Hierarchical Interface-based Supervisory Control: Command-pair Interfaces

R.J. Leduc

Dept. of Computing and Software, McMaster University
leduc@mcmaster.ca

Version 1.1,
June 15, 2004

Abstract—In this paper we extend our previous results in which we presented a hierarchical method that decomposed a system into a high level subsystem which communicated with $n \geq 1$ parallel low level subsystems through separate interfaces. This method offered potentially significant computational savings as the complete system model never needed to be constructed.

We introduce a new interface structure called command-pair interfaces which is capable of representing state information about the low levels, and extend the previous results to include the more general command-pair interfaces. We then illustrate the new approach by re-visiting a large manufacturing example. Finally, we present a complexity analysis showing that the algorithm's time complexity is $\mathbf{O}(m^2)$, where $m = n + 1$ is the total number of subsystems.

I. Introduction

In the area of Discrete-Event Systems (DES), two common tasks are to verify that a composite system, based on a cartesian product of subsystems, is (i) nonblocking and (ii) controllable. The main obstacle to performing these tasks is the combinatorial explosion of the product statespace. Although many methods have been developed to deal with this problem (modular control [1, 25, 33, 36, 39], decentralized control [3, 28, 35], vector DES/Petri Nets [8, 9, 27, 31, 43], model aggregation methods [6, 7, 10, 16, 18, 32, 38, 40, 42], and multi-level hierarchy [4, 14, 29, 37]), large-scale systems are still problematic, particularly for verification of nonblocking.

One exception is the recent work of Zhang [41] who have recently developed algorithms that use Integer Decision Diagrams to verify centralized DES systems on the order of 10^{23} states. This builds upon the work by the model checking/temporal logic community [2, 12, 11, 5, 19, 30]. However, Zhang's work is a more efficient way to represent DES and verify properties, not a hierarchical method.

To deal with the complexity of large scale systems, the software engineering community has long advocated the decomposition of software into modules (components) that interact via well defined interfaces (e.g., [17]). Recently the supervisory control community has begun to advocate a similar approach [13, 21, 22, 24]. These approaches develop well defined interfaces between components to provide the structure to allow local checks to guarantee global properties such as controllability [13, 22, 24] or nonblocking [21, 22, 24].

In this paper, we extend the work of [21, 22, 24] which introduced a hierarchical method, called *hierarchical interface-based supervisory control* (HISC), that decomposes a system into a *high level subsystem* which communicates with $n \geq 1$ parallel *low level subsystems* through separate interfaces. We do this by introducing a new type of interface called *command-pair interfaces* that is similar, but is able to represent state information about the low levels.

We illustrate the use of command-pair interfaces by revisiting a large manufacturing example [20] with an estimated closed-loop statespace size of 7×10^{21}. Finally, we present a complexity analysis for the method.

II. Serial Case and Command-pair Interfaces

Before we introduce command-pair interfaces, we must first introduce the setting that they are defined in. We will do this by discussing the serial case of HISC. In the serial case, we are restricting ourselves to only one *low level* ($n = 1$). In this setting, we have a master-slave system, where a high level subsystem sends a command to a low level subsystem, which then performs the indicated task and sends back a reply. Figure 1 shows conceptually the structure and information flow of the system. We call this the serial case as communication occurs in a serial fashion between the two subsystems.

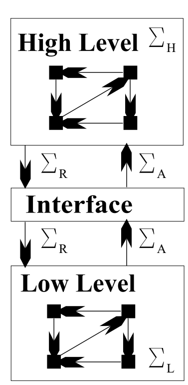

Fig. 1. Interface Block Diagram.

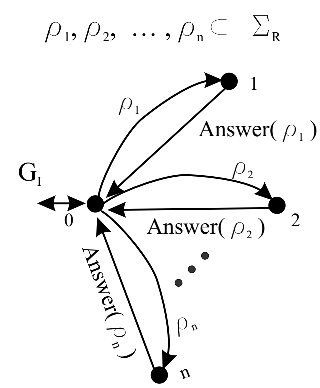

Fig. 2. Interface Specification.

To capture the restriction of the flow of information imposed by the *interface,* the alphabet of the plant (Σ) is split into four disjoint alphabets: Σ_H, Σ_L, Σ_R, and Σ_A. The events in Σ_H are called *high level events* and the events in Σ_L *low level events* as these events appear only in the high level and low level models, respectively.

The alphabets Σ_R and Σ_A are called collectively *interface events.* These events are common to both levels of the hierarchy and represent communication between the two subsystems. The events in Σ_R, called *request events*, represent commands sent from the high level subsystem to the low level subsystem. The events in Σ_A are *answer events* and represent the low level's responses to the request events.

In [21], Leduc et al. introduced the concept of *star interfaces.*[1] This interface structure was useful as it has a regular structure and is thus easy to construct. To define a star interface, the designer selects a set of request events, and then for each request event, the designer defines a set of answer events that can follow it. In essence, the designer defines a map **Answer** : $\Sigma_R \rightarrow \mathrm{Pwr}(\Sigma_A)$. We add the constraints that the *low level subsystem* must provide at least one response for each request it receives, and that Σ_A does not contain any unused events. Figure 2, shows how a star interface, with $n = |\Sigma_R|$ $(n \geq 0)$, is expressed as a DES. The required structure for a star interface is given by DES G_I.

Command-pair interfaces are similar to star interfaces, the key difference being that the "star" shape is no longer required. A command-pair interface still has a request event followed by an answer event, but it can now contain additional state information. With a command-pair interface we can have a DES structure as in Figure 3. Request events ρ_1 and ρ_2 might represent the regular behaviour of the system, while α_3 and ρ_3 represent breakdown and repair of the system.

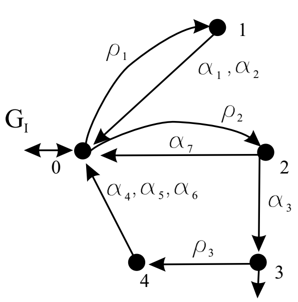

Fig. 3. Example Command-pair Interface.

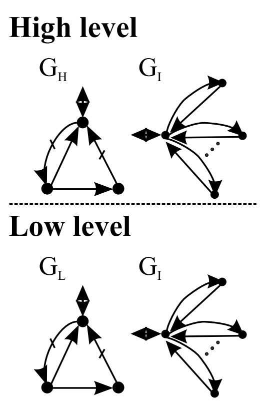

Fig. 4. Two Tiered Structure of the System.

Definition: A DES $G_I = (X, \Sigma_I, \xi, x_o, X_m)$ is a command-pair interface if the following conditions are satisfied:

(A) $\Sigma_I = \Sigma_R \dot\cup \Sigma_A$

(B) $(\forall s \in L(G_I))(\forall \rho \in \Sigma_R)\ s\rho \in L(G_I) \Rightarrow s \in L_m(G_I)$

(C) $(\forall s \in L_m(G_I))(\forall \sigma \in \Sigma_I)\ s\sigma \in L(G_I) \Rightarrow \sigma \notin \Sigma_A$

(D) $L_m(G_I) = \{\epsilon\} \cup (\Sigma_I^*.\Sigma_A \cap L(G_I))$

(E) $L(G_I) \subseteq \overline{(\Sigma_R.\Sigma_A)^*}$

Finally, we show that star interfaces are a special case of command-pair interfaces.

Proposition 1 *If DES $G_I = (X, \Sigma_I, \xi, x_o, X_m)$ is a star interface, then G_I is a command-pair interface.*

Proof: See proof in [26].

A. Definitions and Notation

For our setting, we assume the high level subsystem is modelled by DES G_H (defined over event set $\Sigma_H \cup \Sigma_R \cup \Sigma_A$), the low level subsystem by DES G_L (defined over event set $\Sigma_L \cup \Sigma_R \cup \Sigma_A$), and the interface by DES G_I (defined over $\Sigma_R \cup \Sigma_A$). Also, the high level will mean $\mathbf{sync}(G_H, G_I)$, and the low level $\mathbf{sync}(G_L, G_I)$.[2] The overall structure of the system is displayed in Figure 4.

To simplify the notation in proofs, we introduce the following event sets, natural projections, and useful languages:

$$\begin{array}{rlr} \Sigma_I := \Sigma_R \dot\cup \Sigma_A, & P_{IH} : \Sigma^* \rightarrow \Sigma_{IH}^* & \\ \Sigma_{IH} := \Sigma_H \dot\cup \Sigma_R \dot\cup \Sigma_A, & P_{IL} : \Sigma^* \rightarrow \Sigma_{IL}^* & \\ \Sigma_{IL} := \Sigma_L \dot\cup \Sigma_R \dot\cup \Sigma_A, & P_I : \Sigma^* \rightarrow \Sigma_I^* & \\ \mathcal{H} := P_{IH}^{-1}(L(G_H)), & \mathcal{H}_m := P_{IH}^{-1}(L_m(G_H)) & \subseteq \Sigma^* \\ \mathcal{L} := P_{IL}^{-1}(L(G_L)), & \mathcal{L}_m := P_{IL}^{-1}(L_m(G_L)) & \subseteq \Sigma^* \\ \mathcal{I} := P_I^{-1}(L(G_I)), & \mathcal{I}_m := P_I^{-1}(L_m(G_I)) & \subseteq \Sigma^* \end{array}$$

Whereas the representation of the system as given in Figure 4 is useful for verifying nonblocking as it simplifies the notation, it ignores the distinctions between plants and supervisors. For controllability, we need to split the subsystems into their plant and supervisor components. We will do so as shown in Figure 5.

We next define the *high level plant* to to be $\mathcal{G}_H$, and the *high level supervisor* to be $\mathcal{S}_H$ (both defined over event set Σ_{IH}). Similarly, the *low level plant* and *supervisor* are $\mathcal{G}_L$ and $\mathcal{S}_L$ (defined over event set Σ_{IL}). To be consistent with the previous form, we define the following identities for the high and low level subsystems as below.

$$G_H := \mathbf{sync}(\mathcal{G}_H, \mathcal{S}_H) \qquad\qquad G_L := \mathbf{sync}(\mathcal{G}_L, \mathcal{S}_L)$$

We can now define our *flat supervisor* and *plant* as well as some useful languages as follows:

$$\begin{array}{llr} \mathbf{Plant} := \mathbf{sync}(\mathcal{G}_H, \mathcal{G}_L) & \mathbf{Sup} := \mathbf{sync}(\mathcal{S}_H, \mathcal{S}_L, G_I) & \\ \mathbf{H} := P_{IH}^{-1} L(\mathcal{G}_H), & \mathbf{H}_{\mathcal{S}} := P_{IH}^{-1} L(\mathcal{S}_H), & \subseteq \Sigma^* \\ \mathbf{L} := P_{IL}^{-1} L(\mathcal{G}_L), & \mathbf{L}_{\mathcal{S}} := P_{IL}^{-1} L(\mathcal{S}_L), & \subseteq \Sigma^* \end{array}$$

Finally, we will be using the eligibility operator in our definitions. For a language $L \subseteq \Sigma^*$ and a string $s \in \Sigma^*$, the operator $\mathrm{Elig}_L : \Sigma^* \rightarrow \mathrm{Pwr}(\Sigma)$ is defined as follows:

$$\mathrm{Elig}_L(s) := \{\sigma \in \Sigma | s\sigma \in L\}$$

[1] Leduc et al. referred to star interfaces as "interfaces." We are introducing the term star interfaces to make it easier to refer to this type of interface structure.

[2] The operation **sync** is the synchronous product operation from CTCT [39].

B. Serial Interface Properties and Theorems

We now present the interface requirements that the system must satisfy to ensure that it interacts with the interface correctly. We then define the nonblocking and controllability requirements each level must satisfy. Refer to [26] for a more detailed explanation of the requirements. The only difference between the definitions and theorems in this paper and the ones in [21, 22] is that the definitions and theorems now all refer to command-pair interfaces.[3]

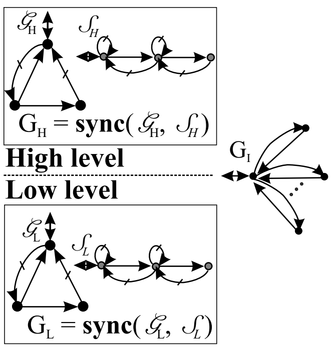

Fig. 5. Plant and Supervisor Subplant Decomposition

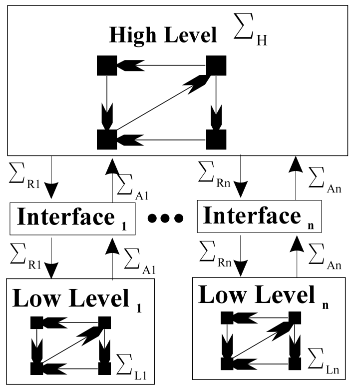

Fig. 6. Parallel Interface Block Diagram.

Serial Interface Consistent: The system composed of DES G_H, G_L and G_I, is *serial interface consistent* with respect to the alphabet partition $\Sigma := \Sigma_H \dot\cup \Sigma_L \dot\cup \Sigma_R \dot\cup \Sigma_A$, if the following properties are satisfied:

Multi-level Properties

1. The event set of G_H is Σ_{IH}, and the event set of G_L is Σ_{IL}.
2. G_I is a command-pair interface for the alphabet partition $\Sigma := \Sigma_H \dot\cup \Sigma_L \dot\cup \Sigma_R \dot\cup \Sigma_A$

High Level Properties

3. $\mathcal{H}\Sigma_A \cap \mathcal{I} \subseteq \mathcal{H}$

Low Level Properties

4. $\mathcal{L}\Sigma_R \cap \mathcal{I} \subseteq \mathcal{L}$
5. $(\forall s \in \Sigma^*.\Sigma_R \cap \mathcal{L} \cap \mathcal{I})$ $[\mathrm{Elig}_{\mathcal{L}\cap\mathcal{I}}(s\Sigma_L^*) \cap \Sigma_A = \mathrm{Elig}_{\mathcal{I}}(s) \cap \Sigma_A]$
   where $\mathrm{Elig}_{\mathcal{L}\cap\mathcal{I}}(s\Sigma_L^*) := \cup_{l\in\Sigma_L^*} \mathrm{Elig}_{\mathcal{L}\cap\mathcal{I}}(sl)$
6. $(\forall s \in \mathcal{L} \cap \mathcal{I})$ $[s \in \mathcal{I}_m \Rightarrow (\exists l \in \Sigma_L^*)\ sl \in \mathcal{L}_m \cap \mathcal{I}_m]$

Serial Level-wise Nonblocking: The system composed of DES G_H, G_L, and G_I, is said to be *serial level-wise nonblocking* if the following conditions are satisfied:

(I) $\overline{\mathcal{H}_m \cap \mathcal{I}_m} = \mathcal{H} \cap \mathcal{I}$ *nonblocking at the high level*

(II) $\overline{\mathcal{L}_m \cap \mathcal{I}_m} = \mathcal{L} \cap \mathcal{I}$ *nonblocking at the low level*

We now define the controllability requirements for each level. We adopt the standard partition $\Sigma = \Sigma_u \dot\cup \Sigma_c$, splitting our alphabet into *uncontrollable* and *controllable events.*

[3] Note, the controllability results in this paper are automatic from [21, 22] as they don't rely on the specific structure of the star interface, just its event set. They are included for completeness.

Serial Level-wise Controllable: The system composed of plant components $\mathcal{G}_H$, $\mathcal{G}_L$, supervisors $\mathcal{S}_H$, $\mathcal{S}_L$, and interface G_I, is said to be *serial level-wise controllable* with respect to the alphabet partition $\Sigma := \Sigma_H \dot\cup \Sigma_L \dot\cup \Sigma_R \dot\cup \Sigma_A$, if the following conditions are satisfied:

(I) The alphabet of $\mathcal{G}_H$ and $\mathcal{S}_H$ is Σ_{IH}, the alphabet of $\mathcal{G}_L$ and $\mathcal{S}_L$ is Σ_{IL}, and the alphabet of G_I is Σ_I

(II) $(\mathbf{L}_{\mathcal{S}} \cap \mathcal{I})\Sigma_u \cap \mathbf{L} \subseteq \mathbf{L}_{\mathcal{S}} \cap \mathcal{I}$

(III) $\mathbf{H}_{\mathcal{S}}\Sigma_u \cap (\mathbf{H} \cap \mathcal{I}) \subseteq \mathbf{H}_{\mathcal{S}}$.

We now present our main results for this section, the *serial interface nonblocking theorem* and the *serial controllability theorem.*

Theorem 1 *If the system composed of DES G_H, G_L, and G_I is serial level-wise nonblocking and serial interface consistent with respect to the alphabet partition $\Sigma := \Sigma_H \dot\cup \Sigma_L \dot\cup \Sigma_R \dot\cup \Sigma_A$, then*
$L(G) = \overline{L}_m(G)$, where $G = \mathbf{sync}(G_H, G_L, G_I)$

Proof: See proof in [26].

Theorem 2 *If the system composed of plant components $\mathcal{G}_H$, $\mathcal{G}_L$, supervisors $\mathcal{S}_H$, $\mathcal{S}_L$, and interface G_I, is serial level-wise controllable with respect to the alphabet partition $\Sigma := \Sigma_H \dot\cup \Sigma_L \dot\cup \Sigma_R \dot\cup \Sigma_A$, then:*

$$(\forall s \in L(\mathbf{Plant}) \cap L(\mathbf{Sup}))\ \mathit{Elig}_{L(\mathrm{Plant})}(s) \cap \Sigma_u \subseteq \mathit{Elig}_{L(\mathrm{Sup})}(s)$$

Proof: See proof in [26].

III. Parallel Case

In Section II, we described the serial case for the HISC method where the number of low levels (n) is restricted to one. We now describe the more general setting where we have $n \geq 1$ low levels. Figure 6 shows conceptually the structure and flow of information of such a system. In this new setting, we still have a single high level, but this time it is interacting with $n \geq 1$ independent low levels, communicating with each low level in parallel through a separate interface. We will refer to the number of low levels, n, as the *degree* of the system.

As in the serial case, in order to capture the restriction of the flow of information imposed by the interface, we partition the alphabet of the system into the following analogous pairwise disjoint alphabets: Σ_H, Σ_{R_j}, Σ_{A_j}, and Σ_{L_j}, with $j = 1, \ldots, n$.

For an n^{th} degree parallel system, we assume the high level subsystem is modelled by DES G_H (defined over event set $\dot\cup_{j\in\{1,\ldots,n\}}[\Sigma_{R_j} \dot\cup \Sigma_{A_j}] \dot\cup \Sigma_H$). For $j \in \{1, \ldots, n\}$, the j^{th} low level subsystem is modelled by DES G_{L_j} (defined over event set $\Sigma_{L_j} \dot\cup \Sigma_{R_j} \dot\cup \Sigma_{A_j}$), the j^{th} interface by DES G_{I_j} (defined over event set $\Sigma_{R_j} \dot\cup \Sigma_{A_j}$), and that the overall system has the structure shown in Figure 7. Furthermore, we will refer to the j^{th} low level to mean $\mathbf{sync}(G_{L_j}, G_{I_j})$ and we will assume that the alphabet partition is specified by $\Sigma := \dot\cup_{j\in\{1,\ldots,n\}}[\Sigma_{L_j} \dot\cup \Sigma_{R_j} \dot\cup \Sigma_{A_j}] \dot\cup \Sigma_H$ and that the flat system is taken to be:

$$G = \mathbf{sync}(G_H, G_{L_1}, \ldots, G_{L_n}, G_{I_1}, \ldots, G_{I_n})$$

In order to simplify the notation in proofs, we now introduce the following event sets, natural projections, and

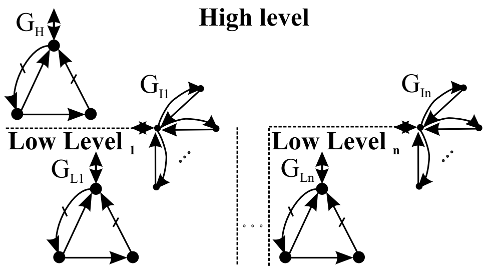

Fig. 7. Two Tiered Structure of Parallel System

useful languages. For the remainder of this section, the index j is defined to have range $\{1, \ldots, n\}$.

$$\begin{array}{rlr} \Sigma_{I_j} := \Sigma_{R_j} \cup \Sigma_{A_j}, & P_{IH} : \Sigma^* \to \Sigma_{IH}^* & \\ \Sigma_{IH} := \cup_{j\in\{1,\ldots,n\}} \Sigma_{I_j} \cup \Sigma_H, & P_{IL_j} : \Sigma^* \to \Sigma_{IL_j}^* & \\ \Sigma_{IL_j} := \Sigma_{L_j} \cup \Sigma_{I_j}, & P_{I_j} : \Sigma^* \to \Sigma_{I_j}^* & \\ \mathcal{H} := P_{IH}^{-1}(L(G_H)), & \mathcal{H}_m := P_{IH}^{-1}(L_m(G_H)) & \subseteq \Sigma^* \\ \mathcal{L}_j := P_{IL_j}^{-1}(L(G_{L_j})), & \mathcal{L}_{m_j} := P_{IL_j}^{-1}(L_m(G_{L_j})) & \subseteq \Sigma^* \\ \mathcal{I}_j := P_{I_j}^{-1}(L(G_{I_j})), & \mathcal{I}_{m_j} := P_{I_j}^{-1}(L_m(G_{I_j})) & \subseteq \Sigma^* \end{array}$$

As in the serial case, we need to be able to decompose the n^{th} degree $(n \geq 1)$ *parallel interface system* into its plant and supervisor components.

We now define the high level plant to to be $\mathcal{G}_H$, and the high level supervisor to be $\mathcal{S}_H$ (both defined over Σ_{IH}). Similarly, the j^{th} *low level plant* and *supervisor* are $\mathcal{G}_{L_j}$ and $\mathcal{S}_{L_j}$(defined over Σ_{IL_j}). We now define the high level subsystem and *the j^{th} low level subsystem* as follows:

$$G_H := \mathbf{sync}(\mathcal{G}_H, \mathcal{S}_H) \qquad G_{L_j} := \mathbf{sync}(\mathcal{G}_{L_j}, \mathcal{S}_{L_j})$$

We can now define our flat supervisor and plant as well as some useful languages as follows:

$$\begin{array}{rcll} \mathbf{Plant} & := & \mathbf{sync}(\mathcal{G}_H, \mathcal{G}_{L_1}, \ldots, \mathcal{G}_{L_n}) & \\ \mathbf{Sup} & := & \mathbf{sync}(\mathcal{S}_H, \mathcal{S}_{L_1}, \ldots, \mathcal{S}_{L_n}, G_{I_1}, \ldots, G_{I_n}) & \\ \mathbf{H} & := & P_{IH}^{-1} L(\mathcal{G}_H), & \mathbf{H}_{\mathcal{S}} := P_{IH}^{-1} L(\mathcal{S}_H), \subseteq \Sigma^* \\ \mathbf{L}_j & := & P_{IL_j}^{-1} L(\mathcal{G}_{L_j}), & \mathbf{L}_{\mathcal{S}_j} := P_{IL_j}^{-1} L(\mathcal{S}_{L_j}), \subseteq \Sigma^* \end{array}$$

A. Serial System Extraction

As the event set of each low level is mutually exclusive from the event sets of the other low levels, we can consider the *parallel interface system* as n *serial interface systems* by choosing one low level and ignoring the others. This will allow us to reuse our existing definitions and results for serial interface systems.

In this section, we introduce the concept of *serial system extractions* for an n^{th} degree $(n \geq 1)$ parallel interface system, shown conceptually in Figure 8 in terms of subsystems. Below we give the general form of the definition. The parallel subsystem form of the definition can be obtained by using the identities $G_H = \mathbf{sync}(\mathcal{G}_H, \mathcal{S}_H)$, $G_L = \mathbf{sync}(\mathcal{G}_L, \mathcal{S}_L)$, and $G_{L_j} = \mathbf{sync}(\mathcal{G}_{L_j}, \mathcal{S}_{L_j})$.

j^{th} **Serial System Extraction:** For the n^{th} degree $(n \geq 1)$ parallel interface system composed of DES $\mathcal{G}_H, \mathcal{G}_{L_1}, \ldots, \mathcal{G}_{L_n}, \mathcal{S}_H, \mathcal{S}_{L_1}, \ldots, \mathcal{S}_{L_n}, G_{I_1}, \ldots, G_{I_n}$, with alphabet partition $\Sigma := \dot\cup_{k\in\{1,\ldots,n\}} [\Sigma_{L_k} \dot\cup \Sigma_{R_k} \dot\cup \Sigma_{A_k}] \dot\cup \Sigma_H$, the j^{th} serial system extraction, denoted by *system*(j), is composed of the following elements:

$$\begin{array}{rcl} \mathcal{G}_H(j) & := & \mathbf{sync}(\mathcal{G}_H, G_{I_1}, \ldots, G_{I_{(j-1)}}, G_{I_{(j+1)}}, \ldots, G_{I_n}) \\ \mathcal{S}_H(j) & := & \mathcal{S}_H,\, \mathcal{G}_L(j) := \mathcal{G}_{L_j},\, \mathcal{S}_L(j) := \mathcal{S}_{L_j},\, G_I(j) := G_{I_j} \\ \Sigma_H(j) & := & \dot\cup_{k\in\{1,\ldots,(j-1),(j+1),\ldots,n\}} \Sigma_{I_k} \dot\cup \Sigma_H \\ \Sigma_L(j) & := & \Sigma_{L_j}\ , \quad \Sigma_R(j) := \Sigma_{R_j}\ , \quad \Sigma_A(j) := \Sigma_{A_j} \\ \Sigma(j) & := & \Sigma_H(j) \dot\cup \Sigma_L(j) \dot\cup \Sigma_R(j) \dot\cup \Sigma_A(j) \end{array}$$

B. Parallel Case Definitions and Theorems

In this section we present a set of properties that are equivalent to their serial interface counterparts.

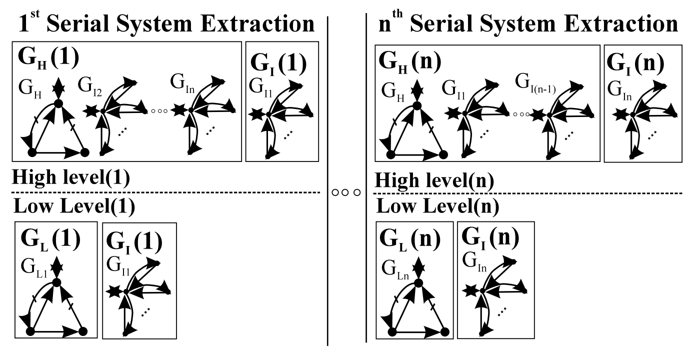

Fig. 8. The Serial System Extraction

Interface Consistent: The n^{th} degree $(n \geq 1)$ parallel interface system composed of DES $G_H, G_{L_1}, \ldots, G_{L_n}, G_{I_1}, \ldots, G_{I_n}$, is *interface consistent* with respect to alphabet partition $\Sigma := \dot\cup_{k\in\{1,\ldots,n\}} [\Sigma_{L_k} \dot\cup \Sigma_{R_k} \dot\cup \Sigma_{A_k}] \dot\cup \Sigma_H$, if:
$(\forall j \in \{1, \ldots, n\})$ The j^{th} serial system extraction of the system is serial interface consistent.

Level-wise Nonblocking: The n^{th} degree $(n \geq 1)$parallel interface system composed of DES $G_H, G_{L_1}, \ldots, G_{L_n}, G_{I_1}, \ldots, G_{I_n}$, is *level-wise nonblocking* with respect to the alphabet partition $\Sigma := \dot\cup_{k\in\{1,\ldots,n\}} [\Sigma_{L_k} \dot\cup \Sigma_{R_k} \dot\cup \Sigma_{A_k}] \dot\cup \Sigma_H$, if:
$(\forall j \in \{1, \ldots, n\})$ The j^{th} serial system extraction of the system is serial level-wise nonblocking.

Level-wise Controllable: The n^{th} degree $(n \geq 1)$ parallel interface system composed of DES $\mathcal{G}_H, \mathcal{G}_{L_1}, \ldots, \mathcal{G}_{L_n}, \mathcal{S}_H, \mathcal{S}_{L_1}, \ldots, \mathcal{S}_{L_n}, G_{I_1}, \ldots, G_{I_n}$, is *level-wise controllable* with respect to alphabet partition $\Sigma := \dot\cup_{k\in\{1,\ldots,n\}} [\Sigma_{L_k} \dot\cup \Sigma_{R_k} \dot\cup \Sigma_{A_k}] \dot\cup \Sigma_H$, if:
$(\forall j \in \{1, \ldots, n\})$ The j^{th} serial system extraction of the system is serial level-wise controllable.

We now present our nonblocking theorem and controllability theorem for parallel interface systems.

Theorem 3 *If the n^{th} degree $(n \geq 1)$ parallel interface system composed of DES $G_H, G_{L_1}, \ldots, G_{L_n}, G_{I_1}, \ldots, G_{I_n}$, is level-wise nonblocking and interface consistent with respect to the alphabet partition $\Sigma := \dot\cup_{k\in\{1,\ldots,n\}} [\Sigma_{L_k} \dot\cup \Sigma_{R_k} \dot\cup \Sigma_{A_k}] \dot\cup \Sigma_H$, then*
$L(G) = \overline{L}_m(G)$,
where $G = \mathbf{sync}(G_H, G_{L1}, \ldots, G_{Ln}, G_{I1}, \ldots, G_{In})$

Proof: See proof in [26].

Theorem 4 *If the n^{th} degree ($n \geq 1$) parallel interface system composed of plant components $\mathcal{G}_H, \mathcal{G}_{L_1}, \ldots, \mathcal{G}_{L_n}$, supervisors $\mathcal{S}_H, \mathcal{S}_{L_1}, \ldots, \mathcal{S}_{L_n}$, and interfaces $G_{I_1}, \ldots, G_{I_n}$, is level-wise controllable with respect to the alphabet partition $\Sigma := \dot{\cup}_{k \in \{1,\ldots,n\}}[\Sigma_{L_k} \dot{\cup} \Sigma_{R_k} \dot{\cup} \Sigma_{A_k}] \dot{\cup} \Sigma_H$, then*

$$(\forall s \in L(\mathbf{Plant}) \cap L(\mathbf{Sup})) \quad Elig_{L(\text{Plant})}(s) \cap \Sigma_u \subseteq Elig_{L(\text{Sup})}(s)$$

Proof: See proof in [26].

IV. Application to the AIP

We now revisit an application to a large manufacturing system, the Atelier Inter-établissement de Productique (AIP) discussed in [20], to illustrate the use of command-pair interfaces. The AIP, shown in Figure 9, is a highly automated manufacturing system consisting of a central loop (CL) and four external loops (EL), three assembly stations (AS), an input/output (I/O) station, and four inter-loop transfer units (TU). The I/O station is where the pallets enter and leave the system. Pallets can be of type 1 or of type 2, and it is assumed that the type of the pallet entering is random.

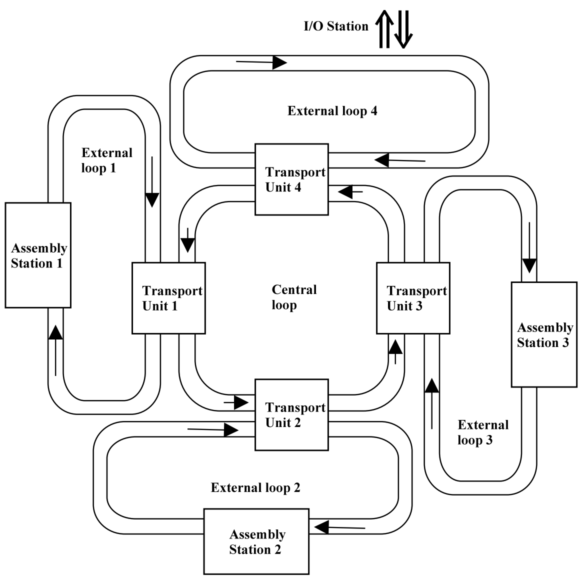

Fig. 9. The Atelier Inter-établissement de Productique

A. Assembly Stations

Each assembly station consists of a robot to perform assembly tasks, an extractor to transfer the pallet from the conveyor to the robot, and sensors, and a read/write (R/W) device to access the pallet's electronic label.

Although the assembly stations are similar, they differ with respect to functionality and reliability. Station 1 is capable of performing task1A and task1B, while station 2 can perform task2A and task2B. Station 3 can perform all four tasks, function as a pallet repair station, and substitute for the other stations when they are down.

B. Transport Units

The transport units are used to transfer pallets between the central loop, and the external loops. Each one consists of a transport drawer, sensors, a R/W device, as well as pallet gates and pallet stops, to control access from the given loop.

C. Using Command-pair Interfaces

In [20], the AIP was modelled using only star interfaces. The system was designed as a 7^{th} degree parallel interface system, with the low levels representing the three assembly stations, and four transfer units. For full design details, refer to [26].

The design of the low level for assembly station 1 was poorly suited to being modelled by star interfaces. This can be seen by examining its star interface, shown in Figure 10. We see that the AS1 has two request events, *ProcPallet.AS1* and *DoRpr.AS1*. Clearly, it only makes sense to do a repair, after the answer event *ASDwn.k* has occurred. Also, it doesn't make sense to try to process a pallet while the AS is down. With command-pair interfaces, modelling this is easily accomplished as in Figure 10.

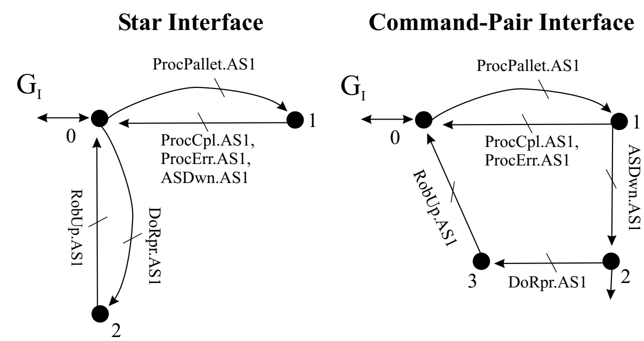

Fig. 10. Interfaces for Assembly Station 1

In particular, representing the assembly station as a star interface is difficult because of **Point 5** of the serial interface consistent definition. This point says that after a request event has occurred (such as *ProcPallet.AS1*), then all answer events that the interface says can follow the event (ie. for star interface, events at *state 1*) must be possible after at most a sequence of low level events. However once the station is down, the physical events *ProcCpl.AS1* and *ProcErr.AS1* can't occur until it has been repaired. How this was resolved in [20] can be seen in Figure 11 where events *RobDwn.AS1, RtasksCpl.AS1, RobUp.AS1,* and *AssmbErrA/B.AS1* roughly correspond to the station's answer events and *ProcType1/2.AS1* and *DoRpr.AS1* roughly correspond to the request events. We see that if the robot is down (*state s11*) and a request to process a pallet is made, all answer events are possible but which one occurs is randomly selected and has no physical meaning. This ugly kludge is unnecessary if command-pair interfaces are used.

V. Complexity Analysis

To aid in investigating HISC, we have developed software routines to verify that a system satisfies the conditions: serial level-wise nonblocking and controllable, and serial interface consistent. The routines were developed by Leduc during his collaboration with Siemens Corporate Research and they use the algorithms described in [26].

Analyzing the steps required to verify the above conditions, we see that they consist of verifying system properties (ie. is G_I a command-pair interface), high level properties and low level properties. We next note that we can treat the high level as a low level for our analysis as the conditions to be checked are equivalent. By grouping the system properties with the low level properties, means verifying two "low levels" (components) can be used as an upper bound for verifying the system.

As we can not perform an analytic analysis as the only sourcecode available is copyrighted by Siemens and cannot be released, we follow the advice of Goodrich et al. [15] and use experimental algorithm analysis to estimate the worse case time complexity for per component analysis. This is sufficient as the per component complexity only contributes a constant term to the complexity of evaluating a system.

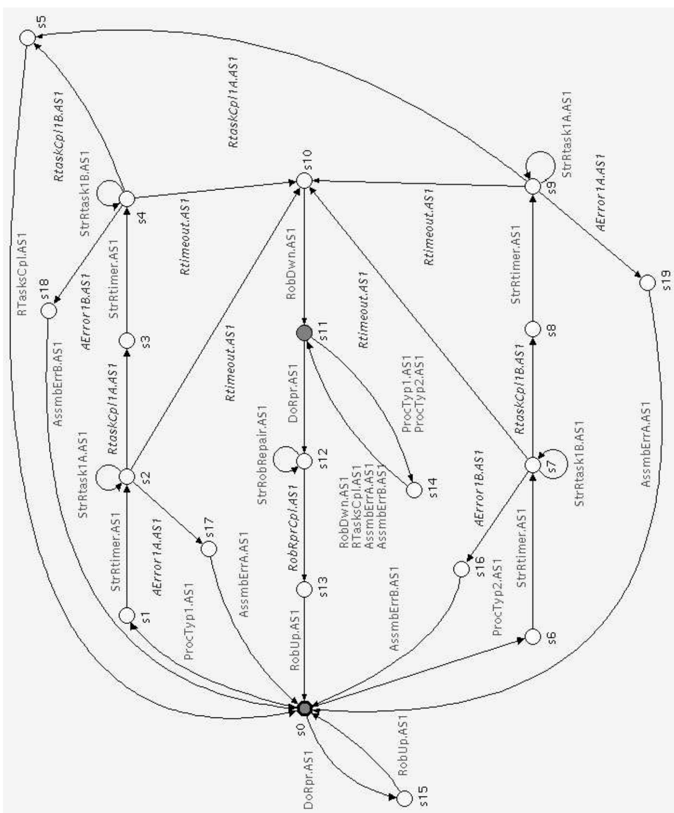

Fig. 11. DoRobotTasks.AS1

To perform this analysis, we will assume that the running time for one component is of the form $t(x) = bx^c$ with x the state size of our component and for some constants $b > 0$ and $c > 0$. We then use the *power test* discussed in [15] to experimentally determine the worst case running time to be $t(x) = (8.56 \times 10^{-9})x^3$ which makes the algorithm $\mathbf{O}(x^3)$ (chapter 6 of [26]).

We next consider verifying an n^{th} degree parallel interface system. To do this, we must check that $3n + 1$ event sets are pairwise disjoint and check that n serial extraction systems are serial level-wise nonblocking and controllable, and serial interface consistent. We let $m = n + 1$ be the number of components to be verified. We also assume that the statespace x of each component and the cardinality of the system's event set (Σ) are bounded with upper bounds $N \geq 0$ and $N_\Sigma \geq 0$, respectively. We further assume that the cardinality of event sets Σ_H, $\Sigma_{L_1}, \ldots, \Sigma_{L_n}$, $\Sigma_{R_1}, \ldots, \Sigma_{R_n}$ and $\Sigma_{A_1}, \ldots, \Sigma_{A_n}$ is each bounded by $N_{\Sigma'} \geq 0$.

It can be shown that verifying $3n+1$ event sets are pairwise disjoint can be performed by $\frac{9}{2}m^2 - \frac{15}{2}m + 3$ empty intersection tests which are each $\mathbf{O}(N_{\Sigma'}^2)$ [34]. The whole process is thus $\mathbf{O}(\frac{9}{2}m^2 N_{\Sigma'}^2 - \frac{15}{2}mN_{\Sigma'}^2 + 3N_{\Sigma'}^2) = \mathbf{O}(m^2)$.

To verify the n serial extraction systems, we must perform the per component analysis $2n$ times giving $\mathbf{O}(2n \cdot x^3) = \mathbf{O}(2mN^3 - 2N^3) = \mathbf{O}(m)$ as N is a constant. Combining the two steps, we have $\mathbf{O}(m^2 + m) = \mathbf{O}(m^2)$. This is only practical as long as N isn't too large.

We next compare the HISC method to verifying nonblocking of flat system. Based on the work of Rudie [34], it can be shown that the monolithic approach is $\mathbf{O}(N^{2m})$ and thus our scales significantly better. Table I illustrates this for terms $T_1 = N^{2m}$, and $T_2 = 2mN^3 - 2N^3 + \frac{9}{2}m^2N_{\Sigma'}^2 - \frac{15}{2}mN_{\Sigma'}^2 + 3N_{\Sigma'}^2$. We see that even for $m = 2$ (serial system) and $N = 10^6$, our approach is six orders of magnitude better. To put this into perspective, if our algorithm ran for one hour, the monolithic algorithm would require 114 years!

		$m=2$		$m=9$	
N	$N_{\Sigma'}$	T_1	T_2	T_1	T_2
10^3	10^2	10^{12}	2×10^9	10^{54}	1.60×10^{10}
10^6	10^2	10^{24}	2×10^{18}	10^{108}	1.60×10^{19}

TABLE I
Parallel Algorithm Comparison

The cost for this increase in computational efficiency is a more restrictive architecture. As similar interface-based approaches are common in both hardware and software, we are confident that our method will be widely applicable.

UPDATE: The analysis presented here relies on the assumption that the statespace of each component is bounded by the constant N. As long as this assumption is reasonable, the analysis is correct. For the DES $G_H, G_{L_1}, \ldots, G_{L_n}, G_{I_1}, \ldots, G_{I_n}$, this assumption is reasonable.

However, when analyzing the conditions *interface consistent, level-wise nonblocking,* and *level-wise controllable,* we must construct *serial extraction systems* (see Section A). to analyze the corresponding serial conditions. For example, to verify that the *parallel interface system* is *interface consistent,* we must verify that all n *serial system extractions (subsystem form)* are *serial interface consistent.* To verify the latter condition, we must use the component $G_H(j) := G_H||_s G_{I_1}||_s \ldots ||_s G_{I_{(j-1)}}||_s G_{I_{(j+1)}}||_s \ldots ||_s G_{I_n}$ with the serial algorithms we developed in [26]. Unlike the DES G_H, component $G_H(j)$ grows proportionally to n, thus the assumption that $G_H(j)$ is bounded by N is questionable. In this view, the above analysis is a bit too optimistic and is thus more in line with an average or best case analysis. This does not mean that the approach does not have great potential to scale. For a good scalability discussion, see [23].

VI. Conclusions

HISC offers an effective method to model systems with a natural client-server architecture. Command-pair interfaces extends the modelling flexibility for interfaces by allowing the representation of low level state information, thus enabling many new systems to be modelled as low levels.

As each requirement can be verified using only one subsystem, the entire plant model never needs to be constructed or traversed, offering potentially significant savings in computation. We have shown this concretely by proving that the time complexity for analyzing a system by our method is $\mathbf{O}(m^2)$, as compared to a monolithic analysis which is $\mathbf{O}(N^{2m})$.

References

[1] N. Alsop. *Formal Techniques for the Procedural Control of Industrial Processes*. PhD thesis, Department of Chemical Engineering and Chemical Technology, Imperial College of Science, Technology and Medicine, London, 1996.

[2] A. Arnold. *Finite Transition Systems*. Prentice Hall, 1994.

[3] George Barrett and Stephane Lafortune. Decentralized supervisory control with communicating controllers. *IEEE Trans. Automatic Control*, 45(9):1620–1638, 2000.

[4] Y. Brave and M. Heymann. Control of discrete event systems modeled as hierarchical state machines. *IEEE Trans. on Automatic Control*, 38(12):1803–1819, Dec 1993.

[5] J.R. Burch, Edmund M. Clarke, and K.L. McMillan. Symbolic model checking: 10^{20} states and beyond. *Information and Computation*, 98:142–170, 1992.

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

[7] Haoxun Chen and Hans-Michael Hanisch. Model aggregation for hierarchical control synthesis of discrete event systems. In *Proc. 39th Conf. Decision Contr.*, pages 418–423, Sydney, Australia, December 2000.

[8] S.-L. Chen. Existence and design of supervisors for vector discrete event systems. Master's thesis, Department of Electrical Engineering, University of Toronto, Toronto, Ont, 1992.

[9] S.-L. Chen. *Control of Discrete-Event Systems of Vector and Mixed Structural Type*. PhD thesis, Department of Electrical and Computer Engineering, University of Toronto, Toronto, Ont, 1996.

[10] Yi-Liang Chen and Feng Lin. Hierarchical modeling and abstraction of discrete event systems using finite state machines with parameters. In *Proc. 40th Conf. Decision Contr.*, pages 4110–4115, Orlando, USA, December 2001.

[11] Edmund M. Clarke, O. Grümberg, and K. Hamaguchi. Another look at LTL model checking. In *Proc. of 6th Conf. on Computer Aided Verification*, number 818 in LNCS, pages 415–427. Springer-Verlag, 1994.

[12] Edmund M. Clarke, Orna Grumberg, and Doron A. Peled. *Model Checking*. MIT Press, 2001.

[13] E. W. Endsley, M. R. Lucas, and D. M. Tilbury. Modular design and verification of logic control for reconfigurable machining systems. Submitted to Discrete Event Dynamic Systems: Theory and Applications.

[14] Peyman Gohari-Moghadam. A linguistic framework for controlled hierarchical DES. Master's thesis, Department of Electrical and Computer Engineering, University of Toronto, Toronto, Ont, 1998.

[15] Michael T. Goodrich and Roberto Tamassia. *Algorithm Design*. Wiley, 2001.

[16] O. Grümberg and D.E. Long. Model checking and modular verification. In *Proc. of CONCOUR'91*, number 527 in LNCS, pages 361–375. Springer-Verlag, 1991.

[17] Daniel M. Hoffman and David M. Weiss, editors. *Software Fundamentals. Collected Papers by David L. Parnas*. Addison Wesley, 2001.

[18] Paul Hubbard and Peter E. Caines. Trace-DC hierarchical supervisory control with applications to transfer-lines. In *Proc. 37th Conf. Decision Contr.*, pages 3293–3298, Tampa, Florida USA, December 1998.

[19] Mark Lawford. *Model Reduction of Discrete Real-Time Systems*. PhD thesis, Department of Electrical and Computer Engineering, University of Toronto, Toronto, Ont, 1997.

[20] R. Leduc, M. Lawford, and W. Murray Wonham. Hierarchical interface-based supervisory control: AIP example. In *Proc. of 39th Annual Allerton Conference on Comm., Contr., and Comp.*, pages 396–405, Oct 2001.

[21] R.J. Leduc, B.A. Brandin, and W. Murray Wonham. Hierarchical interface-based non-blocking verification. In *Proceedings of the Canadian Conference on Electrical and Computer Engineering*, pages 1–6, May 2000.

[22] R.J. Leduc, B.A. Brandin, W. Murray Wonham, and M. Lawford. Hierarchical interface-based supervisory control: Serial case. In *Proc. of 40th Conf. Decision Contr.*, pages 4116–4121, Orlando, USA, December 2001.

[23] R.J. Leduc, M. Lawford, and W. Murray Wonham. Hierarchical interface-based supervisory control, part II: Parallel case. Submitted to IEEE Trans. Automatic Control, Aug, 2003. An earlier version is available as Software Quality Research Laboratory Report No. 13, Dept. of Computing and Software, McMaster University, Hamilton, ON. [ONLINE] Available: http://www.cas.mcmaster.ca/sqrl/sqrl_reports.html.

[24] R.J. Leduc, W. Murray Wonham, and M. Lawford. Hierarchical interface-based supervisory control: Parallel case. In *Proc. of 39th Annual Allerton Conference on Comm., Contr., and Comp.*, pages 386–395, Oct 2001.

[25] Ryan Leduc. PLC implementation of a DES supervisor for a manufacturing testbed: An implementation perspective. Master's thesis, Department of Electrical and Computer Engineering, University of Toronto, Toronto, Ont, 1996.

[26] Ryan Leduc. *Hierarchical Interface-based Supervisory Control*. PhD thesis, Department of Electrical and Computer Engineering, University of Toronto, Toronto, Ont, 2002.

[27] Y. Li. *Control of Vector Discrete-Event Systems*. PhD thesis, Department of Electrical Engineering, University of Toronto, Toronto, Ont, 1991.

[28] F. Lin and W. Murray Wonham. Decentralized control and coordination of discrete-event systems with partial observations. In *Proc. 27th IEEE Conf. Decision Contr.*, pages 1125–1130, Dec 1988.

[29] Hong Liu, Jun-Cheol Park, and Raymond E. Miller. On hybrid synthesis for hierarchical structured petri nets. Technical report, Department of Computer Science, University of Maryland, College Park, MD, 1996.

[30] K.L. McMillan. *Symbolic Model Checking*. Kluwer, 1992.

[31] John O. Moody and Panos J. Antsaklis. *Supervisory Control of Discrete Event Systems using Petri Nets*. Kluwer Academic Publishers, 1998.

[32] Ken Qian Pu. Modeling and control of discrete-event systems with hierarchical abstraction. Master's thesis, Dept. of Electrical and Computer Engineering, University of Toronto, Toronto, Ont, 2000.

[33] M.H. de Queiroz and J.E.R. Cury. Modular supervisory control of large scale discrete event systems. In *Proceedings of WODES 2000*, pages 103–110, Ghent, Belgium, Aug 2000.

[34] K. Rudie. Software for the control of discrete-event systems: A complexity study. Master's thesis, Dept. of Electrical and Computer Engineering, University of Toronto, Toronto, Ont, 1988.

[35] Karen Rudie and W. Murray Wonham. Think globally, act locally: decentralized supervisory control. *IEEE Trans. on Automatic Control*, 37(11):1692–1708, Nov 1992.

[36] G. Stremersch and R.K. Boel. Decomposition of the supervisory control problem for Petri nets under preservation of maximal permissiveness. *IEEE Trans. Automatic Control*, 46(9):1490–1496, 2001.

[37] Bing Wang. Top-down design for RW supervisory control theory. Master's thesis, Department of Electrical and Computer Engineering, University of Toronto, Toronto, Ont, 1995.

[38] K.C. Wong. *Discrete-Event Control Architecture: An Algebraic Approach*. PhD thesis, Department of Electrical and Computer Engineering, University of Toronto, Toronto, Ont, 1994.

[39] W. Murray Wonham. *Notes on Control of Discrete-Event Systems*. Department of Electrical and Computer Engineering, University of Toronto, 2002.

[40] Weimin Wu, Hongye Su, Jian Chu, and Haifeng Zhai. Hierarchical control of DES based on colored petri nets. In *Proc. of IEEE Systems, Man, and Cybernetics*, volume 3, pages 1571–1576, 2001.

[41] Z.H. Zhang. Smart TCT: an efficient algorithm for supervisory control design. Master's thesis, Dept. of Electrical and Computer Engineering, University of Toronto, Toronto, Ont, 2001.

[42] H. Zhong and W. Murray Wonham. On the consistency of hierarchical supervision in discrete-event systems. *IEEE Trans. on Automatic Control*, 35(10):1125–1134, Oct 1990.

[43] Meng Chu Zhou, David T. Wang, and Israel Mayk. Using petri nets for object-oriented design of command and control systems. *International Journal of Intelligent Control and Systems*, 2(2):287–300, 1998.