

Hierarchical Interface-based Supervisory Control - Part II: Parallel Case

R.J. Leduc, *Member, IEEE* M. Lawford, *Member, IEEE* and
W.M. Wonham *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. We first review the setting for the serial case ($n = 1$) [1], and then generalize it for $n \geq 1$. 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 closed-loop state space size of 2.9×10^{21} .

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], [2].¹ In the present 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

R.J. Leduc and M. Lawford are with the Dept. of Computing and Software, McMaster University, McMaster University, 1280 Main Street West, Hamilton, ON, Canada L8S 4K1. Email: leduc,lawford@mcmaster.ca

W.M. Wonham is with the Dept. of Electrical and Computer Engineering, University of Toronto, Toronto, ON, Canada, M5S 3G4. Email: wonham@control.utoronto.ca

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

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 following we first review the setting for the serial case ($n = 1$), discussed in [1]. We define an interface, and a set of (local) consistency properties that can be used to verify if a DES is globally nonblocking and controllable. We then generalize our definitions to $n \geq 1$. Next we present algorithms for verifying these new properties, briefly discussing computational complexity. We close with an application to a manufacturing example with an estimated closed-loop state space size of 2.9×10^{21} .

II. REVIEW OF HISC SERIAL CASE

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. 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. Only the primary serial case definitions required to explain the parallel case are restated here. Appendix II restates the serial case propositions needed for parallel case proofs. Details of the serial case definitions and propositions are given in Part I of this paper [1].

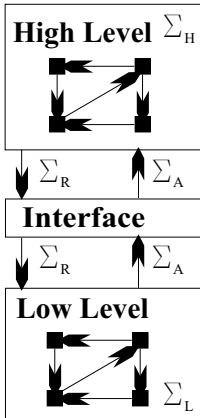


Fig. 1. Interface Block Diagram.

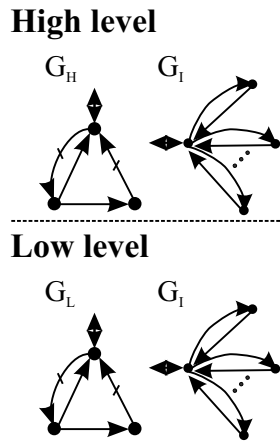


Fig. 2. Two Tiered Structure of the System.

To capture the restriction on information flow imposed by the interface, the alphabet Σ is split into four disjoint alphabets: Σ_H , Σ_L , Σ_R , and Σ_A . The events in Σ_H are *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 together form the *interface events*. These events are common to both levels of the hierarchy and facilitate 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 subsystem's responses to the request events.

A. Interface Definitions and Notation

We will review two interface definitions: *star interfaces* and the more general *command-pair interfaces*. To define a star interface, the designer selects a set of request events, and for each request event, a set of answer events; namely a map $\mathbf{Answer} : \Sigma_R \rightarrow \text{Pwr}(\Sigma_A)$. For $\rho \in \Sigma_R$, $\mathbf{Answer}(\rho)$ is the set of possible answers the low level subsystem could provide after receiving request ρ . We require that the low level subsystem provide at least one response for each request it receives, and that Σ_A does not contain any unused events. Figure 3 shows the required structure for a star interface, represented as a DES.

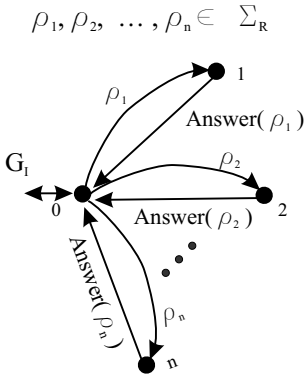


Fig. 3. Star Interface.

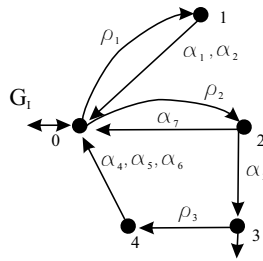


Fig. 4. Example Command-pair Interface.

Command-pair interfaces are a generalization of star interfaces. The key difference is that the “star” shape is no longer required allowing DES structures such as the one in Figure 4. In the sequel, by an interface we will mean explicitly a command-pair interface, and use the two synonymously.

Definition 1: A DES $G_I = (X, \Sigma_R \dot{\cup} \Sigma_A, \xi, x_o, X_m)$ is a *command-pair interface* if the following conditions are satisfied:

- (A) $L(G_I) \subseteq \overline{(\Sigma_R \cdot \Sigma_A)^*}$
- (B) $L_m(G_I) = (\Sigma_R \cdot \Sigma_A)^* \cap L(G_I)$

We assume the high level subsystem is modelled by DES G_H (defined over $\Sigma_H \dot{\cup} \Sigma_R \dot{\cup} \Sigma_A$), the low level subsystem by DES G_L (defined over event set $\Sigma_L \dot{\cup} \Sigma_R \dot{\cup} \Sigma_A$), and the interface by DES G_I (defined over $\Sigma_R \dot{\cup} \Sigma_A$). Also, the *high level* will mean the synchronous product $G_H || G_I$, and the *low level* $G_L || G_I$. The overall structure of the system is displayed in Figure 2.

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

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

The representation of the system given in Figure 2 simplifies notation when verifying non-blocking by ignoring the distinction between plants and supervisors. For controllability, we need to split the subsystems into their plant and supervisor components, as shown in Figure 5.

We next define the *high level plant* 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 notation, we identify the high and low level subsystems as below.

$$G_H := \mathcal{G}_H || \mathcal{S}_H \quad G_L := \mathcal{G}_L || \mathcal{S}_L$$

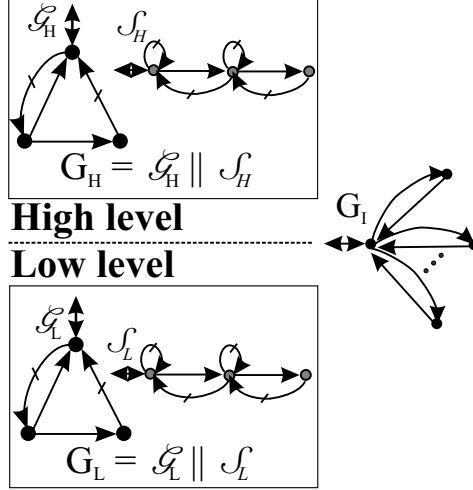


Fig. 5. Plant and Supervisor Subplant Decomposition

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

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

Finally, we need the eligibility function. For a language $L \subseteq \Sigma^*$ and a string $s \in \Sigma^*$, the function $\text{Elig}_L : \Sigma^* \rightarrow \text{Pwr}(\Sigma)$ is given by

$$\text{Elig}_L(s) := \{\sigma \in \Sigma \mid s\sigma \in L\}$$

B. Serial Interface Properties

We now present the interface requirements that the system must satisfy to ensure that it interacts with the interface correctly. Then we define the nonblocking and controllability requirements each level must satisfy.

Definition 2: 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 hold,

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.

High Level Properties

- 3) $(\forall s \in \mathcal{H} \cap \mathcal{I}) \text{Elig}_{\mathcal{I}}(s) \cap \Sigma_A \subseteq \text{Elig}_{\mathcal{H}}(s)$

Low Level Properties

- 4) $(\forall s \in \mathcal{L} \cap \mathcal{I}) \text{Elig}_{\mathcal{I}}(s) \cap \Sigma_R \subseteq \text{Elig}_{\mathcal{L}}(s)$
- 5) $(\forall s \in \Sigma^* \cdot \Sigma_R \cap \mathcal{L} \cap \mathcal{I})$

$$\text{Elig}_{\mathcal{L} \cap \mathcal{I}}(s \Sigma_L^*) \cap \Sigma_A = \text{Elig}_{\mathcal{I}}(s) \cap \Sigma_A \quad \text{where}$$

$$\text{Elig}_{\mathcal{L} \cap \mathcal{I}}(s \Sigma_L^*) := \bigcup_{l \in \Sigma_L^*} \text{Elig}_{\mathcal{L} \cap \mathcal{I}}(sl)$$

- 6) $(\forall s \in \mathcal{L} \cap \mathcal{I})$

$$s \in \mathcal{I}_m \Rightarrow (\exists l \in \Sigma_L^*) sl \in \mathcal{L}_m \cap \mathcal{I}_m$$

Definition 3: 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*

For the controllability requirements at each level, we adopt the standard partition $\Sigma = \Sigma_u \dot{\cup} \Sigma_c$, splitting our alphabet into *uncontrollable* and *controllable events*.

Definition 4: 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) $(\forall s \in \mathbf{L} \cap \mathbf{L}_S \cap \mathcal{I}) \text{Elig}_{\mathbf{L}}(s) \cap \Sigma_u \subseteq \text{Elig}_{\mathbf{L}_S \cap \mathcal{I}}(s)$
- (III) $(\forall s \in \mathbf{H} \cap \mathcal{I} \cap \mathbf{H}_S) \text{Elig}_{\mathbf{H} \cap \mathcal{I}}(s) \cap \Sigma_u \subseteq \text{Elig}_{\mathbf{H}_S}(s)$

We now present the *serial interface strict marking* condition, a restriction of the *serial interface consistency* definition.

Definition 5: The system composed of DES G_H , G_L and 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 $\mathcal{L} \cap \mathcal{I}_m = \mathcal{L}_m \cap \mathcal{I}_m$.

The above statement is equivalent to **Property 6** of the *serial interface consistency* definition above, with string l restricted to the empty string. It is useful as it implies **Property 6** of the *serial interface consistency* definition (see below), but is less expensive to evaluate.

Proposition 1: If the system composed of DES G_H , G_L and 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 proof in [6]. ■

III. PARALLEL CASE

In Section II, 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. Figure 6 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.

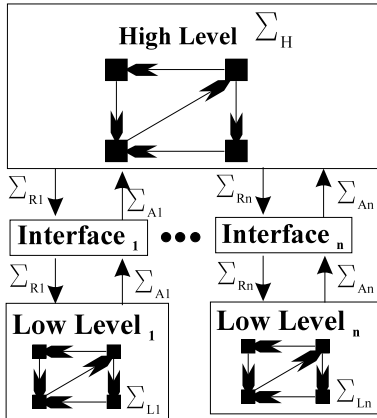


Fig. 6. Parallel Interface Block Diagram.

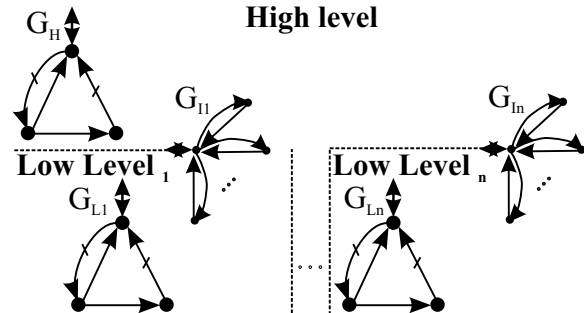


Fig. 7. Two Tiered Structure of Parallel System

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: Σ_H , Σ_{R_j} , Σ_{A_j} , and Σ_{L_j} , with $j = 1, \dots, n$.

For an n^{th} degree parallel system, the high level subsystem is modelled by DES G_H (defined over event set $\dot{\cup}_{j \in \{1, \dots, n\}} [\Sigma_{R_j} \dot{\cup} \Sigma_{A_j}] \dot{\cup} \Sigma_H$). For $j \in \{1, \dots, n\}$, 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 $\Sigma_{R_j} \dot{\cup} \Sigma_{A_j}$), so that the overall system has the structure shown in Figure 7. By the j^{th} low level we mean $G_{L_j} \parallel G_{I_j}$, assume that the alphabet partition is given by $\Sigma := \dot{\cup}_{j \in \{1, \dots, n\}} [\Sigma_{L_j} \dot{\cup} \Sigma_{R_j} \dot{\cup} \Sigma_{A_j}] \dot{\cup} \Sigma_H$, and take the flat system to be

$$G = G_H \parallel G_{L_1} \parallel \dots \parallel G_{L_n} \parallel G_{I_1} \parallel \dots \parallel G_{I_n}$$

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 $\{1, \dots, n\}$.

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

A. General Form

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 designate the high level plant as \mathcal{G}_H , and the high level supervisor as \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}). The high level subsystem and the j^{th} low level subsystem are

$$G_H := \mathcal{G}_H \parallel \mathcal{S}_H \qquad G_{L_j} := \mathcal{G}_{L_j} \parallel \mathcal{S}_{L_j}$$

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 G_H and 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:

$$\begin{aligned}
 \mathbf{Plant} &:= \mathcal{G}_H \parallel \mathcal{G}_{L_1} \parallel \dots \parallel \mathcal{G}_{L_n} & \mathbf{Sup} &:= \mathcal{S}_H \parallel \mathcal{S}_{L_1} \parallel \dots \parallel \mathcal{S}_{L_n} \parallel \mathcal{G}_{I_1} \parallel \dots \parallel \mathcal{G}_{I_n} \\
 \mathbf{H} &:= P_{IH}^{-1}L(\mathcal{G}_H), & \mathbf{H}_S &:= P_{IH}^{-1}L(\mathcal{S}_H), \quad \subseteq \Sigma^* \\
 \mathbf{L}_j &:= P_{IL_j}^{-1}L(\mathcal{G}_{L_j}), & \mathbf{L}_{S_j} &:= P_{IL_j}^{-1}L(\mathcal{S}_{L_j}), \quad \subseteq \Sigma^*
 \end{aligned}$$

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 n^{th} degree ($n \geq 1$) parallel interface system, shown conceptually in Figure 8 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 j^{th} *serial system extraction*, as the type of the parallel system will make clear which definition is intended.

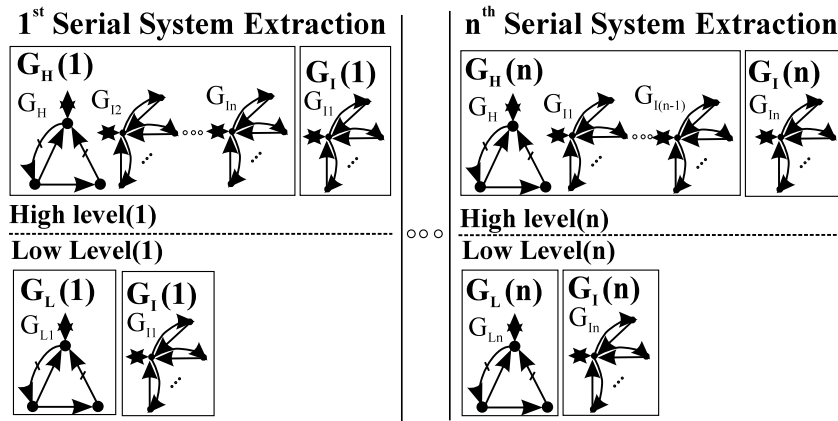


Fig. 8. The Serial System Extractions

Definition 6: For the n^{th} degree ($n \geq 1$) parallel interface system composed of DES G_H , $G_{L_1}, \dots, G_{L_n}, G_{I_1}, \dots, G_{I_n}$, with alphabet partition $\Sigma := \dot{\cup}_{k \in \{1, \dots, n\}} (\Sigma_{L_k} \dot{\cup} \Sigma_{R_k} \dot{\cup} \Sigma_{A_k}) \dot{\cup} \Sigma_H$, the j^{th} *serial system extraction* (subsystem form), denoted by $system(j)$, is composed of the

following elements:

$$\begin{aligned}
G_H(j) &:= G_H || G_{I_1} || \dots || G_{I_{(j-1)}} || G_{I_{(j+1)}} || \dots || G_{I_n} \\
G_L(j) &:= G_{L_j}, \quad G_I(j) := G_{I_j} \\
\Sigma_H(j) &:= \dot{\cup}_{k \in \{1, \dots, (j-1), (j+1), \dots, n\}} \Sigma_{I_k} \dot{\cup} \Sigma_H \\
\Sigma_L(j) &:= \Sigma_{L_j}, \quad \Sigma_R(j) := \Sigma_{R_j}, \quad \Sigma_A(j) := \Sigma_{A_j} \\
\Sigma(j) &:= \Sigma_H(j) \dot{\cup} \Sigma_L(j) \dot{\cup} \Sigma_R(j) \dot{\cup} \Sigma_A(j) \\
&= \Sigma - \dot{\cup}_{k \in \{1, \dots, (j-1), (j+1), \dots, n\}} \Sigma_{L_k}
\end{aligned}$$

Definition 7: For the n^{th} degree ($n \geq 1$) parallel interface system composed of DES $\mathcal{G}_H, \mathcal{G}_{L_1}, \dots, \mathcal{G}_{L_n}, \mathcal{S}_H, \mathcal{S}_{L_1}, \dots, \mathcal{S}_{L_n}, G_{I_1}, \dots, G_{I_n}$, with alphabet partition $\Sigma := \dot{\cup}_{k \in \{1, \dots, n\}} (\Sigma_{L_k} \dot{\cup} \Sigma_{R_k} \dot{\cup} \Sigma_{A_k}) \dot{\cup} \Sigma_H$, the j^{th} serial system extraction (general form), denoted by $\text{system}(j)$, is composed of the following elements:

$$\begin{aligned}
\mathcal{G}_H(j) &:= \mathcal{G}_H || G_{I_1} || \dots || G_{I_{(j-1)}} || G_{I_{(j+1)}} || \dots || G_{I_n} \\
\mathcal{S}_H(j) &:= \mathcal{S}_H, \mathcal{G}_L(j) := \mathcal{G}_{L_j}, \mathcal{S}_L(j) := \mathcal{S}_{L_j}, G_I(j) := G_{I_j}
\end{aligned}$$

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. They all involve interpreting the parallel system as n serial systems by using the serial system extraction definition.

Definition 8: The n^{th} degree ($n \geq 1$) parallel interface system composed of DES $G_H, G_{L_1}, \dots, G_{L_n}, G_{I_1}, \dots, G_{I_n}$, is *interface consistent* with respect to alphabet partition $\Sigma := \dot{\cup}_{k \in \{1, \dots, n\}} [\Sigma_{L_k} \dot{\cup} \Sigma_{R_k} \dot{\cup} \Sigma_{A_k}] \dot{\cup} \Sigma_H$, if for all $j \in \{1, \dots, n\}$ the j^{th} serial system extraction of the system is serial interface consistent.

Definition 9: The n^{th} degree ($n \geq 1$) parallel interface system composed of DES $G_H, G_{L_1}, \dots, G_{L_n}, G_{I_1}, \dots, G_{I_n}$, is *level-wise nonblocking* with respect to the alphabet partition $\Sigma := \dot{\cup}_{k \in \{1, \dots, n\}} [\Sigma_{L_k} \dot{\cup} \Sigma_{R_k} \dot{\cup} \Sigma_{A_k}] \dot{\cup} \Sigma_H$, if for all $j \in \{1, \dots, n\}$ the j^{th} serial system extraction of the system is serial level-wise nonblocking.

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 10: The n^{th} degree ($n \geq 1$) parallel interface system composed of DES $\mathcal{G}_H, \mathcal{G}_{L_1}, \dots, \mathcal{G}_{L_n}, \mathcal{S}_H, \mathcal{S}_{L_1}, \dots, \mathcal{S}_{L_n}, G_{I_1}, \dots, G_{I_n}$, is *level-wise controllable* with respect to alphabet partition $\Sigma := \dot{\cup}_{k \in \{1, \dots, n\}} [\Sigma_{L_k} \dot{\cup} \Sigma_{R_k} \dot{\cup} \Sigma_{A_k}] \dot{\cup} \Sigma_H$, if for all $j \in \{1, \dots, n\}$ the j^{th} 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 j^{th} serial system extraction in terms of the original parallel system. First we need the new natural projection, $P_j : \Sigma^* \rightarrow \Sigma(j)^*$.

Proposition 2: If the n^{th} degree ($n \geq 1$) parallel interface system composed of DES $G_H, G_{L_1}, \dots, G_{L_n}, G_{I_1}, \dots, G_{I_n}$, is interface consistent with respect to the alphabet partition $\Sigma := \dot{\cup}_{k \in \{1, \dots, n\}} (\Sigma_{L_k} \dot{\cup} \Sigma_{R_k} \dot{\cup} \Sigma_{A_k}) \dot{\cup} \Sigma_H$, then for the j^{th} serial system extraction, $\text{system}(j)$, the following is true:

- (i) The flat system is: $G(j) = G_H || G_{L_j} || G_{I_1} || \dots || G_{I_n}$
- (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 $G_H(j)$ is $\Sigma_{IH}(j)$, the event set of $G_L(j)$ is $\Sigma_{IL}(j)$, and the event set of $G_I(j)$ is $\Sigma_I(j)$.
- (v) The indicated languages satisfy the following

$$\begin{aligned} \mathcal{H}(j) &= P_j(\mathcal{H}) \cap [\cap_{k \in \{1, \dots, (j-1), (j+1), \dots, n\}} P_j(\mathcal{I}_k)] \\ \mathcal{H}_m(j) &= P_j(\mathcal{H}) \cap [\cap_{k \in \{1, \dots, (j-1), (j+1), \dots, n\}} P_j(\mathcal{I}_{m_k})] \\ \mathcal{L}(j) &= P_j(\mathcal{L}_j) \\ \mathcal{L}_m(j) &= P_j(\mathcal{L}_{m_j}) \\ \mathcal{I}(j) &= P_j(\mathcal{I}_j) \\ \mathcal{I}_m(j) &= P_j(\mathcal{I}_{m_j}) \end{aligned}$$

- (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 proof in [6]. ■

Proposition 3: If the n^{th} degree ($n \geq 1$) parallel interface system composed of DES $\mathcal{G}_H, \mathcal{G}_{L_1}, \dots, \mathcal{G}_{L_n}, \mathcal{S}_H, \mathcal{S}_{L_1}, \dots, \mathcal{S}_{L_n}, G_{I_1}, \dots, G_{I_n}$, is level-wise controllable with respect to the alphabet partition $\Sigma := \dot{\cup}_{k \in \{1, \dots, n\}} (\Sigma_{L_k} \dot{\cup} \Sigma_{R_k} \dot{\cup} \Sigma_{A_k}) \dot{\cup} \Sigma_H$, then for the j^{th} serial system extraction, $\text{system}(j)$, the following is true:

- (i) The flat plant is $\mathbf{Plant}(j) = \mathcal{G}_H \| G_{I_1} \| \dots \| G_{I_{(j-1)}} \| G_{I_{(j+1)}} \| \dots \| G_{I_n} \| \mathcal{G}_{L_j}$ and the flat supervisor is $\mathbf{Sup}(j) = \mathcal{S}_H \| \mathcal{S}_{L_j} \| G_{I_j}$
- (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 $\mathcal{G}_H(j)$ and $\mathcal{S}_H(j)$ is $\Sigma_{IH}(j)$, the alphabet of $\mathcal{G}_L(j)$ and $\mathcal{S}_L(j)$ is $\Sigma_{IL}(j)$, and the alphabet of $G_I(j)$ is $\Sigma_I(j)$
- (v) The indicated languages satisfy the following statements:

$$\mathbf{H}(j) = P_j(\mathbf{H}) \cap [\cap_{k \in \{1, \dots, (j-1), (j+1), \dots, n\}} P_j(\mathcal{I}_k)]$$

$$\mathbf{H}_S(j) = P_j(\mathbf{H}_S)$$

$$\mathbf{L}(j) = P_j(\mathbf{L}_j)$$

$$\mathbf{L}_S(j) = P_j(\mathbf{L}_{S_j})$$

$$\mathcal{I}(j) = P_j(\mathcal{I}_j)$$

$$L(\mathbf{Plant}(j)) = P_j(\mathbf{H}) \cap [\cap_{k \in \{1, \dots, (j-1), (j+1), \dots, n\}} P_j(\mathcal{I}_k)] \cap P_j(\mathbf{L}_j)$$

$$L(\mathbf{Sup}(j)) = P_j(\mathbf{H}_S) \cap P_j(\mathbf{L}_{S_j}) \cap P_j(\mathcal{I}_j)$$

- (vi) Languages $\mathbf{H}(j), \mathbf{H}_S(j), \mathbf{L}(j), \mathbf{L}_S(j), \mathcal{I}(j), L(\mathbf{Plant})(j)$, and $L(\mathbf{Sup})(j)$ are closed.

Proof: See 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 $P_k : \Sigma^* \rightarrow \Sigma_k^*$, where $k = a, b$.

Proposition 4: 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 proof in [6]. ■

D. Nonblocking Propositions and Theorem

We will now present **Propositions 5-7**, 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 **Proposition 10** for the serial case. It asserts that the low levels are not dependent on high level events to reach a marked state.

Proposition 5: If the n^{th} degree ($n \geq 1$) parallel interface system composed of DES $G_H, G_{L_1}, \dots, G_{L_n}, G_{I_1}, \dots, G_{I_n}$, is level-wise nonblocking and interface consistent with respect to the alphabet partition $\Sigma := \dot{\cup}_{k \in \{1, \dots, n\}} (\Sigma_{L_k} \dot{\cup} \Sigma_{R_k} \dot{\cup} \Sigma_{A_k}) \dot{\cup} \Sigma_H$, then

$$(\forall s \in \mathcal{H} \cap [\cap_{j \in \{1, \dots, n\}} (\mathcal{L}_j \cap \mathcal{I}_j)])$$

$$(\exists l \in \Sigma_{IL}^*) (sl \in \mathcal{H} \cap [\cap_{j \in \{1, 2, \dots, n\}} (\mathcal{L}_{m_j} \cap \mathcal{I}_{m_j})])$$

Proof: See Appendix I. ■

We group the last two propositions together as **Proposition 7** builds upon **Proposition 6**. The first proposition asserts that any string accepted by the system can always be extended to a string marked by the high level.

Proposition 6: If the n^{th} degree ($n \geq 1$) parallel interface system composed of DES $G_H, G_{L_1}, \dots, G_{L_n}, G_{I_1}, \dots, G_{I_n}$, is level-wise nonblocking and interface consistent with respect to the alphabet partition $\Sigma := \dot{\cup}_{k \in \{1, \dots, n\}} (\Sigma_{L_k} \dot{\cup} \Sigma_{R_k} \dot{\cup} \Sigma_{A_k}) \dot{\cup} \Sigma_H$, then

$$(\forall s \in \mathcal{H} \cap [\cap_{j \in \{1, \dots, n\}} (\mathcal{L}_j \cap \mathcal{I}_j)])$$

$$(\exists h \in \Sigma_{IH}^*) (sh \in \mathcal{H}_m \cap [\cap_{j \in \{1, \dots, n\}} \mathcal{I}_{m_j}])$$

Proof: See Appendix I. ■

Our last proposition is analogous to **Proposition 11** for the serial case. 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 7: If the n^{th} degree ($n \geq 1$) parallel interface system composed of DES $G_H, G_{L_1}, \dots, G_{L_n}, G_{I_1}, \dots, G_{I_n}$, is level-wise nonblocking and interface consistent with respect to the alphabet partition $\Sigma := \dot{\cup}_{k \in \{1, \dots, n\}} (\Sigma_{L_k} \dot{\cup} \Sigma_{R_k} \dot{\cup} \Sigma_{A_k}) \dot{\cup} \Sigma_H$, then

$$(\forall s \in \mathcal{H} \cap [\cap_{j \in \{1, \dots, n\}} (\mathcal{L}_{m_j} \cap \mathcal{I}_{m_j})]) (\forall h \in \Sigma_{IH}^*)$$

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

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 n^{th} degree ($n \geq 1$) parallel interface system composed of DES $G_H, G_{L_1}, \dots, G_{L_n}, G_{I_1}, \dots, G_{I_n}$, is level-wise nonblocking and interface consistent with respect to the alphabet partition $\Sigma := \dot{\cup}_{k \in \{1, \dots, n\}} (\Sigma_{L_k} \dot{\cup} \Sigma_{R_k} \dot{\cup} \Sigma_{A_k}) \dot{\cup} \Sigma_H$, then $L(G) = \overline{L_m(G)}$, where $G = G_H || G_{L_1} || \dots || G_{L_n} || G_{I_1} || \dots || G_{I_n}$

Proof:

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

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

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

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

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

We first apply **Proposition 5** and conclude:

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

We note that (3) implies that $sl \in \mathcal{H} \cap [\cap_{j \in \{1, \dots, n\}} (\mathcal{L}_j \cap \mathcal{I}_j)]$. Now we can apply **Proposition 6**, taking sl to be string s in that proposition, and conclude:

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

Combining with (3), we can now apply **Proposition 7**, taking sl to be string s in that proposition, and conclude:

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

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

E. Controllability Propositions and Theorem

We will now present **Propositions 8** and **9**, followed by our main result for this section, **Theorem 2**. The following propositions are analogous to the serial controllability propositions.

We start with **proposition 8**. 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 8: If the n^{th} degree ($n \geq 1$) parallel interface system composed of plant components $\mathcal{G}_H, \mathcal{G}_{L_1}, \dots, \mathcal{G}_{L_n}$, supervisors $\mathcal{S}_H, \mathcal{S}_{L_1}, \dots, \mathcal{S}_{L_n}$, and interfaces G_{I_1}, \dots, G_{I_n} , is level-wise controllable with respect to the alphabet partition $\Sigma := \dot{\cup}_{k \in \{1, \dots, n\}} (\Sigma_{L_k} \dot{\cup} \Sigma_{R_k} \dot{\cup} \Sigma_{A_k}) \dot{\cup} \Sigma_H$, then

$$(\forall j \in \{1, \dots, n\}) (\forall s \in L(\mathbf{Plant}) \cap \mathbf{L}_{S_j} \cap \mathcal{I}_j) \quad \text{Elig}_{L(\mathbf{Plant})}(s) \cap \Sigma_u \subseteq \text{Elig}_{\mathbf{L}_{S_j} \cap \mathcal{I}_j}(s)$$

Proof: See Appendix I. ■

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

Proposition 9: If the n^{th} degree ($n \geq 1$) parallel interface system composed of plant components $\mathcal{G}_H, \mathcal{G}_{L_1}, \dots, \mathcal{G}_{L_n}$, supervisors $\mathcal{S}_H, \mathcal{S}_{L_1}, \dots, \mathcal{S}_{L_n}$, and interfaces G_{I_1}, \dots, G_{I_n} , is level-wise controllable with respect to the alphabet partition $\Sigma := \dot{\cup}_{k \in \{1, \dots, n\}} (\Sigma_{L_k} \dot{\cup} \Sigma_{R_k} \dot{\cup} \Sigma_{A_k}) \dot{\cup} \Sigma_H$, then

$$(\forall s \in L(\mathbf{Plant}) \cap \mathbf{H}_S \cap [\cap_{k \in \{1, \dots, n\}} \mathcal{I}_k]) \quad \text{Elig}_{L(\mathbf{Plant}) \cap [\cap_{k \in \{1, \dots, n\}} \mathcal{I}_k]}(s) \cap \Sigma_u \subseteq \text{Elig}_{\mathbf{H}_S}(s)$$

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

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

Proof:

Assume that the n^{th} degree ($n \geq 1$) parallel interface system is level-wise controllable. (1)

Let $s \in L(\mathbf{Plant}) \cap L(\mathbf{Sup})$, and $\sigma \in \text{Elig}_{L(\mathbf{Plant})}(s) \cap \Sigma_u$ (2)

We show this implies $\sigma \in \text{Elig}_{L(\mathbf{Sup})}(s)$. It is sufficient to show $s\sigma \in L(\mathbf{Sup}) = \mathbf{H}_S \cap [\bigcap_{k \in \{1, \dots, n\}} (\mathbf{L}_{S_k} \cap \mathcal{I}_k)]$

Note $s, s\sigma \in \mathbf{H} \cap [\bigcap_{k \in \{1, \dots, n\}} \mathbf{L}_k] = L(\mathbf{Plant})$ and $s \in \mathbf{H}_S \cap [\bigcap_{k \in \{1, \dots, n\}} (\mathbf{L}_{S_k} \cap \mathcal{I}_k)]$ by (2). (3)

By (1), we can apply **Proposition 8** and conclude:

$$s\sigma \in \bigcap_{k \in \{1, \dots, n\}} (\mathbf{L}_{S_k} \cap \mathcal{I}_k) \quad (4)$$

All that remains is to show that $s\sigma \in \mathbf{H}_S$

From (3), we have $s \in L(\mathbf{Plant}) \cap \mathbf{H}_S \cap [\bigcap_{k \in \{1, \dots, n\}} \mathcal{I}_k]$

From (3) and (4), we have $s\sigma \in L(\mathbf{Plant}) \cap [\bigcap_{k \in \{1, \dots, n\}} \mathcal{I}_k]$

$$\Rightarrow \sigma \in \text{Elig}_{L(\mathbf{Plant}) \cap [\bigcap_{k \in \{1, \dots, n\}} \mathcal{I}_k]}(s) \cap \Sigma_u$$

We can now apply **Proposition 9**, and conclude $\sigma \in \text{Elig}_{\mathbf{H}_S}(s)$

$$\Rightarrow s\sigma \in \mathbf{H}_S$$

Combining with (4), we can now conclude $s\sigma \in \mathbf{H}_S \cap [\bigcap_{k \in \{1, \dots, n\}} (\mathbf{L}_{S_k} \cap \mathcal{I}_k)]$, as required. ■

IV. 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 (Def. 3) and controllable (Def. 4), and serial interface consistent (Def. 2). 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 CTCT's **condat** function [7]. For example, in the case of **Point 3**, we simply define **ThePlant** = G_I , **TheSpec** = 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 $G_{I_j} || G_{L_j}$. If we assume that the sizes of the state sets of G_{I_j} and G_{L_j} are bounded by N_I and N_L , respectively, then this operation is $\mathcal{O}(N_I^3 N_L^3)$ for each low-level component [8] and hence $\mathcal{O}(N_I^3 N_L^3 n)$ for the system. An algorithm for checking these two points is presented in Appendix III.

All of the routines used on the example in Sec. V 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**). Further details of the algorithms can be found in [6].

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

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

If we let N_H denote the size of the state space of G_H , the product state space is bounded by $N_H N_L^n$.

From Def. 7 we see that the high-level of the j^{th} serial system extraction is given by:

$$G_H(j) := G_H || G_{I_1} || \dots || G_{I_{(j-1)}} || G_{I_{(j+1)}} || \dots || G_{I_n}$$

which has a state space size bound of $N_H N_I^{n-1}$. This system is then used to check serial level-wise non-blocking (Def. 3-I) and controllability (Def. 4-III) which require an additional language intersection with \mathcal{I}_j effectively requiring computation of $G_H(j) || G_{I_j}$, 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 nonblock-ing 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 trade-off 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.

V. 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 [10] and [11]. The AIP, shown in Figure 9, 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 type 1 or of type 2, chosen at random.

A. Assembly Stations

The assembly stations are shown in Figure 10. 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 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 tasks 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.

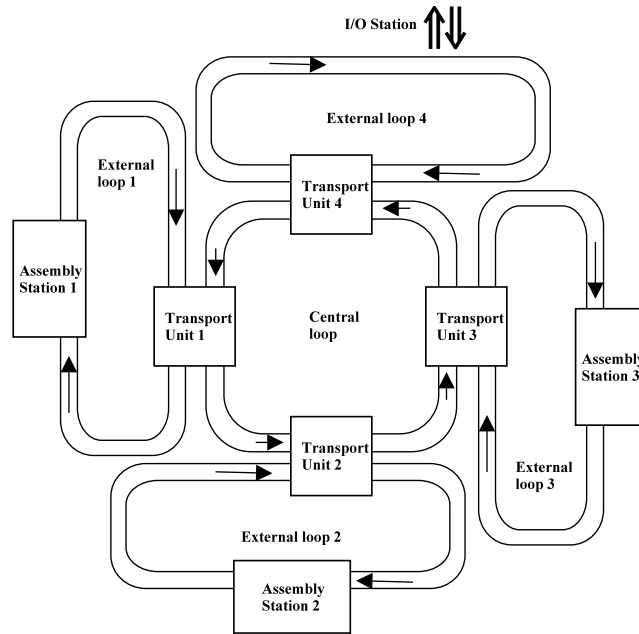


Fig. 9. The Atelier Inter-établissement de Productique

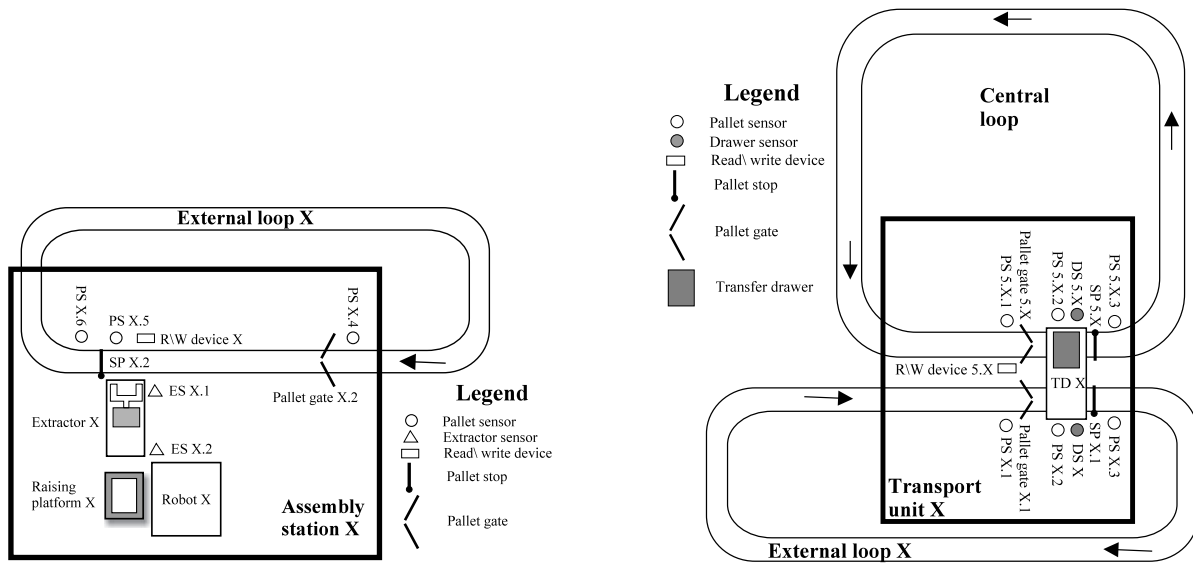


Fig. 10. Assembly Station of External Loop $X = 1, 2, 3$.

Fig. 11. Transport Unit for External Loop $X = 1, 2, 3, 4$

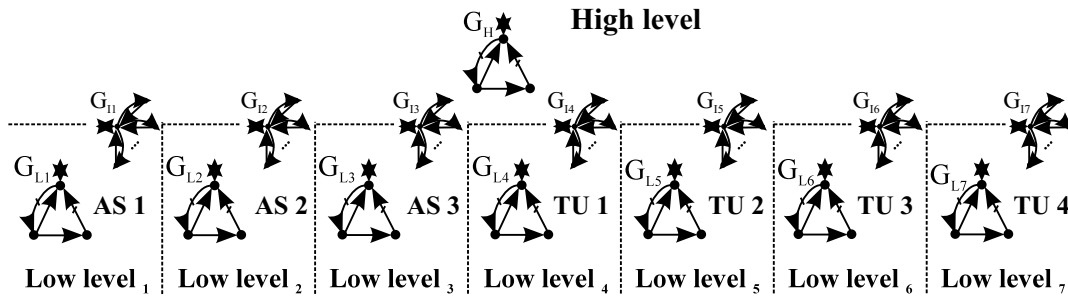


Fig. 12. Structure of Parallel System

B. Transport Units

The structure of the four identical transport units is shown in Figure 11. The transport units are used to transfer pallets between 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 [10] and [11] and restated below. To this we add **Specification 7** to make the assembly stations more interesting.

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

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 down into a high level, and seven low levels corresponding to the three assembly stations and four transport units, as shown in Figure 12. We select a few representative subsystems to describe in the following sections. As this example contains 181 DES, 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 [10] and [11]. 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) *The 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 Figure 13. 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 $k = AS1, AS2$.

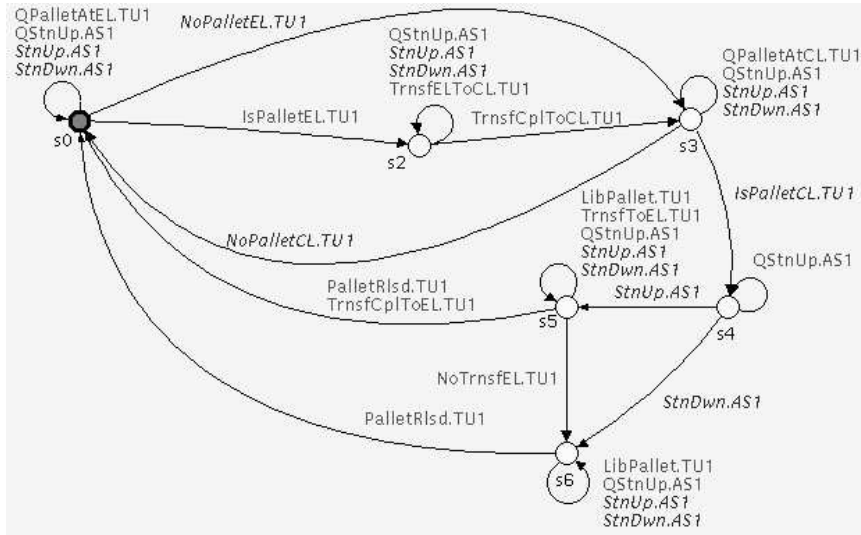


Fig. 13. Supervisor ManageTU1.

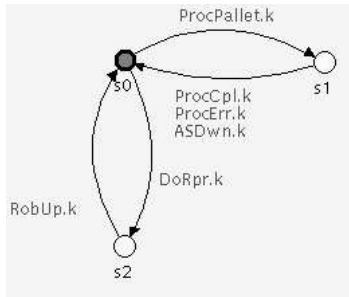


Fig. 14. Interface to low level $k = AS1, AS2$.

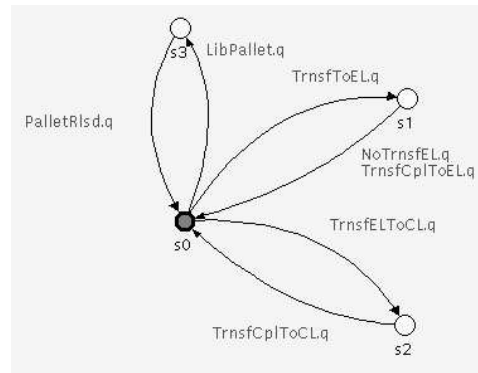


Fig. 15. Interface to low level $q = TU1, TU2, TU4$.

Subsystem k contains 17 DES and provides the functionality specified in its interface, shown in Figure 14. An assembly station accepts the pallet at its gate, and presents it to the robot 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 Figure 16, 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.

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 $r = TU1, TU2$.

Subsystem r contains 25 DES and provides the functionality specified in its interface, shown in Figure 15. 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 task required by the pallet is performed by the external loop's assembly station.

Supervisor *HndlTrnsfToEL.r*, shown in Figure 17, 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 non-blocking, serial level-wise controllable, and serial interface consistent. Thus we conclude that the system is level-wise non-blocking, 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 minutes, using 760MB of memory. The machine used was a 750MHz Athlon system, with 512MB of RAM, 2GB of swap, running Redhat Linux 6.2. A standard

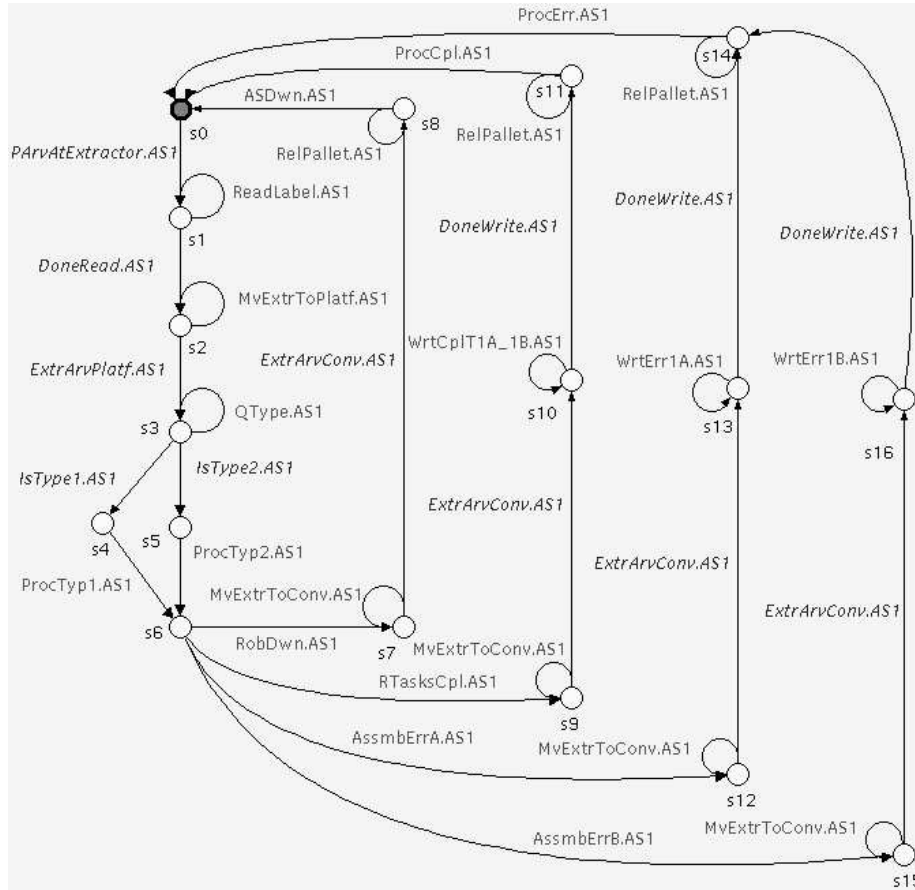


Fig. 16. Supervisor HndIPallet.AS1

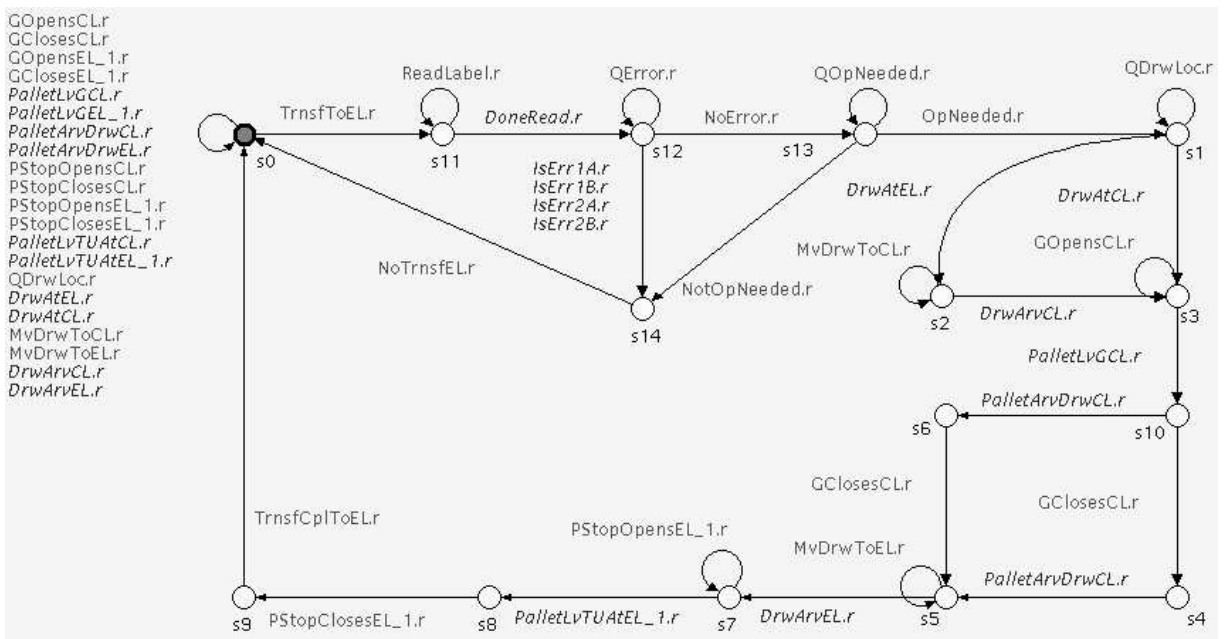


Fig. 17. Supervisor HndTrnsfToEL.r

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,

Subsystem	Standalone		with G_{I_j}		Size of G_{I_j}	
	States	Transitions	States	Transitions	States	Transitions
G_H	1,480,864	14,419,512	3,306,240	28,442,432	8,192	104,448
AS1	1,795	4,418	120	191	4	6
AS2	1,795	4,418	120	191	4	6
AS3	1,199	2,448	203	330	2	4
TU1	98	120	98	120	4	7
TU2	98	120	98	120	4	7
TU3	204	266	204	266	4	10
TU4	152	180	152	180	4	7

TABLE I

SIZE OF AIP SUBSYSTEM AUTOMATA MODELS

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 (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 IV, 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 G_H , while N_I and N_L are the bounds for G_{I_j} and G_{L_j} ($j = 1, \dots, n$), 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 $N_I^n = (4)^2(2)(4)^4 = 8192$. This is a potential savings of 11 orders of magnitude! In fact, instead of multiplying $N_H = 1,480,864$ by a factor of $N_I^n = 8192$, 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 1,795 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 [12], [13]. 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.

VI. CONCLUSIONS

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

Finally, we discussed a large example based on the automated manufacturing system of the Atelier Inter-établissement de Productique (AIP). As the example contains 181 DES with an estimated 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

Proof for **Proposition 5** in Section III-D: *If the n^{th} degree ($n \geq 1$) parallel interface system composed of DES $G_H, G_{L_1}, \dots, G_{L_n}, G_{I_1}, \dots, G_{I_n}$, is level-wise nonblocking and interface consistent with respect to the alphabet partition $\Sigma := \dot{\cup}_{k \in \{1, \dots, n\}} (\Sigma_{L_k} \dot{\cup} \Sigma_{R_k} \dot{\cup} \Sigma_{A_k}) \dot{\cup} \Sigma_H$, then*

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

Proof:

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

Let $s \in \mathcal{H} \cap [\bigcap_{j \in \{1, \dots, n\}} (\mathcal{L}_j \cap \mathcal{I}_j)]$ (2)

We will now show this implies: $(\exists l \in \Sigma_{IL}^*) (sl \in \mathcal{H} \cap [\bigcap_{j \in \{1, \dots, n\}} (\mathcal{L}_{m_j} \cap \mathcal{I}_{m_j})])$

To do this, we will use an inductive proof. We define $\Sigma_{IL_0}^* = \Sigma^*$, $\mathcal{L}_{m_0} = \mathcal{I}_{m_0} = \Sigma^*$ and $\mathcal{L}_{n+1} = \mathcal{I}_{n+1} = \Sigma^*$.

Claim to be proven:

For $k \in \{0, 1, \dots, n\}$, there exist strings $l_i \in \Sigma_{IL_i}^*$, $i \in \{0, 1, \dots, k\}$, such that:

$$sl_0 l_1 \dots l_k \in \mathcal{H} \cap [\bigcap_{v \in \{0, 1, \dots, k\}} (\mathcal{L}_{m_v} \cap \mathcal{I}_{m_v})] \cap [\bigcap_{w \in \{k+1, \dots, n+1\}} (\mathcal{L}_w \cap \mathcal{I}_w)] \quad (3)$$

Initial Case: $k = 0$

We take $l_0 = \epsilon \in \Sigma_{IL_0}^* = \Sigma^*$. We immediately have $sl_0 = s \in \mathcal{H} \cap [\bigcap_{j \in \{1, \dots, n\}} (\mathcal{L}_j \cap \mathcal{I}_j)]$ (by (2))

We have automatically $sl_0 \in (\mathcal{L}_{m_0} \cap \mathcal{I}_{m_0} \cap \mathcal{L}_{n+1} \cap \mathcal{I}_{n+1}) = \Sigma^*$

Initial case complete.

Inductive Step:

Let $k \in \{1, \dots, n\}$. Assume there exist strings $l_i \in \Sigma_{IL_i}^*$, $i \in \{0, 1, \dots, k-1\}$, and that they satisfy (3) when $k-1$ is substituted for k .

$$\Rightarrow sl_0 l_1 \dots l_{k-1} \in \mathcal{H} \cap [\bigcap_{v \in \{0, 1, \dots, (k-1)\}} (\mathcal{L}_{m_v} \cap \mathcal{I}_{m_v})] \cap [\bigcap_{w \in \{k, \dots, n+1\}} (\mathcal{L}_w \cap \mathcal{I}_w)] \quad (4)$$

We will show this implies that we can construct string $l_k \in \Sigma_{IL_k}^*$, such that $sl_0 l_1 \dots l_k \in \mathcal{H} \cap [\bigcap_{v \in \{0, 1, \dots, k\}} (\mathcal{L}_{m_v} \cap \mathcal{I}_{m_v})] \cap [\bigcap_{w \in \{k+1, \dots, n+1\}} (\mathcal{L}_w \cap \mathcal{I}_w)]$

We first note that (4) implies $sl_0 l_1 \dots l_{k-1} \in \mathcal{H} \cap [\bigcap_{w \in \{1, \dots, n\}} (\mathcal{L}_w \cap \mathcal{I}_w)]$

$\Rightarrow P_k(sl_0 l_1 \dots l_{k-1}) \in P_k(\mathcal{H}) \cap P_k(\mathcal{L}_k) \cap [\bigcap_{w \in \{1, \dots, n\}} P_k(\mathcal{I}_w)] = \mathcal{H}(k) \cap \mathcal{L}(k) \cap \mathcal{I}(k)$
by **Proposition 2**

We apply **Proposition 10** to system (k) by taking $P_k(sl_0 l_1 \dots l_{k-1})$ to be string s in that proposition and conclude:

$$(\exists l_k \in \Sigma_{IL_k}^*) P_k(sl_0 l_1 \dots l_{k-1}) l_k \in \mathcal{H}(k) \cap \mathcal{L}_m(k) \cap \mathcal{I}_m(k)$$

Noting that $P_k(l_k) = l_k$ as $\Sigma_{IL_k} \subseteq \Sigma(k)$, we can conclude:

$$P_k(sl_0l_1 \dots l_{k-1}l_k) \in \mathcal{H}(k) \cap \mathcal{L}_m(k) \cap \mathcal{I}_m(k) \quad (5)$$

Substituting into (5) for $\mathcal{H}(k)$, $\mathcal{L}_m(k)$, and $\mathcal{I}_m(k)$ (by **Proposition 2**), we have:

$$\begin{aligned} P_k(sl_0l_1 \dots l_{k-1}l_k) \in & P_k \cdot P_{IH}^{-1}(L(G_H)) \cap [\cap_{w \in \{1, \dots, k-1, k+1, \dots, n\}} P_k \cdot P_{I_w}^{-1}(L(G_{I_w}))] \\ & \cap P_k \cdot P_{IL_k}^{-1}(L_m(G_{L_k})) \cap P_k \cdot P_{I_k}^{-1}(L_m(G_{I_k})) \end{aligned} \quad (6)$$

From **Proposition 2**, we have $\Sigma_{IH} \subseteq \Sigma(k)$, $\Sigma_{I_w} \subseteq \Sigma(k)$ (for $w \in \{1, \dots, k-1, k+1, \dots, n\}$), $\Sigma_{IL_k} \subseteq \Sigma(k)$, and $\Sigma_{I_k} \subseteq \Sigma(k)$. We can now apply **Proposition 4** four times by taking $[\Sigma_a = \Sigma(k)$, $\Sigma_b = \Sigma_{IH}$, and $L_b = L(G_H)]$, $[\Sigma_a = \Sigma(k)$, $\Sigma_b = \Sigma_{I_w}$, and $L_b = L(G_{I_w})]$, $[\Sigma_a = \Sigma(k)$, $\Sigma_b = \Sigma_{IL_k}$, and $L_b = L_m(G_{L_k})]$, and $[\Sigma_a = \Sigma(k)$, $\Sigma_b = \Sigma_{I_k}$, and $L_b = L_m(G_{I_k})]$ and conclude:

$$\begin{aligned} sl_0l_1 \dots l_{k-1}l_k \in & P_{IH}^{-1}(L(G_H)) \cap [\cap_{w \in \{1, \dots, k-1, k+1, \dots, n\}} P_{I_w}^{-1}(L(G_{I_w}))] \\ & \cap P_{IL_k}^{-1}(L_m(G_{L_k})) \cap P_{I_k}^{-1}(L_m(G_{I_k})) \end{aligned}$$

$$\Rightarrow sl_0l_1 \dots l_{k-1}l_k \in \mathcal{H} \cap [\cap_{w \in \{1, \dots, k-1, k+1, \dots, n\}} \mathcal{I}_w] \cap \mathcal{L}_{m_k} \cap \mathcal{I}_{m_k} \quad (7)$$

$$\text{We have automatically } sl_0l_1 \dots l_{k-1}l_k \in (\mathcal{L}_{m_0} \cap \mathcal{I}_{m_0} \cap \mathcal{L}_{n+1} \cap \mathcal{I}_{n+1}) = \Sigma^* \quad (8)$$

We next note that by (4), we have:

$$sl_0l_1 \dots l_{k-1} \in \mathcal{H} \cap [\cap_{v \in \{0, 1, \dots, (k-1)\}} (\mathcal{L}_{m_v} \cap \mathcal{I}_{m_v})] \cap [\cap_{w \in \{k+1, \dots, n\}} \mathcal{L}_w] \quad (9)$$

From (1) we have $\Sigma_{IL_k} \cap \Sigma_{IL_v} = \emptyset$, for $v \in \{1, \dots, (k-1)\}$. As $l_k \in \Sigma_{IL_k}$, we have $P_{IL_v}(sl_0l_1 \dots l_{k-1}l_k) = P_{IL_v}(sl_0l_1 \dots l_{k-1})$. By (9), this implies:

$$P_{IL_v}(sl_0l_1 \dots l_{k-1}l_k) \in L_m(G_{L_v})$$

$$\Rightarrow sl_0l_1 \dots l_{k-1}l_k \in \mathcal{L}_{m_v} \quad (10)$$

Similarly we have $P_{I_v}(sl_0l_1 \dots l_{k-1}l_k) = P_{I_v}(sl_0l_1 \dots l_{k-1})$ and $P_{IL_w}(sl_0l_1 \dots l_{k-1}l_k) = P_{IL_w}(sl_0l_1 \dots l_{k-1})$, for $w \in \{k+1, \dots, n\}$. We thus conclude:

$$sl_0l_1 \dots l_{k-1}l_k \in [\cap_{v \in \{0, 1, \dots, (k-1)\}} \mathcal{I}_{m_v}] \cap [\cap_{w \in \{k+1, \dots, n\}} \mathcal{L}_w]$$

Combining with (10), we have:

$$sl_0l_1 \dots l_{k-1}l_k \in \cap_{v \in \{0, 1, \dots, k-1\}} (\mathcal{L}_{m_v} \cap \mathcal{I}_{m_v}) \cap [\cap_{w \in \{k+1, \dots, n\}} \mathcal{L}_w]$$

Combining with (7) and (8), we have:

$$sl_0l_1 \dots l_{k-1}l_k \in \mathcal{H} \cap [\cap_{v \in \{0, 1, \dots, k\}} (\mathcal{L}_{m_v} \cap \mathcal{I}_{m_v})] \cap [\cap_{w \in \{k+1, \dots, n+1\}} (\mathcal{L}_w \cap \mathcal{I}_w)], \text{ as}$$

required.

Inductive step complete.

We have now proven the **Initial case** and the **Inductive step**. We now conclude that the **Claim** is true, by induction.

Taking $k = n$ and using fact that $l_0 = \epsilon$, we thus can conclude there exists strings $l_i \in \Sigma_{IL_i}^*$, $i \in \{1, \dots, n\}$, such that: $sl_1 \dots l_n \in \mathcal{H} \cap [\bigcap_{j \in \{1, \dots, n\}} (\mathcal{L}_{m_j} \cap \mathcal{I}_{m_j})]$

We thus take $l = l_1 \dots l_n$ and we have $l \in \Sigma_{IL}^*$ and $sl \in \mathcal{H} \cap [\bigcap_{j \in \{1, \dots, n\}} (\mathcal{L}_{m_j} \cap \mathcal{I}_{m_j})]$, as required. ■

Proof for **Proposition 6** in Section III-D: *If the n^{th} degree ($n \geq 1$) parallel interface system composed of DES $G_H, G_{L_1}, \dots, G_{L_n}, G_{I_1}, \dots, G_{I_n}$, is level-wise nonblocking and interface consistent with respect to the alphabet partition $\Sigma := \dot{\cup}_{k \in \{1, \dots, n\}} (\Sigma_{L_k} \dot{\cup} \Sigma_{R_k} \dot{\cup} \Sigma_{A_k}) \dot{\cup} \Sigma_H$, then*

$$\begin{aligned} & (\forall s \in \mathcal{H} \cap [\bigcap_{j \in \{1, \dots, n\}} (\mathcal{L}_j \cap \mathcal{I}_j)]) \\ & \quad (\exists h \in \Sigma_{IH}^*) (sh \in \mathcal{H}_m \cap [\bigcap_{j \in \{1, \dots, n\}} \mathcal{I}_{m_j}]) \end{aligned}$$

Proof:

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

Let $s \in \mathcal{H} \cap [\bigcap_{j \in \{1, \dots, n\}} (\mathcal{L}_j \cap \mathcal{I}_j)]$ (2)

We will now show this implies:

$$(\exists h \in \Sigma_{IH}^*) (sh \in \mathcal{H}_m \cap [\bigcap_{j \in \{1, \dots, n\}} \mathcal{I}_{m_j}])$$

We start by examining the 1^{st} serial extraction system of the parallel system.

From (2) and **Proposition 2**, we have:

$$P_1(s) \in P_1(\mathcal{H}) \cap [\bigcap_{j \in \{2, \dots, n\}} P_1(\mathcal{I}_j)] \cap P_1(\mathcal{L}_1) \cap P_1(\mathcal{I}_1) = \mathcal{H}(1) \cap \mathcal{L}(1) \cap \mathcal{I}(1) \quad (3)$$

We note that system(1) is serial level-wise nonblocking, as the parallel system is level-wise nonblocking (by (1)). This implies:

$$(\exists h' \in \Sigma(1)^*) P_1(s)h' \in \mathcal{H}_m(1) \cap \mathcal{I}_m(1)$$

$$\Rightarrow P_{IH}(1) (P_1(s)h') \in L_m(G_H(1))$$

We next note that:

$$\begin{aligned}
P_{IH}(1) (P_1(s)h') &= P_{IH}(1) \cdot P_1(s) P_{IH}(1)(h') \\
&= P_{IH}(1) \cdot P_1(s) P_{IH}(1) \cdot P_{IH}(1)(h') \\
&= P_{IH}(1) (P_1(s)P_{IH}(1)(h'))(4)
\end{aligned}$$

$$\Rightarrow P_1(s)P_{IH}(1)(h') \in \mathcal{H}_m(1) \quad (5)$$

As $\Sigma_I(1) \subseteq \Sigma_{IH}(1)$, we can conclude by (4) that $P_I(1) (P_1(s)h') = P_I(1) (P_1(s)P_{IH}(1)(h'))$ and thus: $P_1(s)P_{IH}(1)(h') \in \mathcal{I}_m(1)$ (6)

We next note that as $\Sigma(1) \subseteq \Sigma$ and $\Sigma_{IH} = \Sigma_{IH}(1)$ (by **Proposition 2**), we can conclude $P_{IH}(h') = P_{IH}(1)(h')$.

$$\Rightarrow P_1(s)P_{IH}(h') \in \mathcal{H}_m(1) \cap \mathcal{I}_m(1)$$

We then take $h = P_{IH}(h')$ and we have: $h \in \Sigma_{IH}^*$ and $P_1(s)h \in \mathcal{H}_m(1) \cap \mathcal{I}_m(1)$ (7)

From (7), substituting for $\mathcal{H}_m(1)$ and $\mathcal{I}_m(1)$ (by **Proposition 2**) and noting $P_1(h) = h$ as $h \in \Sigma_{IH} \subseteq \Sigma(1)$, we have:

$$P_1(sh) \in P_1 \cdot P_{IH}^{-1}(L_m(G_H)) \cap [\bigcap_{j \in \{2, \dots, n\}} P_1 \cdot P_{I_j}^{-1}(L_m(G_{I_j}))] \cap P_1 \cdot P_{I_1}^{-1}(L_m(G_{I_1}))$$

From **Proposition 2**, we have $\Sigma_{IH} \subseteq \Sigma(1)$ and $\Sigma_{I_j} \subseteq \Sigma(1)$ (for $j \in \{1, \dots, n\}$). We can now apply **Proposition 4** twice by taking $[\Sigma_a = \Sigma(1), \Sigma_b = \Sigma_{IH}, \text{ and } L_b = L_m(G_H)]$ and then $[\Sigma_a = \Sigma(1), \Sigma_b = \Sigma_{I_j}, \text{ and } L_b = L_m(G_{I_j})]$ and conclude:

$$sh \in P_{IH}^{-1}(L_m(G_H)) \cap [\bigcap_{j \in \{1, \dots, n\}} P_{I_j}^{-1}(L_m(G_{I_j}))]$$

$$\Rightarrow sh \in \mathcal{H}_m \cap [\bigcap_{j \in \{1, \dots, n\}} \mathcal{I}_{m_j}]$$

Combining with (7), we thus have $h \in \Sigma_{IH}^*$ with the required property that $sh \in \mathcal{H}_m \cap [\bigcap_{j \in \{1, \dots, n\}} \mathcal{I}_{m_j}]$ ■

Proof for Proposition 7 in Section III-D: *If the n^{th} degree ($n \geq 1$) parallel interface system composed of DES $G_H, G_{L_1}, \dots, G_{L_n}, G_{I_1}, \dots, G_{I_n}$, is level-wise nonblocking and interface consistent with respect to the alphabet partition $\Sigma := \dot{\cup}_{k \in \{1, \dots, n\}} (\Sigma_{L_k} \dot{\cup} \Sigma_{R_k} \dot{\cup} \Sigma_{A_k}) \dot{\cup} \Sigma_H$, then*

$$(\forall s \in \mathcal{H} \cap [\bigcap_{j \in \{1, \dots, n\}} (\mathcal{L}_{m_j} \cap \mathcal{I}_{m_j})]) (\forall h \in \Sigma_{IH}^*)$$

$$(sh \in \mathcal{H}_m \cap [\bigcap_{j \in \{1, \dots, n\}} \mathcal{I}_{m_j}]) \Rightarrow (\exists u \in \Sigma^*)(su \in \mathcal{H}_m \cap [\bigcap_{j \in \{1, \dots, n\}} (\mathcal{L}_{m_j} \cap \mathcal{I}_{m_j})] \wedge P_{IH}(u) = h)$$

Proof:

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

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

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

Iterative step:

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

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

As $P_i(h) = h$ since $h \in \Sigma_{IH}^* \subseteq \Sigma(i)$ (by **Proposition 2**), we can conclude:

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

$\Rightarrow P_i(s)h \in \mathcal{H}_m(i) \cap \mathcal{I}_m(i)$ (by **Proposition 2**) (3)

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

From **Proposition 2**, we thus have:

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

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 **Proposition 11** 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 2**), we can conclude $P_{IH}(u_i) = P_{IH}(i)(u_i) = h$. (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 2**), and $\mathcal{I}_m(i)$, we have:

$$P_i(su_i) \in P_i \cdot P_{IH}^{-1}(L_m(G_H)) \cap [\bigcap_{j \in \{1, \dots, n\}} P_i \cdot P_{I_j}^{-1}(L_m(G_{I_j}))] \cap P_i \cdot P_{IL_i}^{-1}(L_m(G_{L_i})) \quad (8)$$

From **Proposition 2**, we have $\Sigma_{IH} \subseteq \Sigma(i)$, $\Sigma_{I_j} \subseteq \Sigma(i)$ (for $j \in \{1, \dots, n\}$), and $\Sigma_{IL_i} \subseteq \Sigma(i)$. We can now apply **Proposition 4** three times by taking $[\Sigma_a = \Sigma(i)]$,

$\Sigma_b = \Sigma_{IH}$, and $L_b = L_m(G_H)$], $[\Sigma_a = \Sigma(i)$, $\Sigma_b = \Sigma_{I_j}$, and $L_b = L_m(G_{I_j})$], and then $[\Sigma_a = \Sigma(i)$, $\Sigma_b = \Sigma_{IL_i}$, and $L_b = L_m(G_{L_i})$], and conclude:

$$\begin{aligned} & su_i \in P_{IH}^{-1}(L_m(G_H)) \cap [\cap_{j \in \{1, \dots, n\}} P_{I_j}^{-1}(L_m(G_{I_j}))] \cap P_{IL_i}^{-1}(L_m(G_{L_i})) \\ \Rightarrow & su_i \in \mathcal{H}_m \cap [\cap_{j \in \{1, \dots, n\}} \mathcal{I}_{m_j}] \cap \mathcal{L}_{m_i} \text{ and } P_{IH}(u_i) = h \text{ (by (7))}, \text{ as required.} \end{aligned}$$

Iterative step complete.

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

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

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

We know that the set is non-empty for the following reasons:

- For each $i \in \{1, \dots, n\}$, we have $u_i \in \Sigma(i)^*$ where:

$$\Sigma(i) := \Sigma_{IH} \dot{\cup} \Sigma_{L_i} = \Sigma - (\dot{\cup}_{j \in \{1, \dots, i-1, i+1, \dots, n\}} \Sigma_{L_j}).$$
- 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 $(\forall i \in \{1, \dots, n\}) P_i(u) = u_i$. As $\Sigma_{IH} \subseteq \Sigma(i)$ (by **Proposition 2**) 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: $sh \in \mathcal{H}_m \cap [\cap_{j \in \{1, \dots, n\}} \mathcal{I}_{m_j}]$

$$\Rightarrow P_{IH}(sh) = P_{IH}(su) \in L_m(G_H) \text{ by (11).}$$

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

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

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

From (9), we have $su_j \in \mathcal{L}_{m_j}$ for $j \in \{1, \dots, n\}$. 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 2**), we can conclude $P_{IL_j}(u) = P_{IL_j}(u_j)$ and thus: $su \in \mathcal{L}_{m_j}$

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

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

Proof for **Proposition 8** in Section III-E: *If the n^{th} degree ($n \geq 1$) parallel interface system composed of plant components $\mathcal{G}_H, \mathcal{G}_{L_1}, \dots, \mathcal{G}_{L_n}$, supervisors $\mathcal{S}_H, \mathcal{S}_{L_1}, \dots, \mathcal{S}_{L_n}$, and interfaces G_{I_1}, \dots, G_{I_n} , is level-wise controllable with respect to the alphabet partition $\Sigma := \dot{\cup}_{k \in \{1, \dots, n\}} (\Sigma_{L_k} \dot{\cup} \Sigma_{R_k} \dot{\cup} \Sigma_{A_k}) \dot{\cup} \Sigma_H$, then*

$$(\forall j \in \{1, \dots, n\}) (\forall s \in L(\mathbf{Plant}) \cap \mathbf{L}_{S_j} \cap \mathcal{I}_j) \quad \text{Elig}_{L(\mathbf{Plant})}(s) \cap \Sigma_u \subseteq \text{Elig}_{\mathbf{L}_{S_j} \cap \mathcal{I}_j}(s)$$

Proof: Assume that the n^{th} degree ($n \geq 1$) parallel interface system is level-wise controllable. **(1)**

$$\text{Let } j \in \{1, \dots, n\}, s \in L(\mathbf{Plant}) \cap \mathbf{L}_{S_j} \cap \mathcal{I}_j, \text{ and } \sigma \in \text{Elig}_{L(\mathbf{Plant})}(s) \cap \Sigma_u. \quad \text{(2)}$$

We will now show that this implies $\sigma \in \text{Elig}_{\mathbf{L}_{S_j} \cap \mathcal{I}_j}(s)$.

It's sufficient to show that $s\sigma \in \mathbf{L}_{S_j} \cap \mathcal{I}_j$.

$$\text{We first note that } s, s\sigma \in \mathbf{H} \cap [\cap_{k \in \{1, \dots, n\}} \mathbf{L}_k] = L(\mathbf{Plant}) \text{ by (2).} \quad \text{(3)}$$

We have two cases: **I)** $\sigma \notin \Sigma_{IL_j}$ and **II)** $\sigma \in \Sigma_{IL_j}$.

case I)

Assume $\sigma \notin \Sigma_{IL_j}$. This implies: $P_{IL_j}(\sigma) = \epsilon$, where ϵ is the empty string.

$$\Rightarrow P_{IL_j}(s\sigma) = P_{IL_j}(s)P_{IL_j}(\sigma) = P_{IL_j}(s), \text{ as the natural projection is catenative. Similarly, we have } P_{I_j}(s\sigma) = P_{I_j}(s) \text{ as } \sigma \notin \Sigma_{I_j} \text{ since } \Sigma_{I_j} \subseteq \Sigma_{IL_j}. \quad \text{(4)}$$

From **(2)**, we have $s \in \mathbf{L}_{S_j} \cap \mathcal{I}_j = P_{IL_j}^{-1}L(\mathcal{S}_{L_j}) \cap P_{I_j}^{-1}(L(G_{I_j}))$.

$$\Rightarrow P_{IL_j}(s) \in L(\mathcal{S}_{L_j}) \text{ and } P_{I_j}(s) \in L(G_{I_j})$$

$$\Rightarrow P_{IL_j}(s\sigma) \in L(\mathcal{S}_{L_j}) \text{ and } P_{I_j}(s\sigma) \in L(G_{I_j}) \text{ by (4).}$$

$$\Rightarrow s\sigma \in \mathbf{L}_{S_j} \cap \mathcal{I}_j.$$

Case I complete.

case II)

Assume $\sigma \in \Sigma_{IL_j}$.

We now examine $\text{system}(j)$, the j^{th} serial system extraction of our parallel system.

We first note that we have $\sigma \in \Sigma_{IL}(j)$ as $\Sigma_{IL}(j) = \Sigma_{IL_j}$ by **Proposition 3**.

$$\Rightarrow \sigma \in \Sigma(j) \supseteq \Sigma_{IL}(j)$$

$\Rightarrow P_j(\sigma) = \sigma$. See Section III-C for the definition of the natural projection P_j .

$$\Rightarrow P_j(s\sigma) = P_j(s)\sigma$$

From **(1)**, we can conclude that $\text{system}(j)$ is serial level-wise controllable. **(5)**

As $s, s\sigma \in \mathbf{H} \cap [\cap_{k \in \{1, \dots, n\}} \mathbf{L}_k]$ by **(3)**, we have $s, s\sigma \in \mathbf{L}_j$.

$$\Rightarrow P_j(s) \in P_j \mathbf{L}_j \text{ and } P_j(s\sigma) = P_j(s)\sigma \in P_j \mathbf{L}_j$$

$$\Rightarrow P_j(s), P_j(s)\sigma \in \mathbf{L}(j), \text{ by } \mathbf{Proposition 3}.$$

As we have $\sigma \in \Sigma_u$ from **(2)**, we can conclude $\sigma \in \text{Elig}_{\mathbf{L}(j)}(P_j(s)) \cap \Sigma_u$.

From **(2)**, we have $s \in L(\mathbf{Sup})$ and thus $s \in \mathbf{L}_{\mathcal{S}_j} \cap \mathcal{I}_j$.

$$\Rightarrow P_j(s) \in P_j \mathbf{L}_{\mathcal{S}_j} \cap P_j \mathcal{I}_j$$

$$\Rightarrow P_j(s) \in \mathbf{L}_{\mathcal{S}}(j) \cap \mathcal{I}(j), \text{ by } \mathbf{Proposition 3}.$$

We now have $P_j(s) \in \mathbf{L}(j) \cap \mathbf{L}_{\mathcal{S}}(j) \cap \mathcal{I}(j)$ and $\sigma \in \text{Elig}_{\mathbf{L}(j)}(P_j(s)) \cap \Sigma_u$ and can conclude by **point II** of the serial level-wise controllable definition that:

$$\sigma \in \text{Elig}_{\mathbf{L}_{\mathcal{S}}(j) \cap \mathcal{I}(j)}(P_j(s)) \text{ and thus } P_j(s)\sigma = P_j(s\sigma) \in \mathbf{L}_{\mathcal{S}}(j) \cap \mathcal{I}(j)$$

Substituting for $\mathbf{L}_{\mathcal{S}}(j)$ and $\mathcal{I}(j)$ (by **Proposition 3**) gives: $P_j(s\sigma) \in P_j P_{IL_j}^{-1} L(\mathcal{S}_{L_j}) \cap P_j P_{I_j}^{-1} L(G_{I_j})$

We note that since $\Sigma_{IL}(j) = \Sigma_{IL_j}$ and $\Sigma_I(j) = \Sigma_{I_j}$ we have $\Sigma_{IL_j} \subseteq \Sigma(j)$ and $\Sigma_{I_j} \subseteq \Sigma(j)$. We can thus apply **Proposition 4** twice, taking first $\Sigma_a = \Sigma(j)$, $\Sigma_b = \Sigma_{IL_j}$, and $L_b = L(\mathcal{S}_{L_j})$ and then $\Sigma_a = \Sigma(j)$, $\Sigma_b = \Sigma_{I_j}$, and $L_b = L(G_{I_j})$. We can thus conclude:

$$s\sigma \in P_{IL_j}^{-1} L(\mathcal{S}_{L_j}) \cap P_{I_j}^{-1} L(G_{I_j}) = \mathbf{L}_{\mathcal{S}_j} \cap \mathcal{I}_j$$

Case II complete.

By **Cases I** and **II**, we have $s\sigma \in \mathbf{L}_{\mathcal{S}_j} \cap \mathcal{I}_j$, as required. ■

Proof for **Proposition 9** in Section III-E: If the n^{th} degree ($n \geq 1$) parallel interface system composed of plant components $\mathcal{G}_H, \mathcal{G}_{L_1}, \dots, \mathcal{G}_{L_n}$, supervisors $\mathcal{S}_H, \mathcal{S}_{L_1}, \dots, \mathcal{S}_{L_n}$, and interfaces G_{I_1}, \dots, G_{I_n} , is level-wise controllable with respect to the alphabet partition $\Sigma :=$

$\dot{\cup}_{k \in \{1, \dots, n\}} (\Sigma_{L_k} \dot{\cup} \Sigma_{R_k} \dot{\cup} \Sigma_{A_k}) \dot{\cup} \Sigma_H$, then

$$(\forall s \in L(\mathbf{Plant}) \cap \mathbf{H}_S \cap [\cap_{k \in \{1, \dots, n\}} \mathcal{I}_k]) \quad \text{Elig}_{L(\mathbf{Plant}) \cap [\cap_{k \in \{1, \dots, n\}} \mathcal{I}_k]}(s) \cap \Sigma_u \subseteq \text{Elig}_{\mathbf{H}_S}(s)$$

Proof:

Assume that the n^{th} degree ($n \geq 1$) parallel interface system is level-wise controllable. (1)

Let $s \in L(\mathbf{Plant}) \cap \mathbf{H}_S \cap [\cap_{k \in \{1, \dots, n\}} \mathcal{I}_k]$, and $\sigma \in \text{Elig}_{L(\mathbf{Plant}) \cap [\cap_{k \in \{1, \dots, n\}} \mathcal{I}_k]}(s) \cap \Sigma_u$ (2)

We will now show that this implies $\sigma \in \text{Elig}_{\mathbf{H}_S}(s)$

It's sufficient to show that $s\sigma \in \mathbf{H}_S$

We first note that:

$$s, s\sigma \in \mathbf{H} \cap [\cap_{k \in \{1, \dots, n\}} (\mathbf{L}_k \cap \mathcal{I}_k)] = L(\mathbf{Plant}) \cap [\cap_{k \in \{1, \dots, n\}} \mathcal{I}_k] \text{ by (2). (3)}$$

By examining the definition of $\Sigma(j)$ for some $j \in \{1, \dots, n\}$ (see definition of j^{th} serial system extraction: general form in Section 7), we see that $\Sigma = \cup_{k \in \{1, \dots, n\}} \Sigma(k)$

$$\Rightarrow (\exists j \in \{1, \dots, n\}) \sigma \in \Sigma(j) \quad (4)$$

We use this j and note that by (1), we can conclude that $\text{system}(j)$, the j^{th} serial system extraction of our parallel system, is serial level-wise controllable. (5)

From (2) and (3), we have $s \in \mathbf{H} \cap \mathbf{H}_S \cap [\cap_{k \in \{1, \dots, n\}} \mathcal{I}_k]$

$$\Rightarrow P_j(s) \in P_j(\mathbf{H}) \cap P_j(\mathbf{H}_S) \cap [\cap_{k \in \{1, \dots, n\}} P_j(\mathcal{I}_k)]$$

See Section III-C for the definition of the natural projection, P_j .

$$\Rightarrow P_j(s) \in \mathbf{H}(j) \cap \mathcal{I}(j) \cap \mathbf{H}_S(j), \text{ by Proposition 3.}$$

Similarly, from (3) we can conclude $P_j(s\sigma) \in \mathbf{H}(j) \cap \mathcal{I}(j)$

We next note that $\sigma \in \Sigma(j)$ (from (4)) implies that $P_j(s\sigma) = P_j(s)\sigma$.

$$\Rightarrow \sigma \in \text{Elig}_{\mathbf{H}(j) \cap \mathcal{I}(j)}(P_j(s)) \cap \Sigma_u$$

We can now conclude by **point III** of the serial level-wise controllable definition that:

$$\sigma \in \text{Elig}_{\mathbf{H}_S(j)}(P_j(s)) \text{ and thus } P_j(s)\sigma = P_j(s\sigma) \in \mathbf{H}_S(j)$$

$$\Rightarrow P_j(s\sigma) \in P_j(\mathbf{H}_S), \text{ by Proposition 3.}$$

$$\Rightarrow P_j(s\sigma) \in P_j P_{IH}^{-1} L(\mathcal{S}_H)$$

As $\Sigma_{IH} = \Sigma_{IH}(j)$ by **Proposition 3**, we have $\Sigma_{IH} \subseteq \Sigma(j)$. We can thus apply **Proposition 4** by taking $\Sigma_a = \Sigma(j)$, $\Sigma_b = \Sigma_{IH}$, and $L_b = L(\mathcal{S}_H)$ and thus conclude:

$$s\sigma \in P_{IH}^{-1}L(\mathcal{S}_H) = \mathbf{H}_S, \text{ as required.} \quad \blacksquare$$

APPENDIX II

SERIAL CASE PROPOSITIONS

We now restate some key propositions for the serial case that we will need in the proofs for the parallel case. For a detailed discussion and proofs, see Part I of this paper [1].

Proposition 10: 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

$$(\forall s \in \mathcal{H} \cap \mathcal{L} \cap \mathcal{I})$$

$$(\exists l \in \Sigma_{IL}^*)(sl \in \mathcal{H} \cap \mathcal{L}_m \cap \mathcal{I}_m)$$

Proposition 11: 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

$$(\forall s \in \mathcal{H} \cap \mathcal{L} \cap \mathcal{I}_m)(\forall h \in \Sigma_{IH}^*)$$

$$sh \in \mathcal{H}_m \cap \mathcal{I}_m \Rightarrow (\exists u \in \Sigma^*)(su \in \mathcal{H}_m \cap \mathcal{L}_m \cap \mathcal{I}_m) \wedge (P_{IH}(u) = h) \wedge (P_I(u) \in \{\epsilon\} \cup \Sigma_R \cdot \Sigma_I^*)$$

APPENDIX III

ALGORITHM FOR SERIAL INTERFACE CONSISTENCY

In this section we present an algorithm for evaluating **Points 5 and 6** of the serial interface consistent definition (Def. 2). We assume that all DES are reachable, deterministic, and have finite state and event sets, and that we are given $G_L := (Y_L, \Sigma_{G_L}, \delta_L, y_{L_o}, Y_{L_m})$, $G_I := (X, \Sigma_{G_I}, \xi, x_o, X_m)$, and the map **Answer** : $X \rightarrow \mathbf{Pwr}(\Sigma_A)$ defined as:

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

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

and $y \in Y$ as follows:

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

To avoid repeating existing algorithms, we will assume we already have the DES $G_{IL} := G_L \parallel G_I = (Y_{IL} \subseteq Y_L \times X, \Sigma_{G_{IL}}, \delta_{IL}, y_{ILo}, Y_{ILm})$ and the inverse transition function δ_{IL}^{-1} , which can be created using CTCT [7]. We will also use the variables Y_{ck_mk} (states of G_{IL} that are in $Y_{IL} \cap [(Y_L - Y_{Lm}) \times X_m]$), Y_{fnd} , Y_{pend} , and Σ_{-fnd} .

$$Y_{ck_mk} := \emptyset$$

for $y \in (Y_L \times X_m) \cap Y_{IL}$

if ($y \notin Y_{ILm}$)

$$Y_{ck_mk} := Y_{ck_mk} \cup \{y\};$$

for ρ **in** Σ_R

if ($\delta_{IL}(y, \rho)!$)

$$(y_L, x) := \delta_{IL}(y, \rho);$$

$$Y_{fnd} := Y_{pend} := \{(y_L, x)\};$$

$$\Sigma_{-fnd} := \mathbf{Answer}(x) ;$$

while ($Y_{pend} \neq \emptyset$ **and** $\Sigma_{-fnd} \neq \emptyset$)

$$y' := \mathbf{pop}(Y_{pend});$$

for σ **in** Σ_{IL}

if ($\delta_{IL}(y', \sigma)!$)

$$y'' := \delta_{IL}(y', \sigma);$$

$$\Sigma_{-fnd} := \Sigma_{-fnd} - \{\sigma\};$$

if ($(\sigma \in \Sigma_L) \wedge (y'' \notin Y_{fnd})$)

$$Y_{fnd} := Y_{fnd} \cup \{y''\};$$

push(y'' , Y_{pend});

if ($\Sigma_{-fnd} \neq \emptyset$) **then**

return “point 5 fails”;

if ($Y_{ck_mk} = \emptyset$)

return “points 5 and 6 pass”;

$$Y_{fnd} := Y_{pend} := Y_{ILm};$$

```

while ( $Y_{pend} \neq \emptyset$ )
   $y := \mathbf{pop}(Y_{pend});$ 
  for  $\sigma$  in  $\Sigma_L$ 
    if ( $Y := \delta_{IL}^{-1}(y, \sigma) \neq \emptyset$ )
      for  $y'$  in  $Y$ 
        if ( $y' \notin Y_{fnd}$ )
           $Y_{fnd} := Y_{fnd} \cup \{y'\};$ 
          push( $y', Y_{pend}$ );
           $Y_{ck\_mk} := Y_{ck\_mk} - \{y'\};$ 
          if ( $Y_{ck\_mk} = \emptyset$ )
            return “points 5 and 6 pass” ;
return “point 6 fails” ;

```

REFERENCES

- [1] R. Leduc, B. Brandin, M. Lawford, and W. M. Wonham, “Hierarchical interface-based supervisory control, part I: Serial case,” submitted to IEEE Trans. Automatic Control, 2003.
- [2] R. Leduc, B. Brandin, W. M. Wonham, and M. Lawford, “Hierarchical interface-based supervisory control: Serial case,” in *Proc. of 40th Conf. Decision Contr.*, Orlando, USA, December 2001, pp. 4116–4121.
- [3] R. Leduc, M. Lawford, and W. M. Wonham, “Hierarchical interface-based supervisory control: AIP example,” in *Proc. of 39th Annual Allerton Conference on Comm., Contr., and Comp.*, Oct 2001, pp. 396–405.
- [4] R. Leduc, W. M. Wonham, and M. Lawford, “Hierarchical interface-based supervisory control: Parallel case,” in *Proc. of 39th Annual Allerton Conference on Comm., Contr., and Comp.*, Oct 2001, pp. 386–395.
- [5] 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.
- [6] R. Leduc, “Hierarchical interface-based supervisory control,” Ph.D. dissertation, Department of Electrical and Computer Engineering, University of Toronto, Toronto, Ont, 2002.
- [7] W. M. Wonham, *Notes on Control of Discrete-Event Systems*, Department of Electrical and Computer Engineering, University of Toronto, July 2003. Notes and CTCT software can be downloaded at <http://www.control.toronto.edu/DES/>.
- [8] S. Warshal, “A theorem on boolean matrices,” *Journal of the ACM*, vol. 9, no. 1, pp. 11–12, January 1962.
- [9] 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.
- [10] B. Brandin and F. Charbonnier, “The supervisory control of the automated manufacturing system of the AIP,” in *Proc. Rensselaer’s 1994 Fourth International Conference on Computer Integrated Manufacturing and Automation Technology*, Troy, Oct 1994, pp. 319–324.
- [11] 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.

- [12] Z. 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.
- [13] Z. Zhang and W. M. Wonham, "STCT: an efficient algorithm for supervisory control design," in *Proc. of SCODES 2001*, INRIA, Paris, July 2001.