15th Triennial World Congress, Barcelona, Spain

HIERARCHICAL SUPERVISORY CONTROL OF DISCRETE EVENT SYSTEMS BASED ON STATE AGGREGATION

César R. C. Torrico *,1 **José E. R. Cury** *,2

* *Departamento de Automação e Sistemas*
Universidade Federal de Santa Catarina
P.O. Box 470- 88040/001 - Florianópolis - SC - Brasil

Abstract: This paper presents a hierarchical control theory for discrete event systems based on state aggregation and advanced control structures. The proposed hierarchical structure consists of two levels, a low-level represented by the classical model of Ramadge-Wonham, and a high-level, obtained by state aggregation. In this model, the high-level events are a subset of the low-level events, but for controller synthesis, this level will be endowed by advanced control structures as previously introduced by the authors. *Copyright ©2002 IFAC*

Keywords: Discrete Event Systems, Supervisory Control, Hierarchical Control.

1. INTRODUCTION

In many application areas, the complexity of the processes has greatly increased during the last decades. This essentially because the integration among the components of the processes constantly grows to allow the resources to be used in a more efficient way. Hierarchical control provides an opportunity to manipulate a complex problem through its decomposition into smaller subproblems for later mount their solutions in a hierarchical structure. The formalism of hierarchical control of Discrete Events Systems (DES) was first introduced by Zhong and Wonham (1990) using Ramadge and Wonham (1989) framework and later extended by Wong and Wonham (1996) to deal with marked languages and blocking issues. In this approach two hierarchical levels are considered, one associated to the operator and another associated to the manager. The hierarchical control problem consisted of designing a high level controller such that the expected behavior in this level equals the behavior obtained by applying the hierarchical control. This condition defines the *hierarchical consistence* and *strong hierarchical consistence*. The difference between the two types of hierarchical consistencies cited above is that, in the first case there may exist implementable behaviors not seen as such in the higher level. In the above cited papers conditions to build consistent abstractions are given. Nevertheless, besides the fact that those conditions may be too conservative, the information channel connecting both low and high level models has in general to be refined several times until hierarchical consistence is obtained. On the other hand Caines and Hubbard (1998) developed an approach for hierarchical control based on state aggregation. This approach also presents conditions for strong hierarchical consistence, but it doesn't present a systematic way to obtain this condition. Furthermore, the approach allows only to solve problems with forbidden states specifications.

In this work a new formalism for hierarchical control of discrete event systems is introduced. We keep on two levels of hierarchy: for the low level we refer to the traditional model of Ramadge and Wonham; on the other hand a model proposed by Cury *et al.* (2001) based on advanced control structures is considered for the high level. The high level problem is defined on specifications built from a sub-set of the low-level set of events considered as relevant to the manager. The objective is to propose a model for hierarchical control based on state aggregation, such that the hierarchical structure, obtained in a simple and straightforward way, with minimum refinement, is

[1] Supported by CAPES
[2] Supported by CNPq under grant number 300953/93-3 and PRONEX 015/98

strong hierarchical consistent. A similar work in a linguistic framework is proposed in (da Cunha and Cury, 2001).

The paper is organized as follows. Section 2 shortly introduces the problem; the idea of state aggregation is presented in Section 3 while the complete high level model is defined in Sec 4; main results are showed in Sec 5; finally Sec 6 illustrates the proposed methodology by a simple example of a transfer line.

2. THE HIERARCHICAL CONTROL PROBLEM

Given a plant G defined over an alphabet Σ, and a set of events $\Sigma^A \subset \Sigma$, the problem is to obtain an aggregated model for the high level, defined over Σ^A, such that: *i)* every realizable specification in the high level equals the image of a realizable language in the low level; *ii)* every realizable language in the low level has a realizable image in the high level.

In the hierarchical structure the low-level consists of a *plant* G and a *supervisor* S defined over Σ, and the high-level (aggregated-level) consists of a plant G^A and a supervisor S^A, defined over $\Sigma^A \subset \Sigma$. These are coupled as shown in Fig. 1.

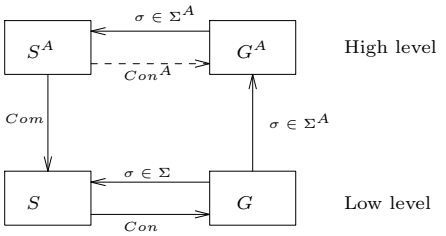


Fig. 1. Hierarchical Structure

In Fig. 1, G is the plant to be controlled in the real world by S, the operator, while G^A is an abstraction of G obtained by state aggregation and controlled by S^A, the manager. The plant G informs to the high-level only a sub-set of events ($\sigma \in \Sigma^A$) considered relevant for the manager, refreshing the model G^A.

To each occurrence of an event $\sigma \in \Sigma^A$, the manager S^A generates a command (Com) to be implemented by the operator S. The operator S receives two information - the command Com of the aggregated level and the occurrence of events $\sigma \in \Sigma$ generated by the plant G - and with these data he applies a control Con to the plant G. The control Con^A results to be a virtual control, since the behavior of G^A is totally determined by the behavior of G.

3. STATE AGGREGATION

In the low level, the plant is represented by a quintuple $G = (\Sigma, X, \delta, q_0, Q_m)$, where, Σ is the set of event labels partitioned into controllable (Σ_c) and uncontrollable events (Σ_u), i.e., $\Sigma = \Sigma_c \cup \Sigma_u$, X is the state set, $\delta : X \times \Sigma \longrightarrow X$ is the (partially-defined) transition function, $q_0 \in X$ is the initial state and $Q_m \subseteq X$ is the marked state set.

To obtain an aggregated model for the high level, a partition of the state set X, called π-partition, is first defined. The π-partition is a collection of subsets of X (called π-blocks) denoted formally by $\pi = \{X_1, X_2, ..., X_N\}$ with $\bigcup X_i = X$, $X_i \neq \emptyset$ and $X_i \cap X_j = \emptyset$ for $i \neq j$, so that two states belong to a same block if and only if they are linked by some string of no relevant events. From this partition, an abstract model is defined over the set of relevant events.

Definition 1. (π-aggregated automaton) Given an automaton G, a set of relevant events $\Sigma^A \subset \Sigma$, and a π-partition $\pi = \{X_i : x_j, x_k \in X_i \leftrightarrow x_j, x_k \in X \wedge \exists s \in (\Sigma - \Sigma^A)^*, \delta(x_j, s) = x_k\}$, the *$\pi$-aggregated automaton* is defined as

$$G^A = (\Sigma^A, \pi, \delta^A, q_0^A)$$

where, Σ^A is the set of events, π is the finite state set corresponding to the π-partition of X, $\delta^A : \pi \times \Sigma^A \longrightarrow \pi$ is the transition function such that $\delta^A(X_i, \sigma) = X_j$ **if** $(\exists x \in X_i, \exists y \in X_j : \delta(x, \sigma) = y)$, and $q_0^A \in \pi$ is the initial state, such that $q_0 \in q_0^A$.

Marking features for the high level model are not considered here. In fact G^A is used only to represent admissible sequences of events. Marking is left to be dynamically defined by the control structure, as in (Cury *et al.*, 2001).

Given an automaton G and the corresponding π-aggregated automaton (G^A), we define the canonical map $\Theta : L(G) \longrightarrow L(G^A)$, as $\Theta(\epsilon) = \epsilon$ and

$$\Theta(s\sigma) = \begin{cases} \Theta(s)\sigma & \textbf{if } \sigma \in \Sigma^A \\ \Theta(s) & \textbf{otherwise} \end{cases}$$

for $s \in \Sigma^*$ and $\sigma \in \Sigma$.

The canonical map is naturally extended for languages as $\Theta : 2^{L(G)} \longrightarrow 2^{L(G^A)}$, $\Theta(K) = \{\Theta(s) : s \in K\}$

Let $L^A = \Theta(L(G))$, the mapping to the high-level of the language $L(G)$, and let $L(G^A)$ be the language generated by the π-aggregated automaton G^A. Notice that in general $L^A \subseteq L(G^A)$ since some strings of $L(G^A)$, may not be the image of feasible strings in the low-level.

Before adhering the necessary control structure to the high-level, we may have to proceed to a refinement in G^A. For this we have to consider the two possible following situations of the aggregated automaton:

i) G^A is non-deterministic;

ii) There exist $X_i, X_j \in \pi$ and $\sigma \in \Sigma^A$ such that $\delta^A(X_i, \sigma) = X_j$, and such that $\exists\ x, y \in X_i$, *and* $z, w \in X_j$, $z \neq w$, with $\delta(x, \sigma) = z$ and $\delta(y, \sigma) = w$.

If any of the above cases arises in G^A, a refinement procedure is done by renaming each event $\sigma \in \Sigma$ that causes i) or ii) to occur, with events $\sigma', \sigma'', ...$ This refinement allows the model to eliminate any possible ambiguity on the information flowing through the low to high level channel. It may be done locally, without considering dependency between possible multiple instances of refinement for the same original event since it is assumed that the high level specifications remain to be given in terms of the original alphabet Σ^A.

4. HIGH LEVEL MODEL

In this section the high level model will be completed by introducing its control structure. As cited before this control structure will also incorporate the marking attributes of the high level plant. The resulting model will be as the one introduced by Cury *et al.* (2001).

4.1 *Preliminaries*

The *In-set states* $I(X_i)$ of an element X_i of a partition π is the set of states in X_i that are, either the initial state q_0 or states directly accessible with a relevant event, i.e.:

$$x \in I(X_i) \longleftrightarrow (x \in X_i) \wedge ([x = q_0] \vee [\exists x' \in X\ ,\ \exists \sigma \in \Sigma^A\ ,\ \delta(x', \sigma) = x]) \quad (1)$$

Every block of the partition results to be a part of the system that can be represented by another automaton. For each block $X_i \in \pi$, *an automaton* H_i is defined as

$$H_i = \big((\Sigma - \Sigma^A), X_i, I(X_i), \delta_i, Q_{im}\big) \quad (2)$$

where, $(\Sigma - \Sigma^A)$ is the set of event labels, X_i is the state set, $I(X_i)$ is the initial state set, $\delta_i : X_i \times (\Sigma - \Sigma^A) \longrightarrow X_i$ is the partial transition function such that $\delta_i(x, \sigma) = \delta(x, \sigma)$ **if** $x \in X_i$, $\sigma \in (\Sigma - \Sigma^A)$ and Q_{im} is the marked state set such that $Q_{im} = X_i \cap Q_m$.

Given an automaton H_i each state $x_j \in I(X_i)$ defines *a subsystem of* H_i is as an automaton

$$H_{ij} = ((\Sigma - \Sigma^A), X_{ij}, x_j, \delta_{ij}, Q_{ijm}) \quad (3)$$

where, $(\Sigma - \Sigma^A)$ is the set of event labels, x_j is the initial state, X_{ij} is the state set such that $X_{ij} = \{x \in X_i : x = \hat{\delta_i}(x_j, u), u \in (\Sigma - \Sigma^A)^*\}$, δ_{ij} is the partial transition function such that $\big(\delta_{ij}(x, \sigma) = \delta_i(x, \sigma)$ **if** $x \in X_{ij}$, $\sigma \in (\Sigma - \Sigma^A)\big)$, and Q_{ijm} is the marked state set such that $Q_{ijm} = Q_{im} \cap X_{ij}$.

Also, an *augmented automaton* H_i^+ is defined by adding an extra marked state to H_i which receives transitions corresponding to the active set of events of X_i in G^A, i.e,

$$H_i^+ = (\Sigma, X_i^+, I(X_i), \delta_i^+, Q_{im}^+) \quad (4)$$

where, Σ is the set of event labels, X_i^+ is the state set such that $X_i^+ = X_i \cup \{x^+\}$, $I(X_i)$ is the initial state set, δ_i^+ is the partial transition function such that $(\delta_i^+(x, \sigma) = \delta(x, \sigma)$ **if** $x \in X_i$, $\sigma \in (\Sigma - \Sigma^A))$ **and** $(\delta_i^+(x, \sigma) = x^+$ **if** $x \in X_i$, $\sigma \in \Sigma^A$, $\delta(x, \sigma)$ is defined), and Q_{im}^+ is the marked state set such that $Q_{im}^+ = Q_{im} \cup \{x^+\}$.

Analogously, a *subsystem of* H_i^+ is defined for each state $x_j \in I(X_i)$ of H_i^+, as an automaton

$$H_{ij}^+ = \big(\Sigma, X_{ij}^+, \delta_{ij}^+, x_j, Q_{ijm}^+\big) \quad (5)$$

where, Σ is the set of event labels, x_j is the initial state, X_{ij}^+ is the state set such that $X_{ij}^+ = \{x \in X_i^+ : x = \hat{\delta_i^+}(x_j, u), u \in \Sigma^*\}$, δ_{ij}^+ is the partial transition function such that $\delta_{ij}^+ = \delta_i^+ / X_{ij}^+$ and Q_{ijm}^+ is the marked state set such that $Q_{ijm}^+ = Q_{im}^+ \cap X_{ij}^+$.

Let H_{ij} and H_{ij}^+ be subsystems of H_i and H_i^+ respectively. The set of *subsystems between* H_{ij} *and* H_{ij}^+, is defined as

$$\mathcal{S}_{ij} = \{H_s : \big[L(H_{ij}) \subseteq L(H_s) \subseteq L(H_{ij}^+)\big] \wedge [L_m(H_s) = L_m(H_{ij}^+) \cap L(H_s)]\} \quad (6)$$

In each automaton $H_s \in \mathcal{S}_{ij}$ the transitions to $x^+ \in X_{ij}^+$ correspond to a subset of the set of active events of X_i in G^A reachable from the input state $x_j \in I(X_i)$.

Two sets of controllable languages with respect to $L(H_{ij}^+)$ are introduced as follows:

$$\mathcal{C}_{ij} = \{K_c \subset \Sigma^* : K_c \neq \emptyset, K_c = \sup \mathcal{CF}(L_m(H_s), H_{ij}^+), H_s \in \mathcal{S}_{ij}\} \quad (7)$$

and

$$\mathcal{D}_{ij} = \{K_d \subset \Sigma^* : K_d \neq \emptyset, K_d = \sup \mathcal{CF}[L_m(H_s) - L_m(H_{ij}), H_{ij}^+], H_s \in \mathcal{S}_{ij}\} \quad (8)$$

where $\sup \mathcal{CF}(\diamond, \square)$ denotes the supremal controllable and $L_m(\square)$-closed language with respect to $L(\square)$, contained in $\diamond$ (Ramadge and Wonham, 1989).

Consider the set

$$E_{ij} = \mathcal{C}_{ij} \cup \mathcal{D}_{ij} \quad (9)$$

Each element $K \in E_{ij}$ corresponds to the least restrictive behavior within the block X_i in the low level that can be controlled from $x_j \in I(X_i)$, to generate $\Theta(K) - \{\epsilon\}$ as the next admissible set of events in the the high level. Behaviors in $\mathcal{D}_{ij}$ all don't pass through marked states in X_i, while behaviors in $\mathcal{C}_{ij}$ that are not in $\mathcal{D}_{ij}$ all pass by at least one marked state of X_i. The computation of E_{ij} provides the base for the definition of the high level control structure to be introduced in the following.

4.2 *High level control structure*

Given E_{ij} a set of pairs $\Gamma_{ij} \in 2^{\Sigma^A} \times \{M, N\}$ is defined as

$\Gamma_{ij} = \{(\gamma, \#) : \gamma = [\Theta(K) - \{\epsilon\}],\ K \in E_{ij},\ \text{and } \# = M \text{ if } (\epsilon \in \Theta(K)),\ \text{else } \# = N\}$

The high level control structure is defined as a map $\Gamma : (\pi, X) \rightarrow 2^{2^{\Sigma^A} \times \{M,N\}}$, which associates to each block $X_i \in \pi$ and input state $x_j \in I(X_i)$ a set of control patterns $\Gamma_{ij} \subseteq 2^{\Sigma^A} \times \{M, N\}$. $(\gamma, \#) \in \Gamma_{ij}$ is a control pattern valid in X_i when entering it through x_j with,

(1) $\gamma \subset \Sigma$ being a set of enabled events after X_i;
(2) $\# = M$ being a marking attribute meaning that the current past string in $L(G^A)$ is considered a task of the system, or equivalently, setting X_i currently as a marked state; and
(3) $\# = N$ meaning that the current past string $L(G^A)$ is not considered a task of the system, or equivalently, setting X_i currently as an unmarked state.

Theorem 2. Γ represents a control structure as defined in (Cury *et al.*, 2001), i. e., each $\Gamma_{ij} \in \Gamma$ satisfies:

(1) $(\gamma_1, N), (\gamma_2, N) \in \Gamma_{ij} \longrightarrow (\gamma_1 \cup \gamma_2, N) \in \Gamma_{ij}$
(2) $(\gamma_1, M), (\gamma_2, \#) \in \Gamma_{ij} \longrightarrow (\gamma_1 \cup \gamma_2, M) \in \Gamma_{ij},\ \# = M, N$ [3]

At this point, the pair $D = (L(G^A), \Gamma)$ provides a complete high level model. Nevertheless the control structure Γ as defined depends on X_i and $x_j \in I(X_i)$. Once the information of the input state cannot be seen in the π-automaton, a refinement in G^A is then proposed in order to turn Γ into a state dependent control structure.

To represent the π-automaton with a state dependent control structure each block X_i is splitted into a set of high level states, each one for a subset of $I(X_i)$ sharing a common set of control patterns. The resulting high level automaton G_s^A is guaranteed to have a state dependent control structure.

[3] The proofs are omitted due to lack of space. Further information available in (Torrico and Cury, 2001)

The number of states of the automaton G_s^A is, in the worst case, the sum of input-states of the blocks defining the states of G^A. From now on G_s^A is referred as G^A as well as the set of control patterns associated to a new state X_i of G^A is referred as Γ_i.

Given a controlled DES $D = (L(G^A), \Gamma)$ representing the high level plant and control structure as defined above, a supervisor f^A for D is defined as a map, $f^A : L(G^A) \rightarrow 2^{\Sigma} \times \{M, N\}$. The behavior of the closed loop system f^A/D is represented by a pair of languages, a closed language $L(f^A/D)$ and a marked language $L_m(f^A/D)$. The closed language $L(f^A/D)$ is defined recursively as

(1) $\epsilon \in L(f^A/D)$
(2) $s\sigma \in L(f^A/D) \longleftrightarrow s \in L(f^A/D) \wedge s\sigma \in L(G^A) \wedge \sigma \in \gamma$ with $f^A(s) = (\gamma, \#)$

and the marked language $L_m(f^A/D)$ as

$s \in L_m(f^A/D) \longleftrightarrow s \in L(f^A/D) \wedge f^A(s) = (\cdot, M)$

where $(\gamma, \#) \in \Gamma_i$ for $\delta^A(q_0^A, s) = X_i$.

In general $\overline{L_m(f^A/D)} \subseteq L(f^A/D)$, and the supervisor is said to be nonblocking if $\overline{L_m(f^A/D)} = L(f^A/D)$.

4.3 *High level supervisory control problem*

In order to propose a solution for a high level supervisory control problem the concept of Γ-compatibility (Cury *et al.*, 2001) is introduced in the context of the aggregated model.

Definition 3. (Γ-compatibility) The language $K \subseteq L(G^A)$ is Γ-*compatible w.r.t* $L(G^A)$ if, and only if, $K = \emptyset$ or

(1) $(\forall s \in K)\ (\exists(\gamma, M) \in \Gamma_i) : \gamma \cap \Sigma_{L(G^A)}(s) = \Sigma_K(s)$, and
(2) $(\forall s \in \overline{K} - K)\ (\exists(\gamma, N) \in \Gamma_i) : \gamma \cap \Sigma_{L(G^A)}(s) = \Sigma_K(s)$

where $\delta^A(q_0^A, s) = X_i$ and $\Sigma_L(s)$ represents the set of active events in L, after s.

The following theorem by Cury *et al.* (2001) shows that Γ-*compatibility* is a necessary and sufficient condition for the existence of a nonblocking supervisor to implement a given language K in a controlled DES $D = (L(G^A), \Gamma)$.

Theorem 4. (Existence of Supervisors) Given $D = (L(G^A), \Gamma)$ and given a language $K \subseteq L(G^A)$, with $K \neq \emptyset$, then there is a nonblocking supervisor f^A for D, such that $L_m(f^A/D) = K$, if only if K is Γ-*compatible w.r.t* $L(G^A)$.

Cury *et al.* (2001) proved the existence of the supremal Γ-compatible language contained in a

given language K, as well as provided an algorithm for its synthesis and the synthesis of the corresponding supervisor.

5. MAIN RESULTS

The following result establishes that the aggregated model as constructed above leads to a strong consistent hierarchical system.

Theorem 5. Given a low level plant G, a set of relevant events $\Sigma^A \subset \Sigma$ and an aggregated system $D = (L(G^A), \Gamma)$ as proposed, there exists $K^A \subseteq L(G^A)$, Γ-compatible w.r.t $L(G^A)$ if and only if there exists $K \subseteq L_m(G)$ controllable w.r.t $L(G)$ and $L_m(G)$-closed such that $\Theta(K) = K^A$.

Given a high level Γ-compatible language $K^A \subseteq L(G^A)$ and a nonblocking supervisor f^A such that $L_m(f^A/D) = K^A$, a procedure to compute a low level nonblocking supervisor f such that $\Theta(L_m(f/G)) = K^A$ is presented in the following.

By construction of the high level control structure, for each high level control pattern $(\gamma, \#) \in \Gamma_{ij}$ as initially introduced there exists in the low level a language $J \subset L(H_{ij}^+)$ such that $J \in E_{ij}$ and $\Theta(J) - \epsilon = \gamma$.

Let $f_{ij}^J : J \longrightarrow 2^\Sigma$ be a nonblocking supervisor such that $L_m(f_{ij}^J/H_{ij}^+) = J$.

The following algorithm translates the implementation of a high level supervisor $f^A : L(G^A) \longrightarrow 2^{2^\Sigma \times \{M,N\}}$ into a low-level supervisor $f : L(G) \longrightarrow 2^\Sigma$.

Algorithm 1. Obtaining the low level supervisor.

Input: f^A

(1) For $s \in L(G)$, such that $\Theta(s) = \epsilon$,
$f(s) = f_{0,0}^J(s)$, where J is such that $(\Theta(J) - \epsilon) = \gamma$ **and** $\Big[(\epsilon \in \Theta(J)$ if $f^A(\epsilon) = (\gamma, M))$ **or** $(\epsilon \notin \Theta(J)$ if $f^A(\epsilon) = (\gamma, N))\Big]$.

(2) For $s \in L(G)$, with $s = s'\alpha s''$, $s' \in \Sigma^*$, $\alpha \in \Sigma^A$, $s'' \in (\Sigma - \Sigma^A)^*$, such that $\Theta(s) \in \overline{K}^A$, and $\delta(x_0, s'\alpha) = x_j \in I(X_i)$,
$f(s) = f_{ij}^J(s'')$, where J is such that $(\Theta(J) - \epsilon) = \gamma$ **and** $\Big[(\epsilon \in \Theta(J)$ if $f^A(\theta(s)) = (\gamma, M))$ **or** $(\epsilon \notin \Theta(J)$ if $f^A(\theta(s)) = (\gamma, N))\Big]$.

Output: f

Theorem 6. Given a nonblocking supervisor $f^A : L(G^A) \longrightarrow 2^\Sigma \times \{M, N\}$ with $L_m(f^A/D) = K^A$, and $f : L(G) \longrightarrow 2^\Sigma$ as computed by algorithm 1, it's true that, $\Theta(L_m(f/G)) = K^A$.

There are two possible mechanisms to implement the hierarchical control scheme. The first one is an off-line procedure that computes the whole low level supervisor f from a given high level supervisor f^A and then shuts off the command channel Com. In this scheme the information channel from the low to high level may be kept just to allow the manager to monitor the resulting abstracted behavior of the system. The second way of implementing the hierarchical control is by the following on-line procedure: the high level supervisor applies through the command channel Com, a control input $(\gamma, \#) \in \Gamma_i$ when entering state X_i after the occurrence of a relevant event $\sigma \in \Sigma^A$; the low level activates the local supervisor f_{ij}^J as a function of the current low level state $x_j \in I(X_i)$ and the local language $J \in \mathcal{D}_{ij}$ such that $\Theta(J) = \gamma$ if $\# = N$, or $J \in \mathcal{C}_{ij} - \mathcal{D}_{ij}$ such that $\Theta(J) - \{\epsilon\} = \gamma$ if $\# = M$.

6. EXAMPLE: TRANSFER LINE WITH RE-ENTRANT FLOW

Consider the transfer line consisting of two Machines M_1, M_2 followed by a test unit TU, linked by buffers B_1 and B_2, as shown in Figure 2. This example was first treated by Wonham (1998).

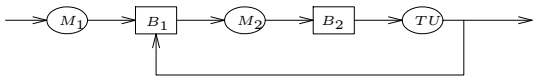
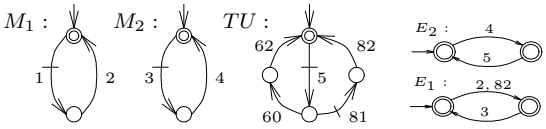


Fig. 2. Transfer line

Automata for the system components are shown in Fig. 3(a).



(a) (b)

Fig. 3. (a)Model of the system components and (b) Specification for the buffers.

The controllable events 1 and 3 represent the start of machines M_1 and M_2, respectively, and the uncontrollable events 2 and 4 represent the end of operations. The start of the test unit is represented by the controllable event 5 and the signs of decision of "passes" or "fails" are represented by the events 60, 81 respectively. In case of pass test, the workpiece is sent to the system output (event 62); in case of fail test, it is returned to B_1(event 82) for reprocessing by M_2. Buffers are considered to be of capacity one. Specifications for no *overflow* and no *underflow* of the buffers are represented by the automata in Fig. 3(b). Fig. 4 presents a possible solution to this problem. This automaton is now adopted as a plant G for hierarchical control purposes.

Let us assume that a manager is interested only in controlling the entrance and exit of pieces in the system. Thus, $1, 60, 81$ are considered as the relevant events for the high level. A first

aggregated model built as in Section 3is shown in Fig. 4 where event 1 after X_0 needed to be redefined as $1'$ and $1''$.

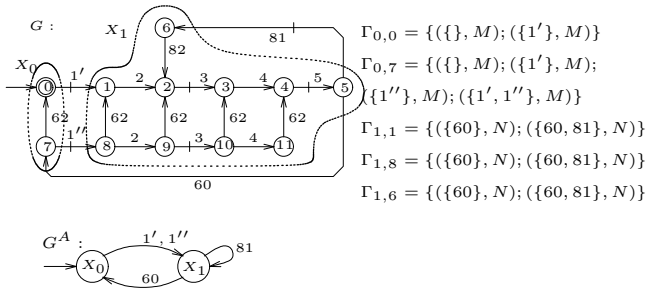


Fig. 4. Plant and its aggregation.

In the high-level automaton (Fig 4) observe that the block X_0 presents two different control patterns for the input states 0 and 7 ($\Gamma_{0,0} \neq \Gamma_{0,7}$). The automaton is then refined following the procedure presented in Section 4. Fig. 5 shows the resulting high level model with its corresponding state dependent control structure.

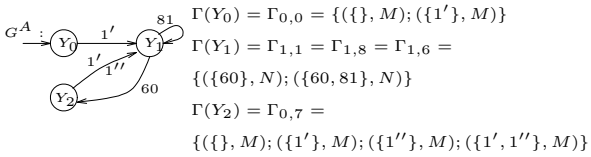


Fig. 5. High level Plant with state dependent control structures.

(Fig 6) represents a high level specification which establishes that if a piece is refused twice by the test unit, it is forced to pass and the system must stop.

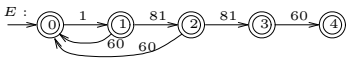


Fig. 6. Specification.

Fig. 7 shows the optimal solution for the high level problem as computed by applying the supremal Γ-*compatible* algorithm as in (Cury *et al.*, 2001).

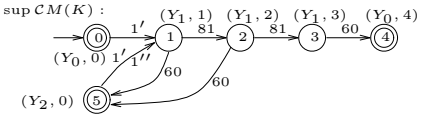


Fig. 7. Supremal Γ-*compatible* language.

Finally, Fig. 8 shows the global low level implementation of the optimal hierarchical supervision.

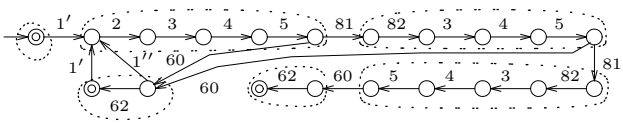


Fig. 8. Low-level implementation

7. CONCLUSIONS

The state aggregation procedure proposed in this work allows to construct hierarchical consistent models for DES supervision in a straightforward way. The limitation of the presented methodology relies mainly in the fact that a large number of local supervisory control problems has to be solved in order to construct the high level control structure, although these are in general performed over small subsystems. Also, modularity may be a way to reduce this complexity. In fact, hierarchical systems are in general large scale systems modelled through the composition of smaller subsystems and characterized by having several specifications. This problem was already treated in (Wong and Wonham, 1998). The authors are currently investigating the possibility of incorporating results on local modularity as proposed in (de Queiroz and Cury, 2000) in the hierarchical control approach as introduced here.

8. REFERENCES

Caines, P.E. and P.J. Hubbard (1998). A state aggregation approach to hierarchical supervisory control with applications to a transferline example. In: *Proceedings of the WODES 98*. Cagliary, Italy. pp. 2–7.

Cury, J.E.R., C.C. Torrico and A.E.C. da Cunha (2001). A new approach for supervisory control of discrete event sistems. In: *Proceedings of the European Control Conference 2001*.

da Cunha, A.E.C. and J.E.R. Cury (2001). Hierarchically consistent controlled discrete event systems. *Article submited to the IFAC World Congress 2002*.

de Queiroz, M. H. and J. E. R. Cury (2000). Modular supervisory control of large scale discrete event systems. *Discrete Event Systems. Analysis and Control (WODES)* pp. 103–110.

Ramadge, P.J. and W.M. Wonham (1989). The control of discrete event systems. *Proceeding of the IEEE* **77**(1), 81–98.

Torrico, C.R.C. and J.E.R. Cury (2001). Controle supervisório hierárquico de seds: Uma abordagem baseada na agregação de estados. *www.lcmi.ufsc.br/∼torrico/trabalhos/Qualify.zip*.

Wong, K.C. and W.M. Wonham (1996). Hierarquical control of discrete-event systems. *Discrete Event Dynamical Systems* **6**, 241–273.

Wong, K.C. and W.M. Wonham (1998). Modular control and coordination of discrete-event systems. *Discrete Event Dynamical Systems* **8**(3), 247–297.

Wonham, W.M. (1998). *Notes on Control of Discrete-Event Systems. Course Notes for ECE 1636F/1637S*. Revision 98.09.01.

Zhong, H. and W.M. Wonham (1990). On the consistency of hierarchical supervision in discrete-event systems. *IEEE Transactions on Automatic Control* **35**(10), 1125–1134.