MODEL AGGREGATION FOR HIERARCHICAL CONTROL SYNTHESIS OF DISCRETE EVENT SYSTEMS

Haoxun Chen[1] Hans-Michael Hanisch[2]
Department of Electrical and Computer Engineering
University of Connecticut, Storrs, Connecticut 06269, USA[1]
Deptartment of Engineering Science
Martin Luther University of Halle-Wittenberg, Halle, D-06099, Germany[2]
{hxchen@engr.uconn.edu[1], Hans-Michael.Hanisch@iw.uni-halle.de[2]}

ABSTRACT. A model aggregation approach is developed for discrete event systems described by condition/event automata, a class of condition/event systems. It is based on input-output equivalence of the systems. With the approach, a higher level model of a plant in the hierarchical control synthesis of discrete event systems can be derived from a lower level model by hiding insignificant states and events while keeping the input-output behaviour of the plant invariant in view of the higher level.

KEYWORDS: Discrete event system, hierarchical control, condition/event system, automaton, model aggregation.

1. INTRODUCTION

Hierarchical structure has long been adopted as a control structure for complex systems. It may be described as a division of control action and information according to scope. The scope of a control action may be defined by the extent of its temporal horizon, or by the depth of its logical dependency in a task decomposition.

Previous work on hierarchical control of discrete event systems (DES) is based on the supervisory control framework of Ramadge and Wonham [5]. In [8], hierarchical structure for control of DESs is formalized, which embodies a low-level real-world model controlled by an operator and a high-level abstract model ‘‘virtually’’ controlled by a manager. The Concept of hierarchical consistency is proposed, relating high-level behaviour required by the manager to low-level behaviour achievable by the operator. In [3] [7], decentralized supervision and coordination of DESs are studied for partially observed discrete event systems, but their scheme is based on a single model of the hierarchical systems. In [1] [2], a hierarchical control theory for supervisory automata is developed based on state set partition and state aggregation. Conditions are determined on state partitions which ensure that the control of transitions between partition blocks in a high level aggregated model, combined with local state-dependent controls, is effective in the sense of achieving specifications given either for the high level model or for the low level model. The work [4] takes a similar framework but is based on formal language. Structured languages are proposed to describe the behaviours of a system at different levels in a hierarchy in a top-down way. Language containment and controllability are defined recursively and an algorithm for the computation of supremal controllable sublanguages is provided.

In this paper, condition/event automata, a class of condition/event systems, are proposed for the modeling of DESs and a model aggregation approach is developed for the systems based on input-output equivalence of condition/event systems. We take condition/event systems rather than finite state automata as a model of DESs because with the modeling framework, both plant and its controller (supervisor) can be symmetrically described by condition/event systems, the interactions between the plant and the controller can be easily described by condition and event signals, and the closed system obtained by coupling the plant with the controller is also a condition/event system. Moreover, condition/event systems provide a modular and input-output modeling framework for DESs and can describe both spontaneous and forced events. The approach developed can be used in the hierarchical of DESs with which a higher level model of a plant can be derived from a lower level model by hiding insignificant states and events while keeping the input-output behaviour of the plant invariant in view of the higher level. In addition, a bottom-up hierarchical control synthesis methodology for DESs is proposed under the framework of condition/event automata and key technical issues for the methodology are addressed.

2. CONDITION/EVENT AUTOMATA

Condition/event systems were introduced in [6] as a framework for modeling and analyzing interconnected discrete event systems where both event and condition signals are presented. The main attractions of this framework to control of discrete event systems are that it is a modular modeling paradigm and that it can describe both spontaneous and forced events. In the following, we introduce a class of condition/event systems, called condition/event automata, which have a finite state automaton realization and a specific input-output structure.

Definition 1: A *condition/event automaton* (CEA) is defined as a tuple (U, V, Y, Z, A, I, O), where

- U is the set of *input condition signals*,
- V is the set of *input event signals*,
- Y is the set of *output condition signals*,
- Z is the set of *output event signals*,

- $A = (Q, \Sigma, \delta, q_0)$ is a finite state automaton, where Q is a set of states, Σ is a set of event labels, δ: $Q \times \Sigma \rightarrow Q$ is the transition function (a partial function), $q_0 \in Q$ is the initial state,
- $I = (I_C, I_E)$ is the *input function*, where I_C: $Q \times \Sigma \rightarrow 2^U$ is the *condition input function*, I_E: $Q \times \Sigma \rightarrow 2^V$ is the *event input function*,
- $O = (O_C, O_E)$ is the *output function*, where O_C: $Q \rightarrow Y \cup \{\varnothing\}$ is the *condition output function*, O_E: $Q \times \Sigma \rightarrow Z \cup \{\varnothing\}$ is the *event output function*.

All condition and event signals in the above definition are supposed to be binary-valued. The two values are 1 (true) and 0 (false). When time is explicitly considered in the model, an input condition signal $u \in U$ is denoted by $u(.)$, U is denoted by $U(.)$. The same convention applies to output condition signal, input and output event signals. Without loss of generality, we assume the values of all input condition signals are zero initially.

The state of the CEA is given by the state of the automaton A. At state $q \in Q$ and given values of the input condition and event signals, an event $\sigma \in \Sigma$ is state enabled if $\delta(q,\sigma)$ is defined, condition enabled if all input condition signals in the set $I_C(q,\sigma)$ have the value 1, event enabled if all input event signals in the set $I_E(q,\sigma)$ have the value 1. An event σ can occur if it is state, condition and event enabled. It is forced to occur immediately if it is forced by an input event signal (i.e., $I_E(q,\sigma) \neq \varnothing$).

An output condition signal $y \in Y$ takes the value 1 if and only if there is $q \in Q$ such that $O_C(q) = y$ and the CEA (the automaton A) is at state q. An output event signal $z \in Z$ takes the value 1 at the instant that an event σ occurs at a state q such that $O_E(q,\sigma) = z$, otherwise it takes the value 0.

A pair (q,σ) is called a transition if $\delta(q,\sigma)$ is defined, $T = \{ (q,\sigma) \mid q \in Q, \sigma \in \Sigma, \delta(q,\sigma) \text{ is defined} \}$ is the set of all transitions of A (the CEA). All input condition and event signals, all output event signals are associated with the transitions, all output condition signals are associated with the states of A.

Figure 1 gives examples of CEAs, where two different types of arcs are used to represent condition and event signals, respectively. For instance, in (a), u and y are condition signals, v and z are event signals. To make the connection between a signal and a transition much clear, we add a bar (a short line) crossing the transition at the connection point as we have done in graphical representation of Petri nets. Some transitions may be connected with both input event and condition signals as shown in (b). Moreover, some input condition or event signals may be connected with more than one transitions, and more than one states (resp. transitions) may be connected with one output condition (resp. event) signal as shown in (c).

For a finite event or signal set X, let $\{0,1\}^X$ denote the set of all mappings of $X \rightarrow \{0,1\}$. Given a CEA S, an input-output trajectory of S is a mapping (time function) $(U(.), V(.), Y(.), Z(.))$: $[0,\infty) \rightarrow \{0,1\}^U \times \{0,1\}^V \times \{0,1\}^Y \times \{0,1\}^Z$ (i.e., $U(.)$: $[0,\infty) \rightarrow \{0,1\}^U$, $V(.)$: $[0,\infty) \rightarrow \{0,1\}^V$, ...) such that at every time $0 \leq t < \infty$, the values of input condition signals U, input event signals V, output condition signals Y and output event signals Z are $U(t)$, $V(t)$, $Y(t)$ and $Z(t)$, respectively.

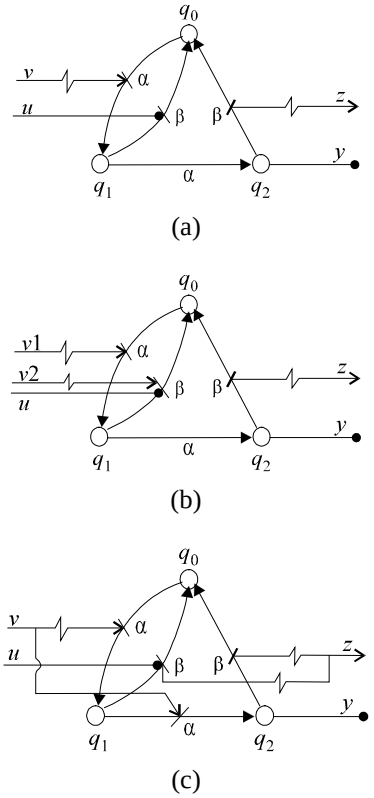


Figure 1: Condition/event automata

Definition 2 (Interconnection of CEAs): Given two CEAs $S_i = (U_i, V_i, Y_i, Z_i, A_i, I_i, O_i)$, i=1,2, with $U_1 \cap U_2 = V_1 \cap V_2 = Y_1 \cap Z_2 = Z_1 \cap Z_2 = \varnothing$, the interconnection of S_1 and S_2 is defined by a *condition connection mapping* (*matrix*) C: $(Y_1 \cup Y_2) \times (U_1 \cup U_2) \rightarrow \{0,1\}$ and *an event connection mapping* (*matrix*) E: $(Z_1 \cup Z_2) \times (V_1 \cup V_2) \rightarrow \{0,1\}$, where $C(y,u)$=1 (resp. 0) means that output condition signal y is (resp. is not) connected with input condition signal u, $E(z,v)$=1 (resp. 0) means that output event signal z is (resp. is not) connected with input event signal v. The resultant system is an interconnected CEA.

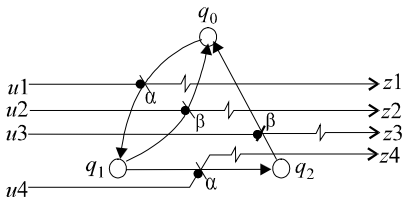
In the interconnected CEA, there may be multiple output condition (resp. event) signals connected with one input condition (resp. event) signal and one output condition (resp. event) signal connected with multiple input condition (resp. event) signals. For the first case, the input condition (resp. event) signal takes the value 1 if and only if one of the output condition (resp. event) signals takes the value 1, i.e., the input condition (resp. event) signal is the disjunction of the output condition (resp. event) signals. For the second case, all of the input condition (resp. event) signals take the same value as the output condition (resp. event) signal. The interconnected CEA, represented by

$S_1{}^C\|^E S_2$ or $S_1\|S_2$ for simplicity, is also a CEA because the mappings C and E can be absorbed into the definitions of automaton A, input function I, and output function O in a condition/event automaton realization for $S_1\|S_2$.

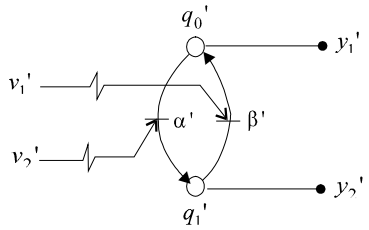
Figure 2 shows two CEAs and their interconnected CEA, where the interconnected CEA in (c) is obtained by connecting the CEA in (a) with the CEA in (b). The condition connection mapping and the event connection mapping are as follows:

$C(y_1', u_1) = C(y_1', u_4) = C(y_2', u_2) = C(y_2', u_3) = 1$, otherwise $C(y', u) = 0$;

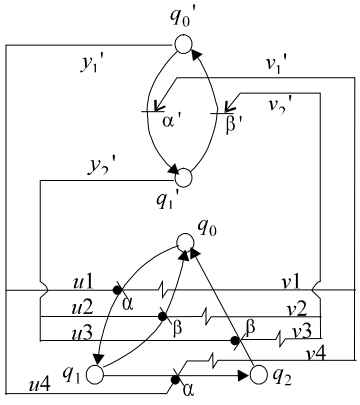
$E(z_1, v_1') = E(z_4, v_1') = E(z_2, v_2') = E(z_3, v_2') = 1$, otherwise $E(z, v') = 0$.



(a)



(b)



(c)

Figure 2: Interconnection of CEAs

The above definition of interconnection can be extended to more than two CEAs.

In [6], C/E languages for C/E systems are introduced in order to study the systems in a purely discrete framework. Similarly, we can define C/E languages for CEAs. However, we take a different formulation by taking account of the binary-value nature of the condition and event signals of the CEAs.

For the set of input condition signals U and the set of output condition signals Y, we introduce two additional event sets (or event signal sets) $U_e = U^+ \cup U^- = \{ u^+ \mid u \in U\} \cup \{ u^- \mid u \in U\}$ and $Y_e = Y^+ \cup Y^- = \{ y^+ \mid y \in Y\} \cup \{ y^- \mid y \in Y\}$, where for each $c \in U \cup Y$, c^+ and c^- denote the event that the value of the condition signal c changes from 0 to 1 and the event that the value changes from 1 to 0, respectively. For a finite set X, let X^* denote the set of all strings of elements in X.

Definition 3: Given a CEA S, the C/E language for the system S, denoted by L_S, is a subset of $(U_e \cup Y_e \cup V \cup Z)^*$, given by

$L_S = \{ w = \sigma_1\sigma_2... \in (U_e \cup Y_e \cup V \cup Z)^* \mid$ there are a nondecreasing time sequence $\{t_i\}$ and an input-output trajectory $(U(.), V(.), Y(.), Z(.))$ of S such that the event signal σ_i comes true at time t_i if $\sigma_i \in V \cup Z$, or the value of the condition signal corresponding to σ_i changes from 0 to 1 if $\sigma_i \in U^+ \cup Y^+$ (resp. changes from 1 to 0 if $\sigma_i \in U^- \cup Y^-$) at time t_i^-, and no changes on the condition and event signals happen at any other time $\}$.

Note that the time involved in the above definition is not essential since the C/E language is only a description of the logic behaviour of the CEA.

For instance, $vu^+u^-vu^+u^-...$, $vy^+zy^-vy^+zy^-...$, $vu^+u^-vy^+zy^-vu^+u^-vy^+zy^-...$ are three words of the C/E language L_S for the CEA in Figure 1(a).

In [6], an equivalence relation was defined for C/E systems by means of C/E languages in order to determine whether two C/E models S_1 and S_2 characterize the same systems. Similarly, we have:

Definition 4: Two CEAs S_1 and S_2 are said to be input-output equivalent if $L_{S_1} = L_{S_2}$.

Figure 3 gives a CEA that is input-output equivalent to the CEA in Figure 1(a).

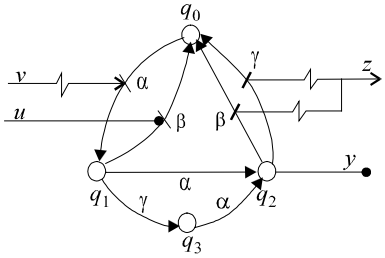


Figure 3

3. MODEL AGGREGATION

Model aggregation has long been known as an approach for simplifying the analysis and synthesis of large-scale systems in continuous system theory. The basic idea is to derive a simplified model of a system with less state variables (or states) from an original model while keeping the input-output behaviours of the two models the same.

Given a CEA S, we want to find a simplified CEA model $\hat{S}$ from S by model aggregation such that $\hat{S}$ has

less states and is input-output equivalent to S. First we will consider the case that S is a CEA as defined in Definition 1 and then consider the case that S is an interconnected CEA consisting of several CEA modules.

3.1. Singular-Module CEAs

For a singular-module CEA, let $Q^s = \{ q\in Q \mid O_C(q) \neq \varnothing \}$, $T = \{ (q,\sigma) \mid q\in Q, \delta(q,\sigma)$ is defined$\}$, $T^s = \{ (q,\sigma) \mid \exists(q,\sigma)\in T, I_C(q,\sigma) \neq \{\varnothing\} \vee I_E(q,\sigma)\neq\{\varnothing\} \vee O_E(q,\sigma) \neq \varnothing \}$, and $T^{us} = T \setminus T^s$. $L(A) = \{ w\in \Sigma^* \mid \delta(q_0,w)$ is defined $\}$ is the language generated by A.

We say that a transition (q,σ) of A in S is controlled by an input condition (event) signal u (v) if $u\in I_C(q,\sigma)$ ($v\in I_E(q,\sigma)$), is observed by an output event signal z if $z\in O_E(q,\sigma)$. T^s is the set of controllable or observable transitions, T^{us} is the set of uncontrollable and unobservable transitions.

Define an equivalence relation, denoted R_{O_C}, on Q:

$$\forall q,\ q'\in Q:\ q \cong q'\,(R_{O_C}) \Leftrightarrow [O_C(q) = O_C(q')]$$

For each $q\in Q$, $[q](R_{O_C}) \subseteq Q$ is used to denote the equivalence class containing the state q. Note that all states not in Q^s form an equivalence class of R_{O_C}.

Two transitions (q,σ), (q',σ') are said to have the same control and observation structure if $I_C(q,\sigma) = I_C(q',\sigma') \wedge I_E(q,\sigma)=I_E(q',\sigma') \wedge O_E(q,\sigma)=O_E(q',\sigma')$.

Define an equivalence relation, denoted $R_{I,O}$, on T:

$$\forall (q,\sigma), (q',\sigma')\in T: (q,\sigma) \cong (q',\sigma')\,(R_{I,O}) \Leftrightarrow$$

((q,σ) and (q',σ') have the same control and observation structure].

For each $(q,\sigma)\in T$, $[q,\sigma](R_{I,O}) \subseteq T$ is used to denote the equivalence class containing the state (q,σ). Note that all transitions not in T^s form an equivalence class of $R_{I,O}$.

From A, we construct an automaton A^s as follows:

$A^s = (Q, \Sigma^s, \delta^s, q_0)$
$\Sigma^s = \{ [q,\sigma](R_{I,O}) \mid (q,\sigma)\in T \}$,
$\delta^s : Q\times\Sigma^{s+}\to 2^Q$,
$\delta^s (q, [q',\sigma'](R_{I,O})) = \{\delta(q,\sigma) \mid (q,\sigma)\in [q',\sigma'](R_{I,O}) \wedge (q,\sigma)\in T \}$, $(q',\sigma')\in T$.

The automaton is obtained by relabelling the transitions of A according to the equivalent relation $R_{I,O}$.

Note that A^s may be a undeterministic automaton. For simplicity, we assume that A^s is a deterministic automaton hereafter, but the following results also apply to the case when A^s is undeterministic.

Given a language $K\subseteq \Sigma^{s*}$, it induces an equivalence relation, denoted R_K, on Σ^{s*}:

$$\forall s, t\in\Sigma^{s*}: s \cong t\,(R_K) \Leftrightarrow [K\setminus\{s\} = K\setminus\{t\}].$$

where for $w\in\Sigma^{s*}$, $K\setminus\{w\} := \{ r\in\Sigma^{s*} \mid wr\in K \}$.

Let $L(A^s)$ be the language generated by A^s. We have $L(A^s) = \theta(L(A))$, where θ is recursively defined as follows:

$\theta(\varepsilon) = \varepsilon$ (ε denotes the empty string),
$\theta((q,\sigma)) = [q,\sigma](R_{I,O})$,
$\theta(s(q,\sigma)) = \theta(s)\theta((q,\sigma))$.

For each $q\in Q^s$, define

$$L_m([q]) = L_m(A^s, [q](R_{O_C})) = \{ w\in\Sigma^{s*} \mid \delta^s(q_0,w) \text{ is defined and } \delta^s(q_0,w)\in[q](R_{O_C}) \}$$

$L_m([q])$ is the language marked by A^s, suppose that the set of marked states of A^s is taken as $[q](R_{O_C})$.

Define an equivalence relation, denoted R_S, on Σ^{s*}:

$$\forall s, t\in\Sigma^{s*}: s \cong t\,(R_S) \Leftrightarrow [s \cong t\,(R_{L(A^s)})] \wedge [\forall q\in Q^s, s \cong t\,(R_{L_m([q])})]$$

For each $s\in \Sigma^{s*}$, $[s](R_S) \subseteq \Sigma^{s*}$ is used to denote the equivalence class containing the string s.

Proposition 1: The number of the equivalence classes of R_S is finite, i.e., R_S is of finite index.

Proof: It can be derived from the facts: 1) $\forall s, t\in\Sigma^{s*}$, $\delta^s(q_0,s) = \delta^s(q_0,t) \Rightarrow s \cong t\,(R_S)$; 2) the number of states of A^s is finite.

Proposition 2: The equivalence relation R_S is a right congruence on Σ^{s*}, i.e., $\forall s, t, u \in \Sigma^{s*}$: $s \cong t\,(R_S) \Rightarrow su \cong tu\,(R_S)$.

It can be derived from the right congruence of both $R_{L(A^s)}$ and $R_{L_m([q])}$.

Proposition 3: For any $s, t\in\Sigma^{s*}$, $[s](R_S) = [t](R_S) \Rightarrow O_C(\delta^s(q_0,s)) = O_C(\delta^s(q_0,t))$.

Proof: Case 1: one of s, t, say, s such that $\delta^s(q_0,s) = q\in Q^s$. For this case, we also have $\delta^s(q_0,s) = q$ since $s \cong t(R_{L_m([q])})$ and $\varepsilon\in R_{L_m([q])}\setminus\{s\}$. Case 2: $\delta^s(q_0,s)\notin Q^s$, $\delta^s(q_0,t)\notin Q^s$. For this case, we have $O_C(\delta^s(q_0,s)) = O_C(\delta^s(q_0,t)) = \varnothing$.

Proposition 4: For any $(q',\sigma')\in[q,\sigma](R_{I,O})$, $I_C(q',\sigma') = I_C(q,\sigma)$, $I_E(q',\sigma') = I_E(q,\sigma)$ and $O_E(q',\sigma') = O_E(q,\sigma)$.

This result directly follows the definition of $R_{I,O}$.

Construct a CEA $\hat{S} = (U, V, Y, Z, \hat{A}, \hat{I}, \hat{O})$ as follows:

$$\hat{A} = (\hat{Q}, \Sigma^s, \hat{\delta}, \hat{q}_0),$$

where

$\hat{Q} = \{ [s](R_S) \mid s\in\Sigma^{s*} \}$,
$\hat{\delta}([s](R_S),\sigma) = [s\sigma](R_S)$, $\forall [s](R_S)\in\hat{Q}$, $\sigma\in\Sigma^s$,
$\hat{q}_0 = [\varepsilon](R_S)$ (ε denotes the empty string),
$\hat{I} = (\hat{I}_C, \hat{I}_E)$,
$\hat{I}_C : \hat{Q}\times\Sigma^s \to 2^U$, $\hat{I}_C(\hat{q},[q,\sigma](R_{I,O})) = I_C(q,\sigma)$,

for any $(\hat{q}, [q,\sigma](R_{I,O})) \in \hat{T}$, where $\hat{T} = \{ (\hat{q}, \hat{\sigma}) \in \hat{Q} \times \Sigma^s \mid \hat{\delta}(\hat{q}, \hat{\sigma}) \text{ is defined}\}$,

$\hat{I}_E : \hat{Q} \times \Sigma^s \to 2^V$, $\hat{I}_E(\hat{q}, [q,\sigma](R_{I,O})) = I_E(q,\sigma)$, for any $(\hat{q}, [q,\sigma](R_{I,O})) \in \hat{T}$,

$\hat{O} = (\hat{O}_C, \hat{O}_E)$,

$\hat{O}_C : \hat{Q} \to Y \cup \{\varnothing\}$, $\hat{O}_C([s](R_S)) = O_C(\delta^s(q_0,s))$, for any $s \in \Sigma^{s*}$,

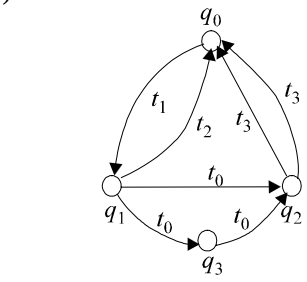
$\hat{O}_E : \hat{Q} \times \Sigma^s \to Z \cup \{\varnothing\}$, $\hat{O}_E(\hat{q}, [q,\sigma](R_{I,O})) = O_E(q,\sigma)$, for any $(\hat{q}, [q,\sigma](R_{I,O})) \in \hat{T}$.

According to Propositions 1-4, the above CEA $\hat{S}$ is well-defined. Note that the automaton $\hat{A}$ defined in the above may be undeterministic, but we can relabel the transitions of $\hat{A}$ to make it deterministic.

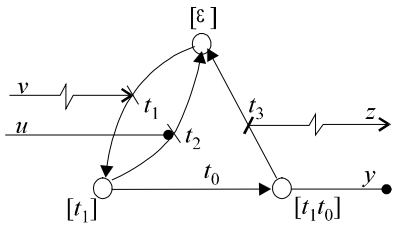
Theorem 1: The CEAs $\hat{S}$ and S are input-output equivalent, i.e., $L_{\hat{S}} = L_S$.

From Theorem 1, we know $\hat{S}$ is an aggregated model of CEA S and the main task for constructing the aggregated model is to compute the equivalence classes of R_S and the equivalence classes of $R_{I,O}$. Since for any $s, t \in \Sigma^{s*}$, $\delta^s(q_0,s) = \delta^s(q_0,t) \Rightarrow s \cong t(R_S)$, in order to identify the equivalence classes, we only need to take one event sequence (word) $s \in \Sigma^{s*}$ that leads to a state q from the initial state q_0 in A^s (i.e., $q=\delta^s(q_0,s)$) for each state of A^s, and to compare if any two of them are equivalent with respect to R_S. The computational complexity of the aggregation is polynomial (a quadratic function) with respect to the numbers of states, transitions, input signals, and output signals of the CEA.

For instance, for the CEA in Figure 3, we have A^s as shown in Figure 4(a) and the aggregated model as shown in Figure 4 (b).



(a)



(b)

Figure 4: Model aggregation

3.2. Multiple-Module CEAs

Now, we consider model aggregation for an interconnected CEA consisting of several CEA modules. The model aggregation consists of three steps. First, for each CEA module of the system, we can simplify it to get its aggregated model by applying the model aggregation method proposed in the above. In the aggregation of the module, we consider both its connected condition/event signals (i.e., condition and event signals that are connected to other modules) and its unconnected condition/event signals (i.e., condition and event signals that are not connected to other modules). With the aggregated models of all the CEA modules, we can get an aggregated interconnected CEA. As soon as the interconnected CEA is obtained, the second step is to find an equivalent CEA realization with singular CEA module (i.e. a CEA with singular module that is equivalent to the interconnected CEA). With the singular-module CEA, the third step is to simplify the CEA furthermore by applying the model aggregation method for singular-module CEAs again to get the final aggregated model of the original system.

4. HIERARCHICAL CONTROL SYNTHESIS

Hierarchical structure is widely used for the control of dynamic systems that perform a range of complex tasks. It represents a divison of information processing and control action according to scope. Several methodologies for hierarchical system modeling and control synthesis have been proposed for discrete event systems, including bottom-up and top-down methodologies. In this paper, we focus on the bottom-up methodology.

The methodology starts with a very detailed plant model at the lowest level. Higher level models of the plant are consecutively obtained by model aggregation from their adjacent lower level models. Control specifications are given in a hierarchical way. At each level, a specification for the controlled behaviour of the plant at the level is given. Such hierarchical specifications usually exhibit information aggregation. Higher level specifications usually address higher level control goals such as resource allocation and coordination, although lower level specifications specify the behaviour of the plant in a rather detailed way. Accordingly, controllers fulfiling the specifications for the plant are also designed and implemented in a hierarchical way. The controllers at different levels usually possess different sets of input signals (observation signals) and output signals (control signals). The signals may be condition or event signals in our condition/event system framework. An illustration of the bottom-up control synthesis methodology is given by Figure 5, where Pi, Si and Ci denote the plant model, the specification, and the controller at the i-th level, respectively.

The tasks for the bottom-up control synthesis (numbered in Figure 5) include:

1. The formal synthesis of a controller from the model of the uncontrolled behaviour of a plant and a given specification for the closed loop behaviour of the plant.

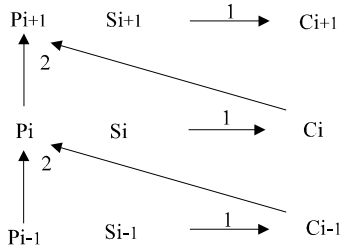


Figure 5: Bottom-up control synthesis

2. The interconnection of the uncontrolled plant model and the controller model and the generation of an aggregated model for the control synthesis at a higher level.

Task 1 is out of the scope of this paper. For Task 2, since both plant and its controller (supervisor) can be symmetrically described by condition/event systems, the interactions between the plant and the controller can be easily described by condition and event signals, and the closed system obtained by coupling the plant with the controller is also a condition/event system, we will focus on the model aggregation in the following.

Given a plant and a level in its hierarchical control structure, the task of model aggregation is to obtain a model of the uncontrolled behaviour of the plant at the level from its adjacent lower level in accordance with the abstraction at the considered level.

As we have mentioned in the above, different levels involves different sets of control action and observation information. In terms of our condition/event system models, it means that plant and controller models at different levels have different sets of input/output condition and event signals. Suppose that the hierarchical structure has the levels numbered by 1, 2, ...,*n* from bottom to top, and that the sets of input condition signals, input events signals, output condition signals and output event signals at level *i* are U_i, V_i, Y_i, and Z_i, respectively. Let $U_{i+}= \cup_{k=i}^{n} U_k$, $V_{i+}= \cup_{k=i}^{n} V_k$, $Y_{i+}= \cup_{k=i}^{n} Y_k$, $Z_{i+}= \cup_{k=i}^{n} Z_k$ be the sets of the four types of signals at the levels no lower than *i* (from level *i* to *n*). The model aggregation procedure for the hierarchical control synthesis is as follows:

We start with a very detailed CEA model of the uncontrolled behaviour of a plant at the lowest level 1 that has the sets of input condition signals, input events signals, output condition signals and output event signals as U_{1+}, V_{1+}, Y_{1+} and Z_{1+} respectively. After synthesizing a controller according to a specification and the CEA model for this level, we can get the closed loop system for the level where all signals of U_i, V_i, Y_i and Z_i are coupled or removed due to the implementation of the controller. The closed loop system is an interconnected CEA with the sets of the four types of signals being U_{2+}, V_{2+}, Y_{2+}, and Z_{2+}, respectively. The interconnected CEA can be aggregated to obtain a model of the uncontrolled behaviour of the plant at level 2 by applying the model aggregation method in Section 3. This process continues until we get a model of the uncontrolled behaviour of the plant at top level *n*. Figure 6 illustrates the model aggregation procedure.

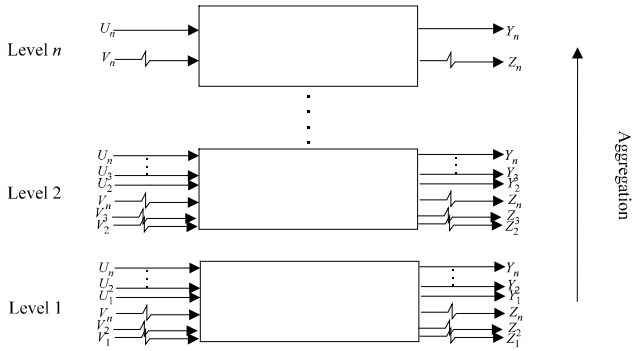


Figure 6: Model aggregation for hierarchical synthesis

6. CONCLUSION

In this paper, we propose a model aggregation approach to bottom-up hierarchical control synthesis of discrete event systems described by condition/event automata. The main advantage of the condition/event system model is that it provides a modular and input-output modeling framework for interconnected discrete event systems and can describe both spontaneous and forced events. Moreover, the model provides a unified framework for supervisory control of discrete event systems under complete observation and under partial observation, because partial observation of events and/or states of a plant can be represented by output event and/or condition signals of the plant.

REFERENCES

[1] Caines, P. E. and Y-J Wei, The hierarchical lattices of a finite machine, *Systems and Control Letters*, 25:257-263, 1995.

[2] Hubbard, P. and P. E. Caines, A state aggregation approach to hierarchical supervisory control, *Proc. of WODES98*, Cagliari, Italy, pp. 2-7, 1998.

[3] Lin F. and W. M. Wonham, Decentralized control and coordination of discrete-event systems with partial observation, *IEEE Trans. Automat. Contr.*, Vol.35, No.12, pp.1330-1337, 1990.

[4] Peyman-Gohari, M. and W. M. Wonham, A linguistic framework for controlled hierarchical DES, *Proc. of WODES98*, Cagliari, Italy, pp. 207-212, 1998.

[5] Ramadge P. J. and W. M. Wonham, Supervisory control of a class of discrete event processes, *SIAM J. Control Optim.* 25: 206-230, 1987.

[6] Sreenivas, R. S. and B. H. Krogh, On condition/event systems with discrete state realizations, *Discrete Event Systems: Theory and Applications*, 2(1):209-236, 1991.

[7] Wong K.C. and W.M. Wonham, Hierarchical control of discrete-event systems, *Discrete Event Dynamical Systems*, 6:241-273, 1996.

[8] Zhang, H. and W. M. Wonham, On the consistency of hierarchical supervision in discrete-event systems, *IEEE Trans. on Automatic Control*, Vol.35, No.10, pp. 1125-1134, 1990.