Hierarchical Interface-based Supervisory Control: Parallel Case

R.J. Leduc,[1,2] W.M. Wonham,[1] and M. Lawford[2]

[1]Dept. of Electrical and Computer Engineering, University of Toronto
[2]Dept. of Computing and Software, McMaster University
email: leduc@control.toronto.edu, wonham@control.toronto.edu, lawford@mcmaster.ca

Abstract

In this paper we present a hierarchical method that decomposes a system 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 define the setting for the serial case ($n = 1$), and then generalise it for $n \geq 1$. We present a definition for an interface, and define a set of interface consistency properties that can be used to verify if a discrete-event system (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 constructed, offering significant savings in computational effort. Additionally, the development of clean interfaces facilitates re-use of the component subsystems.

1 Introduction

In the area of Discrete-Event Systems (DES), two common tasks are to verify that a composite system, based on a cartesian product of subsystems, is (i) nonblocking and (ii) controllable. The main obstacle to performing these tasks is the combinatorial explosion of the product state space. Although many methods have been developed to deal with this problem (modular control [1, 20, 24], decentralised control [14, 21, 25], model aggregation methods [2, 3, 6, 23, 26], and multi-level hierarchy [5, 9, 15, 16, 22, 27]), large-scale systems are still problematic, particularly for verification of nonblocking.

To deal with the complexity of large scale systems, the software engineering community has long advocated the decomposition of software into modules (components) that interact via well defined interfaces (e.g., [17, 18, 19]). Recently the supervisory control community has begun to advocate a similar approach [10, 13, 8, 11]. These approaches develop well defined interfaces between components to provide the structure to allow local checks to guarantee global properties such as controllability [8] or nonblocking [10].

In this paper, we present an interface-based hierarchical method to verify if a system is nonblocking and controllable, extending the work in [11]. For the purposes of the present paper, we restrict ourselves to bi-level systems where the system is split into a *high level subsystem* which interacts with $n \geq 1$ parallel *low level subsystems* via separate interface DES, which regulates the subsystems' interaction. The most significant feature that distinguishes the work from [8] is the results regarding nonblocking.

In the remainder of the paper we first describe the setting for the serial case ($n = 1$), which was introduced in [11]. We present a definition for an interface, and define a set of (local) consistency properties that can be used to verify if a discrete-event system is globally nonblocking and controllable. We then extend our definitions to the general case of $n \geq 1$. In the companion paper [12] we discuss the application of the method to a large manufacturing example with an estimated closed-loop state space size of 7×10^{21}.

2 Serial Case

With the serial case of *hierarchical interface-based supervisory control,* what we are proposing is a master-slave system, where a *high level subsystem* sends a command to a *low level subsystem,* which then performs the indicated task and sends back a reply. Figure 1 shows conceptually the structure and information flow of the system. We call this the serial case as communication occurs in a serial fashion between the two subsystems.

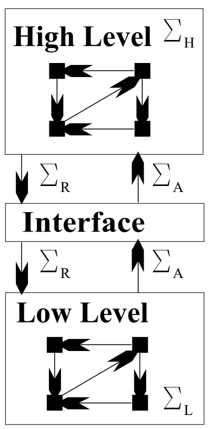

Figure 1: Interface Block Diagram.

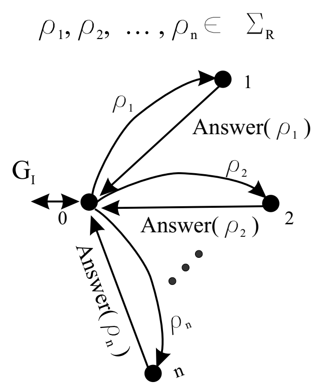

Figure 2: Interface Specification.

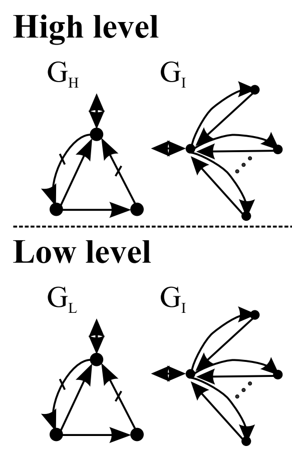

Figure 3: Two Tiered Structure of the System.

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

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

2.1 Interface Definitions and Notation

To define an *interface,* the designer selects a set of *request events*, and then for each *request event*, the designer defines a set of *answer events.* In essence, the designer defines a map **Answer** : $\Sigma_R \to \mathrm{Pwr}(\Sigma_A)$. For $\rho \in \Sigma_R$, **Answer**(ρ) is the set of possible answers the *low level subplant* could provide after receiving request ρ. For consistency, we add the constraints that the *low level subsystem* must provide at least one response for each request it receives, and that Σ_A does not contain any unused events. Figure 2 shows how an *interface* is expressed as a DES. The required structure for an *interface* is given by DES G_I.

For our setting, we assume the *high level subsystem* is modelled by DES G_H (defined over event set $\Sigma_H \cup \Sigma_R \cup \Sigma_A$), the *low level subsystem* by DES G_L (defined over event set $\Sigma_L \cup \Sigma_R \cup \Sigma_A$), and the interface by DES G_I (defined over $\Sigma_R \cup \Sigma_A$). Also, the *high level* will mean $\mathbf{sync}(G_H, G_I)$,[1] and the *low level* $\mathbf{sync}(G_L, G_I)$. The overall structure of the system is displayed in Figure 3.

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

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

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

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

2.2 Serial Interface Consistency and Nonblocking

We now present the interface requirements that the *system* must satisfy to ensure that it interacts with the *interface* correctly. We then define the nonblocking requirements each level must satisfy. Refer to [11] for a more detailed explanation of the requirements.

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

Multi-level Properties

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

High Level Properties

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

Low Level Properties

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

[1]The operation **sync** is the synchronous product operation from CTCT [24].

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

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

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

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

2.3 Serial Level-wise Controllability

For nonblocking we were only concerned with the *high* and *low level subsystems,* ignoring distinctions between plants and supervisors. For controllability, we need to split the subsystems into their plant and supervisor components. We will do so as shown in Figure 4.

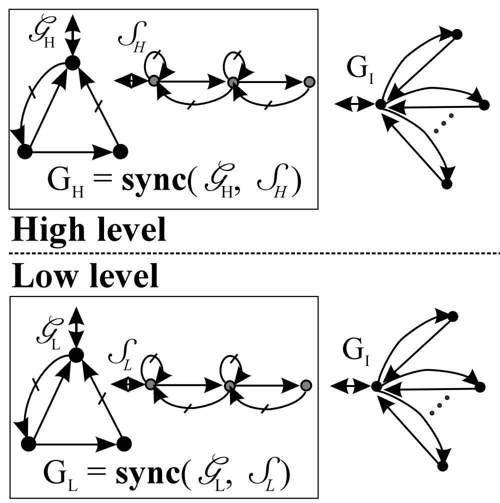

Figure 4: Plant and Supervisor Subplant Decomposition

We next define the *high level plant* to to be $\mathcal{G}_H$, and the *high level supervisor* to be $\mathcal{S}_H$ (both defined over event set Σ_{IH}). Similarly, the *low level plant* and *supervisor* are $\mathcal{G}_L$ and $\mathcal{S}_L$ (defined over event set Σ_{IL}). We can now define our *flat supervisor* and *plant* as well as some useful languages as follows:

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

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

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

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

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

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

3 Parallel Case

In Section 2, we described our method for the serial case where the number of *low levels* (n) is restricted to one. We now extend our work to the more general setting where we have $n \geq 1$ *low levels*. Figure 5 shows conceptually the structure and flow of information of such a system. In this new setting, we still have a single *high level,* but this time it is interacting with $n \geq 1$ independent low levels, communicating with each *low level* in parallel through a separate *interface.* We will refer to the number of *low levels*, n, as the *degree* of the system.

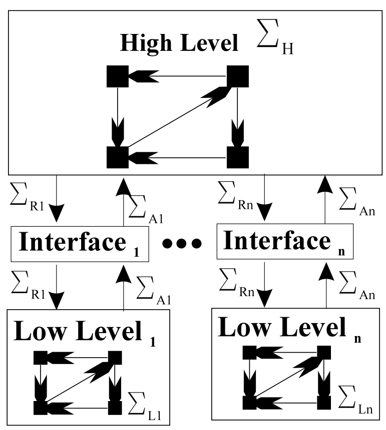

Figure 5: Parallel Interface Block Diagram.

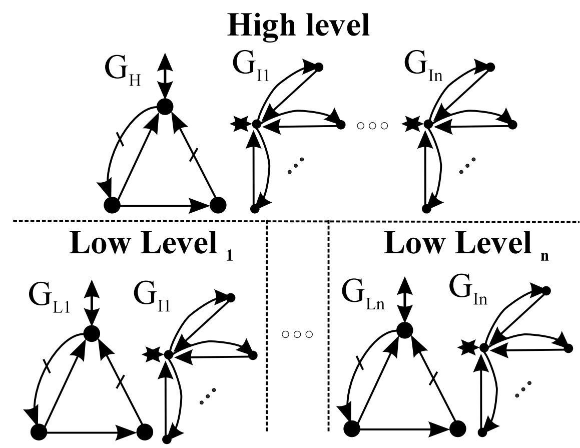

Figure 6: Two Tiered Structure of Parallel System

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

For an n^{th} degree parallel system, we assume the *high level subsystem* is modelled by DES G_H (defined over event set $\dot\cup_{j\in\{1,\ldots,n\}}[\Sigma_{R_j}\dot\cup\Sigma_{A_j}] \,\dot\cup\, \Sigma_H$). For $j \in \{1, \ldots, n\}$, the j^{th} *low level subsystem* is modelled by DES G_{L_j} (defined over event set $\Sigma_{L_j} \,\dot\cup\, \Sigma_{R_j} \,\dot\cup\, \Sigma_{A_j}$), the j^{th} *interface* by DES G_{I_j} (defined over event set $\Sigma_{R_j} \,\dot\cup\, \Sigma_{A_j}$), and that the overall system has the structure shown in Figure 6. Furthermore, we will refer to the j^{th} *low level* to mean $\mathbf{sync}(G_{L_j}, G_{I_j})$ and we will assume that the alphabet partition is specified by $\Sigma := \dot\cup_{j\in\{1,\ldots,n\}}[\Sigma_{L_j}\dot\cup\Sigma_{R_j}\dot\cup\Sigma_{A_j}] \,\dot\cup\, \Sigma_H$ and that the *flat system* is taken to be:

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

In order to simplify the notation in proofs, we now introduce the following event sets, natural projections, and useful languages. For the remainder of this section, the index j is defined to have range $\{1, \ldots, n\}$.

$$\begin{aligned}
\Sigma_{I_j} &:= \quad \Sigma_{R_j} \cup \Sigma_{A_j} \\
\Sigma_{IH} &:= \quad \cup_{j\in\{1,\ldots,n\}}\Sigma_{I_j} \,\cup\, \Sigma_H \\
\Sigma_{IL_j} &:= \quad \Sigma_{L_j} \cup \Sigma_{I_j}
\end{aligned}$$

$$
\begin{aligned}
P_{IH} : \Sigma^* \rightarrow & \quad \Sigma_{IH}^* \\
P_{IL_j} : \Sigma^* \rightarrow & \quad \Sigma_{IL_j}^* \\
P_{I_j} : \Sigma^* \rightarrow & \quad \Sigma_{I_j}^* \\
\mathcal{H} := P_{IH}^{-1}(L(G_H)), & \quad \mathcal{H}_m := P_{IH}^{-1}(L_m(G_H)) \quad \subseteq \Sigma^* \\
\mathcal{L}_j := P_{IL_j}^{-1}(L(G_{L_j})), & \quad \mathcal{L}_{m_j} := P_{IL_j}^{-1}(L_m(G_{L_j})) \quad \subseteq \Sigma^* \\
\mathcal{I}_j := P_{I_j}^{-1}(L(G_{I_j})), & \quad \mathcal{I}_{m_j} := P_{I_j}^{-1}(L_m(G_{I_j})) \quad \subseteq \Sigma^*
\end{aligned}
$$

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

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

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, achieved by 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:

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

3.2 Serial System Extraction

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

In this section, we introduce the concept of *serial system extractions* for an n^{th} degree $(n \geq 1)$ *parallel interface system,* shown conceptually in Figure 7 in terms of subsystems. Below we give the general form of the definition. The parallel subsystem form of the definition can be obtained by using the identities $G_H = \mathbf{sync}(\mathcal{G}_H, \mathcal{S}_H)$, $G_L = \mathbf{sync}(\mathcal{G}_L, \mathcal{S}_L)$, and $G_{L_j} = \mathbf{sync}(\mathcal{G}_{L_j}, \mathcal{S}_{L_j})$.

j^{th} **Serial System Extraction:** For the n^{th} degree $(n \geq 1)$ *parallel interface system* composed of DES $\mathcal{G}_H, \mathcal{G}_{L_1}, \ldots, \mathcal{G}_{L_n}, \mathcal{S}_H, \mathcal{S}_{L_1}, \ldots, \mathcal{S}_{L_n}, G_{I_1}, \ldots, G_{I_n}$, with alphabet

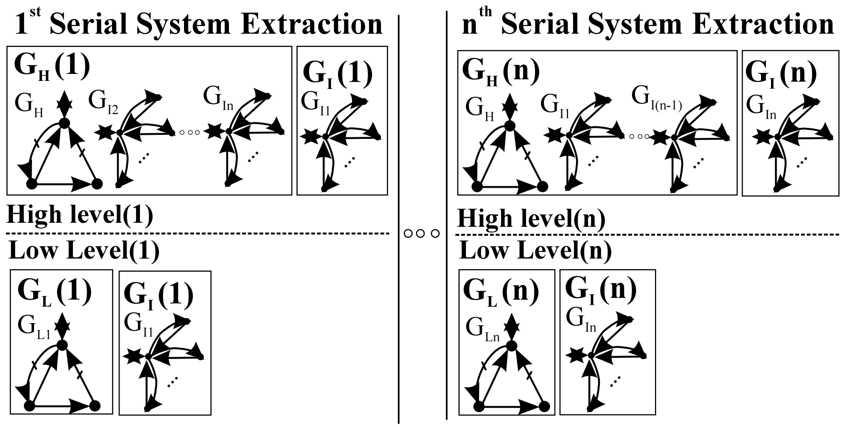

Figure 7: The Serial System Extraction

partition $\Sigma := \dot\cup_{k\in\{1,\ldots,n\}}[\Sigma_{L_k}\dot\cup\Sigma_{R_k}\dot\cup\Sigma_{A_k}] \,\dot\cup\, \Sigma_H$, the j^{th} *serial system extraction,* denoted by *system*(j), is composed of the following elements:

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

After examining the definition of the *serial system extraction,* we see that for $n = 1$, a *parallel interface system* reduces to a single *serial interface system.* We thus see that a serial system is a special case of a parallel system.

3.3 Parallel Case Definitions and Theorems

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

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

$(\forall j \in \{1,\ldots,n\})$ The j^{th} *serial system extraction* of the system is *serial interface consistent.*

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

$(\forall j \in \{1,\ldots,n\})$ The j^{th} *serial system extraction* of the system is *serial level-wise nonblocking.*

We now extend *serial level-wise controllability* to the parallel system case. We adopt the standard partition $\Sigma = \Sigma_u \dot\cup \Sigma_c$, splitting our alphabet into *uncontrollable* and *controllable events.*

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

$(\forall j \in \{1,\ldots,n\})$ The j^{th} *serial system extraction* of the system is *serial level-wise controllable.*

We now present our nonblocking theorem 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.*

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

$L(G) = \overline{L}_m(G)$, *where* $G = \textbf{sync}(G_H, G_{L1}, \ldots, G_{Ln}, G_{I1}, \ldots, G_{In})$

Proof: See [13].

Next, we present our 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.*

Theorem 2 *If the n^{th} degree $(n \geq 1)$* parallel interface system *composed of plant components $\mathcal{G}_H$, $\mathcal{G}_{L_1}, \ldots, \mathcal{G}_{L_n}$, supervisors $\mathcal{S}_H$, $\mathcal{S}_{L_1}, \ldots, \mathcal{S}_{L_n}$, and interfaces $G_{I_1}, \ldots, G_{I_n}$, is* level-wise controllable *with respect to the alphabet partition $\Sigma := \dot\cup_{k\in\{1,\ldots,n\}}[\Sigma_{L_k} \dot\cup \Sigma_{R_k} \dot\cup \Sigma_{A_k}]$ $\dot\cup\, \Sigma_H$, then*

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

where $\textbf{Plant} := \textbf{sync}(\mathcal{G}_H, \mathcal{G}_{L_1}, \ldots, \mathcal{G}_{L_n})$ *is the system's* flat plant, *and*
$\textbf{Sup} := \textbf{sync}(\mathcal{S}_H, \mathcal{S}_{L_1}, \ldots, \mathcal{S}_{L_n}, G_{I_1}, \ldots, G_{I_n})$ *is the system's* flat supervisor.

Proof: See [13].

4 Conclusions

Hierarchical interface-based supervisory control offers an effective method to model systems with a natural client-server architecture. The method offers an intuitive way to model and design the system. Using multiple *low level subsystems* allows the subsystems to be independently modelled and verified, but still allowing a high degree of concurrent operation. As each requirement can be verified using only one subsystem, the entire plant model never needs to be constructed or traversed (in computer memory), offering potentially significant savings in computation.

It is clear from the definitions in Sections 2, and 3, that once we have defined our *interface* and event partition, evaluating our *high* and *low level subsystems* for compliance can be done independently of each other. This means we can evaluate one *high (low) level subsystem* and use it with any *low (high) level subsystem* that satisfies the low (high) level portion of our definitions for the given *interface* and event partition. This provides us with the infrastructure required for component reuse.

We present a full example application of the theory based on the automated manufacturing system of the Atelier Inter-établissement de Productique (AIP) [4, 7] in the companion paper [12]. The AIP system is broken down into a *high level* and seven *low levels* corresponding to the three assembly stations and four transport Units. In total, the example contains 181 DES, with an estimated closed-loop state space of 7×10^{21}.

The analysis in [12] finds the system to be *interface consistent, level-wise nonblocking,* and *level-wise controllable.* Thus we can conclude by Theorems 1 and 2, that the *flat system* is nonblocking and that the system's *flat supervisor* is controllable for the *flat plant.* For further details of the application, we refer the reader to [12].

References

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

[2] R. Alur and T. Henzinger. Local liveness for compositional modelling of fair reactive systems. In *Proc. of 7th CAV*, LNCS, pages 166–179, 1995.

[3] A. Aziz, V. Singhal, and G. Swamy. Minimizing interacting finite state machines: A compositional approach to language containment. In *Proc. of IEEE Int. Conf. on Computer Design: VLSI in Computers and Processors*, pages 255–261, Cambridge, MA, Oct 1994.

[4] B. Brandin and F. Charbonnier. The supervisory control of the automated manufacturing system of the AIP. In *Proc. Rensselaer's 1994 4th Intl. Conf. on Computer Integrated Manufacturing and Automation Technology*, pages 319–324, Troy, Oct 1994.

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

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

[7] F. Charbonnier. Commande par supervision des systèmes à éénements discrets application à un site expérimental l'Atelier Inter-établissement de Productique. Tech. report, Laboratoire d'Automatique de Grenoble, Grenoble, France, 1994.

[8] 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.

[9] P. Gohari. A linguistic framework for controlled hierarchical DES. Master's thesis, Dept. of Electrical and Computer Engineering, University of Toronto, Toronto, Ont, 1998.

[10] R. Leduc, B. Brandin, and W.M. Wonham. Hierarchical interface-based non-blocking verification. In *Proc. of the Canadian Conf. on Elec. and Comp. Engineering*, pages 1–6, May 2000.

[11] R. Leduc, B. Brandin, W.M. Wonham, and M. Lawford. Hierarchical interface-based supervisory control: Serial case. Accepted to *40th Conf. Decision Contr.*, 2001.

[12] R. Leduc, M. Lawford, and W.M. Wonham. Hierarchical interface-based supervisory control: AIP example. Accepted to *39th Annual Allerton Conf. on Comm., Contr., and Comp.*, 2001.

[13] R.J Leduc, W.M. Wonham, and M. Lawford. Hierarchical interface based supervisory control: Bi-level systems. Tech. report, Systems Control Group, University of Toronto, Toronto, ON, Canada, 2001. To appear.

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

[15] H. Liu, J-C. Park, and R.E. Miller. On hybrid synthesis for hierarchical structured petri nets. Tech. report, Dept. of Comp. Sci., University of Maryland, College Park, MD, 1996.

[16] C. Ma. A computational approach to top-down hierarchical supervisory control of DES. Master's thesis, Dept of Electrical and Computer Engineering, University of Toronto, Toronto, Ont, 1999.

[17] D. L. Parnas. On the criteria to be used in decomposing systems into modules. *Communications of the ACM*, December:1053–1058, December 1972.

[18] D. L. Parnas. Use of abstract interfaces in the development of software for embedded computer systems. NRL Report 8047, Naval Research Laboratory, 1977.

[19] D. L. Parnas, P. C. Clements, and D. M. Weiss. The modular structure of complex systems. *IEEE Trans. on Software Engineering*, SE-11(3):259–66, March 1985.

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

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

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

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

[24] W.M. Wonham. *Notes on Control of Discrete-Event Systems.* Dept. of Electrical and Computer Engineering, University of Toronto, 1999. http://odin.control.toronto.edu/DES/.

[25] T. Yoo and S. Lafortune. A general architecture for decentralized supervisory control of discrete-event systems. In *Proceedings of WODES 2000*, pages 111–118, Ghent, Belgium, Aug 2000.

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

[27] M. Zhou, D. Wang, and I. Mayk. Using petri nets for object-oriented design of command and control systems. *Intl. Journal of Intelligent Control and Systems*, 2(2):287–300, 1998.