F and look at the second event. If the first event of s is some ψ_i, then we force τ_i^F and wait until ψ_i. When ψ_i occurs, we look at the second event. In both cases, when we look at the second event, we repeat the same process. It then follows that A_{CD} is restricted as desired. Therefore, A_{TSF} is an H_e-model of A_{CD}. □

5. CONCLUSIONS

In this paper, we have introduced concepts of higher-level modeling for DEDS based on a given set of primitive event sequences corresponding to tasks which the system may perform. Through our investigation of task reachability we constructed task compensators and this, together with task observability allowed us to construct simple high-level models of automata so that events in the high-level model correspond to set-up and completion of tasks. The aggregation scheme presented in this paper provides one tool that can be of use in combatting the computational complexity in DEDS problems of interest. In particular, the cardinality of the overall state space of a system consisting of many subsystems can be extremely large. However, in many applications, the coordination of the subsystems is only required at the task level, allowing the possibility of aggregating the subsystems individually before

considering their composite, thereby reducing complexity significantly. In Özveren (1989) we present an example illustrating this idea involving the multi-level control of workstations connected by buffers. Such a framework opens up numerous questions for the future, ranging from the use of higher-level models as the basis for quantitative system analysis to the careful examination of the information and control required at various levels of a hierarchical control system of the type described here.

REFERENCES

Cieslak, R., C. Desclaux, A. Fawaz and P. Varaiya (1988). Supervisory control of discrete-event processes with partial observations. *IEEE Trans. on Automatic Control*, **33**, 249–260.

Golazewski, C. H. and P. J. Ramadge (1988). Mutual exclusion problems for discrete event systems with shared events. *Proceedings of 27th Conference on Decision and Control.*

Hopcroft, J. E. and J. D. Ullman (1979). *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, Reading, MA.

Inan, K. and P. Varaiya (1988). Finitely recursive process models for discrete event systems. *IEEE Trans. on Automatic Control*, **33**(7).

Kumar, R., V. Garg and S. I. Marcus (1990). Language stability of deds. *Conf on Math. Theory of Control, Bombay, India.*

Lin, F. and W. N. Wonham (1988a). Controllability and observability in the statefeedback control of discrete-event systems. *Proceedings of 27th Conference on Decision and Control.*

Lin, F. and W. N. Wonham (1988b). On observability of discrete event systems. *Information Sciences*, **44**, 173–198.

Özveren, C. M. (1989). Analysis and control of discrete event dynamic systems: a state space aproach. Ph.D. thesis, MIT, Cambridge, MA. Laboratory for Information and Decision Systems Report, LIDS-TH-1907.

Özveren, C. M. and A. S. Willsky (1990). Observability of discrete event dynamic systems. *IEEE Trans. on Automatic Control*, **35**, 797–806.

Özveren, C. M. and A. S. Willsky (1991). Output stabilizability of discrete event dynamic systems. *IEEE Trans. on Automatic Control*, **36**, 925–935.

Özveren, C. M. and A. S. Willsky (1992). Tracking and restrictability in discrete event dynamic systems. *SIAM J. Control Optimiz.*, to appear.

Özveren, C. M., A. S. Willsky and P. J. Antsaklis (1991). Stability and stabilizability of discrete event dynamic systems. *J. ACM*, **38**, 730–752.

Ramadge, P. J. (1989). Some tractable supervisory control problems for discrete event systems modeled by Buchi automata. *IEEE Trans. on Aut. Control*, **34**, 10–19.

Ramadge, P. J. and W. M. Wonham (1987a). Modular feedback logic for discrete event systems. *SIAM J. Control Optimiz.* **25**(5).

Ramadge, P. J. and W. M. Wonham (1987b). Supervisory control of a class of discrete event processes. *SIAM J. Control Optimiz.* **25**(1).

Vaz, A. F. and W. M. Wonham (1986). On supervisor reduction in discrete event systems. *Int. J. Control.*

Zhong, H. and W. M. Wonham (1990). On the consistency of hierarchical supervision in discrete event systems. *IEEE Trans. on Aut. Control*, **35**(10).