Hierarchical Interface-Based Supervisory Control—Part II: Parallel Case

Ryan J. Leduc, *Member, IEEE*, Mark Lawford, *Member, IEEE*, and W. M. Wonham, *Life Fellow, IEEE*

***Abstract*—In this paper, we present a hierarchical method that decomposes a discrete-event system (DES) into a high-level subsystem which communicates with $n \geq 1$ parallel low-level subsystems, through separate interfaces which restrict the interaction of the subsystems. It is a generalization of the serial case $(n = 1)$ described in Part I of this paper, where we define an interface and a set of interface consistency properties that can be used to verify if a DES is nonblocking and controllable. Each clause of the definition can be verified using a single subsystem; thus the complete system model never needs to be stored in memory, offering potentially significant savings in computational resources. We provide algorithms for verifying these new properties, and briefly discuss the computational complexity of the method. Finally, we present an application to a large manufacturing example with an estimated worst-case closed-loop state–space size of 2.9×10^{21}.**

***Index Terms*—Automata, discrete-event systems (DESs), formal methods, hierarchical systems, interfaces.**

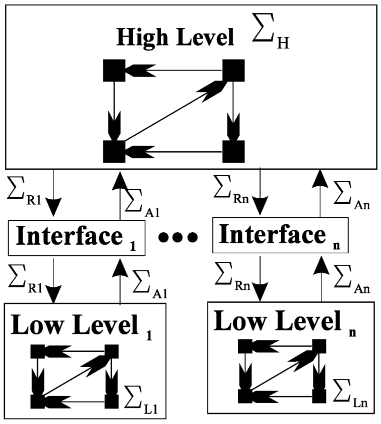

Fig. 1. Parallel interface block diagram.

I. Introduction

In this paper, we present an interface-based hierarchical method to verify if a system is nonblocking and controllable, extending the work in [1] and [2].[1] In this paper, we restrict attention to two-level systems where the system is split into a high-level subsystem interacting with $n \geq 1$ parallel low-level subsystems via a separate interface DES. The most significant feature which distinguishes this paper from some current work along similar lines (e.g., [5]) is the results on nonblocking. A more detailed literature review, motivation for hierarchical interface-based supervisory control (HISC) and discussion of DES preliminaries, are given in Part I of this paper [1].

In the serial case of HISC, we propose a master-slave system, where a *high-level subsystem* sends a command to a *low-level subsystem*, which then performs the indicated task and returns a reply. Fig. 1 in [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. In this case n, the number of low-level systems, is equal to 1.

Referring to the primary definitions of [1] as needed, we generalize the set of (local) consistency properties that can be used to verify if a DES is globally nonblocking and controllable to $n \geq 1$ low-level subsystems in Section II. Next, we present algorithms for verifying these new properties, briefly discussing computational complexity in Section III. We close with an application to a manufacturing example with an estimated worst-case closed-loop state–space size of 2.9×10^{21} in Section IV.

Manuscript received August 27, 2003. Recommended by Associate Editor R. Boel.

R. J. Leduc and M. Lawford are with the Department of Computing and Software, McMaster University, Hamilton, ON L8S 4K1, Canada (e-mail: leduc@mcmaster.ca; lawford@mcmaster.ca).

W. M. Wonham is with the Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON M5S 3G4, Canada (e-mail: wonham@control.utoronto.ca).

Digital Object Identifier 10.1109/TAC.2005.854612

[1]Early results of this work can be found in [3] and [4].

II. Parallel Case

In [1], we described our method for the serial case where the number of low-level subsystems or *degree* of the system (n) is restricted to one. Now we extend to the more general setting with $n \geq 1$ low-level subsystems. Fig. 1 shows conceptually the structure and flow of information. In this new setting, a single high-level subsystem, interacts with $n \geq 1$ independent low-level subsystems, communicating with each low-level subsystem in parallel through a separate interface.

As in the serial case, to restrict the flow of information imposed by the interface, we partition the system alphabet into the following analogous pairwise disjoint alphabets:

$$\Sigma := \Sigma_H \dot{\cup} \left(\dot{\bigcup_{k=1,\ldots,n}} [\Sigma_{L_k} \dot{\cup} \Sigma_{R_k} \dot{\cup} \Sigma_{A_k}] \right). \tag{1}$$

For an nth degree parallel system, the high-level subsystem is modeled by DES $\mathbf{G}_H$ (defined over event set $\dot{\cup}_{k \in \{1,\ldots,n\}} [\Sigma_{R_k} \dot{\cup} \Sigma_{A_k}] \dot{\cup} \Sigma_H$). For $j \in \{1,\ldots,n\}$, the jth low-level subsystem is modeled by DES $\mathbf{G}_{L_j}$ (defined over event set $\Sigma_{L_j} \dot{\cup} \Sigma_{R_j} \dot{\cup} \Sigma_{A_j}$), the jth interface by DES $\mathbf{G}_{I_j}$ (defined over event set $\Sigma_{R_j} \dot{\cup} \Sigma_{A_j}$), so that the overall system has the structure shown in Fig. 2. By the jth *low-level* we mean

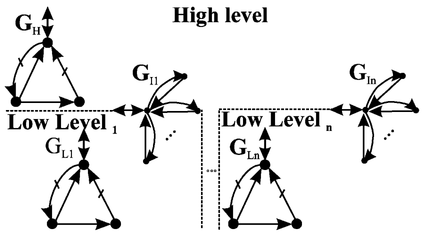

Fig. 2. Two-tiered structure of parallel system.

$\mathbf{G}_{L_j}\|\mathbf{G}_{I_j}$, assume that the alphabet partition is given by (1), and take the *flat system* to be

$$\mathbf{G} = \mathbf{G}_H\|\mathbf{G}_{L_1}\|\ldots\|\mathbf{G}_{L_n}\|\,\mathbf{G}_{I_1}\|\ldots\|\mathbf{G}_{I_n}.$$

To simplify notation in proofs, we bring in the following event sets, natural projections, and languages. For the remainder of this section, the index j has range $\{1,\ldots,n\}$

$$\begin{aligned}\Sigma_{I_j} &:= \Sigma_{R_j}\dot\cup\Sigma_{A_j} \quad P_{I_j}:\Sigma^*\to\Sigma_{I_j}^*\\ \Sigma_{IL_j} &:= \Sigma_{L_j}\dot\cup\Sigma_{I_j} \quad P_{IL_j}:\Sigma^*\to\Sigma_{IL_j}^*\\ \Sigma_{IH} &:= \Sigma_H\cup\bigcup_{k\in\{1,\ldots,n\}}\Sigma_{I_k} \quad P_{IH}:\Sigma^*\to\Sigma_{IH}^*\\ \mathcal{H} &:= P_{IH}^{-1}(L(\mathbf{G}_H)),\quad \mathcal{H}_m := P_{IH}^{-1}(L_m(\mathbf{G}_H))\subseteq\Sigma^*\\ \mathcal{L}_j &:= P_{IL_j}^{-1}\left(L\left(\mathbf{G}_{L_j}\right)\right),\quad \mathcal{L}_{m_j} := P_{IL_j}^{-1}\left(L_m\left(\mathbf{G}_{L_j}\right)\right)\subseteq\Sigma^*\\ \mathcal{I}_j &:= P_{I_j}^{-1}\left(L\left(\mathbf{G}_{I_j}\right)\right),\quad \mathcal{I}_{m_j} := P_{I_j}^{-1}\left(L_m\left(\mathbf{G}_{I_j}\right)\right)\subseteq\Sigma^*.\end{aligned}$$

A. General Form

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

We designate the *high-level plant* as $\mathbf{G}_H^p$, and the *high-level supervisor* as $\mathbf{S}_H$ (both defined over Σ_{IH}). Similarly, the jth *low-level plant* and *supervisor* are $\mathbf{G}_{L_j}^p$ and $\mathbf{S}_{L_j}$ (defined over Σ_{IL_j}). The *high-level subsystem* and *the* jth *low-level subsystem* are

$$\mathbf{G}_H := \mathbf{G}_H^p\|\mathbf{S}_H \qquad \mathbf{G}_{L_j} := \mathbf{G}_{L_j}^p\|\mathbf{S}_{L_j}.$$

The reader should note that the definition of a parallel interface system that we present here in terms of plant and supervisor components, is the *general form* of such systems. The form we defined above (in terms of high and low-level subsystems) is a special case of the general form, on applying the above identities for $\mathbf{G}_H$ and $\mathbf{G}_{L_j}$. We will refer to the original form, used to simplify nonblocking definitions and proofs, as the *parallel subsystem based form.*

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

$$\begin{aligned}\mathbf{Plant} &:= \mathbf{G}_H^p\left\|\mathbf{G}_{L_1}^p\right\|\ldots\|\mathbf{G}_{L_n}^p\\ \mathbf{Sup} &:= \mathbf{S}_H\|\mathbf{S}_{L_1}\|\ldots\|\mathbf{S}_{L_n}\|\,\mathbf{G}_{I_1}\|\ldots\|\mathbf{G}_{I_n}\\ \mathcal{H}^p &:= P_{IH}^{-1}L\left(\mathbf{G}_H^p\right),\quad \mathcal{S}_H := P_{IH}^{-1}L(\mathbf{S}_H),\subseteq\Sigma^*\\ \mathcal{L}_j^p &:= P_{IL_j}^{-1}L\left(\mathbf{G}_{L_j}^p\right),\quad \mathcal{S}_{L_j} := P_{IL_j}^{-1}L\left(\mathbf{S}_{L_j}\right),\subseteq\Sigma^*.\end{aligned}$$

B. Serial System Extraction

As the event set of each low-level is disjoint 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 allows reuse of our previous setup for serial interface systems.

In this section, we introduce the concept of *serial system extractions* for an nth degree $(n\geq 1)$ parallel interface system, shown conceptually in Fig. 3 in terms of subsystems. We first give the subsystem form of the definition, and then the general form (event set definitions not repeated are unchanged). We refer to the jth *serial system extraction*, as the type of the parallel system will make clear which definition is intended.

Definition 1: For the nth degree $(n\geq 1)$ parallel interface system composed of DES $\mathbf{G}_H,\mathbf{G}_{L_1},\ldots,\mathbf{G}_{L_n},\mathbf{G}_{I_1},\ldots,\mathbf{G}_{I_n}$, with alphabet partition (1), the jth *serial system extraction* (subsystem form), denoted by *system*(j), is composed of the following elements:

$$\begin{aligned}\mathbf{G}_H(j) &:= \mathbf{G}_H\|\mathbf{G}_{I_1}\|\ldots\left\|\mathbf{G}_{I_{(j-1)}}\right\|\mathbf{G}_{I_{(j+1)}}\|\ldots\|\mathbf{G}_{I_n}\\ \mathbf{G}_L(j) &:= \mathbf{G}_{L_j}\quad \mathbf{G}_I(j) := \mathbf{G}_{I_j}\\ \Sigma_H(j) &:= \dot\cup_{k\in\{1,\ldots,(j-1),(j+1),\ldots,n\}}\Sigma_{I_k}\dot\cup\Sigma_H\\ \Sigma_L(j) &:= \Sigma_{L_j}\quad \Sigma_R(j) := \Sigma_{R_j}\quad \Sigma_A(j) := \Sigma_{A_j}\\ \Sigma(j) &:= \Sigma_H(j)\dot\cup\Sigma_L(j)\dot\cup\Sigma_R(j)\dot\cup\Sigma_A(j)\\ &= \Sigma-\dot\cup_{k\in\{1,\ldots,(j-1),(j+1),\ldots,n\}}\Sigma_{L_k}.\end{aligned}$$

Definition 2: For the nth degree $(n\geq 1)$ parallel interface system composed of DES $\mathbf{G}_H^p,\mathbf{G}_{L_1}^p,\ldots,\mathbf{G}_{L_n}^p,\mathbf{S}_H,\mathbf{S}_{L_1},\ldots,\mathbf{S}_{L_n},\mathbf{G}_{I_1},\ldots,\mathbf{G}_{I_n}$, with alphabet partition (1), the jth *serial system extraction* (general form), denoted by *system*(j), is composed of the following elements:

$$\begin{aligned}\mathbf{G}_H^p(j) &:= \mathbf{G}_H^p\|\mathbf{G}_{I_1}\|\ldots\left\|\mathbf{G}_{I_{(j-1)}}\right\|\mathbf{G}_{I_{(j+1)}}\|\ldots\|\mathbf{G}_{I_n}\\ \mathbf{S}_H(j) &:= \mathbf{S}_H\quad \mathbf{G}_L^p(j) := \mathbf{G}_{L_j}^p\\ \mathbf{S}_L(j) &:= \mathbf{S}_{L_j}\quad \mathbf{G}_I(j) := \mathbf{G}_{I_j}.\end{aligned}$$

It can be shown that for $n=1$, a parallel interface system reduces to a single serial interface system, and thus a serial system is a special case of a parallel system.

C. Parallel Case Definitions

In this section, we present a set of properties that match their serial interface counterparts from [1]. They all involve interpreting the parallel system as n serial systems by using the serial system extraction definition.

Definition 3: The nth degree $(n\geq 1)$ parallel interface system composed of DES $\mathbf{G}_H,\mathbf{G}_{L_1},\ldots,\mathbf{G}_{L_n},\mathbf{G}_{I_1},\ldots,\mathbf{G}_{I_n}$, is *interface consistent* with respect to alphabet partition (1), if for all $j\in\{1,\ldots,n\}$ the jth serial system extraction of the system is serial interface consistent.

Definition 4: The nth degree $(n\geq 1)$ parallel interface system composed of DES $\mathbf{G}_H,\mathbf{G}_{L_1},\ldots,\mathbf{G}_{L_n},\mathbf{G}_{I_1},\ldots,\mathbf{G}_{I_n}$, is *level-wise nonblocking* with respect to the alphabet partition (1), if for all $j\in\{1,\ldots,n\}$ the jth serial system extraction of the system is serial level-wise nonblocking.

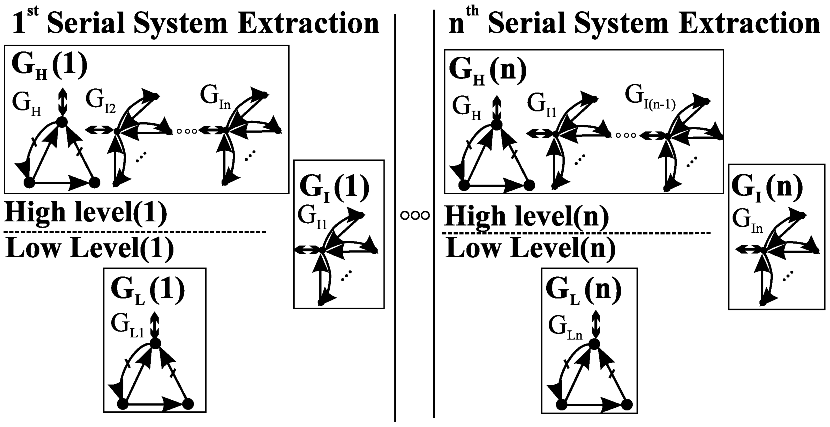

Fig. 3. Serial system extractions.

We now extend serial level-wise controllability to the parallel case, using the standard partition $\Sigma = \Sigma_u \dot{\cup} \Sigma_c$, into uncontrollable and controllable events.

Definition 5: The nth degree $(n \geq 1)$ parallel interface system composed of DES $\mathbf{G}_H^p, \mathbf{G}_{L_1}^p, \ldots, \mathbf{G}_{L_n}^p, \mathbf{S}_H, \mathbf{S}_{L_1}, \ldots, \mathbf{S}_{L_n}, \mathbf{G}_{I_1}, \ldots, \mathbf{G}_{I_n}$, is *level-wise controllable* with respect to alphabet partition (1), if for all $j \in \{1, \ldots, n\}$ the jth serial system extraction of the system is serial level-wise controllable.

We present two propositions that will aid in the use of serial system extractions in proofs. The propositions interpret terminology for the jth serial system extraction in terms of the original parallel system. First, we need the new natural projection, $P_j : \Sigma^* \to \Sigma(j)^*$.

Proposition 1: If the nth degree $(n \geq 1)$ parallel interface system composed of DES $\mathbf{G}_H, \mathbf{G}_{L_1}, \ldots, \mathbf{G}_{L_n}, \mathbf{G}_{I_1}, \ldots, \mathbf{G}_{I_n}$, is interface consistent with respect to the alphabet partition (1), then for the jth serial system extraction, system(j), the following is true.

i) The flat system is: $\mathbf{G}(j) = \mathbf{G}_H \| \mathbf{G}_{L_j} \| \mathbf{G}_{I_1} \| \ldots \| \mathbf{G}_{I_n}$.

ii) The following event sets are: $\Sigma_I(j) = \Sigma_{I_j}, \Sigma_{IH}(j) = \Sigma_{IH}$, and $\Sigma_{IL}(j) = \Sigma_{IL_j}$.

iii) The following inverse natural projections are: $P_{IH}(j)^{-1} = P_j \cdot P_{IH}^{-1}, P_{IL}(j)^{-1} = P_j \cdot P_{IL_j}^{-1}$, and $P_I(j)^{-1} = P_j \cdot P_{I_j}^{-1}$.

iv) The event set of $\mathbf{G}_H(j)$ is $\Sigma_{IH}(j)$, the event set of $\mathbf{G}_L(j)$ is $\Sigma_{IL}(j)$, and the event set of $\mathbf{G}_I(j)$ is $\Sigma_I(j)$.

v) The indicated languages satisfy the following:

$$\mathcal{H}(j) = P_j(\mathcal{H}) \cap [\cap_{k \in \{1,\ldots,(j-1),(j+1),\ldots,n\}} P_j(\mathcal{I}_k)]$$
$$\mathcal{H}_m(j) = P_j(\mathcal{H}_m) \cap [\cap_{k \in \{1,\ldots,(j-1),(j+1),\ldots,n\}} P_j(\mathcal{I}_{m_k})]$$
$$\mathcal{L}(j) = P_j(\mathcal{L}_j) \quad \mathcal{L}_m(j) = P_j(\mathcal{L}_{m_j})$$
$$\mathcal{I}(j) = P_j(\mathcal{I}_j) \quad \mathcal{I}_m(j) = P_j(\mathcal{I}_{m_j}).$$

vi) Languages $\mathcal{H}(j), \mathcal{L}(j)$, and $\mathcal{I}(j)$ are closed.

vii) $\mathcal{H}_m(j) \subseteq \mathcal{H}(j), \mathcal{L}_m(j) \subseteq \mathcal{L}(j)$, and $\mathcal{I}_m(j) \subseteq \mathcal{I}(j)$.

Proof: See the proof in [6]. ■

Proposition 2: If the nth degree $(n \geq 1)$ parallel interface system composed of DES $\mathbf{G}_H^p, \mathbf{G}_{L_1}^p, \ldots, \mathbf{G}_{L_n}^p, \mathbf{S}_H, \mathbf{S}_{L_1}, \ldots, \mathbf{S}_{L_n}, \mathbf{G}_{I_1}, \ldots, \mathbf{G}_{I_n}$, is level-wise controllable with respect to the alphabet partition (1), then for the jth serial system extraction, system (j), the following is true.

i) The flat plant is $\mathbf{Plant}(j) = \mathbf{G}_H^p \| \mathbf{G}_{I_1} \| \ldots \| \mathbf{G}_{I_{(j-1)}} \| \mathbf{G}_{I_{(j+1)}} \| \ldots \| \mathbf{G}_{I_n} \| \mathbf{G}_{L_j}^p$ and the flat supervisor is $\mathbf{Sup}(j) = \mathbf{S}_H \| \mathbf{S}_{L_j} \| \mathbf{G}_{I_j}$.

ii) The following event sets are: $\Sigma_I(j) = \Sigma_{I_j}, \Sigma_{IH}(j) = \Sigma_{IH}$, and $\Sigma_{IL}(j) = \Sigma_{IL_j}$.

iii) The following inverse natural projections are: $P_{IH}(j)^{-1} = P_j \cdot P_{IH}^{-1}, P_{IL}(j)^{-1} = P_j \cdot P_{IL_j}^{-1}$, and $P_I(j)^{-1} = P_j \cdot P_{I_j}^{-1}$.

iv) The alphabet of $\mathbf{G}_H^p(j)$ and $\mathbf{S}_H(j)$ is $\Sigma_{IH}(j)$, the alphabet of $\mathbf{G}_L^p(j)$ and $\mathbf{S}_L(j)$ is $\Sigma_{IL}(j)$, and the alphabet of $\mathbf{G}_I(j)$ is $\Sigma_I(j)$.

v) The indicated languages satisfy the following statements:

$$\mathcal{H}^p(j) = P_j(\mathcal{H}^p) \cap [\cap_{k \in \{1,\ldots,(j-1),(j+1),\ldots,n\}} P_j(\mathcal{I}_k)]$$
$$\mathcal{I}(j) = P_j(\mathcal{I}_j) \quad \mathcal{S}_H(j) = P_j(\mathcal{S}_H)$$
$$\mathcal{L}^p(j) = P_j(\mathcal{L}_j^p) \quad \mathcal{S}_L(j) = P_j(\mathcal{S}_{L_j})$$
$$L(\mathbf{Plant}(j)) = P_j(\mathcal{H}^p) \cap [\cap_{k \in \{1,\ldots,(j-1),(j+1),\ldots,n\}} P_j(\mathcal{I}_k)] \cap P_j(\mathcal{L}_j^p)$$
$$L(\mathbf{Sup}(j)) = P_j(\mathcal{S}_H) \cap P_j(\mathcal{S}_{L_j}) \cap P_j(\mathcal{I}_j).$$

vi) Languages $\mathcal{H}^p(j), \mathcal{S}_H(j), \mathcal{L}^p(j), \mathcal{S}_L(j), \mathcal{I}(j), L(\mathbf{Plant})(j)$, and $L(\mathbf{Sup})(j)$ are closed.

Proof: See the proof in [6]. ■

We now introduce a proposition that provides a useful relationship for natural projections. In the parallel case, we will see several instances of this relationship. First, we need different notation to avoid confusion with that used later. Let $\Sigma_a, \Sigma_b \subseteq \Sigma$ and natural projections $P_k : \Sigma^* \to \Sigma_k^*$, where $k = a, b$.

Proposition 3: If $\Sigma_b \subseteq \Sigma_a$ and $L_b \subseteq \Sigma_b^*$ then $(\forall s \in \Sigma^*) P_a(s) \in P_a \cdot P_b^{-1}(L_b) \Rightarrow s \in P_b^{-1}(L_b)$.

Proof: See the proof in [6]. ■

D. Nonblocking Propositions and Theorem

We will now present **Propositions 4–6**, followed by our main result for this section, **Theorem 1**. The following propositions are analogous to their serial case counterparts.

Our first proposition is analogous to [1, Prop. 1]. It asserts that the low-levels are not dependent on high-level events to reach a marked state.

Proposition 4: If the nth degree $(n \geq 1)$ parallel interface system composed of DES $\mathbf{G}_H, \mathbf{G}_{L_1}, \ldots, \mathbf{G}_{L_n}, \mathbf{G}_{I_1}, \ldots, \mathbf{G}_{I_n}$, is level-wise nonblocking and interface consistent with respect to the alphabet partition (1), then

$$(\forall s \in \mathcal{H} \cap [\cap_{j\in\{1,\ldots,n\}}(\mathcal{L}_j \cap \mathcal{I}_j)])$$
$$(\exists l \in \Sigma_{IL}^*) sl \in \mathcal{H} \cap \left[\cap_{j\in\{1,2,\ldots,n\}} \left(\mathcal{L}_{m_j} \cap \mathcal{I}_{m_j}\right)\right].$$

Proof: See the proof in [6]. ■

We group the last two propositions together as **Proposition 6** builds upon **Proposition 5**. The first proposition asserts that any string accepted by the system can always be extended to a string marked by the high-level.

Proposition 5: If the nth degree $(n \geq 1)$ parallel interface system composed of DES $\mathbf{G}_H, \mathbf{G}_{L_1}, \ldots, \mathbf{G}_{L_n}, \mathbf{G}_{I_1}, \ldots, \mathbf{G}_{I_n}$, is level-wise nonblocking and interface consistent with respect to the alphabet partition (1), then

$$(\forall s \in \mathcal{H} \cap [\cap_{j\in\{1,\ldots,n\}}(\mathcal{L}_j \cap \mathcal{I}_j)])$$
$$(\exists h \in \Sigma_{IH}^*)\ sh \in \mathcal{H}_m \cap \left[\cap_{j\in\{1,\ldots,n\}}\mathcal{I}_{m_j}\right].$$

Proof: See the proof in [6]. ■

Our last proposition is analogous to [1, Prop. 5]. It asserts that we can use string h as a basis to construct string u by adding low-level events so that each low-level subsystem will accept the request and answer event contained in h. As these events are common to both levels, they must agree on their occurrence.

Proposition 6: If the nth degree $(n \geq 1)$ parallel interface system composed of DES $\mathbf{G}_H, \mathbf{G}_{L_1}, \ldots, \mathbf{G}_{L_n}, \mathbf{G}_{I_1}, \ldots, \mathbf{G}_{I_n}$, is level-wise nonblocking and interface consistent with respect to the alphabet partition (1), then

$$\left(\forall s \in \mathcal{H} \cap \left[\cap_{j\in\{1,\ldots,n\}} \left(\mathcal{L}_{m_j} \cap \mathcal{I}_{m_j}\right)\right]\right) (\forall h \in \Sigma_{IH}^*)$$
$$\left(sh \in \mathcal{H}_m \cap \left[\cap_{j\in\{1,\ldots,n\}}\mathcal{I}_{m_j}\right]\right)$$
$$\Rightarrow (\exists u \in \Sigma^*) \left(su \in \mathcal{H}_m \cap \left[\cap_{j\in\{1,\ldots,n\}} \left(\mathcal{L}_{m_j} \cap \mathcal{I}_{m_j}\right)\right] \wedge P_{IH}(u) = h\right).$$

Proof: See Appendix I. ■

We are now ready to present our nonblocking theorem for parallel interface systems. It states that, to verify if a parallel system is nonblocking, it is sufficient to check that each of its serial system extractions is serial level-wise nonblocking and serial interface consistent. As the level-wise nonblocking and interface consistency definitions can be evaluated by examining only one level (the high-level or one of the low-levels) of the system at a time, we now have a means of verifying nonblocking of the parallel system using local checks.

Theorem 1: If the nth degree $(n \geq 1)$ parallel interface system composed of DES $\mathbf{G}_H, \mathbf{G}_{L_1}, \ldots, \mathbf{G}_{L_n}, \mathbf{G}_{I_1}, \ldots, \mathbf{G}_{I_n}$, is level-wise nonblocking and interface consistent with respect to the alphabet partition (1), then $L(\mathbf{G}) = \overline{L_m(\mathbf{G})}$, where

$$\mathbf{G} = \mathbf{G}_H \| \mathbf{G}_{L_1} \| \ldots \| \mathbf{G}_{L_n} \| \mathbf{G}_{I_1} \| \ldots \| \mathbf{G}_{I_n}.$$

Proof: Assume system is level-wise nonblocking and interface consistent. (1)

As $\overline{L_m(\mathbf{G})} \subseteq L(\mathbf{G})$ is automatic, it suffices to show $L(\mathbf{G}) \subseteq \overline{L_m(\mathbf{G})}$

$$\text{Let } s \in L(\mathbf{G}) = \mathcal{H} \cap [\cap_{w\in\{1,\ldots,n\}}(\mathcal{L}_w \cap \mathcal{I}_w)]. \qquad (2)$$

We show this implies $s \in \overline{L_m(\mathbf{G})}$

It is sufficient to show: $(\exists u \in \Sigma^*) su \in L_m(\mathbf{G}) = \mathcal{H}_m \cap [\cap_{w\in\{1,\ldots,n\}}(\mathcal{L}_{m_w} \cap \mathcal{I}_{m_w})]$

We first apply **Proposition 4** and conclude

$$(\exists l \in \Sigma_{IL}^*) \left(sl \in \mathcal{H} \cap \left[\cap_{j\in\{1,\ldots,n\}} \left(\mathcal{L}_{m_j} \cap \mathcal{I}_{m_j}\right)\right]\right). \qquad (3)$$

We note that (3) implies that $sl \in \mathcal{H} \cap [\cap_{j\in\{1,\ldots,n\}}(\mathcal{L}_j \cap \mathcal{I}_j)])$. Now we can apply **Proposition 5**, taking sl to be string s in that proposition, and conclude

$$(\exists h \in \Sigma_{IH}^*) \left(slh \in \mathcal{H}_m \cap \left[\cap_{j\in\{1,\ldots,n\}}\mathcal{I}_{m_j}\right]\right). \qquad (4)$$

Combining with (3), we can now apply **Proposition 6**, taking sl to be string s in that proposition, and conclude

$$(\exists u' \in \Sigma^*)(slu' \in \mathcal{H}_m \cap \left[\cap_{j\in\{1,\ldots,n\}} \left(\mathcal{L}_{m_j} \cap \mathcal{I}_{m_j}\right)\right].$$

We take string $u = lu'$ and we have $su \in \mathcal{H}_m \cap [\cap_{j\in\{1,\ldots,n\}}(\mathcal{L}_{m_j} \cap \mathcal{I}_{m_j})] = L_m(\mathbf{G})$, as required. ■

E. Controllability Propositions and Theorem

We will now present **Propositions 7** and **8**, followed by our main result for this section, **Theorem 2**. The following propositions are analogous to the serial controllability propositions.

We start with **Proposition 7**. It asserts that if the system is level-wise controllable, then each pair of low-level supervisor and interface is controllable for the flat plant.

Proposition 7: If the nth degree $(n \geq 1)$ parallel interface system composed of plant components $\mathbf{G}_H^p, \mathbf{G}_{L_1}^p, \ldots, \mathbf{G}_{L_n}^p$, supervisors $\mathbf{S}_H, \mathbf{S}_{L_1}, \ldots, \mathbf{S}_{L_n}$, and interfaces $\mathbf{G}_{I_1}, \ldots, \mathbf{G}_{I_n}$, is level-wise controllable with respect to the alphabet partition (1), then

$$(\forall j \in \{1, \ldots, n\}) \left(\forall s \in L(\mathbf{Plant}) \cap \mathcal{S}_{L_j} \cap \mathcal{I}_j\right)$$
$$\mathrm{Elig}_{L(\mathbf{Plant})}(s) \cap \Sigma_u \subseteq \mathrm{Elig}_{\mathcal{S}_{L_j} \cap \mathcal{I}_j}(s).$$

Proof: See the proof in [6]. ■

The following proposition asserts that if the system is level-wise controllable, then $\mathbf{S}_H$ is controllable for the flat plant when the flat plant is already under the control of the interfaces.

Proposition 8: If the nth degree $(n \geq 1)$ parallel interface system composed of plant components $\mathbf{G}_H^p, \mathbf{G}_{L_1}^p, \ldots, \mathbf{G}_{L_n}^p$, supervisors $\mathbf{S}_H, \mathbf{S}_{L_1}, \ldots, \mathbf{S}_{L_n}$, and interfaces $\mathbf{G}_{I_1}, \ldots, \mathbf{G}_{I_n}$, is level-wise controllable with respect to the alphabet partition (1), then

$$(\forall s \in L(\mathbf{Plant}) \cap \mathcal{S}_H \cap [\cap_{k\in\{1,\ldots,n\}}\mathcal{I}_k])$$
$$\mathrm{Elig}_{L(\mathbf{Plant}) \cap [\cap_{k\in\{1,\ldots,n\}}\mathcal{I}_k]}(s) \cap \Sigma_u \subseteq \mathrm{Elig}_{\mathcal{S}_H}(s)$$

Proof: See Appendix I. ■

Next, we present a controllability theorem for parallel interface systems. It states that, to verify if a parallel system is controllable, it is sufficient to check that each of its serial system extractions is serial level-wise controllable. As in the case of nonblocking, we now have a means of verifying controllability of the flat supervisor using local checks.

Theorem 2: If the nth degree $(n \geq 1)$ parallel interface system composed of plant components $\mathbf{G}_H^p, \mathbf{G}_{L_1}^p, \ldots, \mathbf{G}_{L_n}^p$, supervisors $\mathbf{S}_H, \mathbf{S}_{L_1}, \ldots, \mathbf{S}_{L_n}$, and interfaces $\mathbf{G}_{I_1}, \ldots, \mathbf{G}_{I_n}$, is level-wise controllable with respect to the alphabet partition (1), then

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

Proof: Assume that the nth degree $(n \geq 10)$ parallel interface system is level-wise controllable. (1)
Let $s \in L(\mathbf{Plant}) \cap L(\mathbf{Sup})$, and $\sigma \in \mathrm{Elig}_{L(\mathbf{Plant})}(s) \cap \Sigma_u$ (2)
We show this implies $\sigma \in \mathrm{Elig}_{L(\mathbf{Sup})}(s)$. It is sufficient to show $s\sigma \in L(\mathbf{Sup}) = \mathcal{S}_H \cap [\cap_{k\in\{1,\ldots,n\}}(\mathcal{S}_{L_k} \cap \mathcal{I}_k)]$
Note $s, s\sigma \in \mathcal{H}^p \cap [\cap_{k\in\{1,\ldots,n\}}\mathbf{L}_k] = L(\mathbf{Plant})$ and $s \in \mathcal{S}_H \cap [\cap_{k\in\{1,\ldots,n\}}(\mathcal{S}_{L_k} \cap \mathcal{I}_k)]$ by (2). (3)
By (1), we can apply **Proposition 7** and conclude

$$s\sigma \in \cap_{k\in\{1,\ldots,n\}} (\mathcal{S}_{L_k} \cap \mathcal{I}_k). \tag{4}$$

All that remains is to show that $s\sigma \in \mathcal{S}_H$
From (3), we have $s \in L(\mathbf{Plant}) \cap \mathcal{S}_H \cap [\cap_{k\in\{1,\ldots,n\}}\mathcal{I}_k]$
From (3) and (4), we have $s\sigma \in L(\mathbf{Plant}) \cap [\cap_{k\in\{1,\ldots,n\}}\mathcal{I}_k]$
$\Rightarrow \sigma \in \mathrm{Elig}_{L(\mathbf{Plant})\cap[\cap_{k\in\{1,\ldots,n\}}\mathcal{I}_k]}(s) \cap \Sigma_u$.
We can now apply **Proposition 8**, and conclude $\sigma \in \mathrm{Elig}_{\mathcal{S}_H}(s)$ $\Rightarrow s\sigma \in \mathcal{S}_H$.
Combining with (4), we can now conclude $s\sigma \in \mathcal{S}_H \cap [\cap_{k\in\{1,\ldots,n\}}(\mathcal{S}_{L_k} \cap \mathcal{I}_k)]$, as required. ■

III. Algorithms and Complexity Analysis

To aid in investigating hierarchical interface-based supervisory control, we have developed software routines to verify that a serial system satisfies the conditions: serial level-wise nonblocking ([1, Def. 4]) and controllable ([1, Def. 5]), and serial interface consistent ([1, Def. 6]). Examining the conditions to be verified, one sees that most of them are either very straightforward (i.e., verifying two sets are disjoint), or can be verified using existing supervisory control algorithms after suitable definitions have been made. **Points 3 and 4** of the serial interface consistency definition can be verified using standard controllability algorithms such as TCTs **condat** function [7]. For example, in the case of **Point 3**, we simply define $\mathbf{ThePlant} = \mathbf{G}_I, \mathbf{TheSpec} = \mathbf{G}_H, \Sigma_u = \Sigma_A$, and $\Sigma_c = \Sigma - \Sigma_A$).

The exceptions are **Points 5 and 6** of the serial interface consistency definition. While these points effectively involve computing a transitive closure over Σ_{L_j} for each of the n low-levels, this only needs to be done for the relatively small systems $\mathbf{G}_{I_j} \| \mathbf{G}_{L_j}$. If we assume that the sizes of the state sets of $\mathbf{G}_{I_j}$ and $\mathbf{G}_{L_j}$ are bounded by N_I and N_L, respectively, then this operation is $\mathbf{O}(N_I^3 N_L^3)$ for each low-level component [8] and hence for a system with n low-level systems is $\mathbf{O}(nN_I^3 N_L^3)$.

The limiting factor in a monolithic verification of the system in which the n low-level subsystems are composed directly with the high-level system (i.e., no interface DES) is the size of the product state space of

$$G_{\text{mono}} := G_H \| G_{L_1} \| \ldots \| G_{L_{(j-1)}} \| G_{L_j} \| G_{L_{(j+1)}} \| \ldots \| G_{L_n}.$$

If we let N_H denote the size of the state space of $\mathbf{G}_H$, the product state–space is bounded by $N_H N_L^n$.

From Definition 2, we see that the high-level of the jth serial system extraction is given by

$$\mathbf{G}_H(j) := \mathbf{G}_H \| \mathbf{G}_{I_1} \| \ldots \| \mathbf{G}_{I_{(j-1)}} \| \mathbf{G}_{I_{(j+1)}} \| \ldots \| \mathbf{G}_{I_n}$$

which has a state–space size bound of $N_H N_I^{n-1}$. This system is then used to check serial level-wise nonblocking ([1, Def. 4–I]) and controllability ([1, Def. 5–III]) which require an additional language intersection with $\mathcal{I}_j$ effectively requiring computation of $\mathbf{G}_H(j) \| \mathbf{G}_{I_j}$, producing a state–space size bounded by $N_H N_I^n$. This indicates that in order to be effective computationally, we need $N_I \ll N_L$, or more generally, interfaces should be designed to be at least an order of magnitude smaller than their respective low-level systems to achieve significant benefit from interface based supervisory control.

The complexity of the supervisory control algorithms involved in controllability and nonblocking also depends upon the product of the sizes of the components' event sets [9]. Therefore, the restriction of the alphabets of interface automata to the interface events provides HISC with further potential computational savings by reducing the number of transitions involved in the computations.

Of course, there is a cost for this increase in computational efficiency. The tradeoff is a more restrictive architecture as the interface approach restricts knowledge about internal details of components, and only allows supervisors to disable local and interface events. As similar interface-based approaches are common in both hardware and software, we are confident that our method will be widely applicable.

All of the routines used on the example in Section IV were developed by Leduc during his collaboration with Siemens Corporate Research. In the algorithms currently implemented for serial interface consistency, the routines actually check that the interface is a star interface (for **Point 2**) and that the system is *serial interface strictly marked* (for **Point 6**).

For completeness, we now present the *serial interface strict marking* condition and a proposition showing that it implies the serial interface consistency definition.

Definition 6: The system composed of DES $\mathbf{G}_H, \mathbf{G}_L$, and $\mathbf{G}_I$, is *serial interface strictly marked* with respect to the alphabet partition $\Sigma := \Sigma_H \dot\cup \Sigma_L \dot\cup \Sigma_R \dot\cup \Sigma_A$, if $\mathcal{L} \cap \mathcal{I}_m = \mathcal{L}_m \cap \mathcal{I}_m$.

This statement is equivalent to **Property 6** of the serial interface consistency definition ([1, Def. 6]), with string l restricted to the empty string. It is useful as it implies **Property 6** of the serial interface consistency definition, but is less expensive to evaluate.

Proposition 9: If the system composed of DES $\mathbf{G}_H, \mathbf{G}_L$, and $\mathbf{G}_I$, satisfies **Properties 1–5** of the serial interface consistency definition and is serial interface strictly marked with respect to the alphabet partition $\Sigma := \Sigma_H \dot\cup \Sigma_L \dot\cup \Sigma_R \dot\cup \Sigma_A$, then the system is serial interface consistent.

Proof: See the proof in [6]. ■

Further details of the algorithms, including discussion of counter example generation when the conditions fail, can be found in Appendix II and [6].

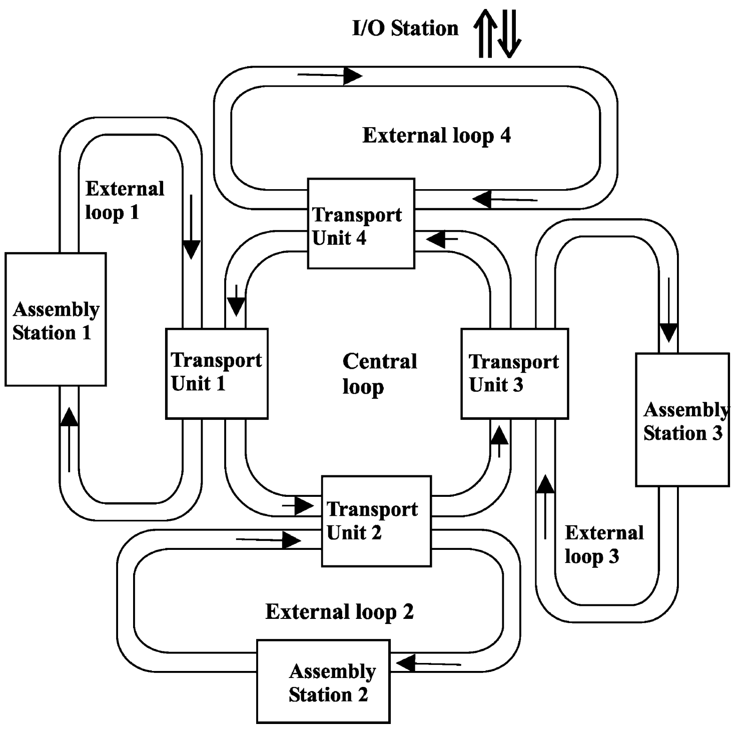

Fig. 4. AIP.

An important topic of current research is how to synthesize local controls that cause an interface inconsistent system to become consistent or produce a controllable nonblocking sublanguage of a specification when these conditions fail. This is currently the topic of [10].

IV. Application to the AIP

To demonstrate the utility of our method, we apply it to a large manufacturing system, the Atelier Inter-établissement de Productique (AIP) as described in [11] and [12]. Here we only present the final system design and the computation results on that model. Details of how the system design was obtained from the initial problem description based upon the HISC theory can be found in [13], [14].

The AIP, shown in Fig. 4, is an 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 types 1 or 2, chosen at random.

A. Assembly Stations

The assembly stations are shown in Fig. 5. Each consists of a robot to perform assembly tasks, an extractor to transfer the pallet from the conveyor loop to the robot, sensors to determine the location of the extractor, and a raising platform to present the pallet to the robot. The station also contains a pallet sensor to detect a pallet at the pallet gate, the pallet stop, and a sensor to detect when a pallet has left the station. Finally, the assembly station contains a read/write (R/W) device to read and write to the pallet's electronic label. The pallet label contains information

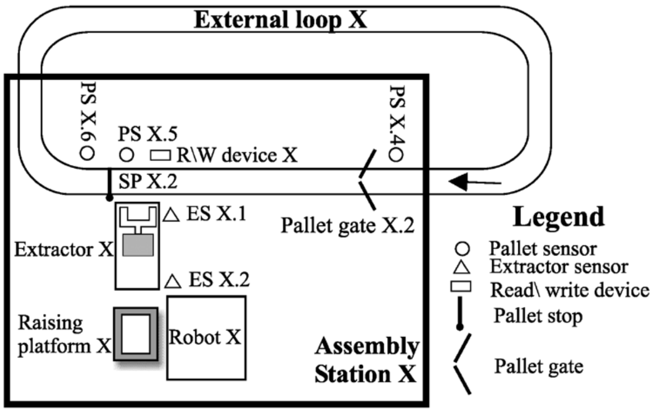

Fig. 5. Assembly station of external loop $X = 1, 2, 3$.

about the pallet type, error status, and assembly status (which tasks have been performed).

Whereas the assembly stations contain the same basic components, they differ with respect to functionality. Station 1 is capable of performing two separate tasks denoted task1A and task1B, while station 2 can perform task task2A and task2B. Station 3 can perform all four of these tasks as well as functioning as a repair station allowing an operator to repair a damaged pallet. The assembly stations also differ with respect to reliability. Stations 1 and 2 can break down and must be repaired, while station 3 is of higher quality and is assumed never to break down. Station 3 is used as a substitute for the other stations when they are down.

B. Transport Units

The structure of the four identical transport units is shown in Fig. 6. The transport units are used to transfer pallets between

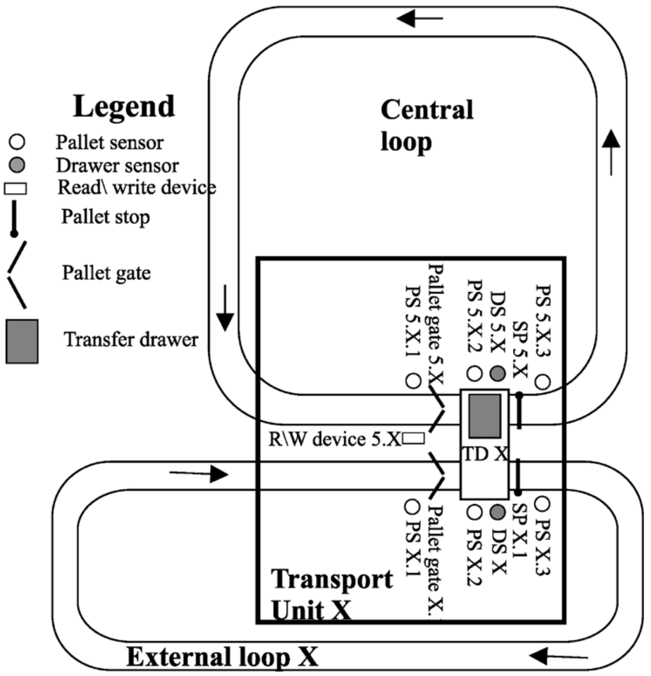

Fig. 6. Transport unit for external loop $X = 1, 2, 3, 4$.

the central loop, and the external loops. Each one consists of a transport drawer which physically conveys the pallet between the two loops, plus sensors to determine the drawer's location. At each loop, the unit contains a pallet gate and a pallet stop, to control access to the unit from the given loop. The unit also contains multiple pallet sensors to detect when a pallet is at a gate, drawer, or has left the unit. Also, each unit contains a R/W device located before the central loop gate.

C. Control Specifications

For this example, we adopt the control specifications and assumptions used in [11] and [12] and restated below. To this we add **Specification 7** to make the assembly stations more interesting and complicated.

Assumptions: We assume that 1) the system is initially empty, 2) two types of pallets are randomly introduced to the system, subjected to assembly operations, and then leave, and 3) pallets enter the system following the order: type 1, type 2, type 1, ...

Specifications:

1) **Routing:** Pallets follow a certain route based on their type. A type 1 pallet must go first to AS1, then AS2 before leaving the system. Type 2 pallets go first to AS2, then AS1 before leaving the system. A pallet is not allowed to leave the system until all four assembly tasks have been successfully performed on it.
2) **Maximum capacity of external loops 1 and 2:** The maximum allowed number of pallets in either loop at a given time is one.
3) **Ordering of pallet exit from system:** The pallets must exit the system in the following order: type 1, type 2, type 1, . . .
4) **Assembly errors:** When a robot makes an assembly error, the pallet is marked damaged and routed to AS3 for maintenance. After maintenance, the pallet is returned to the original assembly station to undergo the assembly operation again.
5) **Assembly station breakdown:** The robots of external loops 1 and 2 are susceptible to breakdowns. When a station is down, pallets are routed to assembly station 3 which is capable of performing all tasks of the other two stations. When the failed station is repaired, all pallets not already in external loop 3 are rerouted to the original station.
6) **Maximum capacity of assembly stations:** To avoid collisions, only one pallet is allowed in a given station at a time.
7) **Assembly task ordering:** Assembly tasks are performed in a different order for pallets of different types. For pallets of type 1, task1A is performed before task1B, and task2A is performed before task2B. For pallets of type 2, task1B is performed before task1A, and task2B is performed before task2A.

D. System Structure

To cast the AIP into a parallel interface system, we break the system into a high-level and seven low-levels, corresponding to the three assembly stations and four transport units, as shown in Fig. 7. We select a few representative subsystems to describe in the following sections. As this example contains 181 DESs, we are not able to describe the design in complete detail, but refer the reader to [6] for a complete description.

The models and supervisors developed for this example are based on the automata presented in [11] and [12]. We have altered them to fit our setting, and extended them to fill in the missing details of several events that were defined as "macro events" in the cited references.

In the diagrams to follow, uncontrollable events are shown in italics; all other events are controllable. Also, initial states can be recognized by a thick outline, and marker states are filled.

1) High-level: The high-level subsystem, which contains 15 DES, keeps track of the breakdown status of assembly stations 1 and 2, and enforces the maximum capacity of external loops 1 and 2. This subsystem controls the operation of all transport units and assembly stations, while tracking the pallets' progress around the manufacturing system.

As an example of the high-level subsystem's behavior, we discuss supervisor *ManageTU1*, shown in Fig. 8. This supervisor controls the transfer of pallets between the central loop and external loop 1. It permits pallets on the central loop to pass through transport unit 1 (to be liberated) without being transferred to the external loop. Pallets are liberated if the attached external loop is at maximum capacity, assembly station 1 is down, or TU1 determines that the pallet is not to be transferred.

2) Low-levels AS1 and AS2: We now describe the low-level subsystems that represent assembly stations 1 and 2. As they are identical, we will describe them collectively as low-level subsystem k, where $k = \mathrm{AS1}, \mathrm{AS2}$.

Subsystem k contains 17 DESs and provides the functionality specified in its interface, shown in Fig. 9. An assembly station accepts the pallet at its gate, and presents it to the robot

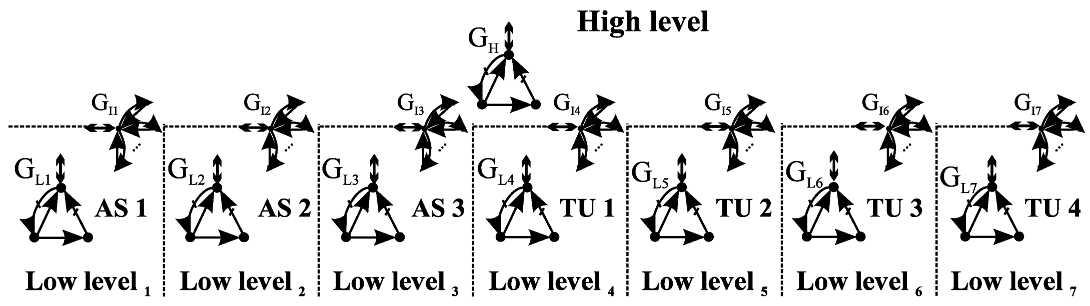

Fig. 7. Structure of parallel system.

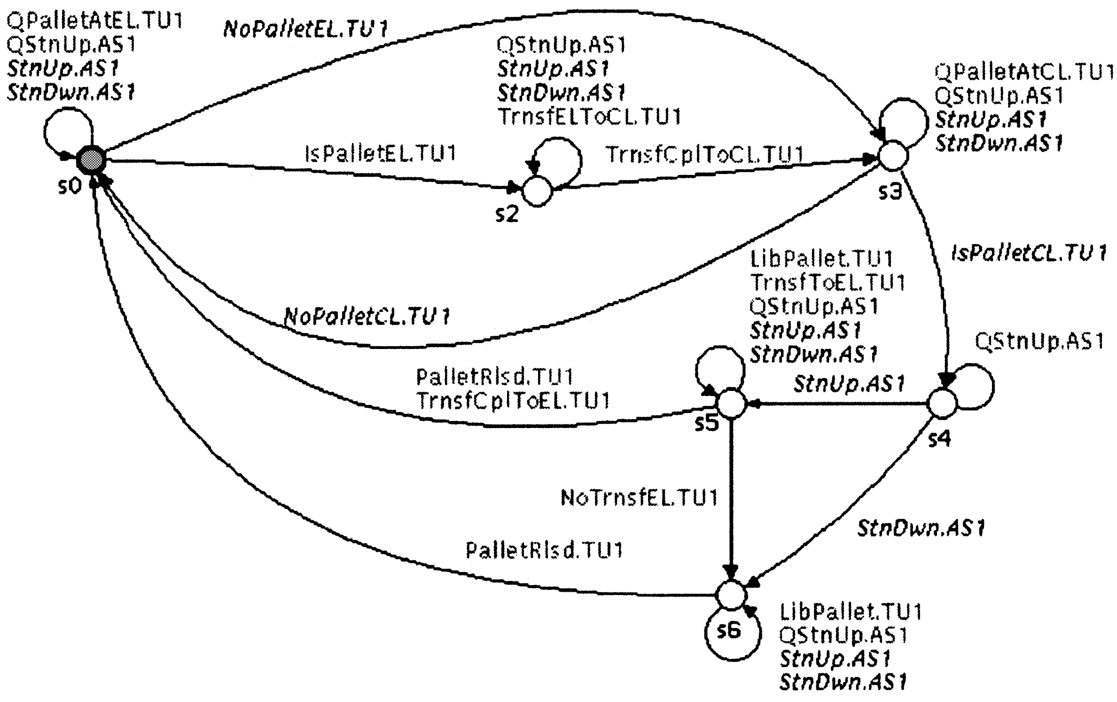

Fig. 8. Supervisor ManageTU1.

for assembly. It then releases the pallet, and reports on the success of the assembly operation. If the robot breaks down, this is reported through the interface, and the pallet is released. Subsystem k then waits for a repair command to return the robot to operation.

Supervisor *HndlPallet.AS1*, shown in Fig. 10, provides an example of low-level subsystem AS1's behavior. *HndlPallet.AS1* handles the task of processing a pallet once it reaches the extractor. It reads the pallet's label, presents the pallet to the robot, and has the robot perform the appropriate tasks on the pallet. The supervisor then allows the pallet to leave the assembly station and reports on the success of the processing operation by updating the pallet's label, and notifies the high-level subsystem through the interface.

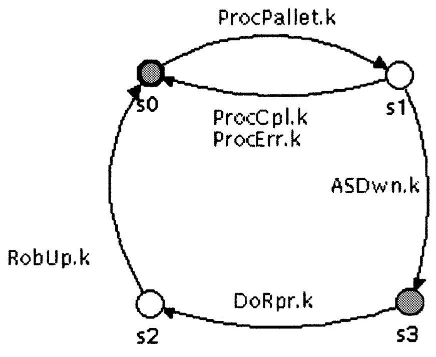

Fig. 9. Interface to low-level $k = \mathrm{AS1}, \mathrm{AS2}$.

3) low-levels TU1 and TU2: We now describe the low-level subsystems that represent transport units 1 and 2. As they are identical, we will describe them collectively as low-level subsystem r, where $r = \mathrm{TU1}, \mathrm{TU2}$.

Subsystem r contains 25 DES and provides the functionality specified in its interface, shown in Fig. 12. The transport units are used to transfer pallets between the central loop, and the external loops (i.e., TU1 transfers pallets between CL and EL1). Subsystem **r** has two entry points for pallets, the central loop gate, and the external loop gate. If a pallet is at the EL gate, subsystem **r** transfers the pallet to the central loop. If a pallet is at the CL gate, subsystem **r** can be requested to liberate the pallet (allow it to pass through and continue on CL), or to transfer the pallet to the EL. When requested to transfer a pallet to the EL, subsystem **r** will only transfer the pallet if the pallet is undamaged and if the next assembly

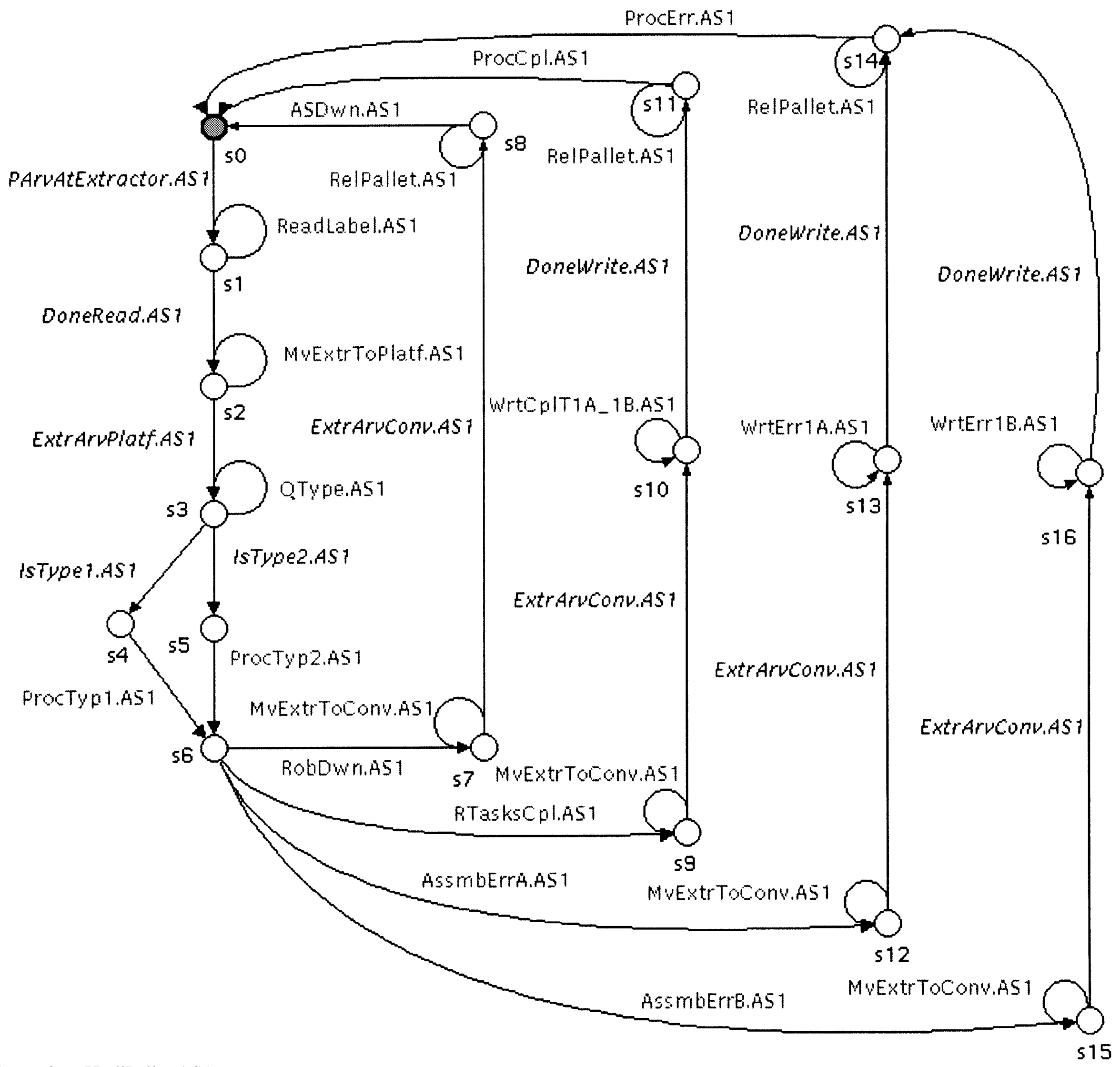

Fig. 10. Supervisor HndlPallet.AS1.

task required by the pallet is performed by the external loop's assembly station.

Supervisor *HndlTrnsfToEL.r*, shown in Fig. 11, provides an example of low-level subsystem r's behavior. *HndlTrnsfToEL.r* handles transporting pallets from the central loop to the external loop. It only transfers pallets if they are undamaged, or if the next assembly task required by the pallet is performed by the external loop's assembly station.

E. Discussion of Results

Applying our research tool to the seven serial extraction systems, we find that they are all serial level-wise nonblocking, serial level-wise controllable, and serial interface consistent. Thus, we conclude that the system is level-wise nonblocking, level-wise controllable, and interface consistent, and hence, by **Theorems 1** and **2**, the flat system is nonblocking and the flat system's supervisor is controllable for the flat plant.

This example contains 181 DES in total, with an estimated closed-loop state space of 2.9×10^{21}. This estimate was calculated by determining the closed-loop state space of the high-level, and each low-level and then multiplying these together to create a worst case state estimate. The computation ran for 25 min, using 760 MB of memory. The machine used was a 750 MHz Athlon system, with 512 MB of RAM, 2 GB of swap, running Redhat Linux 6.2. A standard nonblocking verification was also attempted on the monolithic system, but it quickly failed due to lack of memory.

Table I shows the size of the various subsystem automata used in the AIP calculations. First, the size of the state space of each component without being synchronized with their respective interfaces (Standalone) is given and then state space size when synchronized with their interface DES ($\mathbf{G}_H$ is synchronized with all seven interfaces). The last two columns give the size of the interfaces for for the high-level and each low-level. From Section III, we saw that the limiting factor for a monolithic algorithm would be $N_H N_L^n$ and similarly $N_H N_I^n$ for the HISC method. N_H denotes the size of the state–space of $\mathbf{G}_H$, while N_I and N_L are the bounds for $\mathbf{G}_{I_j}$ and $\mathbf{G}_{L_j} (j = 1, \ldots, n)$, respectively. If we substitute actual data from Table I, we get $N_L^n = (120)^2(203)(98)^2(204)(152) = 8.71 \times 10^{14}$ and $N_I^n = (4)^2(2)(4)^4 = 8\,192$. This is a potential savings of eleven orders of magnitude. In fact, instead of multiplying $N_H = 1\,480\,864$

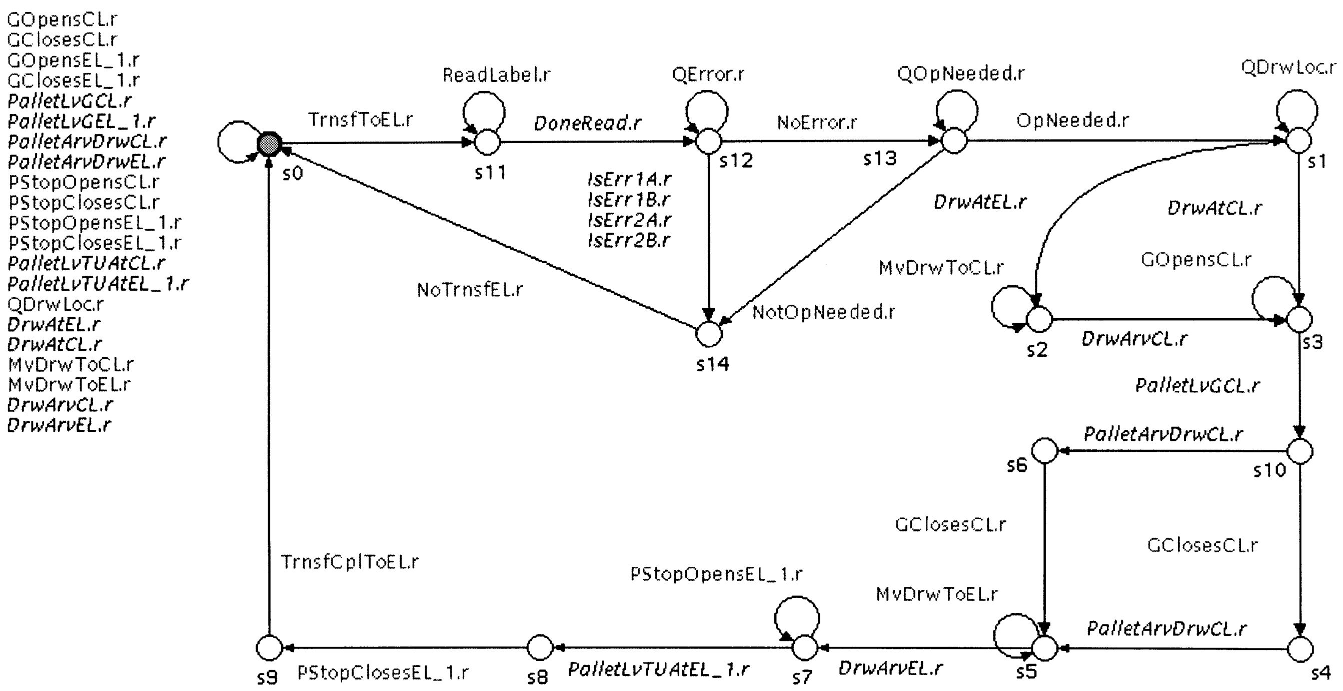

Fig. 11. Supervisor HndlTrnsfToEL.r.

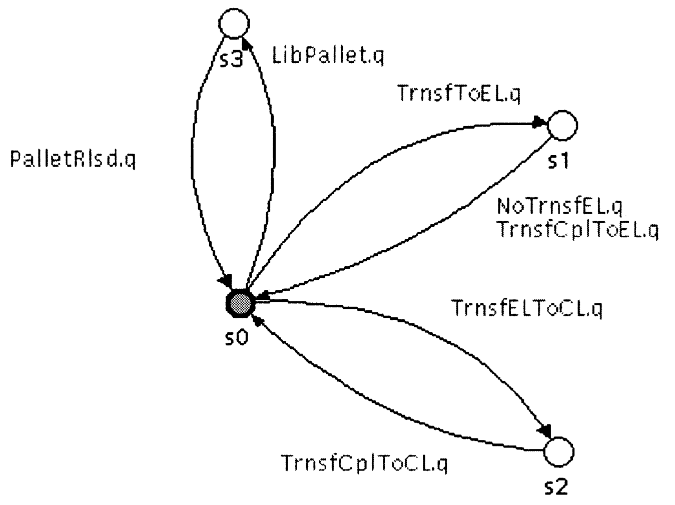

Fig. 12. Interface to low-level $q = \mathrm{TU1}, \mathrm{TU2}, \mathrm{TU4}$.

by a factor of $N_I^n = 8\,192$, adding the interfaces only doubles the state space of the high-level. For low-level AS1, synchronizing with its interface actually causes the state–space to decrease from 1795 to 120 states, an order of magnitude reduction.

We note that the prototype tool used for these calculations did not make use of IDDs and symbolic techniques such as those used in [15] and [16]. We conjecture that using HISC methods with tools utilizing symbolic techniques should allow the method to scale up to considerably larger systems as has been the case with the application of symbolic techniques to monolithic supervisory control calculations.

V. Conclusion

Hierarchical interface-based supervisory control offers an effective means to model systems with a natural master-slave structure. It provides an intuitive way to model and design the system. Using multiple $(n \geq 1)$ low-level subsystems allows the subsystems to be independently modeled and verified, while still allowing a high degree of concurrent operation. As each low-level requirement can be verified using only one subsystem and its interface, the entire plant model never needs to be constructed or traversed (in computer memory), offering potentially significant savings in computation. However, the limiting factor is the size of the high-level as the high-level requirements depend upon the high-level subsystem and the interfaces to all of the low-level components. When the interfaces are designed to have smaller state–spaces than the low-level components, the verification of the high-level requirements will require considerably less space, though in the worst case the space required for the verification still grows exponentially

TABLE I
Size of AIP Subsystem Automata Models

	States		
Subsystem	**Standalone**	**|| with $\mathbf{G}_{I_j}$**	**Size of $\mathbf{G}_{I_j}$**
$\mathbf{G}_H$	1,480,864	3,306,240	8,192
AS1	1,795	120	4
AS2	1,795	120	4
AS3	1,199	203	2
TU1	98	98	4
TU2	98	98	4
TU3	204	204	4
TU4	152	152	4

with the number of components. To address this problem, future research will focus on extending the method to a multilevel hierarchy.

Finally, we discussed a large example based on the automated manufacturing system of the AIP. As the example contains 181 DES with an estimated worst-case closed-loop state–space of size 2.9×10^{21}, it demonstrates that the HISC method can be applied to interesting systems of realistic complexity even though symbolic techniques have not yet been incorporated into our approach.

Appendix I
Proofs of Selected Propositions

Proposition 6:

Proof: Assume system is level-wise nonblocking and interface consistent. (1)

Let $s \in \mathcal{H} \cap [\cap_{j\in\{1,\ldots,n\}}(\mathcal{L}_{m_j} \cap \mathcal{I}_{m_j})], h \in \Sigma_{IH}^*$, and $sh \in \mathcal{H}_m \cap [\cap_{j\in\{1,\ldots,n\}}\mathcal{I}_{m_j}]$ (2)

We will now show this implies $(\exists u \in \Sigma^*)(su \in \mathcal{H}_m \cap [\cap_{j\in\{1,\ldots,n\}}(\mathcal{L}_{m_j} \cap \mathcal{I}_{m_j})]) \wedge (P_{IH}(u) = h)$

Iterative step:

• For each $i \in \{1,\ldots,n\}$, construct u_i, with properties $su_i \in \mathcal{H}_m \cap [\cap_{j\in\{1,\ldots,n\}}\mathcal{I}_{m_j}] \cap \mathcal{L}_{m_i}$ and $P_{IH}(u_i) = h$, as follows.

• From (2), we have $sh \in \mathcal{H}_m \cap [\cap_{j\in\{1,\ldots,n\}}\mathcal{I}_{m_j}]$.

• As $P_i(h) = h$ since $h \in \Sigma_{IH}^* \subseteq \Sigma(i)$ (by **Proposition 1**), we can conclude

$$P_i(sh) = P_i(s)h \in P_i(\mathcal{H}_m) \cap \left[\cap_{j\in\{1,\ldots,i-1,i+1,\ldots,n\}}P_i\left(\mathcal{I}_{m_j}\right)\right] \cap P_i\left(\mathcal{I}_{m_i}\right).$$

• $\Rightarrow P_i(s)h \in \mathcal{H}_m(i) \cap \mathcal{I}_m(i)$ (by **Proposition 1**).(3)

• From (2), we have $s \in \mathcal{H} \cap [\cap_{j\in\{1,\ldots,n\}}(\mathcal{L}_j \cap \mathcal{I}_j)]$. (4)

• From **Proposition 1**, we thus have

$$P_i(s) \in P_i(\mathcal{H}) \cap [\cap_{j\in\{1,\ldots,i-1,i+1,\ldots,n\}}P_i(\mathcal{I}_j)] \cap P_i(\mathcal{L}_i) \cap P_i(\mathcal{I}_i) = \mathcal{H}(i) \cap \mathcal{L}(i) \cap \mathcal{I}(i). \quad (5)$$

• From (2), we have $s \in \mathcal{I}_{m_i}$. This implies $P_i(s) \in P_i(\mathcal{I}_{m_i}) = \mathcal{I}_m(i)$

• Combining with (2), (3), and (5), we now apply [1, Prop. 5] by taking $P_i(s)$ to be string s in that proposition and conclude

$$(\exists u_i \in \Sigma(i)^*)P_i(s)u_i \in (\mathcal{H}_m(i) \cap \mathcal{L}_m(i) \cap \mathcal{I}_m(i)) \wedge (P_{IH}(i)(u_i) = h). \quad (6)$$

• We next note that as $\Sigma(i) \subseteq \Sigma$ and $\Sigma_{IH} = \Sigma_{IH}(i)$ (by **Proposition 1**), we can conclude

$$P_{IH}(u_i) = P_{IH}(i)(u_i) = h. \quad (7)$$

• We now note that $u_i \in \Sigma(i)^*$ implies that $P_i(u_i) = u_i$. Combining with (6) and substituting for $\mathcal{H}_m(i), \mathcal{L}_m(i)$ (using **Proposition 1**), and $\mathcal{I}_m(i)$, we have

$$P_i(su_i) \in P_i \cdot P_{IH}^{-1}(L_m(\mathbf{G}_H)) \cap \left[\cap_{j\in\{1,\ldots,n\}}P_i \cdot P_{I_j}^{-1}\left(L_m\left(\mathbf{G}_{I_j}\right)\right)\right] \cap P_i \cdot P_{IL_i}^{-1}\left(L_m\left(\mathbf{G}_{L_i}\right)\right). \quad (8)$$

• From **Proposition 1**, we have $\Sigma_{IH} \subseteq \Sigma(i), \Sigma_{I_j} \subseteq \Sigma(i)$ (for $j \in \{1,\ldots,n\}$), and $\Sigma_{IL_i} \subseteq \Sigma(i)$. We can now apply **Proposition 3** three times by taking $[\Sigma_a = \Sigma(i), \Sigma_b = \Sigma_{IH}$, and $L_b = L_m(\mathbf{G}_H)]$, $[\Sigma_a = \Sigma(i), \Sigma_b = \Sigma_{I_j}$, and $L_b = L_m(\mathbf{G}_{I_j})]$, and then $[\Sigma_a = \Sigma(i), \Sigma_b = \Sigma_{IL_i}$, and $L_b = L_m(\mathbf{G}_{L_i})]$, and conclude

$$su_i \in P_{IH}^{-1}(L_m(\mathbf{G}_H)) \cap \left[\cap_{j\in\{1,\ldots,n\}}P_{I_j}^{-1}\left(L_m\left(\mathbf{G}_{I_j}\right)\right)\right] \cap P_{IL_i}^{-1}\left(L_m\left(\mathbf{G}_{L_i}\right)\right).$$

• $\Rightarrow su_i \in \mathcal{H}_m \cap [\cap_{j\in\{1,\ldots,n\}}\mathcal{I}_{m_j}] \cap \mathcal{L}_{m_i}$ and $P_{IH}(u_i) = h$ (by (7), as required.

• **Iterative step** complete.

Now that we have completed the **iterative step**, we have shown the following:

$$(\forall i \in \{1,\ldots,n\})(\exists u_i \in \Sigma(i)^*) (su_i \in \mathcal{H}_m \cap [\cap_{j\in\{1,\ldots,n\}}\mathcal{I}_{m_j}] \cap \mathcal{L}_{m_i}) \wedge (P_{IH}(u_i) = h). \quad (9)$$

We take u to be any string in set $\cap_{i\in\{1,\ldots,n\}}P_i^{-1}(u_i)$ (10)

We know that the set is nonempty for the following reasons.

- For each $i \in \{1,\ldots,n\}$, we have $u_i \in \Sigma(i)^*$ where: $\Sigma(i) := \Sigma_{IH}\dot{\cup}\Sigma_{L_i} = \Sigma - (\dot{\cup}_{j\in\{1,\ldots,i-1,i+1,\ldots,n\}}\Sigma_{L_j})$.
- The only events strings u_i have in common are $\sigma \in \Sigma_{IH}$.
- All strings u_i agree on common events as $P_{IH}(u_i) = h$.

From (10), we have $(\forall i \in \{1,\ldots,n\})P_i(u) = u_i$. As $\Sigma_{IH} \subseteq \Sigma(i)$ (by **Proposition 1**) and $h \in \Sigma_{IH}^*$ (by (2)), we can conclude:

$$P_{IH}(u) = P_{IH}(u_i) = h = P_{IH}(h). \quad (11)$$

From (2), we have: $sh \in \mathcal{H}_m \cap [\cap_{j\in\{1,\ldots,n\}}\mathcal{I}_{m_j}]$

$\Rightarrow P_{IH}(sh) = P_{IH}(su) \in L_m(\mathbf{G}_H)$ by (11).

$\Rightarrow su \in \mathcal{H}_m$.

Similarly, as $\Sigma_{I_j} \subseteq \Sigma_{IH}$ for $j \in \{1,\ldots,n\}$, we can conclude (by (11)) that $P_{I_j}(u) = P_{I_j}(h)$ and thus: $su \in \mathcal{I}_{m_j}$

$\Rightarrow su \in \mathcal{H}_m \cap [\cap_{j\in\{1,\ldots,n\}}\mathcal{I}_{m_j}]$.

From (9), we have $su_j \in \mathcal{L}_{m_j}$ for $j \in \{1,\ldots,n\}$. From (10), we have $P_j(u) = u_j = P_j(u_j)$ as $u_j \in \Sigma(j)^*$. As $\Sigma_{IL_j} \subseteq \Sigma(j)$ (by **Proposition 1**), we can conclude $P_{IL_j}(u) = P_{IL_j}(u_j)$ and thus: $su \in \mathcal{L}_{m_j}$

$\Rightarrow su \in \mathcal{H}_m \cap [\cap_{j\in\{1,\ldots,n\}}(\mathcal{L}_{m_j} \cap \mathcal{I}_{m_j})]$

From (11), we have $P_{IH}(u) = h$, as required. ■

Proposition 8:

Proof: Assume that the nth degree $(n \geq 1)$ parallel interface system is level-wise controllable. (1)

Let $s \in L(\mathbf{Plant}) \cap \mathcal{S}_H \cap [\cap_{k\in\{1,\ldots,n\}}\mathcal{I}_k]$, and $\sigma \in \mathrm{Elig}_{L(\mathbf{Plant})\cap[\cap_{k\in\{1,\ldots,n\}}\mathcal{I}_k]}(s) \cap \Sigma_u$ (2)

We will now show that this implies $\sigma \in \mathrm{Elig}_{\mathcal{S}_H}(s)$

It's sufficient to show that $s\sigma \in \mathcal{S}_H$.

We first note that

$$s, s\sigma \in \mathcal{H}^p \cap [\cap_{k\in\{1,\ldots,n\}}(\mathbf{L}_k \cap \mathcal{I}_k)] = L(\mathbf{Plant}) \cap [\cap_{k\in\{1,\ldots,n\}}\mathcal{I}_k] \quad (3)$$

by (2).

By examining the definition of $\Sigma(j)$ for some $j \in \{1,\ldots,n\}$

(see definition of jth serial system extraction: general form in Section 2), we see that $\Sigma = \cup_{k\in\{1,\ldots,n\}}\Sigma(k)$
$\Rightarrow\ (\exists j \in \{1,\ldots,n\})\ \sigma \in \Sigma(j)$. (4)
We use this j and note that by (1), we can conclude that $\mathrm{system}(j)$, the jth serial system extraction of our parallel system, is serial level-wise controllable. (5)
From (2) and (3), we have $s \in \mathcal{H}^p \cap \mathcal{S}_H \cap [\cap_{k\in\{1,\ldots,n\}}\mathcal{I}_k]$
$\Rightarrow P_j(s) \in P_j(\mathcal{H}^p) \cap P_j(\mathcal{S}_H) \cap [\cap_{k\in\{1,\ldots,n\}}P_j(\mathcal{I}_k)]$.
See Section II-C for the definition of the natural projection, P_j.
$\Rightarrow P_j(s) \in \mathcal{H}^p(j) \cap \mathcal{I}(j) \cap \mathcal{S}_H(j)$, by **Proposition 2**.
Similarly, from (3) we can conclude $P_j(s\sigma) \in \mathcal{H}^p(j) \cap \mathcal{I}(j)$
We next note that $\sigma \in \Sigma(j)$ [from (4)] implies that $P_j(s\sigma) = P_j(s)\sigma$.
$\Rightarrow \sigma \in \mathrm{Elig}_{\mathcal{H}^p(j)\cap\mathcal{I}(j)}(P_j(s)) \cap \Sigma_u$
We can now conclude by **point III** of the serial level-wise controllable definition that:
$\sigma \in \mathrm{Elig}_{\mathcal{S}_H(j)}(P_j(s))$ and thus $P_j(s)\sigma = P_j(s\sigma) \in \mathcal{S}_H(j)$
$\Rightarrow P_j(s\sigma) \in P_j(\mathcal{S}_H)$, by **Proposition 2**.
$\Rightarrow P_j(s\sigma) \in P_j P_{IH}^{-1} L(\mathbf{S}_H)$
As $\Sigma_{IH} = \Sigma_{IH}(j)$ by **Proposition 2**, we have $\Sigma_{IH} \subseteq \Sigma(j)$. We can thus apply **Proposition 3** by taking $\Sigma_a = \Sigma(j), \Sigma_b = \Sigma_{IH}$, and $L_b = L(\mathbf{S}_H)$ and thus conclude:
$s\sigma \in P_{IH}^{-1} L(\mathbf{S}_H) = \mathcal{S}_H$, as required. ■

APPENDIX II
ALGORITHM FOR SERIAL INTERFACE CONSISTENCY

In this section, we present an algorithm for evaluating **points 5) and 6)** of the serial interface consistent definition [1, Def. 6] and discuss counter example generation when the conditions fail. We assume that all DES are reachable, deterministic, and have finite state and event sets, and that we are given $\mathbf{G}_L := (Y_L, \Sigma_{G_L}, \delta_L, y_{L_o}, Y_{L_m})$, $\mathbf{G}_I := (X, \Sigma_{G_I}, \xi, x_o, X_m)$, and the map $\mathbf{Answer} : X \rightarrow \mathrm{Pwr}(\Sigma_A)$ defined as:

$$(\forall x \in X)\ \mathbf{Answer}(x) := \{\alpha \in \Sigma_A | \xi(x,\alpha)!\}$$

For a given DES $\mathbf{G} = (Y, \Sigma, \delta, y_o, Y_m)$, we use the standard definition for its transition function $\delta : Y \mathrm{x} \Sigma^* \rightarrow Y$, and define the inverse transition function $\delta^{-1} : Y \mathrm{x} \Sigma^* \rightarrow \mathbf{Pwr}(Y)$, for $s \in \Sigma^*$ and $y \in Y$ as follows:

$$\delta^{-1}(y, s) := \{y' \in Y | \delta(y', s) = y\}.$$

To avoid repeating existing algorithms, we will assume we already have the DES $\mathbf{G}_{IL} := \mathbf{G}_L || \mathbf{G}_I = (Y_{IL} \subseteq Y_L \times X, \Sigma_{G_{IL}}, \delta_{IL}, y_{IL_o}, Y_{IL_m})$ and the inverse transition function δ_{IL}^{-1}, which can be created using TCT [7]. We will also use the variables Y_{ck_mk} (states of $\mathbf{G}_{IL}$ that are in $Y_{IL} \cap [(Y_L - Y_{L_m}) \times X_m]$), Y_{fnd}, Y_{pend}, and $\Sigma_{\neg fnd}$. The complete algorithm appears in Fig. 13.

A. Counter Example Generation

Considering the algorithm in Fig. 13, we see that if **point 5** fails, the variables y_l and x define the state tuple (y_l, x) in $\mathbf{G}_L || \mathbf{G}_I$, reached by the request event contained in variable ρ. This is the state from which we failed to find a low-level path to the answer events remaining in $\Sigma_{\neg Fnd}$, whose transitions are possible at state x in the interface.

We also see that when **Point 6** fails, the variable Y_{ck_mk} contains the set of reachable states of $\mathbf{G}_L || \mathbf{G}_I$ that are marked by $\mathbf{G}_I$, but do not have continuations at the low-level to a marked state.

As mentioned in Section II, all of the remaining conditions that need to be verified to prove global nonblocking and controllability are based upon standard supervisory control algorithms, hence counter examples can be easily generated using standard techniques.

```
$Y_{ck\_mk} := \emptyset$
for $y \in (Y_L \times X_m) \cap Y_{IL}$ // Request events possible
  if $(y \notin Y_{IL_m})$
    $Y_{ck\_mk} := Y_{ck\_mk} \cup \{y\}$; // Flag to check Point 6
  for $\rho$ in $\Sigma_R$ // Checking Point 5
    if $(\delta_{IL}(y,\rho)!)$
      $(y_L, x) := \delta_{IL}(y,\rho)$; // State $(y_L, x)$ start of search
      $Y_{fnd} := Y_{pend} := \{(y_L, x)\}$;
      $\Sigma_{\neg fnd} :=$ Answer$(x)$; // Set of answer events to reach
      while $(Y_{pend} \neq \emptyset$ and $\Sigma_{\neg fnd} \neq \emptyset)$
        $y' :=$ pop$(Y_{pend})$;
        for $\sigma$ in $\Sigma_{IL}$
          if $(\delta_{IL}(y',\sigma)!)$ // $\sigma$ transition found
            $y'' := \delta_{IL}(y',\sigma)$;
            $\Sigma_{\neg fnd} := \Sigma_{\neg fnd} - \{\sigma\}$; // Remove $\sigma$ if present
            if $((\sigma \in \Sigma_L) \wedge (y'' \notin Y_{fnd}))$ // Only follow $\Sigma_L$
                                                   // transitions
              $Y_{fnd} := Y_{fnd} \cup \{y''\}$;
              push$(y'', Y_{pend})$;
        end for
      end while
      if $(\Sigma_{\neg fnd} \neq \emptyset)$ then // couldn't reach some events
        return "point 5 fails";
  end for
end for // If exit this loop, Point 5 passed
if $(Y_{ck\_mk} = \emptyset)$ // No State failed Point 6
  return "points 5 and 6 pass";

$Y_{fnd} := Y_{pend} := Y_{IL_m}$; // Check flagged states for Point 6
while $(Y_{pend} \neq \emptyset)$
  $y :=$ pop$(Y_{pend})$;
  for $\sigma$ in $\Sigma_L$ // Only follow $\Sigma_L$ transitions
    if $(Y := \delta_{IL}^{-1}(y,\sigma) \neq \emptyset)$ // States that reach $y$ via $\sigma$
      for $y'$ in $Y$
        if $(y' \notin Y_{fnd})$
          $Y_{fnd} := Y_{fnd} \cup \{y'\}$;
          push$(y', Y_{pend})$;
          $Y_{ck\_mk} := Y_{ck\_mk} - \{y'\}$; // Remove $y'$ if present
          if $(Y_{ck\_mk} = \emptyset)$ // All states now pass Point 6
            return "points 5 and 6 pass" ;
      end for
  end for
end while // If exit while, then $(Y_{ck\_mk} \neq \emptyset)$
return "point 6 fails" ;
```

Fig. 13. Algorithm to compute **points 5 and 6** of serial interface consistency.

REFERENCES

[1] R. Leduc, B. Brandin, M. Lawford, and W. M. Wonham, "Hierarchical interface-based supervisory control, Part I: Serial case," *IEEE Trans. Autom. Control*, vol. 50, no. 9, pp. 1322–1335, Sep. 2005.

[2] R. Leduc, B. Brandin, W. M. Wonham, and M. Lawford, "Hierarchical interface-based supervisory control: Serial case," in *Proc. 40th Conf. Decision Control*, Orlando, FL, Dec. 2001, pp. 4116–4121.

[3] R. Leduc, M. Lawford, and W. M. Wonham, “Hierarchical interface-based supervisory control: AIP example,” in *Proc. 39th Allerton Conf. Communication, Control, and Computing*, Oct. 2001, pp. 396–405.
[4] R. Leduc, W. M. Wonham, and M. Lawford, “Hierarchical interface-based supervisory control: Parallel case,” in *Proc. 39th Allerton Conf. Communication, Control, and Computing*, Oct. 2001, pp. 386–395.
[5] E. W. Endsley, M. R. Lucas, and D. M. Tilbury. (2000, Oct.) Modular design and verification of logic control for reconfigurable machining systems. [Online]http://www-personal.engin.umich.edu/~tilbury/papers.html
[6] R. Leduc, “Hierarchical interface-based supervisory control,” Ph.D. dissertation, Dept. Elect. Comput. Eng., Univ. Toronto, Toronto, ON, Canada, 2002.
[7] W. M. Wonham. (2004, Jul.) Supervisory control of discrete-event systems. Dept. Elect. Comput. Eng., Univ. Toronto, Toronto, ON, Canada. [Online]. Available: http://www.control.toronto.edu/DES/
[8] S. Warshal, “A theorem on boolean matrices,” *J. ACM*, vol. 9, no. 1, pp. 11–12, Jan. 1962.
[9] K. Rudie, “Software for the control of discrete-event systems: A complexity study,” M.A.Sc. thesis, Dept. Elect. Comput. Eng., Univ. Toronto, Toronto, ON, Canada, 1988.
[10] P. Dai, “Synthesis method for hierarchical interface-based supervisory control,” M.A.Sc. thesis, Dept. Comput. Software, McMaster Univ., Hamilton, Ont, Canada, 2005.
[11] B. Brandin and F. Charbonnier, “The supervisory control of the automated manufacturing system of the AIP,” in *Proc. Rensselaer’s 1994 4th Int. Conf. Computer Integrated Manufacturing and Automation Technology*, Troy, NY, Oct. 1994, pp. 319–324.
[12] F. Charbonnier, “Commande par supervision des systèmes à événements discrets: Application à un site expérimental l’Atelier Inter-établissement de Productique,” Laboratoire d’Automatique de Grenoble, Grenoble, France, Tech. Rep., 1994.
[13] R. Leduc and M. Lawford, “Hierarchical interface-based supervisory control of a flexible manufacturing system,” *IEEE Trans. Control Syst. Technol.* , 2005, submitted for publication.
[14] R. Leduc, M. Lawford, and W. M. Wonham. (2001, Nov.) Hierarchical interface based supervisory control: AIP example for parallel case. Software Quality Research Laboratory, Dept. Comput. Software, McMaster Univ., Hamilton, ON, Canada. [Online]. Available: Tech Rep. no. 2, http://www.cas.mcmaster.ca/sqrl/sqrl_reports.html
[15] Z. Zhang, “Smart TCT: An efficient algorithm for supervisory control design,” M.A.Sc. thesis, Dept. Elect. Comput. Eng., Univ. Toronto, Toronto, ON, Canada, 2001.
[16] Z. Zhang and W. M. Wonham, “STCT: An efficient algorithm for supervisory control design,” in *Proc. SCODES 2001*. Paris, France, Jul. 2001, pp. 82–93.

Ryan Leduc (M’02) received the B.Eng. degree in electrical engineering from the University of Victoria, Victoria, BC, Canada, in 1993, and the M.A.Sc. and Ph.D. degrees in electrical engineering from the University of Toronto, Toronto, ON, Canada, in 1996 and 2002, respectively.

In 1997 and 1998, he was a Guest Scientist at Siemens Corporate Technology, Munich, Germany. In 2001, he joined McMaster University, Hamilton, ON, Canada, where he is currently an Assistant Professor of Software Engineering. His research interests include supervisory control of discrete-event systems (DES) hierarchical structure, concurrency and implementation issues, and DES as software and hardware. He is also interested in hierarchical approaches to formal verification of software and hardware.

Dr. Leduc is the Chair of the IEEE Control Systems Society Technical Committee on Discrete-Event Systems.

Mark Lawford (S’88–M’97) received the B.Sc. degree in engineering mathematics from Queen’s University, Kingston, ON, Canada, in 1989 (receiving the University Medal in engineering mathematics), and the M.A.Sc. and Ph.D. degrees in electrical engineering at the University of Toronto, Toronto, ON, Canada, in 1992 and 1997, respectively.

From 1997 to 1998, he was with Ontario Hydro as a consultant on the Darlington Nuclear Generating Station Shutdown Systems Redesign project, where he was a co-recipient of an Ontario Hydro New Technology Award. Currently, he is an Associate Professor in the Department of Computing and Software at McMaster University, Hamilton, ON, Canada, where he has helped develop the Software Engineering programs. His research interests include discrete-event systems, formal methods for real-time systems, and computer aided inspection of safety critical software. He is a licensed Professional Engineer in the province of Ontario.

W. M. Wonham (M’64–SM’76–F’77) received the B.Eng. degree in engineering physics from McGill University, Montreal, QC, Canada, in 1956, and the Ph.D. degree in control engineering from the University of Cambridge, Cambridge, U.K., in 1961.

From 1961 to 1969, he was associated with several U.S. research groups in control. Since 1970, he has been a Faculty Member in Systems Control with the Department of Electrical and Computer Engineering of the University of Toronto, Toronto, ON, Canada. In addition, he has held visiting lectureships at Washington University, St. Louis, MO, the Massachusetts Institute of Technology, Cambridge, the Institute of System Science of the Academia Sinica, Beijing, China, and other institutions. His research interests have included stochastic control and filtering, geometric multivariable control, and discrete-event systems. He is the author of *Linear Multivariable Control: A Geometric Approach* (New York: Springer-Verlag, 1985) and the coauthor (with C. Ma) of *Hierarchical Control of State Tree Structures* (New York: Springer-Verlag, 2005).

Dr. Wonham is a Fellow of the Royal Society of Canada, and (2005) a Foreign Associate of the (U.S.) National Academy of Engineering. In 1987, he received the IEEE Control Systems Science and Engineering Award, and, in 1990, was a Brouwer Medallist of the Netherlands Mathematical Society. In 1996, he was appointed University Professor in the University of Toronto, and in 2000, University Professor Emeritus.