March 1990 LIDS-P-1949

AGGREGATION AND MULTI-LEVEL CONTROL IN DISCRETE EVENT DYNAMIC SYSTEMS

Cüneyt M. Özveren and Alan S. Willsky
MIT, Cambridge, MA 02139.

Summary

In this paper we consider higher-level aggregate modelling and control of discrete-event dynamic systems (DEDS). 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. A complete treatment of the topics presented here can be found in [5].

1 Background and Preliminaries

The class of systems we consider are defined over $G = (X, \Sigma, \Phi, \Gamma, \Xi)$, where X is the set of states, with $n = |X|$, Σ is the finite 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. Also, $U = 2^\Phi$ denotes the set of admissible inputs. The dynamics on G are:

$$x[k+1] \in f(x[k], \sigma[k+1]) \quad (1.1)$$

$$\sigma[k+1] \in (d(x[k]) \cap u[k]) \cup (d(x[k]) \cap \overline{\Phi}) \quad (1.2)$$

The set-valued function d specifies the set of possible events defined at each state, and the state transition function f is also set-valued. We assume that $\Phi \subset \Gamma$. Whenever an event in Γ occurs, we observe it; otherwise, we see nothing. Thus, our output equation is

$$\gamma[k+1] = h(\sigma[k+1]) \quad (1.3)$$

where h is the projection map from Σ^* to Γ^*, obtained by deleting all events not in Γ.

The set Ξ, denotes the tracking alphabet, and $t : \Sigma^* \rightarrow \Xi^*$, denotes 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. We assume that this is *not* possible. $A = (G, f, d, h, t)$ represents our system.

We say that $x \in X$ is alive if $\forall y \in R(A,x)$, $d(y) \neq \emptyset$. A set $Q \subset X$ is alive if all $x \in Q$ are alive, and A is alive if X is alive. We will assume this. The composition of two automata which share some common events operates as it would with each system in isolation except that when a shared event occurs, it *must* occur in both systems.

Let L be a regular language with minimal recognizer (A_L, x_0). Given a string $s \in L$, if $s = pqr$, then p is a prefix of s, $r \triangleq s/pq$ is a suffix of s, and q is a substring of s.

Definition 1.1 *Given L, $s \in L$* has an infinite extension *in L if for all integers $i \geq |s|$, there exists $r \in L$, $|r| = i$ such that s is a prefix of r. L is* prefix closed *if all the prefixes of any $s \in L$ are also in L. L is a* complete *language if each string in L has an infinite extension in L and L is prefix closed. For any L, we let L^c denote its prefix closure.*

In our development we will construct automata in which certain events can be *forced* to occur. It is straightforward to capture forced events in our present context.

Definition 1.2 *A state x is E-*pre-stable *if every trajectory starting from x passes through E. A state x is E-*stable *if A is alive and every state reachable from x is E-pre-stable. The DEDS is E-*stable *if every x is E-stable.*

A feedback map $K : X \to U$ yields a closed-loop system $A_K = (G, f, d_K, h, t)$ with

$$d_K(x) = (d(x) \cap K(x)) \cup (d(x) \cap \overline{\Phi}) \tag{1.4}$$

Definition 1.3 *A state x is E-*pre-stabilizable *(E-*stabilizable*) if there exists a K such that x is E-pre-stable (E-stable) in A_K. The DEDS is E-*stablilizable *if every x is.*

Definition 1.4 *A subset Q is* f-invariant *if $f(Q,d) \subset Q$ where $f(Q,d) = \bigcup_{x \in Q} f(x, d(x))$.*

The maximal stable set is the maximal f-invariant set in the maximal pre-stable set [4].

Definition 1.5 *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. 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.*

Given any set $V \subset X$, there is a maximal sustainably (f,u)-invariant subset W of V with a corresponding unique *minimally restrictive* feedback K.

A system is *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.

For any set $Q \subset X$ we define the reach of Q in A as:

$$R(A,Q) = \{y \in X \mid \exists x \in Q \text{ such that } x \to^* y\} \tag{1.5}$$

where $x \to^* y$ denotes that y is reachable from x. Let $Y = Y_0 \cup Y_1$, where

$$Y_0 = \{x \in X \mid \nexists y \in X, \sigma \in \Sigma, \text{ such that } x \in f(y,\gamma)\} \tag{1.6}$$

$$Y_1 = \{x \in X \mid \exists y \in X, \gamma \in \Gamma, \text{ such that } x \in f(y,\gamma)\} \tag{1.7}$$

Let $L(A, x)$ denote the event string language generated by A from the state $x \in X$. Also, let $L(A) = \bigcup_{x \in X} L(A, x)$.

In [1], we present an observer in which each observer estimate is a subset of Y corresponding to the set of possible states following the last observable event. The observer is a DEDS with state space $Z \subseteq 2^Y$ and with Γ as its set of events:

$$\hat{x}[k+1] = w(\hat{x}[k], \gamma[k+1]) \triangleq \bigcup_{x \in R(A|\overline{\Gamma}, \hat{x}[k])} f(x, \gamma[k+1]) \quad (1.8)$$

$$\gamma[k+1] \in v(\hat{x}[k]) \triangleq h(\bigcup_{x \in R(A|\overline{\Gamma}, \hat{x}[k])} d(x)) \quad (1.9)$$

In some cases, we treat the observer as a controlled system. A system is observable iff O is stable with respect to its singleton states [1]. Also a state is a *recurrent* if it can be reached by an arbitrarily long event string. Z_r denotes the set of recurrent states of O.

In [3], we define a compensator $C : X \times \Sigma^* \rightarrow U$ which specifies the set of controllable events that are enabled. The closed loop system A_C is the same as A but with

$$\sigma[k+1] \in d_C(x[k], s[k]) \triangleq (d(x[k]) \cap C(x[k], s[k])) \cup (d(x) \cap \overline{\Phi}) \quad (1.10)$$

where $s[k] = \sigma[0] \cdots \sigma[k]$ with $\sigma[0] = \epsilon$. As shown in [3] we can restrict attention to compensators which can be realized by finite state machines.

In [2] we define an output compensator as $C : \Gamma^* \rightarrow U$ so that

$$\sigma[k+1] \in d_C(x[k], s[k]) \triangleq (d(x[k]) \cap C(h(s[k]))) \cup (d(x) \cap \overline{\Phi}) \quad (1.11)$$

The following guarantees that our compensators preserve liveness.

Definition 1.6 *Given $Q \subset X$, $F \subset \Phi$, F is Q-*compatible *if for all $x \in R(A|\overline{\Gamma}, Q)$, $(d(x) \cap F) \cup (d(x) \cap \overline{\Phi}) \neq \emptyset$. An observer feedback $K : Z \rightarrow U$ is A-*compatible *if for all $\hat{x} \in Z$ $K(\hat{x})$ is $\hat{x}$-compatible. A compensator $C : \Gamma^* \rightarrow U$ is A-*compatible *if for all $s \in h(L(A))$, $C(s)$ is $\hat{\mathbf{x}}(s)$-compatible.*

Definition 1.7 *A is* output stabilizable *(*output pre-stabilizable*) with respect to E if there exists an output compensator C such that A_C is E-stable (E-*pre-stable*). We term such a compensator an* output stabilizing *(*output pre-stabilizing*)* compensator.

This definition implies that there exists an integer n_s such that the trajectories in A_C go through E in at most n_s observable transitions, with $n_s \leq q^3$ [2]. Also output pre-stabilizability and liveness are necessary and sufficient for output stabilizability [2].

The following properties relate to a system's ability to generate particular strings [3].

Definition 1.8 *Given $x \in X$ and a complete language L over Ξ, x is L-*restrictable *if there exists a compensator $C : X \times \Sigma^* \rightarrow U$ such that the closed loop system A_C is alive and $t(L(A_C, x)) \subset L$. Given $Q \subset X$, Q is L-*restrictable *if all $x \in Q$ are L-restrictable. Finally, A is L-*restrictable *if X is L-restrictable.*

Definition 1.9 *Given $x \in X$ and a complete language L over Ξ, x is* eventually L-restrictable *if there exists an integer n_a and a compensator $C : X \times \Sigma^* \rightarrow U$ such that the closed loop system A_C is alive and $t(L(A_C, x)) \subset (\Xi \cup \{\epsilon\})^{n_a} L$. Given $Q \subset X$, Q is* eventually L-restrictable *if all $x \in Q$ are eventually L-restrictable. Finally, A is* eventually L-restrictable *if X is eventually L-restrictable. Here $(\Xi \cup \{\epsilon\})^{n_a}$ denotes the set of strings over Ξ that have length at most n_a.*

Definition 1.10 *Given a complete language L over Ξ we say that A is* eventually L-restrictable by output feedback *if there exists an integer n_o and an output compensator $C : \Gamma^* \rightarrow U$ such that A_C is alive and for all $x \in X$, $t(L(A_C, x)) \subset (\Xi \cup \{\epsilon\})^{n_o} L$. Such a C is called an* L-restrictability compensator.

To test for this, let (A_L, x_0^L) be a minimal recognizer for L with state space Z_L. A'_L is the same as A_L except that $Z'_L = Z_L \cup \{b\}$ where b is a state used to signify that the event trajectory is no longer in L. Also, $d'_L(x) = \Xi$ for all $x \in Z'_L$, and

$$f'_L(x, \sigma) = \begin{cases} f_L(x, \sigma) & \text{if } x \neq b \text{ and } \sigma \in d_L(x) \\ \{b\} & \text{otherwise} \end{cases} \tag{1.12}$$

Let O be the observer for A, $A(L) = A \parallel A'_L$, and $O(L) = (G(L), w_L, v_L)$ the observer for $A(L)$; since we know that we will start A'_L in x_0^L, we take the state space of $O(L)$ as $Z(L) = R(O(L), \{\{x_0^L\} \times \hat{x} | \hat{x} \in Z\})$. Let $V_o = \{\hat{z} \in Z(L) |$ for all $(x_L, x_A) \in \hat{z}, x_L \neq b\}$ Let $E(L)$ be the largest subset of V_o 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 $\hat{x}(\hat{z}) = \{x \in X | \exists x_L \in Z_L$ such that $(x_L, x) \in \hat{z}\}$. The construction of $E(L)$ and K^{EL} is a slight variation of the algorithm in [4] for the construction of maximal sustainably (f,u)-invariant subsets. Consider next the set

$$E_o(L) = \{\hat{x} \in Z | x_0^L \times \hat{x} \in E(L)\} \tag{1.13}$$

Proposition 1.11 *A is eventually L-restrictable by output feedback iff there exists an A-compatible feedback $K : Z \rightarrow U$ such that the closed loop system O_K is $E_o(L)$-pre-stable.*

2 Characterizing Higher-Level Models

We now consider higher-level modelling based on a given set of primitives, each of which consists of a finite set of tracking event strings. Given alphabets Σ' and Ξ, a *primitive map* $H_e : \Sigma' \rightarrow 2^{\Xi^*}$, is such that for all $\sigma \in \Sigma'$ $H_e(\sigma)$ is a collection of *finite* length strings. Here $\sigma \in \Sigma'$ is the macroscopic event corresponding to the *set* of tracking strings $H_e(\sigma)$ in the original model. Given H_e we extend it to act on strings over Σ'.

Definition 2.1 *A primitive map H_e is termed* minimal *if for all, not necessarily distinct, $\sigma_1, \sigma_2 \in \Sigma'$ and for all $s \in H_e(\sigma_1)$, no proper suffix of s is in $H_e(\sigma_2)$.*

Proposition 2.2 *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 of r_2, $\Xi^* H_e(r_1) \cap \Xi^* H_e(r_2) = \emptyset$.*

Given A and A', we wish to specify when A' is an *H_e-model* of A, where $H_e : \Sigma' \rightarrow \Xi$ is a minimal primitive map. Two important properties that we require are:

Restrictability: If we can restrict the behavior of the macroscopic model to some complete language $L \subset \Sigma'^*$, then we can also restrict the original system to $H_e(L)^c$.

Detectability: For any lower-level string s in $L(A)$ such that $t(s)$ is in $H_e(p)$ for some string p in the macroscopic system, then (a) we can reconstruct p, after some delay, using the lower-level observation $h(s)$ of s; and (b) for *any* string r so that s is a suffix of r, the reconstruction acting on $h(r)$ results in a string that ends with the reconstruction of $h(s)$. Thanks to minimality, $H_e^{-1}(t(s))$ is single valued. Thus, in order to satisfy the first condition of detectability, we need to be able to reconstruct $H_e^{-1}(t(s))$ from $h(s)$. What (b) requires is that the reconstruction can recognize and "reject" finite length start-up strings that do not correspond to any primitive.

Proposition 2.3 *If A' is an H_e-model of A then for any compensator $C' : \Gamma'^* \to U'$ for A', there exists $C : \Gamma^* \to U$ for A such that $A'_{C'}$ is an H_e-model of A_C with the same H_o.*

3 Aggregation

Suppose that our system is capable of performing a set of primitive tasks. What we would like to do 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.

Let $\mathbf{T}$ be the index set of a collection of tasks, i.e., for any $i \in \mathbf{T}$ there is a finite set L_i of strings over Ξ that represents task i. We let $L_T = \cup_{i \in \mathbf{T}} L_i$.

Definition 3.1 *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 which substring corresponds to which task. Note that if $\mathbf{T}$ is independent, then the minimal recognizer for all of L_T has a single final state as does the minimal recognizer for each L_i.

Definition 3.2 *A task $i \in \mathbf{T}$ is* reachable *(by output feedback) if A is eventually L_i^{*c}-restrictable (by output feedback). $\mathbf{T}$ is* reachable *(by output feedback) if each $i \in \mathbf{T}$ is.*

Given a task $i \in \mathbf{T}$ reachable by output feedback, let $C_i : \Gamma^* \to U$ be an L_i^{*c}-restrictability compensator. Note that states in $E_o(L_i^{*c})$ are guaranteed to generate a sublanguage of L_i^{*c} in the closed loop system. However, for any other state $\hat{x} \in Z$, 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 before the trajectory in O reaches $E_o(L_i^{*c})$. If this happens, then task j will have been completed while the compensator was trying to set-up the system for task i. The following requires that this cannot happen:

Definition 3.3 *An L_i^{*c}-restrictability compensator C_i for a reachable $i \in \mathbf{T}$ is* consistent *with $\mathbf{T}$ if for all $\hat{x} \in Z_r \cap \overline{E_o(L_i^{*c})}$, for all $x \in \hat{x}$, and for all $s \in L(A_{C_i}, x)$, $t(s) \notin L_T$.*

Consider testing the existence of and constructing consistent restrictability compensators. Note that we only need to worry about forcing the trajectory in O into $E_o(L_i^{*c})$ *without* completing any task along the way. Once that is done, restricting the behavior can be achieved by the compensator defined in Proposition 1.11. First, we need a mechanism to recognize that a task is completed. Thus, let (A_T, x_0) be a minimal recognizer for L_T with the final state x_f and state space X_T. We add a new state, g, to the state space of A_T, and for each event that is not previously defined at states in X_T we define a transition to state g. 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 $A \parallel A'_T$ with state space Z'. The initial states of O' are $Z'_0 = \{\hat{x} \times \{x_0\} | \hat{x} \subset Z_r\}$ so that $Z' = R(O', Z_0)$. Let $p : Z' \to Z_r$ be the projection of Z' into Z_r, i.e., $p(\hat{z}) = \bigcup_{(x_1,x_2)\in\hat{z}}\{x_1\}$. Also, let $E'_o = \{\hat{z} \in Z' | p(\hat{z}) \in E_o(L_i^{*c})\}$. Our goal is to reach E'_o from the initial states Z'_0 while avoiding the completion of any task. So, we remove all transitions from states in E'_o and instead create self loops in order to preserve liveness. Let $O'' = (G', w'', v'')$ represent the modified automaton. Consider the set of states in which we need to keep the trajectory, i.e. those that cannot correspond to a completion of any task:

$$E'' = \{\hat{z} \in Z' | \forall (x_1, x_2) \in \hat{z}, x_2 \neq x_f\} \tag{3.1}$$

Let V' be the maximal (f,u)-invariant subset of E', and let $K^{V'}$ be 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'_o while keeping them in V'. Thus, we need to find $K'' : Z' \to U$ so that Z' is E'_o-pre-stable in $O''_{K^{V'}}$ and so that the combined feedback $K(\hat{z}) = K^{V'}(\hat{z}) \cap K''(\hat{z})$ for all $\hat{z} \in Z'$ is A-compatible. The construction of such a K, if it exists, proceeds much as in Section 2. Since $K^{V'}$ is unique, if we cannot find such a feedback, then a consistent restrictability compensator cannot exist. Let us assume that a consistent compensator exists.

Let us outline how we construct a compensator C_i for task i: Given an observation sequence s, let $\hat{x}$ be the current state of O. There are three possibilities:

1. Suppose that $\hat{x} \notin Z_r$ and the trajectory has not entered $E_o(L_i^{*c})$. Then, we use O and an $E_o(L_i^{*c})$-pre-stabilizing feedback to construct $C_i(s)$ as in Proposition 1.11.

2. Suppose that $\hat{x} \in Z_r$ and the trajectory has not entered $E_o(L_i^{*c})$. Let $\hat{x}'$ be the state in 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 $\hat{x}' \times x_o$. Suppose that s/s' takes $\hat{x}' \times x_0$ to $\hat{z}$ in O''. Then, with K as defined above

$$C_i(s) = (v''(\hat{z}) \cap K(\hat{z})) \cup (v''(\hat{z}) \cap \overline{\Phi}) \tag{3.2}$$

3. On entering $E_o(L_i^{*c})$, we switch to using $O(L_i^{*c})$ and the (f,u)-invariance feedback $K^{L_i^{*c}}$.

Given a set of p tasks $\mathbf{T}$, reachable by output feedback, let $C_i : \Gamma^* \to U$ denote the compensator corresponding to task i. The overall compensator C that we construct admits events corresponding to requests for tasks as inputs and switches between C_i. In order to model this, we use an automaton illustrated in Figure 3.1, which has p states, where state i corresponds to using the compensator C_i to control A. For each i, τ_i^F is a forced event, corresponding to switching to C_i. Let $\Phi_T = \{\tau_1^F, \ldots, \tau_p^F\}$ and $U_T = 2^{\Phi_T}$. The input to C is a subset of Φ_T, representing the set of requested tasks. Suppose that 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; (2) if the input contains τ_i^F, then C continues performing

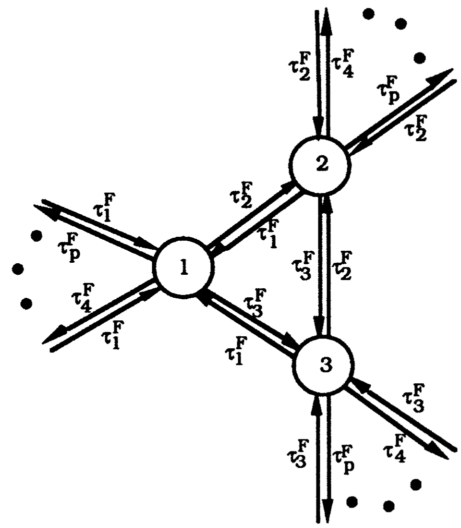

Figure 3.1: An Automaton to Construct C

task i; (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, initializing the corresponding task compensator.

We define a notion of observability for tasks after an initial start-up transient.

Definition 3.4 *A task $i \in \mathbf{T}$ is* observable *if there exists a function $\mathcal{I} : Z_r \times L(O, Z_r) \rightarrow \{\epsilon, \psi_i^F\}$ so that for all $\hat{x} \in Z_r$ and for all $x \in \hat{x}$, $\mathcal{I}$ satisfies*

1. *$\mathcal{I}(\hat{x}, h(s)) = \psi_i^F$ for all $s \in L(A, x)$ such that $s = p_1p_2p_3$ with $t(p_2) \in L_i$, and*
2. *$\mathcal{I}(\hat{x}, h(s)) = \epsilon$ for all other $s \in L(A, x)$.*

We construct a test for the observability of task i assuming that it is reachable and that we are given an L_i^{*c}-restrictability compensator C_i which is consistent with $\mathbf{T}$. Furthermore, thanks to consistency, we only need to construct $\mathcal{I}$ for $\hat{x} \in E_o(L_i^{*c})$ and for strings s such that $t(s) \in L_i^{*c}$. First, 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 $A'_{L_i} \parallel A$. Finally, let $O_Q = (F_Q, w_Q, v_Q)$ be the observer for Q with state space Z_Q that is the reach of

$$Z_{Q0} = \bigcup_{\hat{x} \in E_o(L_i^{*c})} (\{x_0^{L_i}\} \times \hat{x}) \cap X_Q \tag{3.3}$$

in O_Q. Note that if i is observable, then the last event of each string in L_i must be an observable event. In this case, let

$$E_Q = \{\hat{z} \in Z_Q | \exists (x, y) \in \hat{z} \text{ such that } x = x_f^{L_i}\} \tag{3.4}$$

Given the observations on A_{C_i}, at some point in time O enters some state $\hat{x} \in E_o(L_i^{*c})$, and 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

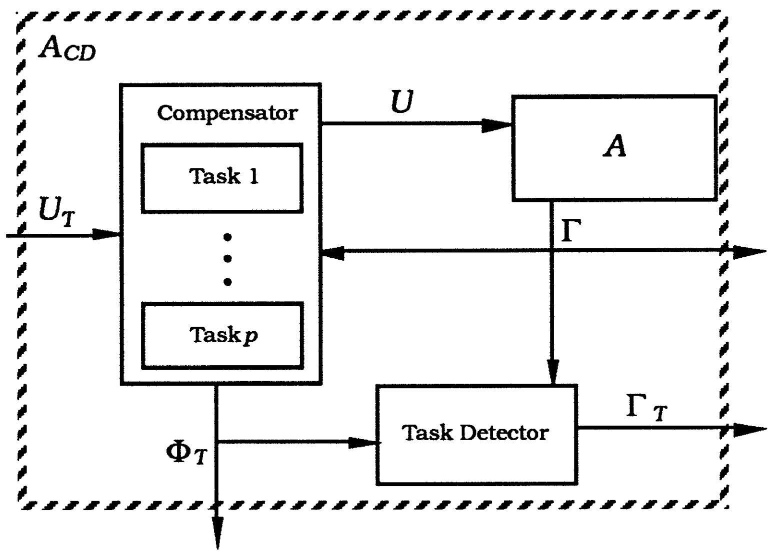

Figure 3.2: The Task-Level Closed-Loop System

enters some $\hat{z} \in E_Q$ at which point we know that task i *may* have been completed. 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 take $\mathcal{I}$ to be ϵ until the trajectory in O_Q enters E_Q and ψ_i^F from that point on.

Suppose that O enters the state $\hat{y}$ when O_Q enters E_Q. Note that $\hat{y} \in E_o(L_i^{*c})$. At this point we detect the first occurrence of task i. In order to detect the next occurrence we immediately re-start O_Q at state $x_0^{L_i} \times \hat{y} \cap X_Q$. The procedure continues in this fashion. 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 consisting of the observer O, the system O_Q which is re-started when a task is detected, and a one-state automaton with self-transition event ψ_i^F, which occurs whenever a task is detected and which is the only observable event for D_i^*. Finally, we define a task detector D from the set of individual 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. The output of D takes values in $\Gamma_T = \{\psi_1^F, \ldots, \psi_p^F\}$.

Figure 3.2 depicts the overall system $A_{CD} = (G_{CD}, f_{CD}, d_{CD}, t_{CD}, h_{CD})$ with

$$G_{CD} = (X_{CD}, \Sigma \cup \Phi_T \cup \Gamma_T, \Phi \cup \Phi_T, \Gamma \cup \Phi_T \cup \Gamma_T, \Xi \cup \Phi_T) \tag{3.5}$$

Note that Φ_T and Γ_T are observable and Φ_T is controllable. We include Φ_T in the tracking events to mark the fact that the system has switched compensators. Also, we impose the restriction (which can be realized since tasks are observable) that events in Φ_T can only be forced right after task completion. Then, A_{CD} can only generate strings s such that

$$t(s) \in (\Xi \cup \{\epsilon\})^{n_t}(L_1^* \cup \cdots \cup L_p^*)(H_e(\tau_1)L_1^* \cup \cdots \cup H_e(\tau_p)L_p^*)^* \tag{3.6}$$

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

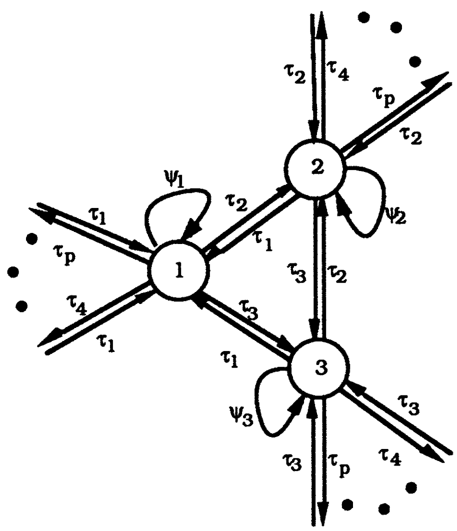

Figure 3.3: Task Standard Form: All events are controllable and observable.

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 is an external command. The task-level behavior of A_{CD} can be modelled by $A_{TSF} = (G_{TSF}, f_{TSF}, d_{TSF})$ of Figure 3.3 where all the events are controllable and observable. We term A_{TSF} the *task standard form.*

We first define $H_e(\epsilon) = \epsilon$ and $H_e(\psi_i) = L_i$. Thanks to the independence of $\mathbf{T}$, for any pair of not necessarily distinct tasks i and j, no suffix of string in $H_e(\psi_i)$ can be in $H_e(\psi_j)$. Defining $H_e(\tau_i)$ we must consider two issues:

1. 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 recurrent 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 recurrent state of O up to and including the transition that takes the trajectory to a state in $E_o(L_i^{*c})$, then $H_e(\tau_i) \subseteq \tau_i^F(\Xi \cup \{\epsilon\})^{n_e}$.

2. We also need to ensure the minimality of H_e. Specifically, 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 $H_e(\tau_i) = (\Xi \cup \{\epsilon\})^{n_e} \cap \overline{(\Xi \cup \{\epsilon\})^{n_e} L_T}$.

Proposition 3.5 *A_{TSF} is an H_e-model of A_{CD}.*

The formalism we have described can be applied to obtain a hierarchy of aggregate models in which words (i.e., tasks) at one level are translated into letters at the next level. In addition, in [5] we develop system-wide task-level models from local task models and consider higher-level coordinated control of the entire system.

ACKNOWLEDGEMENT

The research reported in this paper was supported in part by AFOSR under Grant AFOSR-88-0032 and by the Army Research Office under Grant DAAL03-86-K0171.

References

[1] C. M. Özveren and A. S. Willsky, "Observability of discrete event dynamic systems," *IEEE Transactions on Automatic Control*, to appear.

[2] C. M. Özveren and A. S. Willsky, "Output stabilizability of discrete event dynamic systems," submitted to *IEEE Transactions on Automatic Control.*

[3] C. M. Özveren and A. S. Willsky, "Tracking and restrictability in discrete event dynamic systems," submitted to *SIAM J. on Control and Opt.*

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

[5] C. M. Özveren, "Analysis and Control of Discrete Event Dynamic Systems: A State Space Approach," Ph.D. Thesis, M.I.T., Aug. 1989.